YAYASAN PRIMA AGUS TEKNIK

DASAR DESAIN GRAFIS

JILID 2

Dr. Mars Caroline Wibowo, ST, M.Mm.Tech.

DASAR DESAIN GRAFIS jilid 2

Penulis :
Dr. Mars Caroline Wibowo. S.T., M.Mm.Tech

ISBN : 9 786235 734477

Editor :
Dr. Mars Caroline Wibowo. S.T., M.Mm.Tech

Penyunting :
Dr. Joseph Teguh Santoso, S.Kom., M.Kom.

Desain Sampul dan Tata Letak :
Irdha Yunianto, S.Ds., M.Kom

Penebit :
Yayasan Prima Agus Teknik Bekerja sama dengan
Universitas Sains & Teknologi Komputer (Universitas STEKOM)

Redaksi :
Jl. Majapahit no 605 Semarang
Telp. (024) 6723456
Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

Distributor Tunggal :
Universitas STEKOM
Jl. Majapahit no 605 Semarang
Telp. (024) 6723456
Fax. 024-6710144
Email : info@stekom.ac.id

KATA PENGANTAR

Puji syukur kami panjatkan atas kehadirat Tuhan karena buku yang berjudul "*Dasar Desain Grafis: Jilid 2*" dapat terselesaikan. Grafis komputer adalah bidang yang berkembang pesat, sehingga spesifikasi pengetahuan itu adalah target yang bergerak. Oleh karena itu, dalam buku ini penulis melakukan yang terbaik untuk menghindari ketergantungan pada perangkat keras atau API tertentu. Pembaca didorong untuk melengkapi teks dengan dokumentasi yang relevan untuk perangkat lunak dan lingkungan perangkat keras mereka. Untungnya, budaya grafis komputer memiliki terminologi dan konsep standar yang cukup sehingga pembahasan dalam buku ini dapat dipetakan dengan baik ke sebagian besar lingkungan.

Buku ini dibagi menjadi 2 Jilid. Jilid 1 terdiri dari 13 Bab, Bab Pertama buku jilid 1 ini memberikan definisi beberapa terminologi dasar dan memberikan beberapa latar belakang sejarah, serta sumber informasi yang berkaitan dengan komputer grafis. Bab 2 ini tidak dimaksudkan untuk membahas materi secara menyeluruh; sebaliknya intuisi dan interpretasi geometris yamh sangat ditekankan. Bab 3 Dalam bab ini, kita membahas dasar-dasar gambar dan tampilan raster, dengan memberikan perhatian khusus pada ketidaklinieran tampilan standar. Detail arti sebenarnya dari gambar. nilai piksel yang berhubungan dengan intensitas cahaya penting untuk diingat ketika kita membahas komputasi gambar di bab selanjutnya.

Bab 4 mencakup metode dasar untuk pembangkitan sinar, perpotongan sinar, dan bayangan, yang cukup untuk mengimplementasikan pelacak sinar demonstrasi sederhana. Untuk sistem yang benar-benar berguna, teknik perpotongan sinar yang lebih efisien. Sedangkan Bab 5 meninjau aljabar linier dasar dari perspektif geometris, dengan fokus pada intuisi dan algoritma yang bekerja dengan baik dalam kasus dua dan tiga dimensi. Bab 6 membahas mengenai transformasi geometris seperti rotasi, translasi, scalling, dan proyeksi dapat diselesaikan dengan perkalian matriks, dan matriks transformasi yang digunakan untuk melakukan ini adalah pokok bahasan bab ini. Bab 7 menjelaskan bagaimana menggunakan transformasi matriks untuk mengekspresikan pandangan paralel atau perspektif. Transformasi dalam bab ini memproyeksikan titik 3D dalam *scene* (ruang dunia) ke titik 2D dalam gambar (ruang gambar), dan mereka akan memproyeksikan titik mana pun pada sinar tampilan piksel tertentu kembali ke posisi piksel tersebut dalam ruang gambar.

Bab 8 akan membahas tentang fokus pada dasar-dasar umum seperti pipeline fan rasterisasi. Dalam bab 9, dimulai dengan meringkas pengambilan sampel dan rekonstruksi menggunakan contoh konkret satu dimensi dari audio digital. Kemudian, kami akan menyajikan matematika dasar dan algoritma yang mendasari pengambilan sampel dan rekonstruksi dalam satu dan dua dimensi. Bab 10 menyajikan metode bayangan heuristik yang paling umum. Diantaranya adalah difus dan phongshading, metode lain nya adalah naungan artistik, menggunakan konvensi artistik untuk memberi warna pada objek. Bab 11 membahas penggunaan tekstur untuk merepresentasikan detail permukaan, bayangan, dan pantulan.

Meskipun ide dasarnya sederhana, beberapa masalah praktis memperumit penggunaan tekstur.

Bab 12 berbicara tentang beberapa kategori struktur data dasar dan tidak terkait yang termasuk di antara yang paling umum dan berguna: struktur mesh, struktur data spasial, grafis *scene,* dan array multidimensi bersusun. Bab terakhir dalam buku jilid 1 ini membahas beberapa teknik yang lebih menarik yang dapat digunakan untuk *ray-trace* berbagai scene yang lebih luas dan untuk memasukkan lebih banyak variasi efek.

Buku Jilid 2 yang berjudul sama ini memiliki 13 bab yang akan membahas implementasi dan permodelan Dasar grafis komputer. Bab pertama dalam buku jilid 2 ini, akan membahas mesin untuk operasi probabilitas. Teknik-teknik ini juga akan terbukti berguna untuk mengevaluasi integral rumit secara numerik menggunakan integrasi Monte Carlo. Bab 2 membahas tentang Kurva, yang mencakup dengan permodelan geometris. Bab ke 3 akan memberikan gambaran tentang teknik dan algoritma yang langsung digunakan untuk membuat dan memanipulasi gerak. Bab 4 menjadi panduan pengantar untuk perangkat keras grafis dan dapat digunakan sebagai dasar untuk serangkaian praktik mingguan yang menyelidiki perangkat keras grafis.

Dalam bab 5 ini, membahas masalah praktis pengukuran cahaya, biasanya disebut radiometri. Bab 6 membahas tentang Warna, yaitu membahas teori dan matematika tentang koreksi warna. Bab 7 memberikan gambaran sebagian tentang apa yang diketahui tentang persepsi visual pada orang. Sedangan bab 8 membahas tentang pemilihan algoritma nada yang dapat digunakan untuk menghitung tingkat adaptasi lokal untuk fungsi kompresi sigmoidal.

Dalam bab 9 ini, membahas metode dan menjelaskan cara membangun model implisit kerangka secara lebih rinci. Bab ke 10 dalam buku ini membahas tentang globaliluminasi, seperti apakah yang dinamakan globaliluminati dan kegunaannya akan dibahas dalam bab ini. Bab 11 membahas beberapa aspek yang paling penting secara visual dari sifat material dan beberapa model yang cukup sederhana yang berguna dalam menangkap sifat-sifat ini. Ada banyak model BRDF yang digunakan dalam grafis, dan model yang disajikan di sini dimaksudkan untuk memberikan gambaran tentang BRDF yang tidak menyebar.

Bab 12 penulis akan merinci pertimbangan khusus yang berlaku untuk grafis dalam pengembangan game, mulai dari platform tempat game dijalankan hingga proses produksi game. Bab akhir dalam buku jilid 2 ini akan menjelaskan tentang visualisasi. Visualisasi dapat digunakan untuk menghasilkan hipotesis baru ketika menjelajahi kumpulan data yang sama sekali tidak dikenal, untuk mengkonfirmasi hipotesis yang ada dalam kumpulan data yang dipahami sebagian, atau untuk menyajikan informasi tentang kumpulan data yang diketahui kepada audiens lain. Akhir kata semoga buku ini bermanfaat bagi para pembaca.

Semarang, Februari 2022

Penulis

Dr. Mars Caroline Wibowo, M.Mm.Tech.

DAFTAR ISI

BAB 14

SAMPLING

Banyak aplikasi dalam grafis memerlukan pengambilan sampel yang "adil" dari ruang yang tidak biasa, seperti ruang dari semua garis yang mungkin. Misalnya, kita mungkin perlu menghasilkan tepi acak dalam piksel, atau titik sampel acak pada piksel yang bervariasi dalam kepadatan sesuai dengan beberapa fungsi kepadatan. Bab ini menyediakan mesin untuk operasi probabilitas seperti itu. Teknik-teknik ini juga akan terbukti berguna untuk mengevaluasi integral rumit secara numerik menggunakan integrasi Monte Carlo, yang juga dibahas dalam bab ini.

14.1 INTEGRASI

Meskipun kata-kata "integral" dan "ukuran" sering kali tampak menakutkan, kata-kata itu berkaitan dengan beberapa konsep paling intuitif yang ditemukan dalam matematika, dan kata itu tidak perlu ditakuti. Untuk tujuan kami yang sangat tidak ketat, ukuran hanyalah fungsi yang memetakan himpunan bagian ke R+ dengan cara yang konsisten dengan gagasan intuitif kami tentang panjang, luas, dan volume. Misalnya, pada 2 Bidang Nyata R^2, kita memiliki ukuran luas A yang memberikan nilai pada sekumpulan titik di bidang tersebut. Perhatikan bahwa A hanyalah sebuah fungsi yang mengambil potongan bidang dan luas balik. Ini berarti domain dari A adalah semua himpunan bagian yang mungkin dari R^2, yang kita nyatakan sebagai himpunan daya $P(R^2)$. Dengan demikian, kita dapat mengkarakterisasi A dalam notasi panah:

$$A : P(\mathbb{R}^2) \to \mathrm{R}+.$$

Contoh penerapan ukuran luas menunjukkan bahwa luas persegi dengan panjang sisi satu adalah satu:

$$A([a, a + 1] \times [b, b+ 1]) = 1,$$

di mana (a,b) hanyalah sudut kiri bawah bujur sangkar. Perhatikan bahwa satu titik seperti (3,7) adalah himpunan bagian yang valid dari R2 dan memiliki luas nol: A((3,7)) = 0. Hal yang sama berlaku untuk himpunan titik S pada sumbu x, S = (x, y) sehingga (x,y) R^2 dan y = 0, yaitu, A(S) = 0. Himpunan seperti itu disebut himpunan ukuran nol.

Untuk mempertimbangkan ukuran, suatu fungsi harus mematuhi properti seperti area tertentu. Misalnya, kita memiliki fungsi μ : P(S) → R+. Agar menjadi besaran, kondisi berikut harus benar:

1. Ukuran himpunan kosong adalah nol: ($\emptyset$) = 0
2. Tema dari dua himpunan berbeda adalah jumlah dari besaran mereka sendiri. Aturan ini dengan persimpangan yang mungkin adalah

$$\mu(A \cup B) = \mu(A) + \mu(B) - \mu(A \cap B),$$

di mana adalah operator gabungan himpunan dan adalah operator persimpangan himpunan. Ketika kami benar-benar menghitung ukuran, kami biasanya menggunakan integrasi. Kita dapat menganggap integrasi sebagai notasi yang benar-benar sederhana:

$$A(S) = \int_{x \in S} dA(x)$$

Anda dapat secara informal membaca ruas kanan sebagai "ambil semua titik x di daerah S, dan jumlahkan daerah diferensial yang terkait." Integral sering ditulis dengan cara lain termasuk

$$\int_S dA, \int_{x \in S} dx, \int_{x \in S} dAx \int_x dx$$

Semua rumus di atas mewakili "luas daerah S." Kami akan tetap menggunakan yang pertama, karena sangat bertele-tele sehingga menghindari ambiguitas. Untuk mengevaluasi integral semacam itu secara analitik, kita biasanya perlu meletakkan beberapa sistem koordinat dan menggunakan trik kalkulus untuk menyelesaikan persamaan. Tetapi jangan takut jika keterampilan itu telah memudar, karena kita biasanya harus memperkirakan integral secara numerik, dan itu hanya membutuhkan beberapa teknik sederhana yang akan dibahas nanti dalam bab ini.

Beri nama sebuah himpunan S, kita selalu dapat membuat pengukuran baru dengan pembobotan dengan fungsi nonnegatif $w : S \rightarrow R^+$. Ini paling baik dinyatakan dalam notasi integral. Sebagai contoh, kita dapat memulai dengan contoh pengukuran luas sederhana pada $[0,1]^2$:

$$\int_{x \in [0,1]^2} dA(x)$$

dan kita dapat menggunakan ukuran "berbobot radial" dengan memasukkan fungsi pembobotan radius kuadrat:

$$\int_{x \in [0,1]^2} ¿ \vee x \vee ¿^2 dA(x)$$

Untuk mengevaluasi ini secara analitik, kita dapat memperluas menggunakan sistem koordinat Cartesian dengan $dA = d_x d_y$:

$$\int_{x \in [0,1]^2} ¿ \vee x \vee ¿^2 dA(x) = \int_{z-0}^{1} \int_{y=0}^{1} \left(x^2 + y^2\right) dx\, dy$$

Kuncinya di sini adalah bahwa Anda memikirkan suku $||x||^2$ yang dikawinkan dengan suku dA, dan bahwa ini bersama-sama membentuk ukuran baru, kita dapat menyebut ukuran itu . Ini akan memungkinkan kita untuk menulis (S) alih-alih integral keseluruhan. Jika ini menurut Anda hanya sebagai sekelompok notasi dan pembukuan, Anda benar. Tapi itu memungkinkan kita untuk menuliskan persamaan yang kompak atau diperluas tergantung pada preferensi kita.

14.2 UKURAN DAN RATA-RATA

Tindakan benar-benar mulai membuahkan hasil ketika mengambil rata-rata suatu fungsi. Anda hanya dapat mengambil rata-rata sehubungan dengan ukuran tertentu, dan Anda ingin memilih ukuran yang "alami" untuk aplikasi atau domain. Setelah suatu ukuran dipilih, rata-rata fungsi dari suatu daerah terhadap ukuran adalah:

$$rata-rata(f) = \frac{\int_{z \in S} f(x) d\mu(x)}{\int_{x \in S} d\mu(x)}$$

Misalnya, rata-rata fungsi $f(x, y) = x^2$ di atas $[0,2]^2$ terhadap luas daerah adalah:

$$rata-rata(f)=\frac{\int_{x-0}^{2}\int_{y-0}^{2} x^2\,dx\,dy}{\int_{z-0}^{2}\int_{y-0}^{2} dx\,dy}=\frac{4}{3}$$

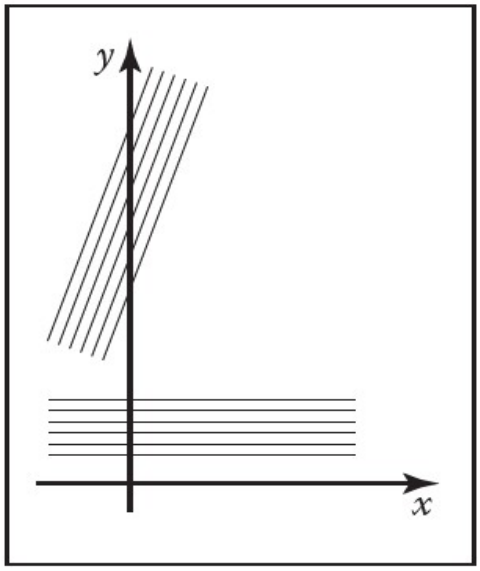
Mesin ini membantu memecahkan masalah yang tampaknya sulit di mana memilih ukuran adalah bagian yang sulit. Masalah seperti itu sering muncul dalam geometri integral, bidang yang mempelajari ukuran pada entitas geometris, seperti garis dan bidang. Misalnya, seseorang mungkin ingin mengetahui panjang rata-rata sebuah garis yang melalui [0,1]2. Artinya, menurut definisi,

$$rata-rata(panjang)=\frac{\int_{garis\ L\ pada\ [0,1]^2} panjang(L)\,d\mu(L)}{\int_{garis\ L\ pada\ [0,1]^2} d\mu(L)}$$

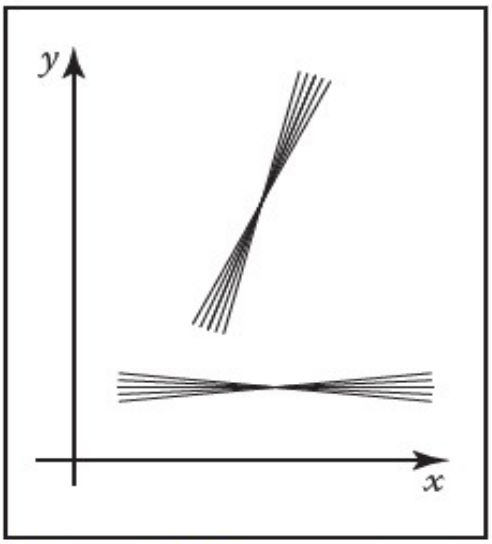
Yang tersisa, setelah kita mengetahuinya, adalah memilih yang sesuai untuk aplikasi tersebut. Ini dibahas untuk baris di bagian berikutnya.

14.3 CONTOH: PENGUKURAN PADA GARIS PADA BIDANG 2D

Apa ukuran yang μ "alami"? Jika Anda membuat parameter garis sebagai *y = mx + b*, Anda mungkin menganggap garis tertentu sebagai titik (m,b) dalam ruang “perpotongan kemiringan”. Ukuran yang mudah digunakan adalah dmdb, tetapi ini tidak akan menjadi ukuran "baik" karena "kumpulan" garis yang tidak sama ukurannya akan memiliki ukuran yang sama. Lebih tepatnya, ukurannya tidak akan invarian terhadap perubahan sistem koordinat. Misalnya, jika Anda mengambil semua garis melalui bujur sangkar $[0,1]^2$, ukuran garis yang melaluinya tidak akan menjadi besaran yang sama melalui satu satuan persegi yang diputar 45 derajat. Apa yang benar-benar kita inginkan adalah ukuran "adil" yang tidak berubah dengan rotasi atau translasi serangkaian garis. Ide ini diilustrasikan pada Gambar 14.1 dan 14.2.



Gambar 14.1. Kedua bundel garis ini harus memiliki ukuran yang sama. Mereka memiliki panjang persimpangan yang berbeda dengan sumbu y sehingga menggunakan db akan menjadi pilihan yang buruk untuk ukuran diferensial.



Gambar 14.2. Kedua bundel garis ini harus memiliki ukuran yang sama. Karena mereka memiliki nilai yang berbeda untuk perubahan kemiringan, menggunakan dm akan menjadi pilihan yang buruk untuk ukuran diferensial.

Untuk mengembangkan ukuran alami pada garis, pertama-tama kita harus mulai menganggapnya sebagai titik dalam ruang ganda. Ini adalah konsep sederhana: garis y = mx + b dapat ditentukan sebagai titik (m,b) dalam ruang perpotongan lereng. Konsep ini diilustrasikan pada Gambar 14.3. Lebih mudah untuk mengembangkan ukuran dalam ruang (ϕ,b). Dalam ruang tersebut b adalah perpotongan y, sedangkan adalah sudut yang dibuat garis dengan sumbu x, seperti yang ditunjukkan pada Gambar 14.4. Di sini, ukuran diferensial $d\phi db$ hampir berhasil, tetapi itu tidak adil karena efek yang ditunjukkan pada Gambar 14.1. Untuk menghitung rentang b yang lebih besar yang dibuat oleh kumpulan lebar konstanta, kita harus menambahkan faktor kosinus:

$$d\mu = \cos\varphi \, d\varphi \, db.$$

Dapat ditunjukkan bahwa besaran ini, hingga konstan, adalah satu-satunya variabel yang berkaitan dengan rotasi dan translasi. Ukuran ini dapat diubah menjadi ukuran yang sesuai untuk parameterisasi saluran lainnya. Misalnya, ukuran yang tepat untuk ruang (m,b) adalah

$$d\mu = \frac{dm\,db}{(1+m¿¿2)^2¿}$$

Untuk ruang garis yang diparameterisasi dalam ruang (u,v)

$$ux + vy + 1 = 0,$$

langkah yang tepat adalah

$$d\mu = \frac{du\,dv}{(u^2+v¿¿2)^{\frac{3}{2}}¿}$$

Untuk garis yang diparameterisasi dalam bentuk (a,b), perpotongan-x dan perpotongan y, pengukurannya adalah

$$d\mu = \frac{ab\,da\,db}{(a^2+b¿¿2)^{\frac{3}{2}}¿}$$

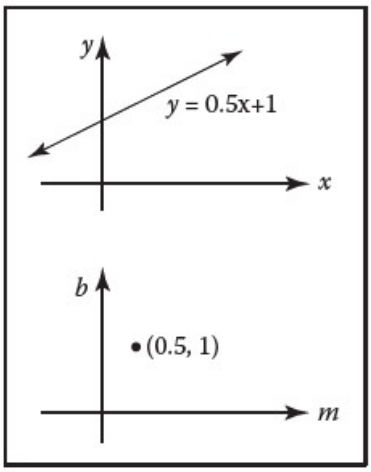
Perhatikan bahwa salah satu dari ruang tersebut adalah cara yang sama-sama valid untuk menentukan garis, dan mana yang terbaik tergantung pada keadaan. Namun, orang mungkin bertanya-tanya apakah ada sistem koordinat di mana ukuran himpunan garis hanya merupakan area di ruang ganda. Faktanya, ada sistem koordinat seperti itu, dan sangat sederhana; itu adalah koordinat normal yang menentukan garis dalam hal jarak normal dari asal ke garis, dan sudut yang dibuat oleh garis normal terhadap sumbu x (Gambar 14.5). Persamaan implisit untuk garis tersebut adalah

$$x \cos \theta + y \sin \theta - p = 0.$$

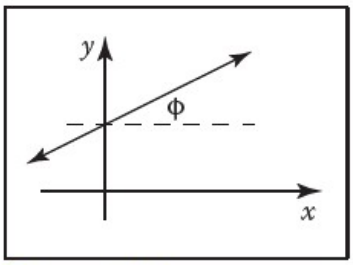
Dan, memang, ukuran di ruang itu adalah

$$d\mu = dp\, d\theta.$$

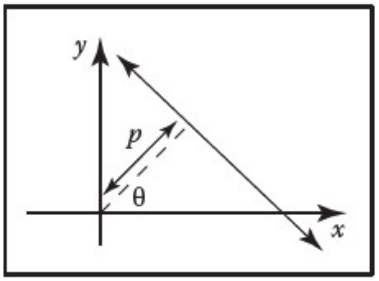
Kami akan menggunakan langkah-langkah ini untuk memilih garis acak yang adil di bagian selanjutnya.



Gambar 14.3. Himpunan titik-titik pada garis y=mx+b dalam ruang (x, y) juga dapat diwakili oleh satu titik di (m,b)ruang sehingga garis atas dan titik bawah mewakili entitas geometris yang sama: garis 2D.



Gambar 14.4. Dalam ruang intersep sudut kita membuat parameter garis dengan sudut $\emptyset \in [-\pi/2, \pi/2)$ daripada kemiringan.



Gambar 14.5. Koordinat normal garis menggunakan jarak normal ke titik asal dan sudut untuk menentukan garis.

14.4 CONTOH: UKURAN GARIS DALAM 3D

Dalam 3D ada banyak cara untuk membuat parameter garis. Mungkin, cara paling sederhana adalah dengan menggunakan perpotongannya dengan bidang tertentu bersama dengan beberapa spesifikasi orientasinya. Sebagai contoh, kita dapat memetakan perpotongan dengan bidang x_y bersama dengan koordinat bola dari orientasinya. Jadi, setiap baris akan ditentukan sebagai (x, y, θ, ϕ) quadruple. Ini menunjukkan bahwa garis dalam 3D adalah entitas 4D, yaitu, mereka dapat digambarkan sebagai titik dalam ruang 4D.

Ukuran diferensial suatu garis tidak boleh berbeda dengan (x,y), tetapi kumpulan garis dengan penampang yang sama harus memiliki ukuran yang sama. Dengan demikian, ukuran diferensial yang adil adalah

$$d\mu = dx\, dy \sin\theta\, d\theta\, d\varphi.$$

Cara lain untuk membuat parameter garis adalah dengan memetakan perpotongan dengan dua bidang sejajar. Misalnya, jika garis memotong bidang z = 0 pada (x = u, y = v) dan bidang z =1 pada (x = s, y = t), maka garis tersebut dapat digambarkan dengan segi empat (u, v, s, t). Perhatikan, bahwa seperti parameterisasi sebelumnya, parameter ini diturunkan untuk garis yang sejajar dengan bidang xy. Ukuran diferensial lebih rumit untuk parameterisasi ini meskipun dapat didekati

$$d\mu \approx du\, dv\, a\, ds\, dt,$$

untuk kumpulan garis yang hampir sejajar dengan sumbu z. Ini adalah ukuran yang sering digunakan secara implisit dalam rendering berbasis gambar.

Untuk himpunan garis yang memotong bola, kita dapat menggunakan parameter dari dua titik di mana garis memotong bola. Jika ini berada dalam koordinat bola, maka titik tersebut dapat digambarkan dengan empat kali lipat (θ_1, ϕ_1, θ_2, ϕ_2) dan ukurannya hanyalah area diferensial yang terkait dengan setiap titik:

$$d\mu = \sin\theta 1\; d\theta 1\; d\varphi 1 \sin\theta 2\; d\theta 2\; d\varphi 2.$$

Ini menyiratkan bahwa memilih dua titik akhir acak yang seragam pada bola menghasilkan garis dengan kerapatan seragam. Pengamatan ini digunakan untuk menghitung faktor bentuk oleh Mateu Sbert dalam disertasinya (Sbert, 1997).

Perhatikan bahwa terkadang kita ingin membuat parameter garis berarah, dan terkadang kita ingin urutan titik akhir tidak menjadi masalah. Ini adalah detail pembukuan yang sangat penting untuk aplikasi rendering di mana jumlah cahaya yang mengalir di sepanjang garis berbeda dalam dua arah di sepanjang garis.

14.5 PROBABILITAS KONTINYU

Banyak algoritma grafis menggunakan probabilitas untuk membangun sampel acak untuk memecahkan masalah integrasi dan rata-rata. Ini adalah domain probabilitas kontinu terapan yang memiliki koneksi dasar untuk mengukur teori.

Fungsi Kepadatan Probabilitas Kontinu Satu Dimensi

Secara longgar, variabel acak kontinu x adalah besaran skalar atau vektor yang "secara acak" mengambil beberapa nilai dari garis nyata R = (−∞, +∞). Perilaku x sepenuhnya dijelaskan oleh distribusi nilai yang dibutuhkan. Distribusi nilai ini dapat dijelaskan secara kuantitatif dengan *probability density function* (pdf), p, terkait dengan x (hubungannya

dilambangkan x ~ p). Probabilitas bahwa x mengasumsikan nilai tertentu dalam beberapa interval [a,b] diberikan oleh integral berikut: (Persamaan 14.1)

$$probabilitas(x \in [a,b]) = \int_a^b p(|x|)\,dx$$

Secara longgar, fungsi densitas probabilitas p menggambarkan kemungkinan relatifvariabel random mengambil nilai tertentu; jika $p(x_1) = 6.0$ dan $p(x_2) = 3.0$, maka variabel acak dengan densitas p dua kali lebih mungkin memiliki nilai "dekat" x_1 daripada nilainya memiliki nilai mendekati x_2. Kepadatan p memiliki dua karakteristik: (Persaman 14.2 (atas) dan persaman 14.3 (bawah)

$$p(x) \geq 0 \text{ (probabilitasnya non-negatif)}$$

$$\int_{-\infty}^{+\infty} p(x)\,dx = 1¿¿$$

Sebagai contoh, variabel acak kanonik mengambil nilai antara nol (inklusif) dan selesai (non-inklusif) dengan probabilitas seragam (di sini seragam berarti setiap nilai untuk memiliki kemungkinan ξ yang sama). Ini menyiratkan bahwa fungsi densitas probabilitas q untuk ξ adalah

$$q(\xi) = \begin{cases} 1 \; jika \leq \xi < 1 \\ 0 \; jika \, tidak \end{cases}$$

Ruang di mana didefinisikan hanyalah interval [0,1). Probabilitas bahwa mengambil suatu nilai dalam selang waktu tertentu $[a,b] \in [0,1)$ adalah

$$probailitas(a \leq \xi \leq b) \int_a^b 1\,dx = b - a$$

Nilai Harapan Satu Dimensi

Nilai rata-rata yang akan diambil oleh fungsi riil f dari variabel acak satu dimensi dengan pdf p yang mendasarinya disebut nilai harapannya, E(f(x)) (kadang-kadang ditulis Ef(x)):

$$E(f(x)) = \int f(x)\,p(x)\,dx$$

Nilai yang diharapkan dari variabel acak satu dimensi dapat dihitung dengan menetapkan f(x) = x. Nilai yang diharapkan memiliki properti yang mengejutkan dan berguna: nilai yang diharapkan dari jumlah dua variabel acak adalah jumlah dari nilai yang diharapkan dari variabel tersebut:

$$E(x + y) = E(x) + E(y),$$

untuk variabel acak x dan y. Karena fungsi dari variabel acak itu sendiri adalah variabel acak, linearitas harapan ini juga berlaku untuknya:

$$E(f(x) + g(y)) = E(f(x)) + E(g(y)).$$

Pertanyaan yang jelas untuk ditanyakan adalah apakah properti ini berlaku jika variabel acak dijumlahkan berkorelasi (variabel yang tidak berkorelasi disebut independen). Properti linearitas ini sebenarnya berlaku apakah variabel independen atau tidak! Properti penjumlahan ini merupakan bentuk penting dari sebagian besar aplikasi Monte Carlo.

14.6 VARIABEL ACAK MULTIDIMENSI

Diskusi variabel acak dan nilai yang diharapkan meluas secara alami ke ruang multidimensi. Sebagian besar masalah grafis akan berada di ruang dimensi yang lebih tinggi. Misalnya, banyak masalah pencahayaan diungkapkan di permukaan belahan bumi. Untungnya, jika kita mendefinisikan ukuran μ pada ruang yang ditempati variabel acak, semuanya sangat mirip dengan kasus satu dimensi. Misalkan ruang S memiliki ukuran terkait ; misalnya S adalah permukaan bola dan mengukur luas. Kita dapat mendefinisikan *pdf* $p : S \to \mathbb{R}$, dan jika x adalah variabel acak dengan x ~ p, maka peluang x akan mengambil suatu nilai di suatu daerah Si S diberikan oleh integral

$$probailitas(x \vee \in S_i) \int_{S_i} p(x)\, d\mu$$

Di sini Probabilitas (kejadian) adalah probabilitas bahwa kejadian itu benar, jadi integralnya adalah probabilitas bahwa x mengambil suatu nilai di daerah Si.

Dalam grafis, S seringkali luas (dμ = dA = dxdy) atau searah (titik pada bola satuan: dμ = dω = sinθ dθdϕ). Sebagai contoh, variabel acak dua dimensi adalah variabel acak terdistribusi seragam pada piringan berjari-jari R. Di sini seragam berarti seragam terhadap luas, misalnya, cara pukulan pemain dart yang buruk akan didistribusikan pada papan dart. Karena seragam, kita tahu bahwa p(α) adalah suatu konstanta. Dari fakta bahwa luas piringan adalah πr^2 dan peluang totalnya adalah satu, kita dapat menyimpulkan bahwa

$$p(\propto) = \frac{1}{\pi R^2}$$

Ini berarti bahwa probabilitas bahwa berada dalam subset tertentu S_1 dari disk adalah adil

$$probailitas(\alpha \in S_i) \int_{S_i} \frac{1}{\pi R^2} dA$$

Ini semua sangat abstrak. Untuk benar-benar menggunakan informasi ini, kita memerlukan integral dalam bentuk yang dapat kita evaluasi. Misalkan Si adalah bagian piringan yang lebih dekat ke pusat daripada keliling. Jika kita mengubah ke koordinat kutub, maka direpresentasikan sebagai pasangan (r, ϕ), dan S_1 adalah daerah di mana r < R/2. Perhatikan, bahwa hanya karena seragam, tidak berarti bahwa atau r adalah tentu seragam (sebenarnya, seragam, dan r tidak seragam). Daerah diferensial *dA* hanya *r dr dϕ.* Jadi,

$$Probabilitas\left(r > \frac{R}{2}\right) = \int_0^{2\pi} \int_0^{\frac{R}{2}} \frac{1}{\pi R^2} r\, dr\, d\phi = 0.25$$

Perbedaan

Varians, V (x), dari variabel acak satu dimensi, menurut definisi, adalah nilai yang diharapkan dari kuadrat selisih antara x dan E(x):

$$V(x) \equiv E([x - E(x)]^2).$$

Beberapa manipulasi aljabar memberikan ekspresi yang tidak jelas:

$$V(x) = E(x2) - [E(x)]^2 .$$

Ekspresi $E([x-E(x)]^2)$ lebih berguna untuk berpikir secara intuitif tentang varians, sedangkan ekspresi ekivalen secara aljabar $E(x^2) - [E(x)]^2$ biasanya sesuai untuk perhitungan. Varians dari jumlah variabel acak adalah jumlah varians jika variabel independen. Properti penjumlahan varians ini adalah salah satu alasan yang sering digunakan dalam analisis model probabilistik. Akar kuadrat dari varians disebut deviasi standar, σ, yang memberikan beberapa indikasi deviasi absolut yang diharapkan dari nilai yang diharapkan.

Perkiraan Berarti

Banyak masalah yang melibatkan jumlah variabel acak yang sangat bergantung xi, di mana variabel tersebut memiliki kesamaan densitas p. Variabel-variabel tersebut dikatakan sebagai variabel-variabel acak yang terdistribusi secara independen/ *independent identically distributed* (iid). Ketika jumlah dibagi dengan jumlah variabel, kita mendapatkan estimasi E(x):

$$E(x) \approx \frac{1}{N} \sum_{i=1}^{N} x_i$$

Ketika N meningkat, varians dari estimasi ini menurun. Kami ingin N cukup besar sehingga kami memiliki keyakinan bahwa perkiraannya "cukup dekat." Namun, tidak ada hal yang pasti di Monte Carlo; kita baru saja memperoleh keyakinan statistik bahwa perkiraan kita baik. Untuk memastikan, kita harus memiliki N =∞. Keyakinan ini dinyatakan oleh*Law of Large Numbers/* Hukum Bilangan Besar:

$$Probabilitas\left[E(x) = \lim_{N \to \infty} \frac{1}{N} \sum_{i=1}^{N} x_i\right] = 1$$

14.7 INTEGRASI MONTE CARLO

Pada bagian ini, metode solusi Monte Carlo dasar untuk integral tertentu diuraikan. Teknik-teknik ini kemudian langsung diterapkan pada masalah integral tertentu. Semua bahan dasar bagian ini juga tercakup dalam beberapa teks klasik Monte Carlo. (Lihat bagian Catatan di akhir bab ini.)

Seperti yang telah dibahas sebelumnya, diberikan fungsi f : S → ℝ dan variabel acak x ~ p, kita dapat memperkirakan nilai yang diharapkan dari f(x) dengan jumlah:

$$E(f(x)) = \int_{x \in S} f(x)\, p(x)\, d\mu \approx \frac{1}{N} \sum_{i=1}^{N} f(x¿¿i)¿$$

Karena nilai yang diharapkan dapat dinyatakan sebagai integral, integral juga didekati dengan jumlah. Bentuk Persamaan (14.4) agak janggal; kami biasanya ingin memperkirakan integral dari satu fungsi g daripada productfp. Kita dapat melakukannya dengan mengganti g = fpas integran:

$$\int_{x \in S} g(x)\, d\mu \approx \frac{1}{N} \sum_{i=1}^{N} \frac{g(x¿¿i)}{p(x¿¿i)} ¿¿$$

Agar rumus ini valid, p harus positif jika g bukan nol.

Jadi untuk mendapatkan estimasi yang baik, kami menginginkan sampel sebanyak mungkin, dan kami ingin g/pto memiliki varians yang rendah (g dan p harus memiliki bentuk yang serupa). Memilih p secara cerdas disebut pengambilan sampel penting, karena jika p besar di mana g besar, akan ada lebih banyak sampel di daerah penting. Persamaan (14.4) juga menunjukkan masalah mendasar dengan integrasi Monte Carlo: hasil yang semakin berkurang. Karena varians estimasi sebanding dengan 1/N, standar deviasi sebanding dengan $\frac{1}{\sqrt{N}}$. Karena kesalahan dalam estimasi berperilaku serupa dengan standar deviasi, kita perlu melipatgandakan N untuk membagi kesalahan menjadi setengahnya.

Cara lain untuk mengurangi varians adalah dengan mempartisi S, domain integral, menjadi beberapa domain yang lebih kecil Si, dan mengevaluasi integral sebagai jumlah integral dari Si. Ini disebut sampling bertingkat, teknik jittering menggunakan sampling piksel (Bab 4). Biasanya hanya satu sampel untuk setiap S_i (dengan densitas p_i), dan dalam hal ini varians penduganya adalah: (Persamaan 14.6)

$$var¿$$

Dapat ditunjukkan bahwa varians dari stratified sampling tidak pernah lebih tinggi dari unstratified jika semua strata memiliki ukuran yang sama:

$$\int_{S_i} p(x)\,d\mu = \frac{1}{N}\int_{s} p(x)\,d\mu$$

Contoh paling umum dari pengambilan sampel bertingkat dalam grafis adalah jittering untuk pengambilan sampel piksel seperti yang dibahas dalam Bagian 13.4. Sebagai contoh solusi Monte Carlo dari integral I, *set g(x)* sama dengan *x* pada interval (0, 4):

$$I = \int_0^4 x\,dx = 8$$

Pengaruh bentuk fungsi p terhadap varians dari estimasi sampel N ditunjukkan pada Tabel 14.1. Perhatikan bahwa varians berkurang ketika bentuk p mirip dengan bentuk g. Varians turun menjadi nol jika p = g/I, tetapi I biasanya tidak diketahui atau kita tidak perlu menggunakan Monte Carlo. Salah satu prinsip penting yang diilustrasikan pada Tabel 14.1 adalah bahwa pengambilan sampel bertingkat seringkali jauh lebih unggul daripada pengambilan sampel kepentingan (Mitchell, 1996). Meskipun varians untuk stratifikasi ini pada I berbanding terbalik dengan pangkat tiga dari jumlah sampel, tidak ada hasil umum untuk perilaku varians di bawah stratifikasi. Ada beberapa fungsi yang stratifikasinya tidak baik. Salah satu contohnya adalah fungsi white noise, di mana variansnya konstan untuk semua wilayah. Di sisi lain, sebagian besar fungsi akan mendapat manfaat dari pengambilan sampel bertingkat, karena varians di setiap subsel biasanya akan lebih kecil daripada varians seluruh domain.

Tabel 14.1. Varians untuk estimasi Monte Carlo sebesar 4 0 xdx, berurutan dari kirii, metode, fungsi sampling, variasi dan contoh sampling

importance	$(6-x)/(16)$	$56.8N^{-1}$	887,500
importance	$1/4$	$21.3N^{-1}$	332,812
importance	$(x+2)/16$	$6.3N^{-1}$	98,437
importance	$x/8$	0	1
stratified	1/4	$21.3N^{-3}$	70

14.8 INTEGRASI KUASI–MONTE CARLO

Metode yang populer untuk kuadratur adalah mengganti titik acak dalam integrasi Monte Carlo dengan titik kuasi-acak. Poin-poin seperti itu bersifat deterministik, tetapi dalam beberapa hal seragam. Misalnya, pada bujur sangkar [0,1]2, himpunan N titik kuasirandom harus memiliki sifat berikut pada daerah dengan luas A di dalam bujur sangkar:

$$\text{Jumlah titik pada bagian} \approx \text{AN}$$

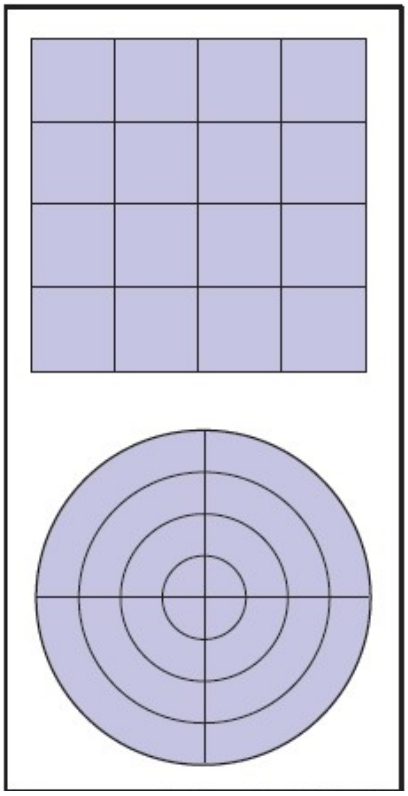
Misalnya, satu set sampel reguler dalam kisi memiliki properti ini. Poin kuasi-acak dapat meningkatkan kinerja di banyak aplikasi integrasi. Terkadang perawatan harus dilakukan untuk memastikan bahwa mereka tidak memperkenalkan aliasing. Sangat bagus bahwa, dalam

aplikasi apa pun di mana panggilan dibuat ke titik acak atau bertingkat dalam $[0,1]^d$, seseorang dapat mengganti titik kuasi-acak dimensi-d tanpa perubahan lain.

Intuisi utama yang memotivasi integrasi kuasi-Monte Carlo adalah bahwa ketika memperkirakan nilai rata-rata integran, setiap set titik sampel akan berhasil, asalkan mereka "adil."

14.9 MEMILIH POIN ACAK

Kita sering ingin menghasilkan kumpulan titik acak atau acak semu pada kuadrat satuan untuk aplikasi seperti pendistribusian ray tracing. Ada beberapa metode untuk melakukan ini, misalnya, jittering (lihat Bagian 13.4). Metode-metode ini memberi kita satu set N titik yang cukup merata pada kuadrat satuan $[0,1]^2$: (u_1, v_1) melalui (u_N, v_N).



Gambar 14.6. Transformasi yang membawa dimensi horizontal dan vertikal seragam ke (r,ϕ) tidak mempertahankan luas relatif; tidak semua area yang dihasilkan sama

Kadang-kadang, ruang pengambilan sampel kami mungkin tidak persegi (misalnya, lensa melingkar), atau mungkin tidak seragam (misalnya, fungsi filter yang berpusat pada piksel). Akan lebih baik jika kita dapat menulis transformasi matematis yang akan mengambil titik-titik terdistribusi setara (u_i, v_i) sebagai input dan output sekumpulan titik dalam ruang sampling yang kita inginkan dengan kepadatan yang kita inginkan. Misalnya, untuk sampel kamera, transformasi akan mengambil (u_i, v_i) dan output (r_i, ϕ_i) sedemikian rupa sehingga titik-titik baru terdistribusi secara merata pada piringan lensa. Meskipun kita mungkin tergoda untuk menggunakan transformasi

$$\varphi_i = 2\pi u_i,$$
$$r_i = v_i R,$$

itu memiliki masalah serius. Sementara titik menutupi lensa, mereka melakukannya secara tidak seragam (Gambar 14.6). Apa yang kita butuhkan dalam kasus ini adalah transformasi yang mengambil daerah dengan luas yang sama ke daerah dengan luas yang sama—

transformasi yang mengambil distribusi sampel seragam pada kuadrat ke distribusi seragam pada domain baru.

Ada beberapa cara untuk menghasilkan titik tidak seragam atau titik seragam pada domain non-persegi panjang, dan bagian berikut meninjau tiga yang paling sering digunakan: inversi fungsi, penolakan, dan Metropolis.

14.10 FUNGSI INVERSI

Jika rapatan f(x) adalah satu dimensi dan didefinisikan pada interval $x \in [x_{min}, x_{max}]$, maka kita dapat membangkitkan bilangan acak α_i yang memiliki rapatan f dari himpunan bilangan acak seragam ξ_i, di mana $\xi_i \in [0,1\ ]$. Untuk melakukan ini, kita memerlukan fungsi distribusi peluang kumulatif P(x):

$$Probabilitas(\alpha < x) = P(x) \int_{x_{min}}^{x} f(x') d\mu$$

Untuk mendapatkan α_i, kita cukup mengubah ξ_i:

$$\alpha i = P^{-1}(\xi^4),$$

di mana P^{-1} adalah invers dari P. Jika P tidak dapat dibalik secara analitik, maka metode numerik akan cukup, karena terdapat invers untuk semua fungsi distribusi probabilitas yang valid.

Perhatikan bahwa secara analitis membalik fungsi lebih membingungkan daripada yang seharusnya karena notasi. Misalnya, jika kita memiliki fungsi

$$y = x^2,$$

untuk x > 0, maka fungsi invers dinyatakan dalam y sebagai fungsi dari x:

$$x = \sqrt{y}.$$

Ketika fungsinya secara analitik dapat dibalik, hampir selalu sesederhana itu. Namun, hal-hal yang sedikit lebih buram dengan notasi standar:

$$f(x) = x^2,$$
$$f^{-1}(x) = \sqrt{x}.$$

Di sini x hanyalah variabel dummy. Anda mungkin merasa lebih mudah menggunakan notasi yang kurang standar:

$$y = x^2,$$
$$x = \sqrt{y},$$

sambil mengingat bahwa ini adalah fungsi terbalik satu sama lain. Misalnya, untuk memilih titik acak xi yang memiliki kerapatan

$$p(x)\frac{3x^2}{2}$$

pada [−1,1], kita melihat bahwa

$$p(x)\frac{3^{2+1}}{2}$$

dan

$$p^{-1} = \sqrt[3]{2x-1}$$

sowecan "warp" aset dari kanonik bilangan acak (ξ_1, ⋯, ξ_N) ke nomor yang didistribusikan dengan benar

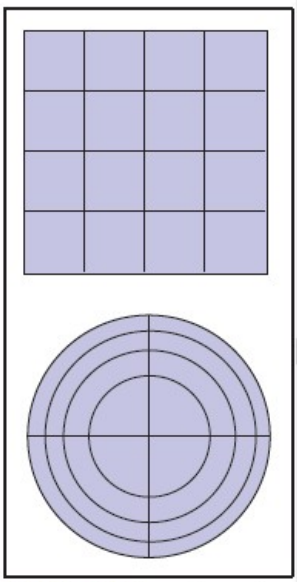
$$(x¿¿1,\ldots,x_N)=\sqrt[3]{2\xi_i-1},\ldots,\sqrt[3]{2\xi_N-1}¿$$

Tentu saja, fungsi warping yang sama ini dapat digunakan untuk mengubah sampel yang "seragam" menjadi sampel yang terdistribusi dengan baik dengan kepadatan yang diinginkan.

Ini berarti bahwa pasangan kanonik (ξ_1, ξ_2) dapat diubah menjadi titik acak yang seragam pada piringan:

$$\varphi = 2\pi\xi_1,$$
$$r = R\ \xi_2.$$

Mapping ini ditunjukkan pada Gambar 14.7.



Gambar 14.7. Mapping yang mengambil daerah dengan luas yang sama dalam satuan persegi ke daerah dengan luas yang sama dalam piringan.

Suku $sin\theta$ muncul karena, pada bola, $d\omega = cos\theta d\theta d\phi$. Ketika kepadatan marjinal ditemukan, p (seperti yang diharapkan) dapat dipisahkan, dan kami menemukan bahwa pasangan (ξ_1, ξ_2) bilangan acak kanonik dapat ditransformasikan ke arah dengan menghormati dasar uvw.

Sekali lagi, hal yang menyenangkan tentang ini adalah bahwa satu set titik jitter pada unit persegi dapat dengan mudah diubah menjadi himpunan titik jitter di belahan bumi dengan distribusi yang diinginkan. Perhatikan bahwa jika n diset ke 1, kita memiliki distribusi difus, seperti yang sering dibutuhkan.

Seringkali kita harus memetakan titik pada bola ke arah yang sesuai dengan dasar dasar. Untuk melakukan ini, pertama-tama kita dapat mengubah sudut menjadi vektor satuan:

$$\mathbf{a} = (\cos\varphi \sin\theta, \sin\varphi \sin\theta, \cos\theta)$$

Sebagai peningkatan efisiensi, kita dapat menghindari mengambil fungsi trigonometri dari fungsi trigonometri terbalik (misalnya, $cos(arccos\theta)$). Misalnya, ketika n =1 (distribusi difus), vektor a disederhanakan menjadi

$$a¿$$

14.11 PENOLAKAN

Metode penolakan memilih titik menurut beberapa distribusi sederhana dan menolak beberapa dari mereka yang berada dalam distribusi yang lebih kompleks. Ada beberapa skenario di mana penolakan digunakan, dan kami menunjukkan beberapa di antaranya dengan contoh

Misalkan kita menginginkan titik acak yang seragam di dalam lingkaran satuan. Pertama-tama kita dapat memilih titik acak seragam $(x, y) \in [-1, 1]^2$ dan menolak titik-titik di luar lingkaran. Jika fungsi r() mengembalikan nomor acak kanonik, maka prosedurnya adalah:

```
done = false
while (not done) do
x = −1+ 2r()
y = −1 + 2r()
if (x2 + y2 < 1) then
done = true
```

Jika kita menginginkan bilangan acak x p dan kita mengetahui bahwa p : [a, b] → R, dan bahwa untuk semua x, p(x) < m, maka kita dapat menghasilkan titik acak dalam persegi panjang [a, b] × [0,m] dan ambil di mana y < p(x):

```
done = false
while (not done) do
x = a + r()(b − a)
y = r()m
if (y < p(x)) then
done = true
```

Ide yang sama dapat diterapkan untuk mengambil titik acak pada permukaan bola. Untuk memilih vektor satuan acak dengan distribusi arah seragam, pertama-tama pilih titik acak dalam bola satuan dan kemudian perlakukan titik itu sebagai vektor arah dengan mengambil vektor satuan dalam arah yang sama:

```
done = false
while (not done) do
    x = −1 + 2r()
    y = −1 + 2r()
    z = −1 + 2r()
    if ((l = √(x² + y² + z²)) < 1)|then
        done = true
x = x/l
y = y/l
z = z/l
```

Meskipun metode penolakan biasanya sederhana untuk dikodekan, metode ini jarang kompatibel dengan stratifikasi. Untuk alasan ini, ia cenderung berkumpul lebih lambat dan karenanya harus digunakan terutama untuk debugging, atau dalam keadaan yang sangat sulit.

14.12 METROPOLIS

Metode Metropolis menggunakan mutasi acak untuk menghasilkan sekumpulan sampel dengan kepadatan yang diinginkan. Konsep ini digunakan secara luas dalam algoritma Metropolis Light Transport yang dirujuk dalam catatan bab. Misalkan kita memiliki titik acak x_0 dalam domain S. Selanjutnya, misalkan untuk sembarang titik x, kita memiliki cara untuk menghasilkan y px acak. Kami menggunakan notasi marginal $p_x(y)$ p(x → y) untuk menyatakan fungsi kerapatan ini. Sekarang, misalkan kita membiarkan x_1 menjadi titik acak di S yang dipilih dengan kepadatan dasar $p(x_0 \rightarrow x_1)$. Kami menghasilkan x_2 dengan kepadatan $p(x_1 \rightarrow x_0)$ dan seterusnya. Dalam batas, di mana kita menghasilkan sejumlah

sampel yang tak terhingga, dapat dibuktikan bahwa sampel akan memiliki beberapa kerapatan dasar yang ditentukan oleh p terlepas dari titik awal x_0.

Sekarang, misalkan kita ingin memilih sedemikian sehingga kerapatan dasar dari sampel yang kita konvergenkan sebanding dengan fungsi f(x) di mana f adalah fungsi nonnegatif dengan domain S. Selanjutnya, anggaplah kita dapat mengevaluasi f, tetapi kita memiliki sedikit atau tidak ada pengetahuan tambahan tentang sifat-sifatnya (fungsi seperti itu umum dalam grafis) . Juga, misalkan kita memiliki kemampuan untuk membuat "transisi" dari xi ke xi+1 dengan fungsi kerapatan yang mendasari t(xi → xi+1). Untuk menambah fleksibilitas, selanjutnya misalkan kita menambahkan probabilitas yang berpotensi bukan nol bahwa xi bertransisi ke dirinya sendiri, yaitu, xi+1 = xi. Kami menyatakan ini sebagai menghasilkan kandidat potensial t(xi → y) dan “menerima” kandidat ini(yaitu, x_{i+1} = y)dengan probabilitas a(x_i → y)dan menolaknya(yaitu, $x_{i+1} = x_i$) dengan probabilitas 1 − a(x_i → y). Perhatikan bahwa barisan x_0, x_1, x_2, ... x_n akan menanggung dan himpunan, tetapi akan ada beberapa korelasi di antara sampel. Mereka akan tetap cocok untuk integrasi Monte Carlo atau estimasi kepadatan, tetapi menganalisis varians dari estimasi tersebut jauh lebih menantang.

Sekarang, misalkan kita diberikan fungsi transisi t(x → y) dan fungsi f(x) yang ingin kita tiru distribusinya, dapatkah kita menggunakan a(y → x) sedemikian rupa sehingga titik-titik terdistribusi dalam bentuk f? Atau lebih tepatnya,

$$\{x_0, x_1, x_2, \ldots\} \quad \frac{f}{f_s}$$

Ternyata ini bisa dipaksakan dengan memastikan xi tidak bergerak dalam arti tertentu. Jika Anda memvisualisasikan kumpulan besar titik sampel x, Anda ingin "aliran" antara dua titik sama di setiap arah. Jika kita menganggap kerapatan titik dekat x dan y masing-masing sebanding dengan f(x) dan f(y), maka aliran dalam dua arah harus sama:

$$\text{flow}(x \to y) = kf(x)t(x \to y)a(x \to y),$$
$$\text{flow}(y \to x) = kf(y)t(y \to x)a(y \to x),$$

di mana k adalah konstanta positif. Menetapkan dua aliran konstan ini memberikan kendala pada:

$$\frac{a(y \to x)}{a(x \to y)} = \frac{f(x)t(x \to y)}{f(y)t(y \to x)}$$

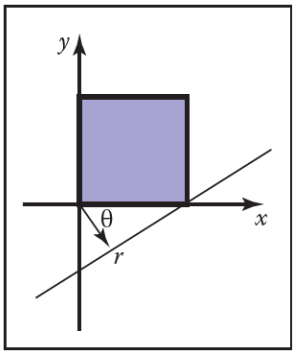
Jadi, jika a(y → x) atau a(x → y) diketahui, maka yang lainnya juga diketahui. Membuatnya lebih besar meningkatkan peluang penerimaan, jadi teknik yang biasa adalah mengatur yang lebih besar dari keduanya menjadi 1.

Kesulitan dalam menggunakan teknik pembuatan sampel Metropolis adalah sulitnya memperkirakan berapa banyak titik yang diperlukan sebelum himpunan titik menjadi "baik". Hal-hal dipercepat jika n poin pertama dibuang, meskipun memilih n dengan bijak bukanlah hal yang sepele.

14.13 CONTOH: MEMILIH GARIS ACAK DI KOTAK

Sebagai contoh dari seluruh proses perancangan strategi pengambilan sampel, pertimbangkan masalah menemukan garis acak yang memotong persegi satuan $[0,1]^2$. Kami ingin proses ini adil; yaitu, kita ingin garis-garis itu terdistribusi secara merata di dalam bujur sangkar. Secara intuitif, kita dapat melihat bahwa ada beberapa kehalusan dalam masalah ini; ada "lebih banyak" garis pada sudut miring daripada di arah horizontal atau vertikal. Hal ini karena penampang persegi tidak seragam.

Tujuan pertama kami adalah menemukan metode inversi fungsi, jika ada, dan kemudian kembali ke penolakan atau Metropolis jika gagal. Hal ini karena kami ingin memiliki sampel bertingkat dalam ruang garis. Kami mencoba menggunakan koordinat normal terlebih dahulu, karena masalah memilih garis acak dalam bujur sangkar hanyalah masalah menemukan titik acak yang seragam di bagian mana pun dari ruang (r, θ) yang sesuai dengan garis dalam bujur sangkar.



Gambar 14.8. Jarak rcorrespondstoaline terbesar yang mengenai bujur sangkar untukθ∈ [− /2, 0]. Karena persegi memiliki panjang sisi satu, r = cosθ.

Pertimbangkan daerah di mana -π/2 < θ < 0. Berapa nilai r yang sesuai dengan garis yang mengenai bujur sangkar? Untuk sudut-sudut tersebut, r < cos θ adalah semua garis yang membentur bujur sangkar seperti yang ditunjukkan pada Gambar 14.8. Penalaran serupa di empat kuadran lainnya menemukan daerah di ruang (r, θ) yang harus dijadikan sampel, seperti yang ditunjukkan pada Gambar 14.9. Persamaan batas daerah tersebut $r_{max}(\theta)$ adalah:

$$r_{\max}(\theta) = \begin{cases} 0 & \text{if } \theta \in [-\pi, -\frac{\pi}{2}], \\ \cos\theta & \text{if } \theta \in [-\frac{\pi}{2}, 0], \\ \sqrt{2}\cos(\theta - \frac{\pi}{4}) & \text{if } \theta \in [0, \frac{\pi}{2}], \\ \sin\theta & \text{if } \theta \in [\frac{\pi}{2}, \pi]. \end{cases}$$

Karena daerah di bawah $r_{max}(\theta)$ adalah fungsi sederhana yang dibatasi oleh r = 0 , kita dapat mengambil sampelnya dengan terlebih dahulu memilih sesuai dengan fungsi kerapatan:

$$p(\theta) = \frac{r_{\max}(\theta)}{\int_{-\pi}^{\pi} r_{\max}(\theta) d\theta}.$$

Penyebut di sini adalah 4. Sekarang, kita dapat menghitung fungsi distribusi probabilitas kumulatif:

$$P(\theta) = \begin{cases} 0 & \text{if } \theta \in [-\pi, -\frac{\pi}{2}], \\ (1 + \sin\theta)/4 & \text{if } \theta \in [-\frac{\pi}{2}, 0], \\ (1 + \frac{\sqrt{2}}{2}\sin(\theta - \frac{\pi}{4}))/2 & \text{if } \theta \in [0, \frac{\pi}{2}], \\ (3 - \cos\theta)/4 & \text{if } \theta \in [\frac{\pi}{2}, \pi]. \end{cases}$$

Kita dapat membalikkan ini dengan memanipulasi 1 = P(θ) ke dalam bentuk = g(ξ1). Ini menghasilkan

$$\theta = \begin{cases} \arcsin(4\xi_1 - 1) & \text{if } \xi_1 < \frac{1}{4}, \\ \arcsin(\frac{\sqrt{2}}{2}(2\xi_1 - 1)) + \frac{\pi}{4} & \text{if } \xi_1 \in [\frac{1}{4}, \frac{3}{4}], \\ \arccos(3 - 4\xi_1) & \text{if } \xi_1 > \frac{3}{4}. \end{cases}$$

Setelah kita mendapatkan , maka r adalah:

$$r = \xi_2 \max(\theta).$$

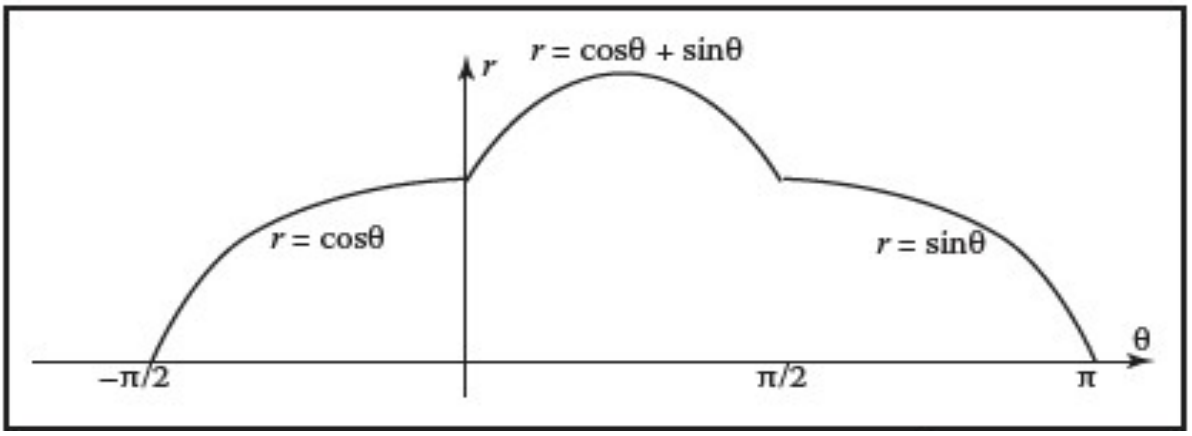
Seperti yang telah dibahas sebelumnya, ada banyak parameterisasi perangkat lunak, dan masing-masing memiliki ukuran "adil" yang terkait. Kami juga dapat menghasilkan garis acak di salah satu ruang ini. Misalnya, dalam ruang perpotongan kemiringan, daerah yang mengenai bujur sangkar ditunjukkan pada Gambar 14.10. Dengan alasan yang mirip dengan ruang normal, fungsi kerapatan untuk lereng adalah

$$p(m) = 1 + |m| \vee \frac{|}{4}|$$

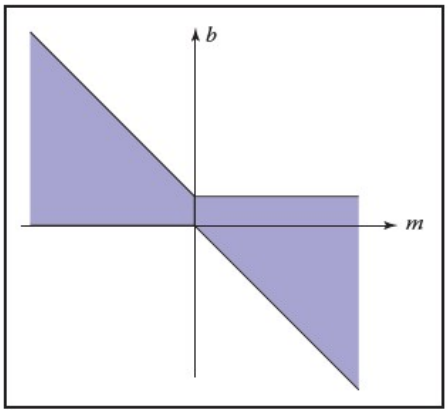
sehubungan dengan ukuran diferensial

$$d\mu = \frac{dm}{(1 + m||2)^{\frac{3}{2}}|}$$

Ini dapat dibalik dengan memecahkan dua persamaan kuadrat. Mengingat m yang dihasilkan menggunakan 1, ini bukan cara yang lebih baik daripada menggunakan koordinat normal; itu hanya cara alternatif.



Gambar 14.9. Jari-jari maksimum untuk garis yang mengenai bujur sangkar [0,1]2 sebagai fungsi dari .



Gambar 14.10. Daerah (m,b) ruang yang memuat garis-garis yang memotong persegi satuan [0,1]2.

Pertanyaan yang Sering Diajukan

- ***Bab ini membahas probabilitas tetapi bukan statistik. Apa perbedaannya?***

Probabilitas adalah studi tentang seberapa besar kemungkinan suatu peristiwa. Statistik menyimpulkan karakteristik besar, tetapi terbatas, populasi dari variabel acak. Dalam pengertian itu, statistik dapat dilihat sebagai jenis probabilitas terapan yang spesifik.

- ***Apakah pengambilan sampel Metropolis sama dengan Algoritma Metropolis Light Transport?***

Tidak. Algoritma Metropolis Light Transport (Veach & Guibas, 1997) menggunakan sampling Metropolis sebagai bagian dari prosedurnya, tetapi secara khusus untuk rendering, dan memiliki langkah-langkah lain juga.

14.14 CATATAN

Rujukan klasik untuk probabilitas geometrik adalah Probabilitas Geometrik (Solomon, 1978). Metode lain untuk memilih tepi acak dalam bujur sangkar diberikan dalam Random–Edge Discrepancy of Supersampling Patterns (Dobkin & Mitchell, 1993). Informasi lebih lanjut tentang metode quasi-Monte Carlo untuk grafis dapat ditemukan di Efficient Multidimensional Sampling (Kollig & Keller, 2002). Tiga buku klasik dan sangat mudah dibaca tentang metode Monte Carlo adalah Metode Monte Carlo (Hammersley & Handscomb, 1964), Metode Monte Carlo, Dasar (Kalos & Whitlock, 1986), dan Metode Monte Carlo (Sobel, Stone, & Messer, 1975).

14.15 LATIHAN

1. Berapakah nilai rata-rata fungsi xyz pada kubus satuan $(x, y, z) \in [0, 1]^3$?
2. Berapakah nilai rata-rata r pada piringan radius satuan: $(r, \phi) \epsilon [0, 1] \times [0, 2\pi)$?
3. Tunjukkan bahwa mapping seragam titik-titik acak kanonik (ξ_1, ξ_2) ke koordinat barysentris dari sembarang segitiga adalah: $\beta = 1 - \sqrt{1 - ¿\,\xi\, 1\, ¿}$ dan $\gamma = (1 - u)\, \xi_2$.
4. Berapa panjang rata-rata garis di dalam persegi satuan? Verifikasi jawaban Anda dengan membuat sepuluh juta garis acak dalam satuan persegi dan rata-rata panjangnya.
5. Berapa panjang rata-rata faline di dalam kubus satuan? Verifikasi jawaban Anda dengan membuat sepuluh juta garis acak dalam kubus satuan dan rata-rata panjangnya.
6. Tunjukkan dari definisi varians bahwa $V(x) = E(x^2) - [E(x)]^2$.

BAB 15
KURVA

15.1 KURVA

ia mengatur titik-titik yang dijiplak pena selama selang waktu tertentu. Sementara kita biasanya memikirkan pena yang menulis di atas kertas (misalnya, kurva yang ada di ruang 2D), pena bisa bergerak dalam 3D untuk menghasilkan kurva ruang, atau Anda bisa membayangkan pena bergerak di ruang lain. Secara matematis, definisi kurva dapat dilihat setidaknya dalam dua cara:

1. bayangan kontinu dari beberapa interval dalam ruang n-dimensi;
2. peta kontinu dari ruang satu dimensi ke ruang dimensi n.

Kedua definisi ini dimulai dengan gagasan tentang rentang interval (waktu di mana pena menelusuri kurva). Namun, ada perbedaan yang signifikan: dalam definisi pertama, kurva adalah kumpulan titik yang dijejak pena (gambar), sedangkan dalam definisi kedua, kurva adalah mapping antara waktu dan kumpulan titik tersebut. Untuk bab ini, kami menggunakan definisi pertama.

Kurva adalah kumpulan titik-titik yang besar tak berhingga. Titik-titik dalam kurva memiliki sifat bahwa setiap titik memiliki dua tetangga, kecuali sejumlah kecil titik yang memiliki satu tetangga (ini adalah titik akhir). Beberapa kurva tidak memiliki titik akhir, baik karena tidak berhingga (seperti garis) atau tertutup (berputar dan terhubung ke dirinya sendiri).

Karena "pena" kurva itu tipis (tak berhingga), sulit untuk membuat daerah yang terisi. Sementara kurva-kurva yang mengisi-ruang dimungkinkan (dengan membuatnya terlipat sendiri berkali-kali), kita tidak perlu mempertimbangkan keanehan-keanehan matematis seperti itu di sini. Umumnya, kita menganggap kurva sebagai garis besar sesuatu, bukan "bagian dalam".

Masalah yang perlu kita atasi adalah bagaimana menspesifikasikan sebuah kurva—memberi nama atau representasi pada sebuah kurva sehingga kita dapat merepresentasikannya di komputer. Untuk beberapa kurva, masalah penamaan mereka mudah karena mereka telah mengetahui bentuk: segmen garis, lingkaran, busur elips, dll. Kurva umum yang tidak memiliki bentuk "bernama" kadang-kadang disebut kurva bentuk bebas. Karena kurva bentuk bebas dapat mengambil hampir semua bentuk, mereka jauh lebih sulit untuk ditentukan. Ada tiga cara utama untuk menentukan kurva secara matematis:

1. Representasi kurva implisit mendefinisikan himpunan titik pada kurva dengan memberikan prosedur yang dapat menguji untuk melihat apakah suatu titik pada kurva. Biasanya, representasi kurva implisit didefinisikan oleh fungsi implisit dari bentuk:

$$f(x, y) = 0,$$

sehingga kurva adalah himpunan titik-titik yang persamaan ini benar. Perhatikan bahwa fungsi implisit f adalah fungsi skalar (mengembalikan satu bilangan real).

2. Representasi kurva parametrik memberikan mapping dari parameter bebas ke himpunan titik pada kurva. Artinya, parameter gratis ini memberikan indeks ke titik-titik pada kurva. Bentuk parametrik dari sebuah kurva adalah sebuah fungsi yang memberikan posisi pada nilai dari parameter bebas. Secara intuitif, jika Anda memikirkan kurva sebagai sesuatu yang dapat Anda gambar dengan pena di atas

selembar kertas, parameter bebasnya adalah waktu, berkisar pada interval dari waktu kita mulai menggambar kurva hingga waktu yang kita selesaikan. Fungsi parametrik kurva ini memberi tahu kita di mana letak pena setiap saat:

$$(x, y) = \mathbf{f}(t).$$

Perhatikan bahwa fungsi parametrik adalah fungsi bernilai vektor. Contoh ini adalah kurva 2D, jadi output dari fungsinya adalah 2-vektor; dalam 3D akan menjadi 3-vektor.

3. Representasi kurva generatif atau prosedural menyediakan prosedur yang dapat menghasilkan titik pada kurva yang tidak termasuk dalam dua kategori pertama. Contoh deskripsi kurva generatif meliputi skema subdivisi dan fraktal.

Ingatlah bahwa kurva adalah kumpulan titik-titik. Representasi ini memberi kita cara untuk menentukan himpunan tersebut. Setiap kurva memiliki banyak kemungkinan representasi. Untuk alasan ini, matematikawan biasanya berhati-hati untuk membedakan antara kurva dan representasinya. Dalam grafis komputer kita sering ceroboh, karena kita biasanya hanya mengacu pada presentasi, bukan kurva yang sebenarnya. Jadi, ketika seseorang mengatakan "kurva implisit", mereka mengacu pada kurva yang diwakili oleh beberapa fungsi implisit atau fungsi implisit yang merupakan salah satu representasi dari beberapa kurva. Perbedaan seperti itu biasanya tidak penting, kecuali jika kita perlu mempertimbangkan representasi berbeda dari kurva yang sama. Kami akan mempertimbangkan representasi kurva yang berbeda dalam bab ini, jadi kami akan lebih berhati-hati. Ketika kita menggunakan istilah seperti "kurva polinomial", yang kita maksud adalah kurva yang dapat diwakili oleh polinomial.

Dengan definisi yang diberikan di awal bab, untuk sesuatu yang menjadi kurva itu harus memiliki representasi parametrik. Namun, banyak kurva memiliki representasi lain. Misalnya, lingkaran 2D dengan pusatnya di asal dan jari-jari sama dengan 1 dapat ditulis dalam bentuk implisit sebagai

$$f(x, y) = x2 + y2 - 1 = 0,$$

atau dalam bentuk parametrik sebagai

$$(x, y) = \mathbf{f}(t) = (\cos t, \sin t),\ t \in [0, 2\pi).$$

Bentuk parametrik tidak perlu menjadi representasi yang paling sesuai untuk kurva yang diberikan. Bahkan, adalah mungkin untuk memiliki kurva dengan representasi implisit atau generatif sederhana yang sulit untuk menemukan representasi parametrik.

Representasi kurva yang berbeda memiliki kelebihan dan kekurangan. Misalnya, kurva parametrik jauh lebih mudah digambar, karena kita dapat mengambil sampel parameter bebas. Umumnya, bentuk parametrik adalah yang paling umum digunakan dalam grafis komputer karena lebih mudah digunakan. Fokus kami akan pada representasi parametrik dari kurva.

15.2 PARAMETERISASI DAN REPARAMETERISASI

Kurva parametrik mengacu pada kurva yang diberikan oleh fungsi parametrik tertentu selama beberapa interval tertentu. Untuk lebih tepatnya, kurva parametrik memiliki fungsi yang diberikan yaitu mapping dari interval parameter. Seringkali lebih mudah untuk memiliki parameter yang berjalan di atas interval satuan dari 0 hingga 1. Ketika parameter bebas bervariasi selama interval satuan, kita sering menyatakan parameter sebagai u.

Jika kita melihat kurva parametrik menjadi garis yang digambar dengan pena, kita dapat menganggap u = 0 sebagai waktu ketika pena pertama kali diletakkan di atas kertas dan satuan waktu untuk menjadi jumlah waktu yang diperlukan untuk menggambar kurva (u =1

adalah akhir kurva). Kurva dapat ditentukan oleh fungsi yang memetakan waktu (dalam koordinat unit ini) ke posisi. Pada dasarnya, spesifikasi kurva adalah fungsi yang dapat menjawab pertanyaan, “Di mana pena pada waktu u?”

Jika kita diberikan fungsi f(t) yang menentukan kurva pada interval [a,b], kita dapat dengan mudah mendefinisikan fungsi baru f2(u) yang menentukan kurva yang sama pada interval satuan. Pertama-tama kita dapat mendefinisikan

$$g(u) = a + (b - a)u,$$

lalu

$$\mathbf{f}2(u) = \mathbf{f}\,(g(u)).$$

Kedua fungsi, f dan f2 keduanya mewakili kurva yang sama, namun memberikan parameterisasi kurva yang berbeda. Proses membuat parameterisasi baru untuk kurva yang ada disebut reparameterisasi, dan mapping dari parameter lama ke yang baru (g, dalam contoh ini) disebut fungsi reparameterisasi.

Jika kita telah mendefinisikan sebuah kurva dengan beberapa parameterisasi, tak terhingga banyak lagi yang ada (karena kita selalu dapat membuat parameter ulang). Mampu memiliki beberapa parameterisasi kurva berguna, karena memungkinkan kita untuk membuat parameterisasi yang nyaman. Namun, ini juga bisa menjadi masalah, karena mempersulit membandingkan dua fungsi untuk melihat apakah keduanya mewakili kurva yang sama.

Inti dari masalah ini lebih umum: keberadaan parameter bebas (atau elemen waktu) menambahkan elemen yang tidak terlihat dan berpotensi tidak diketahui pada representasi kurva kita. Ketika kita melihat kurva setelah digambar, kita belum tentu tahu waktunya. Pena mungkin bergerak dengan kecepatan konstan sepanjang interval waktu, atau mungkin mulai perlahan dan dipercepat. Misalnya, sementara u =0,5 setengah jalan melalui ruang parameter, mungkin tidak setengah jalan di sepanjang kurva jika gerakan pena mulai perlahan dan dipercepat di akhir. Pertimbangkan representasi berikut dari kurva yang sangat sederhana:

$$(x, y) = \mathbf{f}\,(u) = (u, u),$$
$$(x, y) = \mathbf{f}\,(u) = (u2, u2),$$
$$(x, y) = \mathbf{f}\,(u) = (u5, u5).$$

Ketiga fungsi tersebut mewakili kurva yang sama pada interval satuan; namun ketika u bukan 0 atau 1, f(u) mengacu pada titik yang berbeda tergantung pada representasi kurva.

Jika kita diberikan parameterisasi kurva, kita dapat menggunakannya secara langsung sebagai spesifikasi kurva kita, atau kita dapat mengembangkan parameterisasi yang lebih nyaman. Biasanya, parameterisasi alami dibuat dengan cara yang nyaman (atau alami) untuk menentukan kurva, jadi kita tidak perlu tahu tentang bagaimana kecepatan berubah di sepanjang kurva.

Jika kita tahu bahwa pena bergerak dengan kecepatan konstan, maka nilai parameter bebas memiliki lebih banyak makna. Setengah jalan parameterruang setengah jalan di sepanjang kurva. Daripada mengukur waktu, parameter dapat dianggap mengukur panjang sepanjang kurva. Parameterisasi seperti itu disebut parameterisasi panjang busur karena mereka mendefinisikan kurva dengan fungsi yang memetakan dari jarak sepanjang kurva (dikenal sebagai panjang busur) ke posisi. Kita sering menggunakan variabel s untuk menunjukkan parameter panjang busur.

Secara teknis, parameterisasi adalah parameterisasi panjang busur jika besar tangennya (yaitu, turunan dari parameterisasi terhadap parameter) memiliki besaran konstan. Dinyatakan sebagai persamaan,

$$\left|\frac{df(s)}{ds}\right|^2 = c$$

Menghitung panjang di sepanjang kurva bisa jadi rumit. Secara umum, itu didefinisikan oleh integral dari besaran turunan (secara intuitif, besaran turunan adalah kecepatan gerak penatasi sepanjang kurva). Jadi, jika diberikan nilai untuk parameter v, Anda dapat menghitung s (jarak panjang busur sepanjang kurva dari titik f(0) ke titik f(v)) sebagai (Persamaan 15.1)

$$s = \int_0^v \left|\frac{df(t)}{dt}\right|^2 dt$$

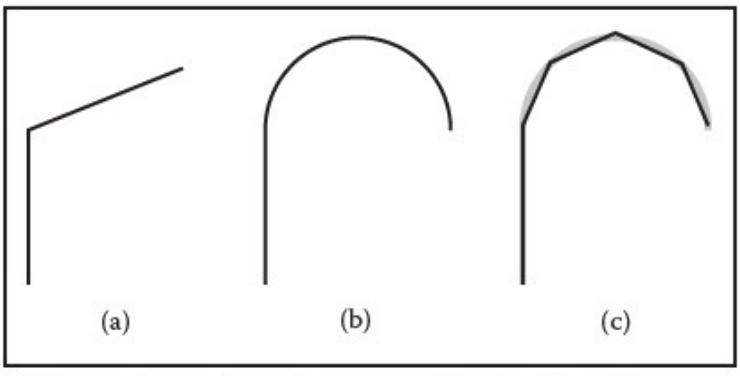
di mana f(t) adalah fungsi yang mendefinisikan kurva dengan parameterisasi alami.

Menggunakan parameterisasi panjang busur membutuhkan kemampuan untuk menyelesaikan Persamaan (15.1) untuk t, diberikan. Untuk banyak jenis kurva yang kita periksa, itu tidak dapat dilakukan dalam bentuk tertutup (sederhana) dan harus dilakukan secara numerik.

Umumnya, kita menggunakan variabel u untuk menyatakan parameter bebas yang berkisar pada interval satuan, s untuk menyatakan parameter bebas panjang busur, dan t untuk menyatakan parameter yang bukan nada dari dua parameter lainnya.

Representasi Parametrik Sepotong

Untuk beberapa kurva, mudah untuk menentukan fungsi parametrik yang mewakili bentuknya. Misalnya, garis, lingkaran, dan elips semuanya memiliki fungsi sederhana yang mendefinisikan titik-titik yang dikandungnya dalam bentuk parameter. Untuk banyak kurva, sulit untuk menemukan fungsi yang menentukan bentuknya. Strategi utama yang kami gunakan untuk membuat kurva kompleks adalah membagi-dan-menaklukkan: kami memecah kurva menjadi beberapa bagian kecil yang lebih sederhana, yang masing-masing memiliki deskripsi sederhana.



Gambar 15.1. (a) Kurva yang dapat dengan mudah direpresentasikan sebagai dua garis; (b) kurva yang dapat dengan mudah direpresentasikan sebagai garis dan busur lingkaran; (c) kurva pendekatan kurva (b) dengan lima segmen garis.

Sebagai contoh, perhatikan kurva pada Gambar 15.1. Dua kurva pertama dengan mudah ditentukan dalam dua bagian. Dalam kasus kurva pada Gambar 15.1(b), kita memerlukan dua jenis potongan yang berbeda: ruas garis dan lingkaran.

Untuk membuat representasi parametrik dari kurva majemuk (seperti kurva pada Gambar 15.1(b)), kita perlu memiliki fungsi parametrik yang beralih di antara fungsi-fungsi yang mewakili potongan. Jika kita mendefinisikan fungsi parametrik pada rentang $0 \leq u \leq 1$, maka kurva pada Gambar 15.1(a)atau (b) dapat didefinisikan sebagai

$$f(u)=\begin{cases} f1(2u)\ jika\ u\le 0.5 \\ jika(2u-1)\ jika\ u>0.5 \end{cases}$$

di mana f_1 adalah parameterisasi bagian pertama, f_2 adalah parameterisasi bagian kedua, dan kedua fungsi ini didefinisikan pada interval satuan. Kita perlu berhati-hati dalam mendefinisikan fungsi f_1 dan f_2 untuk memastikan bahwa potongan-potongan kurva saling cocok. Jika $f_1(1) \neq f_2(0)$, maka potongan kurva kita tidak akan terhubung dan tidak akan membentuk satu kurva kontinu.

Untuk merepresentasikan kurva pada Gambar 15.1(b), kita perlu menggunakan dua jenis potongan yang berbeda: segmen garis dan busur lingkaran. Demi kesederhanaan, kami mungkin lebih suka menggunakan satu jenis bagian. Jika kita mencoba untuk merepresentasikan kurva pada Gambar 15.1(b) dengan hanya satu jenis potongan (segmen garis), kita tidak dapat secara tepat membuat kembali kurva (kecuali jika kita menggunakan jumlah potongan yang tak terhingga). Sementara kurva baru yang terbuat dari segmen garis (seperti pada Gambar 15.1(c)) mungkin tidak persis sama bentuknya seperti pada Gambar 15.1(b), mungkin cukup dekat untuk kita gunakan. Dalam kasus seperti itu, kita mungkin lebih suka kesederhanaan menggunakan potongan segmen garis yang lebih sederhana daripada memiliki kurva yang lebih akurat mewakili bentuk.

Juga, perhatikan bahwa ketika kita menggunakan peningkatan jumlah potongan, kita bisa mendapatkan perkiraan yang lebih baik. Dalam limit (menggunakan jumlah potongan yang tak terhingga), kita dapat dengan tepat merepresentasikan bentuk aslinya.

Satu keuntungan menggunakan representasi sepotong-sepotong adalah memungkinkan kita untuk membuat pertukaran antara

1. seberapa baik kurva yang kita wakili mendekati bentuk sebenarnya yang kita coba wakili;
2. seberapa rumit potongan yang kita gunakan;
3. seberapa banyak potongan yang kita gunakan.

Jadi, jika kita mencoba untuk merepresentasikan bentuk yang rumit, kita mungkin memutuskan bahwa pendekatan kasar dapat diterima dan menggunakan sejumlah kecil potongan sederhana. Untuk meningkatkan aproksimasi, kita dapat memilih antara menggunakan lebih banyak potongan dan menggunakan potongan yang lebih rumit. Dalam praktik grafis komputer, kita cenderung lebih suka menggunakan potongan kurva yang relatif sederhana (baik segmen garis, busur, atau segmen polinomial).

15.3 SPLINES

Sebelum komputer, ketika juru gambar ingin menggambar kurva yang mulus, satu alat yang mereka gunakan adalah sepotong logam kaku yang akan mereka tekuk ke dalam bentuk yang diinginkan untuk dijiplak. Karena logam akan menekuk, tidak melipat, itu akan memiliki bentuk yang halus. Kekakuan berarti bahwa logam akan ditekuk sesedikit mungkin untuk membuat bentuk yang diinginkan. Potongan logam kaku ini disebut spline.

Matematikawan menemukan bahwa mereka dapat mewakili kurva yang dibuat oleh spline juru gambar dengan fungsi polinomial sepotong-sepotong. Awalnya, mereka menggunakan istilah spline yang berarti fungsi polinomial yang halus dan sepotong-sepotong. Baru-baru ini, istilah spline telah digunakan untuk menggambarkan fungsi polinomial sepotong-sepotong. Kami lebih suka definisi yang terakhir ini. Bagi kami, spline adalah fungsi polinomial sepotong-sepotong. Fungsi tersebut sangat berguna untuk mewakili kurva.

15.4 PROPERTIES KURVA

Untuk menggambarkan suatu kurva, kita perlu memberikan beberapa fakta tentang sifat-sifatnya. Untuk kurva “bernama”, properti biasanya spesifik sesuai dengan jenis kurva. Misalnya, untuk menggambarkan sebuah lingkaran, kita dapat memberikan jari-jarinya dan posisi pusatnya. Untuk sebuah elips, kita mungkin juga memberikan orientasi sumbu utama dan rasio panjang sumbu. Namun, untuk kurva bentuk bebas, kita perlu memiliki kumpulan properti yang lebih umum untuk menggambarkan kurva individual.

Beberapa sifat kurva yang diatribusikan hanya pada satu lokasi pada kurva, sedangkan sifat lainnya memerlukan pengetahuan tentang seluruh kurva. Untuk mengetahui perbedaannya, bayangkan kurva itu adalah jalur kereta api. Jika Anda berdiri di lintasan pada hari yang berkabut, Anda dapat mengetahui bahwa lintasan itu lurus atau melengkung dan apakah Anda berada di titik akhir atau tidak. Ini adalah properti lokal. Anda tidak dapat membedakan apakah lintasan itu kurva tertutup atau tidak, atau melintasi dirinya sendiri, atau berapa panjangnya. Kami menyebut jenis properti ini, properti global.

Studi tentang sifat-sifat lokal objek geometris (kurva dan permukaan) dikenal sebagai geometri diferensial. Secara teknis, untuk menjadi properti diferensial, ada beberapa batasan matematis tentang properti (secara kasar, dalam analogi jalur kereta, Anda tidak akan dapat memiliki GPS atau kompas). Daripada khawatir tentang perbedaan ini, kita akan menggunakan istilah properti lokal daripada properti diferensial.

Sifat-sifat lokal adalah alat yang penting untuk menggambarkan kurva karena mereka tidak memerlukan pengetahuan tentang seluruh kurva. Properti lokal termasuk

- kontinuitas,
- posisi di tempat tertentu pada kurva,
- arah pada tempat tertentu pada kurva,
- kelengkungan (dan turunan lainnya).

Seringkali, kami ingin menentukan bahwa kurva mencakup titik tertentu. Suatu kurva dikatakan menginterpolasi suatu titik jika titik tersebut merupakan bagian dari kurva. Sebuah fungsi f menginterpolasi nilai v jika ada beberapa nilai parameter u yang f(t)=v. Kami menyebut tempat interpolasi, yaitu nilai t, situs.

15.5 KONTINUITAS

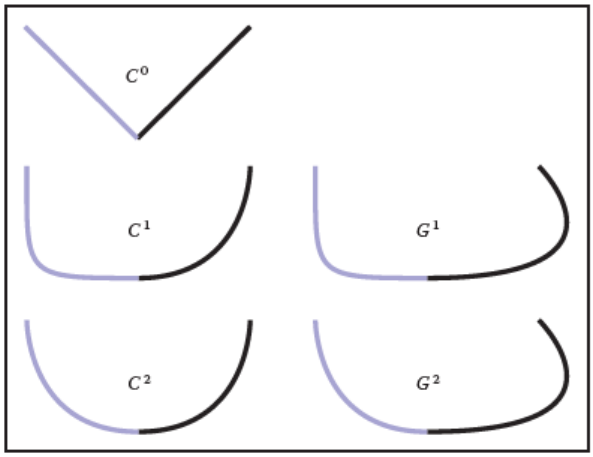
Akan sangat penting untuk memahami sifat lokal dari kurva di mana dua potongan parametrik berkumpul. Jika kurva didefinisikan menggunakan persamaan seperti Persamaan (15.2), maka kita perlu berhati-hati tentang bagaimana potongan didefinisikan. Jika $f_1(1) \neq f_2(0)$, maka kurva akan “patah”—kita tidak akan dapat menggambar kurva dengan goresan pena yang berkelanjutan. Kami menyebut kondisi bahwa potongan-potongan kurva cocok bersama dengan kondisi kontinuitas karena jika mereka bertahan, kurva dapat digambarkan sebagai potongan kontinu. Karena definisi kita tentang "kurva" di awal bab ini membutuhkan kurva yang kontinu, secara teknis "kurva patah" bukanlah kurva.

Selain posisi, kita juga dapat memeriksa apakah perangkat lunak turunan cocok dengan benar. Jika $f'_1(1) \neq f'_2(0)$, maka kurva gabungan akan mengalami perubahan mendadak pada turunan pertamanya pada titik peralihan; turunan pertama tidak kontinu. Secara umum, kita mengatakan bahwa sebuah kurva adalah C^n kontinu jika semua turunannya memenuhi kecocokan lintas bagian. Kami menyatakan posisi itu sendiri sebagai turunan ke-nol, sehingga kondisi kontinuitas C^0 berarti bahwa posisi kurva kontinu, dan kontinuitas C^1 berarti posisi dan turunan pertama kontinu. Definisi kurva membutuhkan kurva menjadi C^0.

Ilustrasi beberapa kondisi kontinuitas ditunjukkan pada Gambar 15.2. Diskontinuitas pada turunan pertama (kurvanya adalah C^0 tetapi bukan C^1) biasanya terlihat karena menunjukkan sudut yang tajam. Diskontinuitas pada turunan kedua terkadang terlihat secara

visual. Diskontinuitas dalam turunan yang lebih tinggi mungkin penting, tergantung pada aplikasinya. Misalnya, jika kurva mewakili suatu gerakan, perubahan mendadak pada turunan kedua terlihat, sehingga kontinuitas turunan ketiga sering berguna. Jika kurva akan memiliki aliran fluida di atasnya (misalnya, jika bentuknya untuk sayap pesawat atau lambung kapal), diskontinuitas pada turunan keempat atau kelima dapat menyebabkan turbulensi.

Jenis kontinuitas yang baru saja kita perkenalkan (C^n) biasanya disebut sebagai kontinuitas parametrik karena bergantung pada parameterisasi dari dua potongan kurva. Jika "kecepatan" masing-masing bagian berbeda, maka mereka tidak akan kontinu. Untuk kasus di mana kita memperhatikan bentuk kurva, dan bukan parameterisasinya, kita mendefinisikan kontinuitas geometrik yang mensyaratkan bahwa turunan dari potongan kurva cocok ketika kurva diparameterisasi secara ekuivalen (misalnya, menggunakan parameterisasi panjang busur). Secara intuitif, ini berarti bahwa turunan-turunan yang bersesuaian harus memiliki arah yang sama, meskipun besarnya berbeda.



Gambar 15.2. Ilustrasi berbagai jenis kontinuitas antara dua segmen kurva.

Jadi, jika kondisi kontinuitas C_1 adalah

$$ff_1'(1)=f_2'(2)$$

kondisi kontinuitas G^1 adalah

$$f_1'(1)=kf_2'(0)$$

untuk beberapa nilai skalar k. Umumnya, kontinuitas geometris kurang restriktif daripada kontinuitas parametrik. Kurva C^n juga G^n kecuali jika turunan parametriknya hilang.

Potongan Polinomial

Representasi kurva yang paling banyak digunakan dalam grafis komputer dilakukan dengan menyatukan elemen-elemen dasar yang didefinisikan oleh polinomial dan disebut potongan polinomial. Misalnya, elemen garis diberikan oleh polinomial linier. Dalam Bagian 15.3.1, kami memberikan definisi formal dan menjelaskan bagaimana menggabungkan potongan-potongan polinomial.

Notasi Polinomial

Polinomial adalah fungsi dari bentuk (Persamaan 15.3)

$$f(t) = a_0 + a_1t + a_2t^2 + \ldots + a_nt^n.$$

Bentuk kanonik tidak selalu memiliki koefisien yang sesuai. Untuk tujuan praktis, di sepanjang bab ini, kita akan menemukan himpunan fungsi basis sedemikian rupa sehingga

koefisien adalah cara yang mudah untuk mengontrol kurva yang diwakili oleh fungsi polinomial.

Untuk menentukan kurva yang tertanam dalam dua dimensi, seseorang dapat menentukan dua polinomial di t: satu untuk bagaimana x bervariasi dengan t dan satu untuk bagaimana y bervariasi dengan t; atau tentukan polinomial tunggal di mana masing-masing ai adalah titik 2D. Situasi analog ada untuk setiap kurva dalam ruang n-dimensi.

15.6 SEGMEN GARIS

Untuk memperkenalkan konsep representasi kurva polinomial sepotong, kita akan membahas segmen garis. Dalam praktiknya, ruas garis sangat sederhana sehingga turunan matematisnya akan tampak berlebihan. Namun, dengan memahami kasus sederhana ini, segalanya akan lebih mudah ketika kita beralih ke polinomial yang lebih rumit.

Pertimbangkan segmen garis yang menghubungkan titik p_0 ke p_1. Kita bisa menulis fungsi parametrik di atas unit domain untuk segmen garis ini sebagai (Persamaan 15.6)

$$\mathbf{f}(u) = (1 - u)\mathbf{p}_0 + u\mathbf{p}_1.$$

Dengan menulis ini dalam bentuk vektor, kita telah menyembunyikan dimensi titik dan fakta bahwa kita berurusan dengan setiap dimensi secara terpisah. Misalnya, jika kita bekerja dalam 2D, kita dapat membuat persamaan terpisah:

$$fx(u) = (1 - u)x_0 + ux_1,$$
$$fy(u) = (1 - u)y_0 + uy_1.$$

Garis yang kita tentukan ditentukan oleh dua titik akhir, tetapi mulai sekarang kita akan tetap menggunakan notasi vektor karena lebih bersih. Kami akan menyebut vektor parameter kontrol, p, titik kontrol, dan setiap elemen p, titik kontrol.

Sementara menggambarkan segmen garis dengan posisi titik akhirnya jelas dan biasanya nyaman, ada cara lain untuk menggambarkan segmen garis. Sebagai contoh,

1. posisi pusat ruas garis, orientasi, dan panjang;
2. posisi satu titik akhir dan posisi titik kedua relatif terhadap yang pertama;
3. posisi tengah ruas garis dan satu titik ujung.

Jelas bahwa dengan memberikan satu jenis deskripsi segmen garis, kita dapat beralih ke yang lain.

Cara lain untuk menggambarkan segmen garis adalah menggunakan bentuk kanonik dari polinomial (seperti yang dibahas dalam Bagian 15.3.1),

Persamaan 15.7

$$\mathbf{f}(u) = \mathbf{a}_0 + u\mathbf{a}_1.$$

Segmen garis apa pun dapat direpresentasikan dengan menentukan a0 dan a1 atau titik akhir (p0 danp1). Biasanya lebih mudah untuk menentukan titik akhir, karena kita dapat menghitung parameter lain dari titik akhir. Untuk menulis bentuk kanonik sebagai ekspresi vektor, kita mendefinisikan vektoru yang merupakan vektor pangkat u:

Persamaan 15.8

$$\mathbf{f}(u) = \mathbf{u} \cdot \mathbf{a}.$$

sehingga Persamaan (15.4) dapat ditulis sebagai

Persamaan 15.9

$$P_0 = f(0) = [1\ 0]\ .\ [a_0\ a_1]$$

$$P_0 = f(0) = [1\ 1]\ .\ [a_0\ a_1]$$

Notasi vektor ini akan mempermudah transformasi antara bentuk kurva yang berbeda.

Persamaan (15.8) menggambarkan segmen kurva dengan himpunan koefisien polinomial untuk bentuk sederhana dari polinomial. Kami menyebut representasi seperti itu bentuk kanonik. Kami akan menyatakan parameter bentuk kanonik dengan a.

Meskipun secara matematis sederhana, bentuk kanonik tidak selalu merupakan cara yang paling mudah untuk menentukan kurva. Misalnya, kita mungkin lebih suka menentukan segmen garis berdasarkan posisi titik akhirnya. Jika kita ingin mendefinisikan p0 sebagai awal segmen (di mana segmen adalah ketika u =0) dan p1 menjadi akhir segmen garis (di mana segmen garis berada di u = 1), kita dapat menulis

$$a_0 = p_0$$

$$a_1 = p_1 - p_0$$

Kita dapat menyelesaikan persamaan ini untuk a0 dan a1:

$$\begin{bmatrix} p0 \\ p1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} a0 \\ a1 \end{bmatrix}$$

Bentuk Matriks untuk Polinomial

Contoh pertama ini cukup mudah untuk diselesaikan, sedangkan contoh yang lebih rumit akan lebih mudah untuk menuliskan Persamaan (15.9) dalam bentuk

$$\mathbf{p} = \mathbf{Ca}$$

di mana kita sebut C, matriks kendala.[1] Jika memiliki vektor titik mengganggu Anda, Anda dapat mempertimbangkan setiap dimensi secara independen (sehingga p adalah $[x_0\ x_1]$ atau $[y_0\ y_1]$ dan a ditangani sesuai).

Kita dapat menyelesaikan Persamaan (15.10) untuk a dengan mencari invers dari C. Matriks invers yang akan kita nyatakan dengan B ini disebut matriks basis. Matriks basis sangat berguna karena memberitahu kita bagaimana mengkonversi antara parameter praktis p dan bentuk kanonik a, dan, oleh karena itu, memberi kita cara mudah untuk mengevaluasi kurva

$$\mathbf{f}\ (u) = \mathbf{u\ B\ p}.$$

Kita dapat menemukan basismatriks untuk bentuk kurva apa pun yang kita inginkan, asalkan tidak ada nonlinier dalam definisi parameter. Contoh parameter yang tidak linier termasuk panjang dan sudut segmen garis.

Sekarang, misalkan kita ingin membuat parameter segmen garis sehingga p_0 adalah titik tengah (u =0,5), dan p_1 adalah titik akhir (u=1). Untuk menurunkan matriks dasar untuk parameterisasi ini, kami menetapkan

$$\mathbf{p}_0 = \mathbf{f}(0.5) = 1\ \mathbf{a}_0 + 0.5\ \mathbf{a}_1,$$
$$\mathbf{p}_1 = \mathbf{f}(1) = 1\ \mathbf{a}_0 + 1\ \mathbf{a}_1.$$

Jadi

$$C = \begin{bmatrix} 1 & 5 \\ 1 & 1 \end{bmatrix}$$

[1] Kami Asumsikan bentuk vektor (baris atau kolom) jelas dari konteksnya, dan kami akan melewatkan semua simbol transpos untuk vektor.

dan oleh karena itu

$$B=C^{-1}=\begin{bmatrix} 2 & -1 \\ -2 & 2 \end{bmatrix}$$

15.7 DI LUAR SEGMEN GARIS

Segmen garis sangat sederhana sehingga menemukan matriks basis adalah hal yang sepele. Namun, itu adalah praktik yang baik untuk kurva dengan derajat yang lebih tinggi. Pertama, mari kita pertimbangkan kuadrat (kurva derajat dua). Keuntungan dari bentuk kanonik (Persamaan (15.4)) adalah bahwa ia bekerja untuk kurva yang lebih rumit ini, hanya dengan membiarkan n menjadi angka yang lebih besar.

Sebuah kuadrat (polinomial derajat-dua) memiliki tiga koefisien, a_0, a_1, dan a_2. Koefisien ini tidak nyaman untuk menggambarkan bentuk kurva. Namun, kita dapat menggunakan metode matriks dasar yang sama untuk merancang parameter yang lebih nyaman. Jika kita mengetahui nilai u, Persamaan (15.4) menjadi persamaan linier dalam parameter, dan aljabar linier dari bagian terakhir masih berfungsi.

Misalkan kita ingin menggambarkan kurva kita dengan posisi awal (u =0), tengah[2] (u =0,5), dan akhir (u=1). Memasukkan nilai yang sesuai ke dalam Persamaan (15.4):

$$\mathbf{p}_0 = \mathbf{f}(0) = \mathbf{a}_0 + 0_1\,\mathbf{a}_1 + 0_2\,\mathbf{a}_2,$$
$$\mathbf{p}_1 = \mathbf{f}(0.5) = \mathbf{a}_0 + 0.5_1\,\mathbf{a}_1 + 0.5_2\,\mathbf{a}_2,$$
$$\mathbf{p}_2 = \mathbf{f}(1) = \mathbf{a}_0 + 1_1\,\mathbf{a}_1 + 1_2\,\mathbf{a}_2.$$

Jadi matriks kendalanya adalah

$$C=\begin{bmatrix} 1 & 0 & 0 \\ 1 & 5 & 25 \\ 1 & 1 & 1 \end{bmatrix}$$

dan matriks basisnya adalah

$$B=C^{-1}=\begin{bmatrix} 1 & 0 & 0 \\ -3 & 4 & -1 \\ 2 & -4 & 2 \end{bmatrix}$$

Ada jenis kendala (atau parameter) tambahan yang terkadang nyaman untuk ditentukan: turunan dari kurva (terhadap parameter bebasnya) pada nilai tertentu. Secara intuitif, turunan memberi tahu kita bagaimana kurva berubah, sehingga turunan pertama memberi tahu kita ke mana arah kurva, turunan kedua memberi tahu kita seberapa cepat kurva berubah arah, dll. Kita akan melihat contoh mengapa berguna untuk menentukan turunanlater.

Untuk kuadrat,

$$\mathbf{f}\,(u) = \mathbf{a}_0 + \mathbf{a}_1 u + \mathbf{a}_2 u_2,$$

turunannya sederhana:

$$f'(u)=\frac{df}{du}=a_1+2a_2u$$

dan

$$f''(u)=\frac{d^2f}{du^2}=\frac{df'}{du}=2a_1$$

[2] Perhatikan bahwa ini adalah bagian tengah dari ruang parameter, yang mungkin bukan bagian tengah dari kurva itu sendiri.

Atau, lebih umum,

$$f'(u)=\sum_{i=1}^{n} iu^{i-1}a_i$$

$$f''(u)=\sum_{i=2}^{n} i(i-1)u^{i-2}a_i$$

Misalnya, pertimbangkan kasus di mana kita ingin menentukan segmen kurva kuadrat dengan posisi, turunan pertama, dan kedua di tengahnya (u = 0,5).

$$\mathbf{p}_0 = \mathbf{f}(0.5) = \mathbf{a}_0 + 0.5_1\,\mathbf{a}_1 + 0.5_2\,\mathbf{a}_2,$$
$$\mathbf{p}_1 = \mathbf{f}\_(0.5) = \mathbf{a}_1 + 2\,0.5\,\mathbf{a}_2,$$
$$\mathbf{p}_2 = \mathbf{f}\_\_(0.5) = 2\,\mathbf{a}_2.$$

matriks kendalanya adalah

$$C=\begin{bmatrix}1 & .5 & .25\\ 0 & 1 & 1\\ 0 & 0 & 2\end{bmatrix}$$

dan matriks basisnya adalah

$$B=C^{-1}=\begin{bmatrix}1 & -.5 & .125\\ 0 & 1 & -.5\\ 0 & 0 & 5\end{bmatrix}$$

15.8 MATRIKS DASAR UNTUK KUBIK

Polinomial kubik populer dalam grafis (Lihat Bagian 15.5). Derivasi untuk berbagai bentuk kubik sama seperti turunan yang telah kita lihat di bagian ini. Kami akan mengerjakan satu contoh lagi untuk latihan.

Bentuk polinomial kubik yang sangat berguna adalah bentuk Hermite, di mana kita menentukan posisi dan turunan pertama di awal dan akhir, yaitu,

$$\mathbf{p}_0 = \mathbf{f}(0) = \mathbf{a}_0 + 0_1\,\mathbf{a}_1 + 0_2\,\mathbf{a}_2 + 0_3\,\mathbf{a}_3,$$
$$\mathbf{p}_1 = \mathbf{f}\_(0) = \mathbf{a}_1 + 2\,0_1\,\mathbf{a}_2 + 3\,0_2\,\mathbf{a}_3,$$
$$\mathbf{p}_2 = \mathbf{f}(1) = \mathbf{a}_0 + 1_1\,\mathbf{a}_1 + 1_2\,\mathbf{a}_2 + 1_3\,\mathbf{a}_3,$$
$$\mathbf{p}_3 = \mathbf{f}\_(1) = \mathbf{a}_1 + 2\,1_1\,\mathbf{a}_2 + 3\,1_2\,\mathbf{a}_3.$$

Jadi, matriks kendalanya adalah

$$\mathrm{C}=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 1 & 1 & 1 & 1\\ 0 & 1 & 2 & 3\end{bmatrix},$$

dan matriks basisnya adalah

$$\mathrm{B}=\mathrm{C}^{-1}=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ -3 & -2 & 3 & -1\\ 2 & 1 & -2 & 1\end{bmatrix}$$

Kami akan membahas splines kubik Hermit di Bagian 15.5.2.

Fungsi Pencampuran

Jika kita mengetahui matriks basis, B, kita dapat mengalikannya dengan vektor parameter, u, untuk mendapatkan vektor fungsi

$$\mathbf{b}(u) = \mathbf{uB}.$$

Perhatikan bahwa kita menyatakan vektor ini dengan b(u) untuk menekankan fakta bahwa nilainya bergantung pada parameter bebas u. Kami menyebut elemen b(u) sebagai fungsi pencampuran, karena elemen tersebut menentukan bagaimana menggabungkan nilai-nilai vektor titik kontrol bersama-sama:

Rumus 15.11

$$f(u)=\sum_{i=0}^{n} b_i(u)\, p_i$$

Penting untuk dicatat bahwa untuk nilai u yang dipilih, Persamaan (15.11) adalah persamaan linier yang menentukan campuran linier (atau rata-rata tertimbang) dari titik kontrol. Ini benar tidak peduli berapa derajat polinomial yang "tersembunyi" di dalam fungsi bi.

Fungsi pencampuran menyediakan abstraksi yang bagus untuk menggambarkan kurva. Setiap jenis kurva dapat direpresentasikan sebagai kombinasi linier dari titik kontrolnya, di mana bobot tersebut dihitung sebagai beberapa fungsi arbitrer dari parameter bebas.

15.9 INTERPOLASI POLINOMIAL

Secara umum, polinomial berderajat n dapat menginterpolasi sekumpulan nilai n +1. Jika kita diberikan vektor p = (p_0, ..., p_n) titik untuk diinterpolasi dan vektor t = (t_0, ..., t_n) dengan nilai parameter yang meningkat, $t_i \neq t_j$, kita dapat menggunakan metode dijelaskan pada bagian sebelumnya untuk menentukan n+1×n+1basismatriksyang memberi kita fungsi f(t) sedemikian rupa sehingga f(ti)=pi. Untuk setiap vektor t yang diberikan, kita perlu menyiapkan dan menyelesaikan sistem linier n =1×n +1. Ini memberi kita satu set n +1 fungsi basis yang melakukan interpolasi:

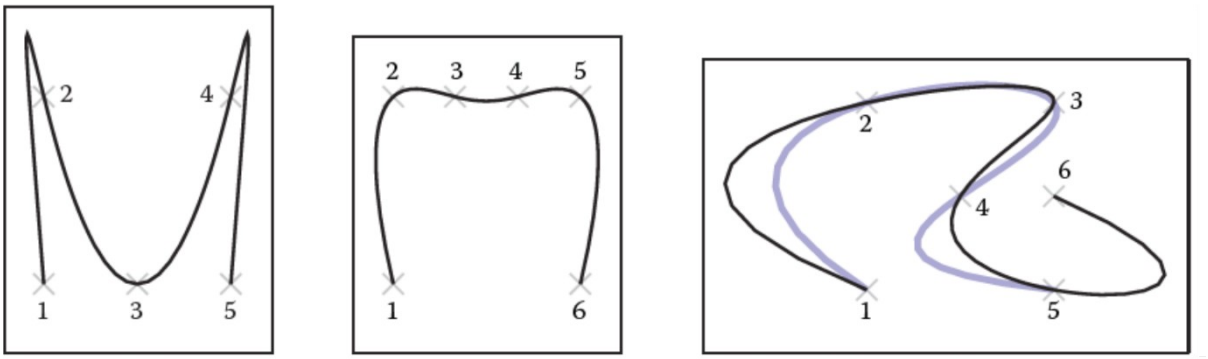
$$f(t)=\sum_{i=0}^{n} p_i b_i(t)$$

Fungsi basis interpolasi ini dapat diturunkan dengan cara lain. Salah satu cara yang sangat elegan untuk mendefinisikannya adalah bentuk Lagrange:

Rumus 15.12

$$b_i=\prod_{j=0,\, j\neq i}^{n} \frac{x-t}{t_i-t_j}$$

Ada cara komputasi yang lebih efisien untuk mengekspresikan fungsi basis interpolasi daripada bentuk Lagrange (lihat De Boor (1978) untuk detailnya).



(a) Interpolasi polinomial melalui lima titik

(b) Interpolasi polinomial melalui enam titik

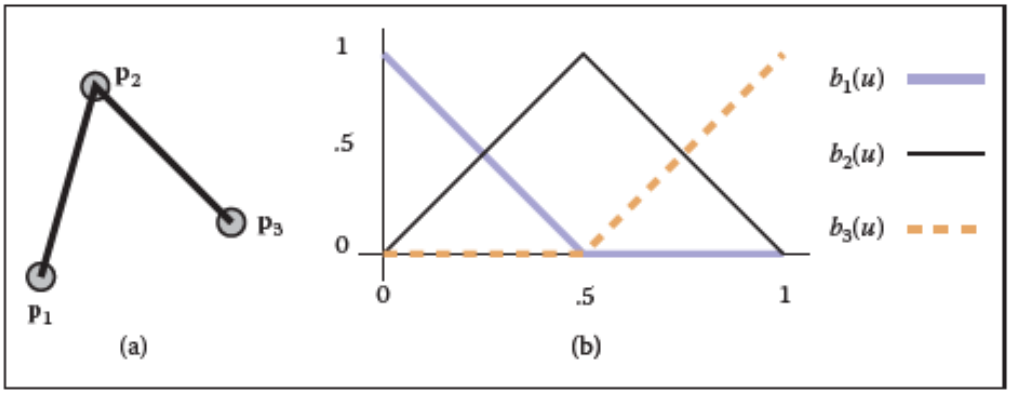
(c) Interpolasi polinomial melalui lima dan enam titik

Gambar 15.3. Interpolasi polinomial melalui beberapa titik. Perhatikan goyangan ekstra dan over-shooting di antara titik-titik. Dalam (c), ketika titik keenam ditambahkan, itu benar-benar mengubah bentuk kurva karena sifat interpolasi polinomial non-lokal.

Interpolasi polinomial menyediakan mekanisme untuk mendefinisikan kurva yang menginterpolasi sekumpulan titik. Gambar 15.3 menunjukkan beberapa contoh. Meskipun dimungkinkan untuk membuat polinomial tunggal untuk menginterpolasi sejumlah titik, kami jarang menggunakan polinomial orde tinggi untuk mewakili kurva dalam grafis komputer. Sebaliknya, interpolasi splines (fungsi polinomial sepotong-sepotong) lebih disukai. Beberapa alasan untuk ini dipertimbangkan dalam Bagian 15.5.3.

Menyatukan Potongan

Sekarang kita telah melihat bagaimana membuat potongan individu dari kurva polinomial, kita dapat mempertimbangkan bagaimana menyatukan potongan-potongan ini.



Gambar 15.4. (a) Dua ruas garis menghubungkan tiga titik; (b) fungsi pencampuran untuk masing-masing titik digambarkan di sebelah kanan.

15.10 SIMPUL

Ide dasar dari fungsi parametrik sepotong-sepotong adalah bahwa setiap bagian hanya digunakan pada beberapa rentang parameter. Sebagai contoh, jika kita ingin mendefinisikan sebuah fungsi yang memiliki dua segmen linier sepotong-sepotong yang menghubungkan tiga titik (seperti yang ditunjukkan pada Gambar 15.4(a)), kita dapat mendefinisikan

$$f(u)=\begin{cases} f_1(2u) \; jika \; 0 \le u < \frac{1}{2} \\ f_2(2u-1) \; jika \; \frac{1}{2} \le u < 1 \end{cases}$$

di mana f_1 dan f_2 adalah fungsi untuk masing-masing dari dua segmen garis. Perhatikan bahwa kami telah mengubah skala parameter untuk masing-masing bagian untuk memudahkan penulisan persamaannya sebagai

$$\mathbf{f}_1(u) = (1 - u)\mathbf{p}_1 + u\mathbf{p}_2.$$

Untuk setiap polinomial dalam fungsi piecewise kami, ada situs (atau nilai parameter) di mana ia dimulai dan diakhiri. Situs di mana fungsi bagian dimulai atau berakhir disebut simpul. Untuk contoh pada Persamaan (15.13), nilai simpulnya adalah 0, 0,5, dan 1.

Kami juga dapat menulis fungsi polinomial sepotong-sepotong sebagai jumlah dari fungsi dasar, masing-masing diskalakan dengan koefisien. Misalnya, kita dapat menulis ulang dua segmen garis Persamaan (15.13) sebagai

$$\mathbf{f}(u) = \mathbf{p}_1 b_1(u) + \mathbf{p}_2 b_2(u) + \mathbf{p}_3 b_3(u),$$

di mana fungsi b1(u) didefinisikan sebagai

$$b_1(u) = \begin{cases} 1 - 2u \ jika \ 0 \le u < \frac{1}{2} \\ 0 \ jika \ tidak \end{cases}$$

dan b2 dan b3 didefinisikan dengan cara yang sama. Fungsi-fungsi ini diplot pada Gambar 15.4(b).

Simpul fungsi polinomial adalah kombinasi simpul dari semua bagian yang digunakan untuk membuatnya. Vektor simpul adalah vektor yang menyimpan semua nilai simpul dalam urutan menaik.

Perhatikan bahwa dalam bagian ini kita telah menggunakan dua mekanisme berbeda untuk menggabungkan potongan polinomial: menggunakan potongan polinomial independen untuk rentang parameter yang berbeda dan memadukan fungsi polinomial sepotong demi sepotong.

Menggunakan Potongan Independen

Dalam Bagian 15.3, kami mendefinisikan potongan polinomi juga di atas rentang parameter unit. Jika kita ingin merakit potongan-potongan ini, kita perlu mengonversi dari parameter fungsi keseluruhan ke nilai parameter untuk potongan. Cara paling sederhana untuk melakukannya adalah dengan menentukan kurva keseluruhan pada rentang parameter[0,n] di mana n adalah jumlah segmen. Bergantung pada nilai parameter, kami dapat menggesernya ke kisaran yang diperlukan.

Menyatukan Segmen

Jika kita ingin membuat kurva tunggal dari dua segmen garis, kita perlu memastikan bahwa ujung segmen garis pertama berada di lokasi yang sama dengan awal segmen berikutnya. Ada tiga cara untuk menghubungkan dua segmen (dalam urutan kesederhanaan):

1. Gambarkan segmen garis dengan dua titik akhir, lalu gunakan titik yang sama untuk keduanya. Kami menyebutnya skema poin bersama.
2. Salin nilai akhir segmen pertama ke awal segmen kedua setiap kali parameter segmen pertama berubah. Kami menyebutnya skema ketergantungan.
3. Tulis persamaan eksplisit untuk koneksi, dan terapkan melalui metode numerik karena parameter lainnya diubah.

Sementara skema yang lebih sederhana lebih disukai karena membutuhkan lebih sedikit pekerjaan, mereka juga menempatkan lebih banyak batasan pada cara segmen garis diparameterisasi. Misalnya, jika kita ingin menggunakan bagian tengah segmen garis sebagai parameter (sehingga pengguna dapat menentukannya secara langsung), kita akan

menggunakan awal setiap segmen garis dan bagian tengah segmen garis sebagai parameternya. Ini akan memaksa kita untuk menggunakan skema ketergantungan.

Perhatikan bahwa jika kita menggunakan skema ketergantungan titik-bersama, jumlah total titik kontrol kurang dari $n*m$, di mana n adalah jumlah segmen dan m adalah jumlah titik kontrol untuk setiap segmen; banyak titik kontrol dari potongan independen akan dihitung sebagai fungsi dari potongan lainnya. Perhatikan bahwa jika kita menggunakan skema titik bersama untuk garis (setiap segmen menggunakan dua titik akhir sebagai parameter dan berbagi titik interior dengan tetangganya), atau jika kita menggunakan skema ketergantungan (seperti contoh dengan titik akhir dan titik tengah pertama), kita berakhir dengan n +1 kontrol untuk kurva n-segmen.

Skema ketergantungan memiliki masalah yang lebih serius. Perubahan di satu tempat dalam kurva dapat merambat melalui seluruh kurva. Ini disebut kurangnya lokalitas. Lokalitas berarti bahwa jika Anda memindahkan titik pada kurva, itu hanya akan mempengaruhi wilayah lokal. Wilayah lokal mungkin besar, tetapi akan terbatas. Jika kontrol kurva tidak memiliki lokalitas, mengubah titik kontrol dapat mempengaruhi titik yang jauh tak terhingga.

Untuk melihat lokalitas, dan kekurangannya, dalam tindakan, pertimbangkan dua rantai segmen garis, seperti yang ditunjukkan pada Gambar 15.5. Satu rantai memiliki bagian-bagiannya yang diparameterisasi oleh titik akhirnya dan menggunakan pembagian titik untuk menjaga kontinuitas. Yang lain memiliki potongannya yang diparameterisasi oleh titik akhir dan titik tengah dan menggunakan propagasi ketergantungan untuk menjaga segmen tetap bersama. Kedua rantai segmen dapat mewakili kurva yang sama: keduanya merupakan himpunan dari n segmen garis yang terhubung. Namun, karena masalah lokalitas, bentuk bersama titik akhir cenderung lebih nyaman bagi pengguna. Pertimbangkan untuk mengubah posisi titik kontrol pertama di setiap rantai. Untuk versi bersama titik akhir, hanya segmen pertama yang akan berubah, sementara semua segmen akan terpengaruh dalam versi titik tengah, seperti pada Gambar 15.5. Faktanya, untuk titik mana pun yang dipindahkan dalam versi bersama titik akhir, paling banyak dua segmen garis akan berubah. Dalam versi titik tengah, semua segmen setelah titik kontrol yang digerakkan akan berubah, bahkan jika rantai panjangnya tak terhingga.

Dalam contoh ini, skema dependensi propagasi adalah yang tidak memiliki kontrol lokal. Hal ini tidak selalu benar. Ada skema direct sharing yang tidak bersifat lokal dan skema propagasi yang bersifat lokal.

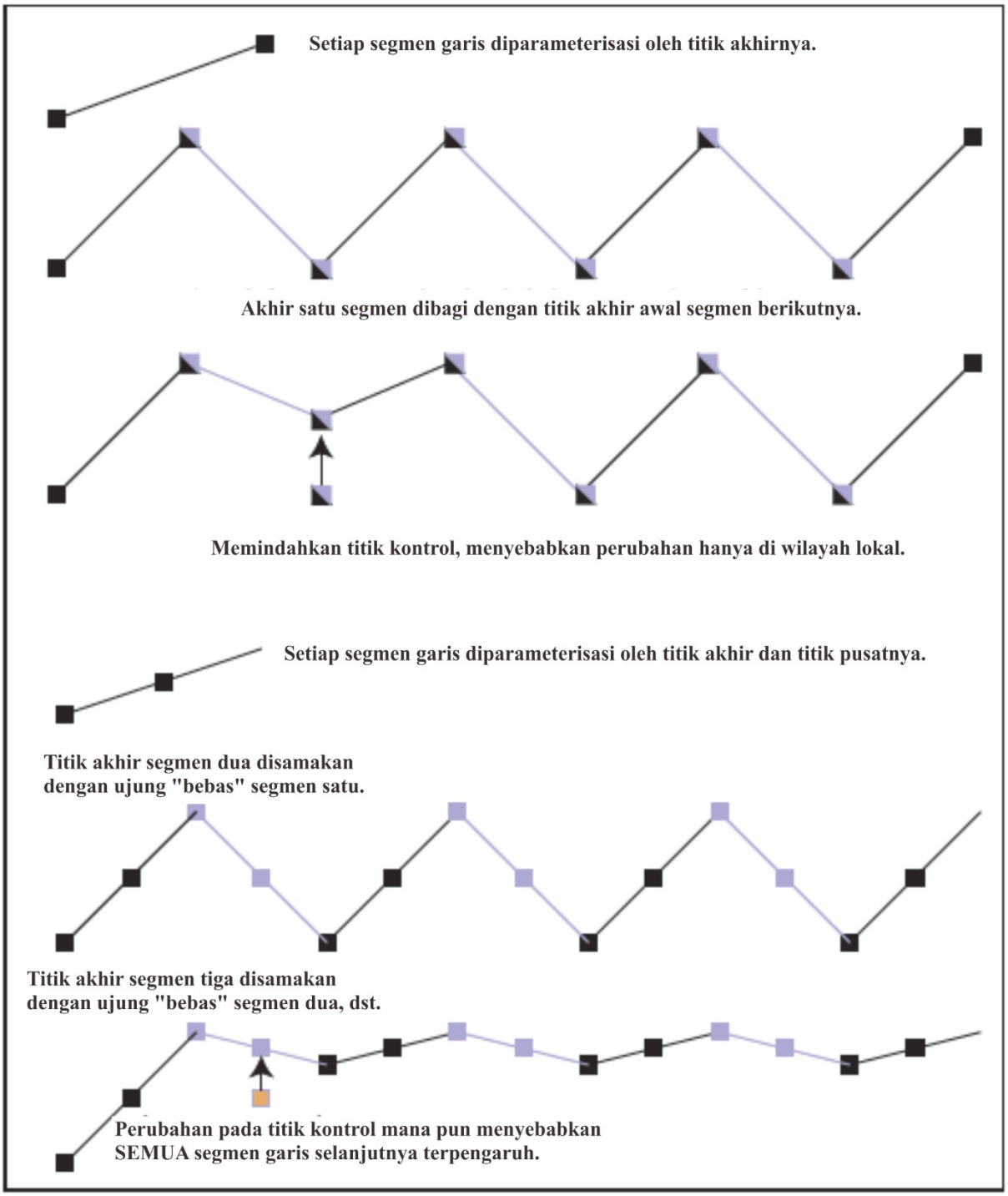
Kami menekankan bahwa lokalitas adalah masalah kenyamanan kontrol. Meskipun tidak nyaman untuk mengubah seluruh kurva setiap saat, perubahan yang sama dapat dilakukan pada kurva. Ini hanya membutuhkan pemindahan beberapa titik secara bersamaan.

15.11 KUBIK

Dalam grafis, ketika kita merepresentasikan kurva menggunakan polinomial sepotong-sepotong, kita biasanya menggunakan segmen garis atau polinomial kubik untuk potongan-potongannya. Ada beberapa alasan mengapa kubik populer dalam grafis komputer:

- Polinomial kubik sepotong-sepotong memungkinkan kontinuitas C^2, yang umumnya cukup untuk sebagian besar tugas visual. Kehalusan C^1 yang ditawarkan kuadrat seringkali tidak mencukupi. Kehalusan yang lebih besar yang ditawarkan oleh polinomial orde tinggi jarang penting.
- Kurva kubik memberikan interpolan kelengkungan minimum untuk menetapkan titik. Yaitu, jika Anda memiliki himpunan n + 3 titik dan mendefinisikan kurva "paling halus" yang melewatinya (yaitu kurva yang memiliki kelengkungan minimum pada panjangnya), kurva ini dapat direpresentasikan sebagai kubus sepotong-potong dengan n segmen .

- Polinomial kubik memiliki simetri simetris dimana posisi dan turunan dapat ditentukan di awal dan akhir.
- Polinomial kubik memiliki tradeoff yang bagus antara masalah numerik dalam komputasi dan kelancaran.



Gambar 15.5. Rantai segmen garis dengan kontrol lokal dan satu dengan kontrol non-lokal

Perhatikan bahwa kita tidak harus menggunakan kubik; mereka hanya cenderung menjadi tradeoff yang baik antara jumlah kelancaran dan kerumitan. Aplikasi yang berbeda mungkin memiliki pengorbanan yang berbeda. Kami fokus pada kubik karena yang paling umum digunakan.

Bentuk kanonik dari polinomialis kubik

$$\mathbf{f}(u) = \mathbf{a}_0 + \mathbf{a}_1 u + \mathbf{a}_2 u^2 + \mathbf{a}_3 u^3.$$

Seperti yang telah kita bahas di Bagian 15.3, koefisien bentuk kanonik ini bukanlah cara yang tepat untuk menggambarkan segmen kubik.

Kami mencari bentuk polinomial kubik yang koefisiennya merupakan cara mudah untuk mengontrol kurva yang dihasilkan yang diwakili oleh kubik. Salah satu kemudahan utama adalah menyediakan cara untuk memastikan keterhubungan potongan dan kontinuitas antar segmen.

Setiap potongan polinomial kubik membutuhkan empat koefisien atau titik kontrol. Itu berarti untuk setiap polinomial dengan n buah, kita mungkin memerlukan hingga 4n titik kontrol jika tidak ada pembagian antar segmen yang dilakukan atau ketergantungan yang digunakan. Lebih sering, beberapa bagian dari setiap segmen dibagi atau bergantung pada segmen yang berdekatan, sehingga jumlah total titik kontrol jauh lebih rendah. Juga, perhatikan bahwa titik kontrol mungkin merupakan posisi atau turunan dari kurva.

Sayangnya, tidak ada representasi "terbaik" tunggal untuk kubik sepotong. Tidak mungkin untuk memiliki representasi kurva polinomial sepotong-sepotong yang memiliki semua properti yang diinginkan berikut ini:

1. setiap bagian kurva adalah kubik;
2. kurva menginterpolasi titik kontrol;
3. kurva memiliki kontrol lokal;
4. kurva memiliki kontinuitas C^2.

Kita dapat memiliki tiga sifat ini, tetapi tidak keempatnya; ada representasi yang memiliki kombinasi tiga. Dalam buku ini, kita akan membahas spline B kubik yang tidak menginterpolasi titik kontrolnya (tetapi memiliki kontrol lokal dan C^2); Spline kardinal dan spline Catmull-Rom yang menginterpolasi titik kontrolnya dan menawarkan kontrol lokal, tetapi bukan C^2; dan kubik alami yang berinterpolasi dan merupakan C^2, tetapi tidak memiliki kontrol lokal.

Sifat kontinuitas kubik mengacu pada kontinuitas antara segmen (pada titik simpul). Potongan kubik itu sendiri memiliki kontinuitas tak terhingga dalam turunannya (seperti yang telah kita bicarakan sejauh ini tentang kontinuitas). Perhatikan bahwa jika Anda memiliki banyak titik kontrol (atau simpul), kurva dapat bergoyang, yang mungkin tidak tampak "halus".

Kubik Alami

Dengan kurva kubik sepotong, dimungkinkan untuk membuat kurva C^2. Untuk melakukan ini, kita perlu menentukan posisi dan turunan pertama dan kedua di awal setiap segmen (sehingga kita dapat memastikan bahwa itu sama dengan di akhir segmen sebelumnya). Perhatikan bahwa setiap segmen kurva menerima tiga dari empat parameternya dari kurva sebelumnya dalam rantai. Rantai kubus kontinu C^2 ini kadang-kadang disebut sebagai garis kubik alami.

Untuk satu segmen kubik alami, kita perlu membuat parameter kubik dengan posisi titik akhir dan turunan pertama dan kedua pada titik awal. Oleh karena itu, titik kontrolnya adalah

$$\begin{aligned}
\mathbf{p}_0 &= \mathbf{f}(0) = \mathbf{a}_0 + 01\mathbf{a}_1 + 0^2\,\mathbf{a}_2 + 0^3\,\mathbf{a}_3, \\
\mathbf{p}_1 &= \mathbf{f}_(0) = 1^1\mathbf{a}_1 + 2\,0^1\,\mathbf{a}_2 + 3\,0^2\,\mathbf{a}_3, \\
\mathbf{p}_2 &= \mathbf{f}__(0) = 2\,1^1\mathbf{a}_2 + 6\,01\,\mathbf{a}_3, \\
\mathbf{p}_3 &= \mathbf{f}(1) = \mathbf{a}_0 + 1^1\,\mathbf{a}_1 + 1^2\,\mathbf{a}_2 + 1^3\,\mathbf{a}_3.
\end{aligned}$$

Oleh karena itu, matriks kendalanya adalah

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix},$$

dan matriks basisnya adalah

$$\mathbf{B} = \mathbf{C}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & .5 & 0 \\ -1 & -1 & -.5 & 1 \end{bmatrix}.$$

Diberikan setofntitik kontrol, spline kubik alami memiliki segmen kubik n-1. Segmen pertama menggunakan titik kontrol untuk menentukan posisi awal, posisi akhir, dan turunan pertama dan kedua di awal. Skema ketergantungan menyalin posisi, dan turunan pertama dan kedua dari akhir segmen pertama untuk digunakan di segmen kedua.

Kerugian dari spline kubik alami adalah tidak bersifat lokal. Setiap perubahan di segmen mana pun mungkin memerlukan seluruh kurva untuk berubah (setidaknya bagian setelah perubahan dilakukan). Lebih buruk lagi, spline kubik alami cenderung tidak berkondisi: perubahan kecil di awal kurva dapat menyebabkan perubahan besar di kemudian hari. Masalah lain adalah bahwa kita hanya memiliki kendali atas turunan kurva pada awalnya. Segmen setelah awal kurva menentukan turunannya dari titik awalnya.

Hermite Cubics

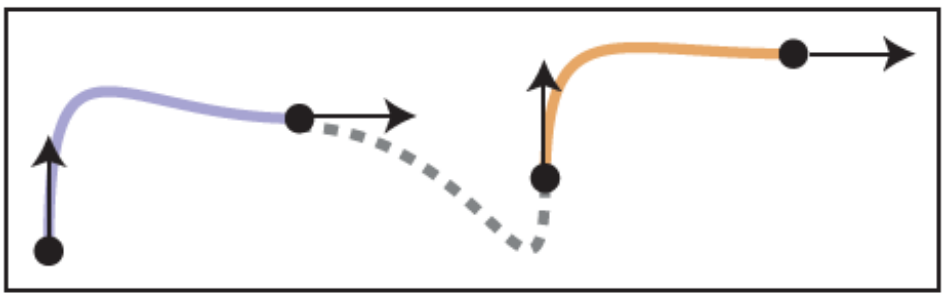
Polinomial kubik pertapa diperkenalkan di Bagian 15.3.4. Segmen dari spline Hermite kubik memungkinkan posisi dan turunan pertama dari kedua titik akhirnya ditentukan. Rantai segmen dapat dihubungkan menjadi spline C^1 dengan menggunakan nilai yang sama untuk posisi dan turunan dari akhir satu segmen dan untuk awal berikutnya.

Diberikan satu set n titik kontrol, di mana setiap titik kontrol lainnya adalah nilai turunan, kubik Hermite spline berisi $(n - 2)/2$ segmen kubik. Spline menginterpolasi titik-titik, seperti yang ditunjukkan pada Gambar 15.6, tetapi hanya dapat menjamin kontinuitas C^1.

Kubus pertapa nyaman karena memberikan kontrol lokal atas bentuk, dan memberikan kontinuitas C^1. Namun, karena pengguna harus menentukan posisi dan turunannya, antarmuka khusus untuk turunan harus disediakan. Salah satu kemungkinan adalah untuk memberikan pengguna dengan titik yang mewakili di mana vektor turunan akan berakhir jika mereka "ditempatkan" pada titik posisi.

Kubik Kardinal

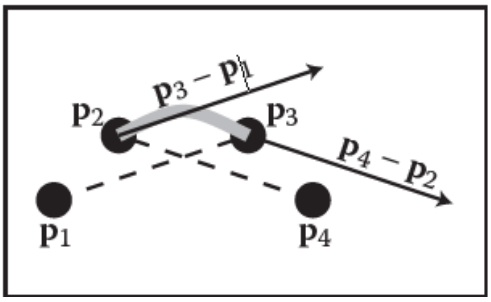
Spline kubik kardinal adalah jenis spline interpolasi C^1 yang terdiri dari segmen polinomial kubik. Diberikan Himpunan n titik kontrol, garis kubik kardinal menggunakan $n-2$ segmen polinomik kubik untuk menginterpolasi semua titik kecuali yang pertama dan terakhir.



Gambar 15.6. Sebuah spline kubik Hermite terdiri dari tiga segmen.

Garis kardinal memiliki parameter yang disebut tegangan yang kontrolnya menunjukkan "ketat" kurva berada di antara titik-titik yang diinterpolasinya. Tegangan adalah angka dalam rentang [0,1) yang mengontrol bagaimana kurva membelok menuju titik kontrol berikutnya. Untuk kasus khusus yang penting dari t =0 , splines disebut spline Catmull-Rom.

Setiap segmen dari spline kardinal menggunakan empat titik kontrol. Untuk segmen i, titik yang digunakan adalah i, i +1 , i +2, dan i +3 karena segmen tersebut berbagi tiga titik dengan tetangganya. Setiap segmen dimulai pada titik kontrol kedua dan berakhir pada titik kontrol ketiga. Turunan pada awal kurva ditentukan oleh vektor antara titik kontrol pertama dan ketiga, sedangkan turunan pada akhir kurva ditentukan oleh vektor antara titik kedua dan keempat, seperti ditunjukkan pada Gambar 15.7.



Gambar 15.7. Segmen dari spline kubik kardinal menginterpolasi titik kontrol kedua dan ketiganya (p2 dan p3), dan menggunakan titik lainnya untuk menentukan turunan di awal dan akhir.

Parameter tegangan menyesuaikan seberapa besar turunannya diskalakan. Secara khusus, turunannya diskalakan dengan (1 t)/2. Oleh karena itu, kendala pada kubik adalah

$$\mathbf{f}(0) = \mathbf{p}_2,$$
$$\mathbf{f}(1) = \mathbf{p}_3,$$
$$\mathbf{f}\_(0) = 1/2\ (1 - t)(\mathbf{p}_3 - \mathbf{p}_1),$$
$$\mathbf{f}\_(1) = 1/2\ (1 - t)(\mathbf{p}_4 - \mathbf{p}_2).$$

Memecahkan persamaan ini untuk titik kontrol (mendefinisikan s = (1−t)/2) memberikan

$$\mathbf{p}0 = \mathbf{f}(1) - {}_{2/1-t}\,\mathbf{f}\_(0) = \mathbf{a}_0 + (1 - {}_{1/s})\,\mathbf{a}_1 + \mathbf{a}_2 + \mathbf{a}_3,$$
$$\mathbf{p}1 = \mathbf{f}(0) = \mathbf{a}_0,$$
$$\mathbf{p}2 = \mathbf{f}(1) = \mathbf{a}_0 + \mathbf{a}_1 + \mathbf{a}_2 + \mathbf{a}_3,$$
$$\mathbf{p}3 = \mathbf{f}(0) + {}_{1/s}\mathbf{f}\_(1) = \mathbf{a}_0 + {}_{1/s}\mathbf{a}1 + 2{}_{1/s}\mathbf{a}_2 + 3{}_{1/s}\mathbf{a}_3.$$

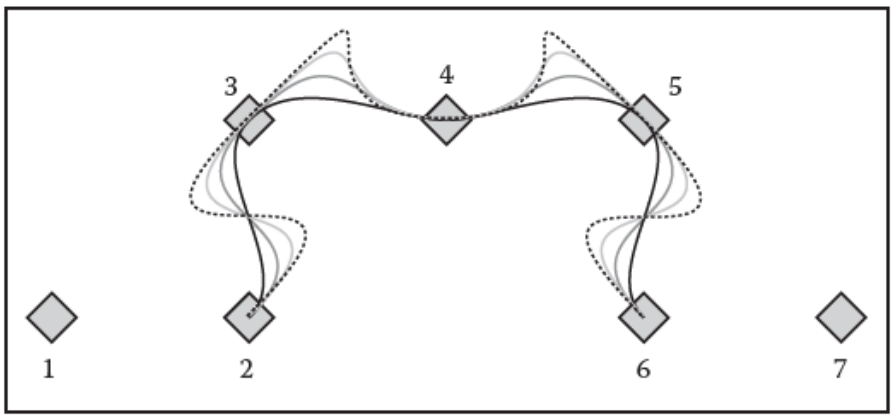
Ini menghasilkan matriks kardinal

$$\mathbf{B} = \mathbf{C}^{-1} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -s & 0 & s & 0 \\ 2s & s-3 & 3-2s & -s \\ -s & 2-s & s-2 & s \end{bmatrix}$$

Karena titik ketiga dari segmen adalah titik kedua dari segmen +1, segmen yang berdekatan dari spline kardinal terhubung. Demikian pula, titik yang sama digunakan untuk menentukan turunan pertama dari setiap segmen, memberikan kontinuitas C^1.

Spline kardinal berguna, karena menyediakan cara mudah untuk menginterpolasi sekumpulan titik dengan kontinuitas C^1 dan kontrol lokal. Mereka hanya C^1, jadi mereka terkadang mendapatkan "ketegaran" di dalamnya. Parameter tegangan memberikan beberapa kontrol atas apa yang terjadi antara titik-titik yang diinterpolasi, seperti yang ditunjukkan pada

Gambar 15.8, di mana satu set splines kardinal melalui satu set poin ditampilkan. Kurva menggunakan titik kontrol yang sama, tetapi menggunakan nilai yang berbeda untuk parameter tegangan. Perhatikan bahwa titik kontrol pertama dan terakhir tidak diinterpolasi.



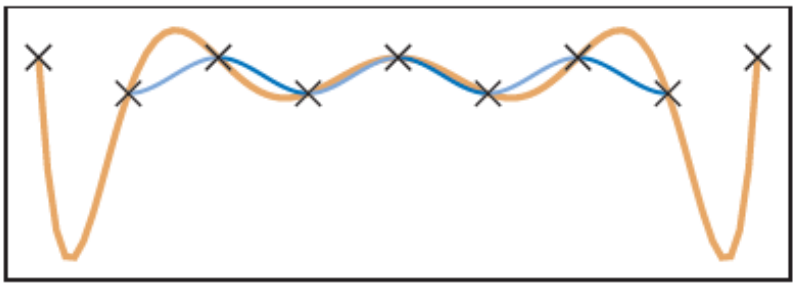
Gambar 15.8. Cardinal splines melalui tujuh titik kontrol dengan nilai parameter tegangan t yang bervariasi.

Mengingat satu set n poin untuk interpolasi, Anda mungkin bertanya-tanya mengapa kita mungkin lebih suka menggunakan spline kubus kardinal (yaitu satu set n − 2 potongan kubik) daripada tunggal, urutan n polinomial seperti yang dijelaskan di Bagian 15.3.6. Beberapa kelemahan polinomial interpolasi adalah:

- Polinomial interpolasi cenderung melampaui titik-titik, seperti terlihat pada Gambar 15.9. Overshooting ini menjadi lebih buruk karena jumlah poin bertambah besar. Garis kardinal cenderung berperilaku baik di antara titik-titik.
- Kontrol polinomialis interpolasi bukan lokal. Mengubah titik di awal spline mempengaruhi seluruh spline. Spline kardinal bersifat lokal: setiap tempat pada spline paling banyak dipengaruhi oleh empat titik tetangganya.
- Evaluasi polinomial interpolasi tidak lokal. Mengevaluasi suatu titik pada polinomial membutuhkan akses ke semua titiknya. Mengevaluasi sebuah titik pada kubus sepotong-sepotong membutuhkan sejumlah kecil perhitungan yang tetap, tidak peduli seberapa besar jumlah titik totalnya.

Ada berbagai masalah numerik dan teknis lainnya dalam menggunakan spline interpolasi karena jumlah poin bertambah besar. Lihat De Boor (2001) untuk informasi lebih lanjut.

Sebuah spline kardinal memiliki kelemahan yang tidak menginterpolasi titik pertama atau terakhir, yang dapat dengan mudah diperbaiki dengan menambahkan titik tambahan di kedua ujung urutan. Spline kardinal juga tidak kontinu—hanya memberikan kontinuitas C1 pada simpul.



Gambar 15.9. Splines interpolasi sembilan titik kontrol (ditandai dengan salib kecil).

Garis oranye tebal menunjukkan polinomial interpolasi. Garis tipis menunjukkan spline Catmull-Rom. Yang terakhir ini terbuat dari tujuh segmen kubik, yang masing-masing ditampilkan dalam nada biru bergantian.

Tampaknya cara termudah untuk mengontrol kurva adalah dengan menentukan sekumpulan titik untuk diinterpolasi. Namun, dalam praktiknya, skema interpolasi sering kali memiliki sifat yang tidak diinginkan karena memiliki kontinuitas yang lebih rendah dan tidak memberikan kendali atas apa yang terjadi di antara titik-titik tersebut. Skema kurva yang hanya mendekati titik sering lebih disukai. Dengan skema aproksimasi, titik kontrol mempengaruhi bentuk kurva, tetapi tidak menentukannya secara tepat. Meskipun kita melepaskan kemampuan untuk secara langsung menentukan titik-titik untuk dilalui kurva, kita memperoleh perilaku kurva dan kontrol lokal yang lebih baik. Jika kita perlu menginterpolasi sekumpulan titik, posisi titik kontrol dapat dihitung sedemikian rupa sehingga kurva melewati titik interpolasi ini. Dua jenis kurva aproksimasi yang paling penting dalam grafis komputer adalah kurva Be´zier dan kurva B-spline.

15.12 KURVA B´EZIER

Kurva Be´zier adalah salah satu representasi paling umum untuk kurva bentuk bebas dalam grafis komputer. Kurva diberi nama untuk Pierre Be´zier, salah satu orang yang berperan penting dalam perkembangannya. Kurva Be´zier memiliki sejarah yang menarik di mana mereka dikembangkan secara bersamaan oleh beberapa kelompok independen.

Kurva Be´zier adalah kurva polinomial yang mendekati titik kontrolnya. Kurva dapat berupa polinomial dengan derajat berapa pun. Kurva derajat d dikendalikan oleh d+1titik kontrol. Kurva menginterpolasi titik kontrol pertama dan terakhirnya, dan bentuknya secara langsung dipengaruhi oleh titik lainnya.

Seringkali, bentuk kompleks dibuat dengan menghubungkan sejumlah kurva Be´zier derajat rendah, dan dalam grafis komputer, kubik(d =3) Kurva Be´zier biasanya digunakan untuk tujuan ini. Banyak program ilustrasi populer, seperti Adobe Illustrator, dan skema representasi font, seperti yang digunakan dalam Postscript, menggunakan kurva Be´zier kubik. Kurva B´ezier sangat populer dalam grafis komputer karena mudah dikendalikan, memiliki sejumlah properti yang berguna, dan sangat efisien algoritma untuk bekerja dengan mereka. Kurva Be´zier dikonstruksi sedemikian rupa sehingga:

- Kurva menginterpolasi titik kontrol pertama dan terakhir, dengan u =0 dan 1, masing-masing.
- Turunan pertama dari kurva di awal (akhir) ditentukan oleh vektor antara titik kontrol pertama dan kedua (di sebelah terakhir dan terakhir). Turunan diberikan oleh vektor-vektor antara titik-titik ini yang diskalakan dengan derajat kurva.
- Turunan yang lebih tinggi pada awal (akhir) kurva bergantung pada titik-titik di awal (akhir) kurva. Turunan ke-n bergantung pada n +1 poin pertama (terakhir).

Misalnya, perhatikan kurva Be´zierderajat3(kubik)seperti pada Gambar 15.10. Kurva memiliki empat (d+1) titik kontrol. Ini dimulai pada titik kontrol pertama (p_0) dan berakhir pada titik terakhir (p_1). Turunan pertama di awal sebanding dengan vektor antara titik kontrol pertama dan kedua ($p_- p_0$). Secara khusus, $f(0) = 3(p_1 - p_0)$. Demikian pula, turunan pertama pada akhir kurva diberikan oleh $f(1) = 3(p_3 - p_2)$. Turunan kedua pada awal kurva dapat ditentukan dari titik kontrol p_0, p_1 dan p_2.

Dengan menggunakan fakta tentang Be´ziercubic dalam paragraf sebelumnya, kita dapat menggunakan metode Bagian 15.5 untuk membuat fungsi parametrik untuknya. Definisi interpolasi dan turunan awal dan akhir memberikan

$$\mathbf{p}_0 = \mathbf{f}(0) = \mathbf{a}_3 0^3 + \mathbf{a}_2 0^2 + \mathbf{a}_1 0 + \mathbf{a}_0,$$

$$\mathbf{p}_3 = \mathbf{f}(1) = \mathbf{a}_3 1^3 + \mathbf{a}_2 1^2 + \mathbf{a}_1 1 + \mathbf{a}_0,$$
$$3(\mathbf{p}_1 - \mathbf{p}_0) = \mathbf{f}\_(0) = 3\mathbf{a}_3 0^2 + 2\mathbf{a}_2 0 + \mathbf{a}_1,$$
$$3(\mathbf{p}_3 - \mathbf{p}_2) = \mathbf{f}\_(1) = 3\mathbf{a}_3 1^2 + 2\mathbf{a}_2 1 + \mathbf{a}_1.$$

Ini dapat diselesaikan untuk matriks dasar

$$\mathrm{B} = \mathrm{C}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix},$$

dan kemudian ditulis sebagai

$$\mathbf{f}(u) = (1-3u+3u2-u3)\mathbf{p}_0 + (3u-6u^2+3u^3)\mathbf{p}_1 + (3u^2-3u^3)\mathbf{p}_2 + (u^3)\mathbf{p}_3,$$

atau

$$f(u) = \sum_{i=0}^{d} b_{i,3}\, p_i$$

di mana $b_{i,3}$ adalah fungsi pencampuran Be´zier dari derajat 3:

$$b_{0,3} = (1-u)^3,$$
$$b_{1,3} = 3u(1-u)^2,$$
$$b_{2,3} = 3u^2(1-u),$$
$$b_{3,3} = u^3.$$

Untungnya, fungsi pencampuran untuk kurva Be´zier memiliki bentuk khusus yang berfungsi untuk semua derajat. Fungsi-fungsi ini dikenal sebagai polinomial basis Bernstein dan memiliki bentuk umum

$$b_{k,n}(u) = C(n, k)\, u^k\, (1-u)^{(n-k)},$$

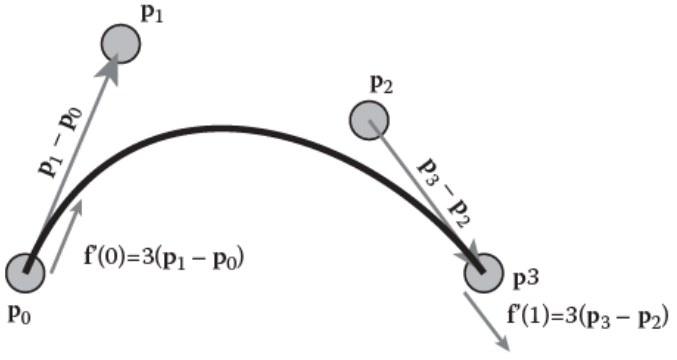
di mana n adalah orde kurva Be´zier, dan k adalah bilangan fungsi campuran antara 0 dan n (inklusif). *C(n,k)* adalah koefisien binomial:

$$C(n,k) = \frac{n!}{k!(n-k)!}$$

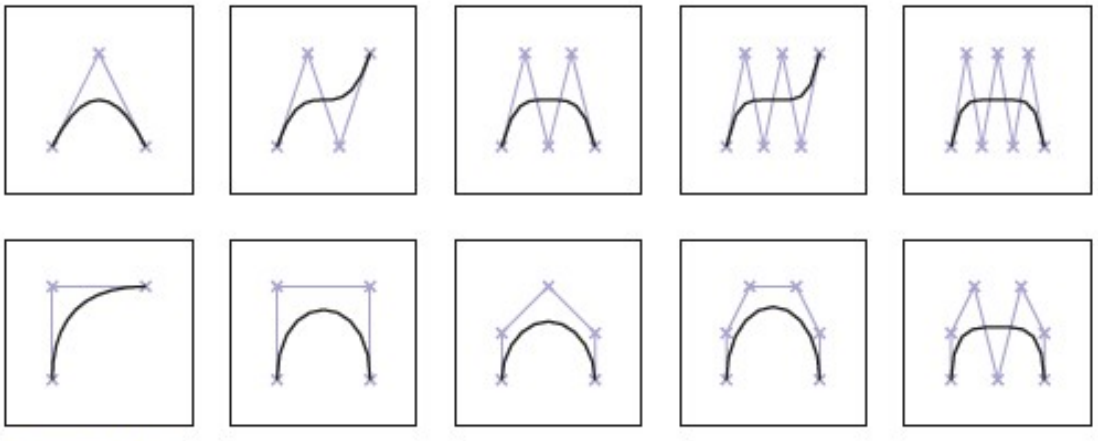
Diberikan posisi titik kontrol p_k, fungsi untuk mengevaluasi kurva Be´zier orde n (dengan n +1 titik kontrol) adalah

$$p(u) = \sum_{k=0}^{n} p_k C(n,k) u^k ¿$$

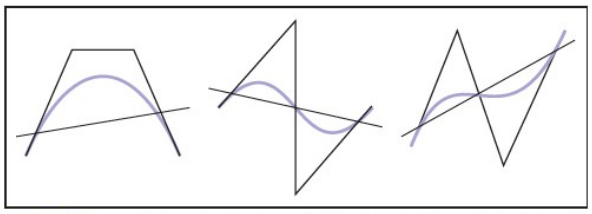
Beberapa segmen Be´zier ditunjukkan pada Gambar 15.11.



Gambar 15.10. Kurva Be´zier kubik dikendalikan oleh empat titik. Ini menginterpolasi yang pertama dan terakhir, dan turunan awal dan akhir adalah tiga kali vektor antara dua titik pertama (atau dua terakhir).



Gambar 15.11. Berbagai segmen Be´zier derajat 2–6. Titik kontrol ditunjukkan dengan tanda silang, dan poligon kontrol (segmen garis yang menghubungkan titik kontrol) juga ditampilkan.



Gambar 15.12. Variasi properti yang semakin berkurang dari kurva Be´zier berarti bahwa kurva tidak melewati garis lebih dari poligon kontrolnya.

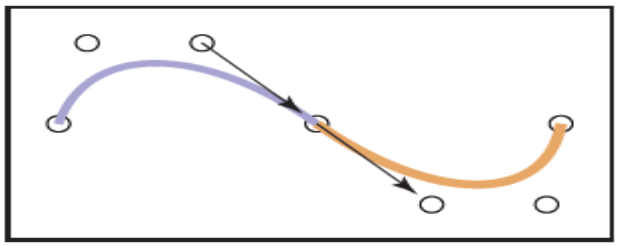
Jika poligon kontrol tidak memiliki "goyangan", kurva juga tidak akan memilikinya. B-splines (Bagian 15.6.2) juga memiliki properti ini.

Segmen Beziers memiliki beberapa sifat yang berguna:

- Kurva dibatasi oleh convexhull dari titik kontrol.
- Setiap garis memotong kurva tidak lebih dari memotong himpunan segmen garis yang menghubungkan titik kontrol. Ini disebut properti yang semakin berkurang variasi. Properti ini diilustrasikan pada Gambar 15.12.
- Kurvanya simetris: membalik urutan titik kontrol menghasilkan kurva yang sama, dengan parameterisasi terbalik.
- Kurvanya sangat invarian. Ini berarti bahwa menerjemahkan, menskalakan, memutar, atau memiringkan titik kontrol sama dengan melakukan operasi tersebut pada kurva itu sendiri.
- Ada algoritme sederhana yang bagus untuk mengevaluasi dan membagi kurva Be´zier menjadi potongan-potongan yang merupakan kurva Be´zier itu sendiri. Karena subdivisi dapat dilakukan secara efektif menggunakan algoritma yang dijelaskan kemudian, pendekatan membagi dan menaklukkan dapat digunakan untuk membuat algoritma yang efektif untuk tugas-tugas penting seperti rendering kurva Be´zier, pendekatan mereka dengan segmen garis, dan menentukan persimpangan antara dua kurva.

Saat segmen Be´zier dihubungkan bersama untuk membuat spline, konektivitas antar segmen dibuat dengan berbagi titik akhir. Namun, kontinuitas turunan harus dibuat dengan memposisikan titik kontrol lainnya. Ini memberi pengguna spline Be´zier kontrol atas kelancaran. Untuk G^1 memberikan pengguna spline Be´zier kontrol atas kelancaran. Untuk kontinuitas G^1, titik kedua hingga terakhir dari kurva pertama dan titik kedua dari kurva kedua

harus kolinear dengan titik-titik akhir yang disamakan. Untuk kontinuitas C^1, jarak antar titik juga harus sama. Hal ini diilustrasikan pada Gambar 15.13. Tingkat kontinuitas yang lebih tinggi dapat dibuat dengan memposisikan lebih banyak titik dengan benar.

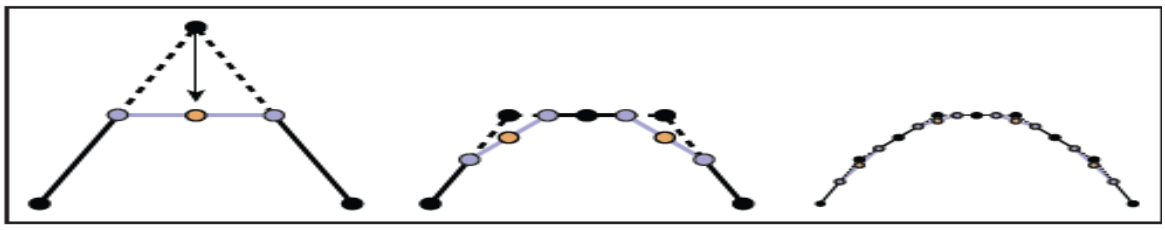


Gambar 15.13. Dua segmen Be´zier terhubung ke sebuah spline C^1, karena vektor antara dua titik terakhir dari segmen pertama sama dengan vektor antara dua titik pertama dari segmen kedua.

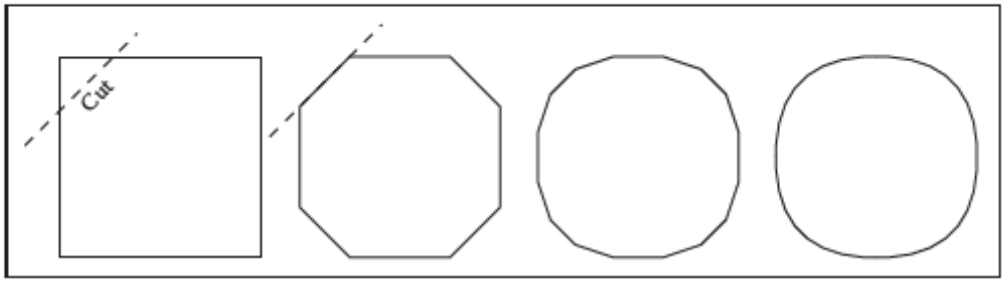
Intuisi Geometris untuk Kurva B´ezier

Beziercurves dapat diturunkan dari prinsip-prinsip geometrik, serta dari metode aljabar yang dijelaskan di atas. Kami menguraikan prinsip-prinsip geometris karena mereka memberikan intuisi tentang cara kerja kurva Be´zier.

Bayangkan bahwa kita memiliki satu set titik kontrol yang darinya kita ingin membuat kurva yang mulus. Cukup menghubungkan titik dengan garis (untuk membentuk poligon kontrol) akan menghasilkan sesuatu yang tidak mulus. Ini akan memiliki sudut tajam. Kita dapat membayangkan "menghaluskan" poligon ini dengan memotong sudut yang tajam, menghasilkan poligon baru yang lebih halus, tetapi masih tidak "halus" dalam pengertian matematis (karena kurva masih berupa poligon, dan karena itu hanya C^1). Kita dapat mengulangi proses ini, setiap kali menghasilkan poligon yang lebih halus, seperti yang ditunjukkan pada Gambar 15.14. Dalam limitnya, yaitu jika kita mengulangi proses itu berkali-kali tak berhingga, kita akan memperoleh kurva mulus C^1.



Gambar 15.14. Prosedur subdivisi untuk Be´ziers kuadrat. Setiap segmen garis dibagi menjadi dua dan titik tengah ini terhubung (titik dan garis biru). Titik kontrol interior dipindahkan ke titik tengah segmen garis baru (titik oranye).



Gambar 15.15. Dengan berulang kali memotong sudut poligon, kita mendekati kurva mulus.

Apa yang telah kami lakukan dengan pemotongan sudut adalah mendefinisikan skema subdivisi. Artinya, kita telah mendefinisikan kurva dengan suatu proses untuk memecah kurva yang lebih sederhana menjadi potongan-potongan yang lebih kecil (misalnya, membaginya). Kurva yang dihasilkan adalah kurva limit yang dicapai dengan menerapkan proses berkali-kali tak berhingga. Jika skema subdivisi didefinisikan dengan benar, hasilnya akan menjadi kurva halus, dan akan memiliki bentuk parametrik.

Mari kita pertimbangkan untuk menerapkan pemotongan sudut ke satu sudut. Diberikan tiga titik (p_0, p_1, p_2), kami berulang kali "memotong sudut" seperti yang ditunjukkan pada Gambar 15.15. Pada setiap langkah, kami membagi setiap segmen garis menjadi dua, menghubungkan titik tengah, dan kemudian memindahkan titik sudut ke titik tengah segmen garis baru. Perhatikan bahwa dalam proses ini, poin baru diperkenalkan, dipindahkan sekali, dan kemudian tetap di posisi ini untuk setiap iterasi yang tersisa. Titik akhir tidak pernah bergerak.

Jika kita menghitung posisi "baru" untuk p_2 sebagai titik tengah dari titik tengah, kita mendapatkan ekspresi:

$$p'_2 = \frac{1}{2}\left(\frac{1}{2}p_0 + \frac{1}{2}p_1\right) + \frac{1}{2}\left(\frac{1}{2}p_1 + \frac{1}{2}p_2\right)$$

Konstruksi sebenarnya bekerja untuk proporsi jarak lain di sepanjang setiap segmen. Jika kita biarkan u menjadi jarak antara awal dan akhir setiap segmen di mana kita menempatkan titik tengah, kita dapat menulis ulang ekspresi ini sebagai

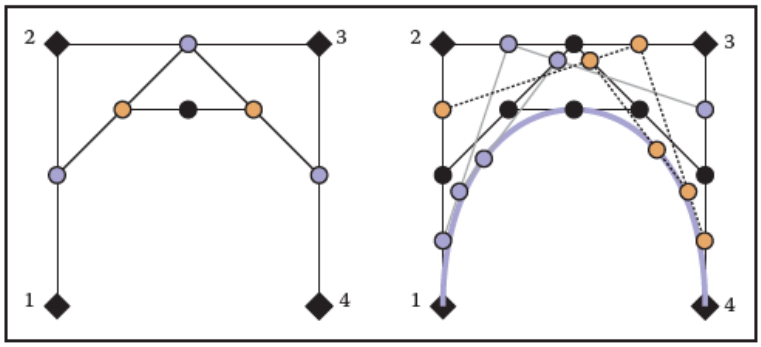
$$\mathbf{p}(u) = (1 - u)((1 - u)\mathbf{p}_0 + u\mathbf{p}_1) + u((1 - u)\mathbf{p}_1 + u\mathbf{p}_2).$$

Regroupingterms memberikan fungsi quadraticBe´zier:

$$\mathbf{B}_2(u) = (1 - u)^2\mathbf{p}_0 + 2u(1 - u)\mathbf{p}_1 + u^2\mathbf{p}_2.$$

Algoritma de Casteljau

Salah satu fitur bagus dari kurva Be´zier adalah bahwa ada metode yang sangat sederhana dan umum untuk menghitung dan membaginya. Metode, yang disebut algoritma *de Casteljau*, menggunakan urutan interpolasi linier untuk menghitung posisi sepanjang kurva Be´zier dari orde biner. Ini adalah generalisasi dari skema subdivisi yang dijelaskan pada bagian sebelumnya.



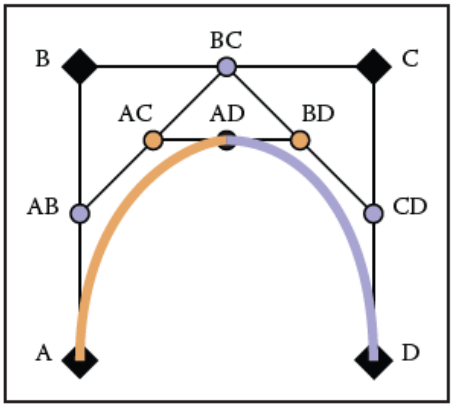
Gambar 15.16. Ilustrasi algoritme de Casteljau untuk Be´zier kubik. Gambar sebelah kiri menunjukkan konstruksi untuk u = 0,5 . Gambar sebelah kanan menunjukkan konstruksi untuk 0,25, 0,5, dan 0,75.

Algoritma de Casteljau dimulai dengan menghubungkan setiap himpunan titik yang berdekatan dengan garis, dan menemukan titik pada garis tersebut yang merupakan interpolasi u, menghasilkan himpunan n−1 titik. Titik-titik ini kemudian dihubungkan dengan garis lurus, garis-garis tersebut diinterpolasi (sekali lagi oleh u), memberikan himpunan n−2 titik. Proses

ini diulang sampai ada satu titik. Sebuah ilustrasi dari proses ini ditunjukkan pada Gambar 15.16.

Proses komputasi sebuah titik pada segmen Be´ziers juga menyediakan metode untuk membagi segmen pada titik tersebut. Titik antara yang dihitung selama algoritma de Casteljau membentuk titik kontrol baru dari segmen baru yang lebih kecil, seperti yang ditunjukkan pada Gambar 15.17.

Adanya algoritma yang baik untuk membagi kurva Be´zier memungkinkan algoritma pembagian dan penaklukan. Misalnya, saat menggambar segmen kurva Be´zier, mudah untuk memeriksa apakah kurvanya mendekati garis lurus karena dibatasi oleh lambungnya yang cembung. Jika titik kendali dari kurva semuanya mendekati kolinear, kurva tersebut dapat digambarkan sebagai garis lurus. Jika tidak, kurva dapat dibagi menjadi bagian-bagian yang lebih kecil, dan prosesnya dapat diulang. Algoritma serupa dapat digunakan untuk menentukan perpotongan antara dua kurva. Karena adanya algoritma tersebut, representasi kurva lainnya sering dikonversi ke bentuk Be´zier untuk diproses.



Gambar 15.17. Algoritma de Casteljau digunakan untuk membagi segmen Be´zier kubik.

Titik awal (intan hitam A, B, C, dan D) diinterpolasi secara linier untuk menghasilkan lingkaran biru (AB, BC, CD), yang diinterpolasi secara linier untuk menghasilkan lingkaran jingga (AC, BD), yang diinterpolasi secara linier untuk memberikan titik pada kubik AD. Proses ini juga telah membagi segmen Be´zier dengan titik kontrol A,B,C,D menjadi dua segmen Be´zier dengan titik kontrol A, AB, AC, AD dan AD, BD, CD, D.

15.13 B-SPLINES

B-splines menyediakan metode untuk mendekati aset dari n titik dengan kurva yang terdiri dari polinomial derajat d yang memberikan kontinuitas $C^{(d-1)}$. Tidak seperti spline Be ´zier pada bagian sebelumnya, spline B memungkinkan kurva dihasilkan untuk tingkat kontinuitas yang diinginkan (hampir sampai jumlah titik). Karena itu, B-spline adalah cara yang lebih disukai untuk menentukan kurva yang sangat halus (tingkat kontinuitas yang tinggi) dalam grafis komputer. Jika kita menginginkan kurva C^2 atau lebih tinggi melalui sejumlah titik yang berubah-ubah, B-spline mungkin merupakan metode yang tepat.

Kita dapat merepresentasikan kurva menggunakan kombinasi linear dari fungsi basis B-spline. Karena fungsi basis ini sendiri adalah splines, kami menyebutnya basis splines atau disingkat B-splines. Setiap B-spline atau fungsi basis terdiri dari himpunan d+1polinomial yang masing-masing berderajat d. Metode B-splines menyediakan prosedur umum untuk mendefinisikan fungsi-fungsi ini.

Istilah B-spline secara khusus mengacu pada salah satu fungsi dasar, bukan fungsi yang diciptakan oleh kombinasi linier dari sekumpulan B-spline. Namun, ada inkonsistensi dalam bagaimana istilah ini digunakan dalam grafis komputer. Umumnya, "kurva Bspline" digunakan untuk mengartikan kurva yang diwakili oleh kombinasi linier B-spline.

Gagasan untuk merepresentasikan polinomial sebagai kombinasi linier dari polinomial lain telah dibahas dalam Bagian 15.3.1 dan 15.3.5. Mewakili spline sebagai kombinasi linier dari spline lain ditunjukkan pada Bagian 15.4.1. Sebenarnya, contoh yang diberikan adalah kasus sederhana dari B-spline.

Notasi umum untuk merepresentasikan suatu fungsi sebagai kombinasi linier dari fungsi lain adalah

Rumus 15.15

$$f(t)=\sum_{i=1}^{n} p_i b_i(t)$$

di mana pi adalah koefisien dan bi adalah fungsi dasar. Jika koefisien adalah titik (misalnya, 2 atau 3 vektor), kami menyebutnya sebagai titik kontrol. Kunci untuk membuat metode seperti itu berhasil adalah dengan mendefinisikan bi dengan tepat. B-splinesmenyediakan cara yang sangat umum untuk melakukan ini.

Satu set B-spline dapat didefinisikan untuk sejumlah koefisien n dan nilai parameter k.[3] Nilai k adalah satu lebih dari derajat polinomial yang digunakan untuk membuat B-splines (k = d +1 .)

B-spline penting karena menyediakan metode yang sangat umum untuk membuat fungsi (yang akan berguna untuk merepresentasikan kurva) yang memiliki sejumlah properti yang berguna. Kurva dengan n titik dibuat dengan B-splines dengan nilai parameter k:

- adalah $C^{(k-2)}$ kontinu;
- dibuat dari polinomial juga derajat k−1;
- memiliki kontrol lokal—setiap lokasi pada kurva hanya bergantung pada k titik kontrol;
- dibatasi oleh convexhull dari poin;
- menunjukkan variasi properti yang semakin berkurang yang diilustrasikan pada Gambar 15.12.

Membuat kurva menggunakan B-splines tidak perlu menginterpolasi titik kontrolnya. Kami akan memperkenalkan B-splines dengan terlebih dahulu melihat kasus spesifik dan sederhana untuk memperkenalkan konsep. Kami kemudian akan menggeneralisasi metode dan menunjukkan mengapa mereka menarik. Karena metode untuk menghitung B-splines sangat umum, kami menunda memperkenalkannya sampai kami menunjukkan apa generalisasi ini.

Seragam Linear B-Splines

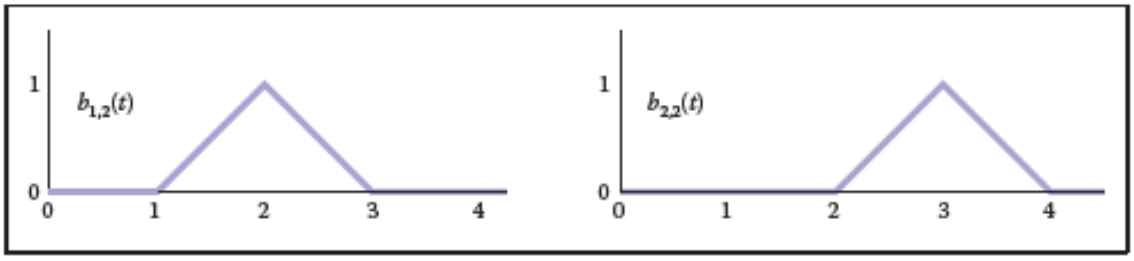
Pertimbangkan satu set fungsi dasar dari bentuk berikut:

$$b_{1,2}(t)=\begin{cases} t-1 \; jika \; i \le t < i+1 \\ 2-1+i \; jika \; i+1 \le t \le i+2 \\ 0 \; jika \; tidak \end{cases}$$

[3] Parameter B-spline sebenarnya adalah urutan polinomial yang digunakan dalam B-splines. Meskipun terminologi ini tidak seragam dalam literatur, penggunaan parameter B-spline k sebagai nilai satu lebih besar dari derajat polinomial banyak digunakan, meskipun beberapa teks (lihat catatan bab) menulis semua persamaan dalam bentuk polinomial. derajat.

Masing-masing fungsi ini terlihat seperti “topi” segitiga kecil antara i dan i+2 dengan puncaknya di i+1 . Masing-masing adalah polinomial piecewise, dengan knotsat i, i+1, andi+2 . Dua di antaranya digambarkan dalam Gambar 15.18.

Masing-masing fungsi ini $b_{i,2}$ adalah derajat pertama (linier) B-spline. Karena kita akan mempertimbangkan B-spline dari nilai parameter lain nanti, kita menyatakan ini dengan 2 dalam subskrip.



Gambar 15.18. B-splines dengan d = 1 atau k = 2.

Perhatikan bahwa kita telah memilih untuk meletakkan tepi bawah dari B-spline (simpul pertamanya) di i. Oleh karena itu, simpul pertama dari B-spline pertama (i =1) adalah di 1. Iterasi pada B-splines atau elemen dari vektor koefisien adalah dari 1 ton (lihat Persamaan 15.15). Ketika B-spline diimplementasikan, serta dalam banyak diskusi lainnya, mereka sering diberi nomor dari 0 hingga n – 1.

Kita dapat membuat fungsi dari himpunan n titik kontrol menggunakan Persamaan 15.15, dengan fungsi ini digunakan untuk bi untuk membuat "fungsi keseluruhan" yang dipengaruhi oleh koefisien. Jika kita menggunakan (k = 2) B-splines ini untuk mendefinisikan fungsi keseluruhan, kita akan mendefinisikan fungsi polinomial sepotong-sepotong yang secara linier menginterpolasi koefisien pi antara t = k dan t = n +1 . Perhatikan bahwa sementara (k = 2) B-splines menginterpolasi semua koefisiennya, B-splines dari derajat yang lebih tinggi melakukan ini di bawah kondisi tertentu yang akan kita bahas di Bagian 15.6.3.

Beberapa sifat B-splines dapat dilihat dalam kasus sederhana ini. Kami akan menulis ini dalam bentuk umum menggunakan k, parameter, dan n untuk jumlah koefisien atau titik kontrol:

- Setiap B-spline memiliki k +1 knot.
- Setiap B-spline adalah nol sebelum simpul pertama dan setelah simpul terakhir.
- Spline keseluruhan memiliki kontrol lokal karena setiap koefisien hanya dikalikan dengan satu B-spline, dan B-spline ini bukan nol hanya antara k +1 knot.
- Spline keseluruhan memiliki n + k knot.
- Setiap spline B adalah kontinu $C^{(k-2)}$, oleh karena itu spline keseluruhan adalah C(k−2) kontinu.
- Jumlah semua nilai parameter B spline antara knot k dan n+1. Rentang ini adalah tempat di mana ada B-splines yang tidak nol. Penjumlahan ke 1 penting karena ini berarti bahwa B-spline adalah shift invariant: menerjemahkan titik kontrol akan menerjemahkan seluruh kurva.
- Di antara setiap simpulnya, B-spline adalah polinomial tunggal dengan derajat d = k−1. Oleh karena itu, kurva keseluruhan (yang menjumlahkan ini bersama-sama) juga dapat dinyatakan sebagai polinomial tunggal derajat d antara simpul yang berdekatan.

Dalam contoh ini, kami telah memilih simpul yang berjarak seragam. Kami akan mempertimbangkan B-splines dengan jarak tidak seragam nanti. Ketika jarak simpul seragam, masing-masing B-splinasi identik kecuali digeser. B-spline dengan jarak simpul yang seragam kadang-kadang disebut *B-spline seragam atau B-spline periodik*.

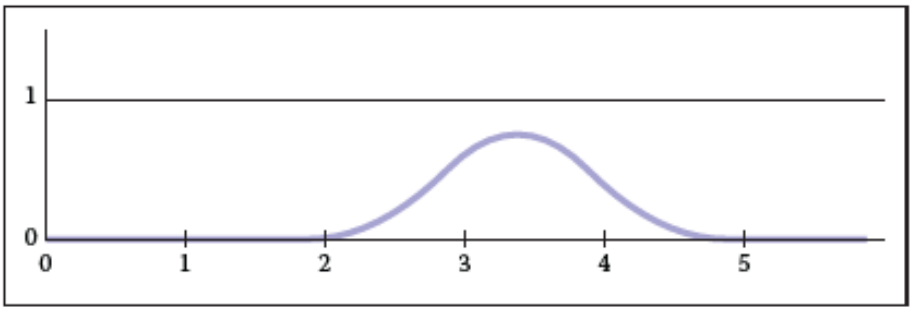
Seragam B-Spline Kuadrat

Sifat-sifat B-spline yang tercantum pada bagian sebelumnya sengaja ditulis untuk n dan k sewenang-wenang. Sebuah prosedur umum untuk membangun B-spline akan diberikan kemudian, tetapi pertama-tama, mari kita pertimbangkan kasus khusus lainnya dengan k =3 .

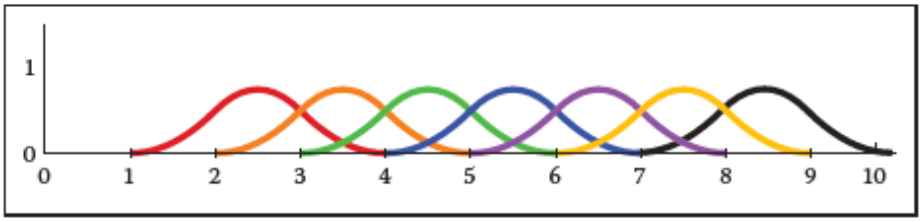
B-spline b2,3 ditunjukkan pada Gambar 15.19. Itu terbuat dari potongan kuadrat (derajat 2), dan memiliki tiga di antaranya. Ini adalah kontinu C1 dan bukan nol hanya dalam empat simpul yang terbentang. Perhatikan bahwa spline B kuadrat dibuat dari tiga bagian, satu di antara simpul 1 dan 2, satu di antara simpul 2 dan 3, dan satu di antara simpul 3 dan 4. Dalam Bagian 15.6.3 kita akan melihat prosedur umum untuk membangun fungsi-fungsi ini. Untuk saat ini, kita cukup memeriksa fungsi-fungsi ini:

$$b_{i,3}(t) = ¿$$

Untuk membuat ekspresi lebih sederhana, kami menulis fungsi untuk setiap bagian seolah-olah diterapkan pada rentang 0 hingga 1.



Gambar 15.19. B-spline $b_{2,3}$ dengan jarak simpul seragam.



Gambar 15.20. Himpunan tujuh B-spline dengan k = 3 dan jarak simpul seragam [1, 2, 3, 4, 5, 6, 7, 8, 10].

Jika kita mengevaluasi fungsi keseluruhan yang dibuat dari penjumlahan bersama B-spline, setiap saat hanya k (3 dalam hal ini) darinya bukan nol. Salah satunya akan menjadi bagian pertama dari Persamaan 15.17, satu akan berada di bagian kedua, dan satu lagi akan berada di bagian ketiga. Oleh karena itu, kita dapat menganggap setiap bagian dari fungsi keseluruhan terdiri dari polinomial derajat d = k 1 yang bergantung pada koefisien k. Untuk kasus k =3, kita dapat menulis

$$f(u) = \frac{1}{2}(1 - u ¿¿¿ 2p_1) + \left(-u^2 + u + \frac{1}{2}p_1\right) + \frac{1}{2}u^2 p_{i+2}$$

Dimana u = t − i. Ini mendefinisikan bagian dari fungsi keseluruhan ketika i $\leq$ t < i + 1.

Jika kita memiliki himpunan n titik, kita dapat menggunakan B-splines untuk membuat kurva. Jika kita memiliki tujuh poin, kita akan membutuhkan satu set tujuh B-spline. Aset tujuhB-splines untuk k =3 ditunjukkan pada Gambar 15.20. Perhatikan bahwa

ada n + k (10) knot, bahwa jumlah dari B-splines adalah 1 pada rentang k sampai n +1(knot 3 sampai 8). Sebuah kurva yang ditentukan dengan menggunakan himpunan titik-titik B-splines dan ditunjukkan pada Gambar 15.21.

Seragam Kubik B-Splines

Karena polinomial kubik sangat populer dalam grafis komputer, kasus khusus B-spline dengan k = 4 cukup penting untuk dipertimbangkan sebelum membahas kasus umum. Sebuah B-spline derajat ketiga didefinisikan oleh empat potongan polinomial kubik. Proses umum di mana potongan-potongan ini ditentukan dijelaskan kemudian, tetapi hasilnya adalah:

$$\begin{cases} \frac{1}{6}u^3 \; jika \; i \le t < i+1 \; u = t-1 \\ \frac{1}{6}\left(-3u^3+3u^2+3u+1\right) jika \; i+1 \le t < i+2 u = t-(1+1) \\ \frac{1}{6}\left(3u^3-6u^2+4\right) jika + 2 \le t < i+3 u = t-(i+2) \\ \frac{1}{6}\left(-u^3-3u^2-3u+1\right) jika + 2 \le t < i+3 u = t-(i+2) \\ 0 \; jikatidak \end{cases}$$

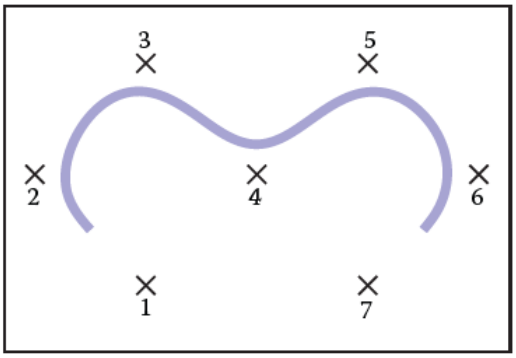
Derajat 3 B-spline ini digambarkan untuk i = 1 pada Gambar 15.22. Kita dapat menulis fungsi untuk kurva keseluruhan antara simpul i +3 dan i +4 sebagai fungsi dari parameter u antara 0 dan 1 dan empat titik kontrol yang mempengaruhinya:

$$\begin{aligned} \mathbf{f}(u) = &\frac{1}{6}(-u^3+3u^2-3u+1)\mathbf{p}_i + \frac{1}{6}(3u^3-6u^2+4)\mathbf{p}_{i+1} \\ &+ \frac{1}{6}(-3u^3+3u^2+3u+1)\mathbf{p}_{i+2} + \frac{1}{6}u^3\mathbf{p}_{i+3}. \end{aligned}$$

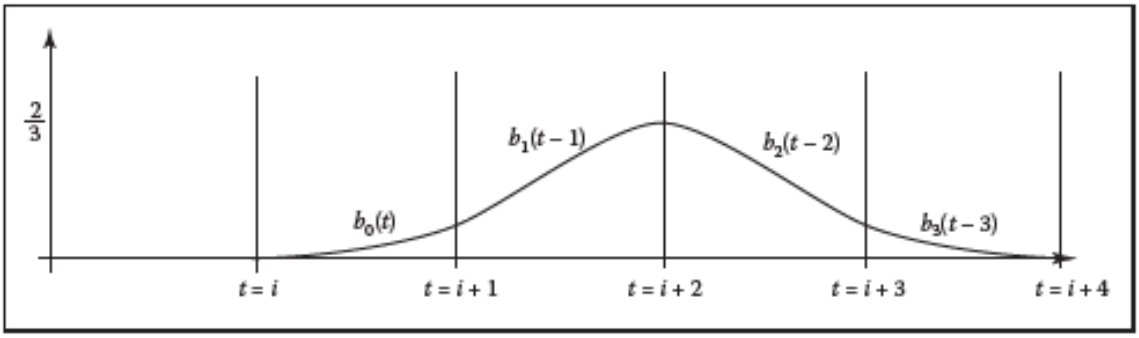
Ini dapat ditulis ulang dengan menggunakan notasi matriks dari bagian sebelumnya, memberikan matriks dasar untuk B-spline kubik dari

$$\mathbf{M}_\mathrm{b} = \frac{1}{6}\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}$$

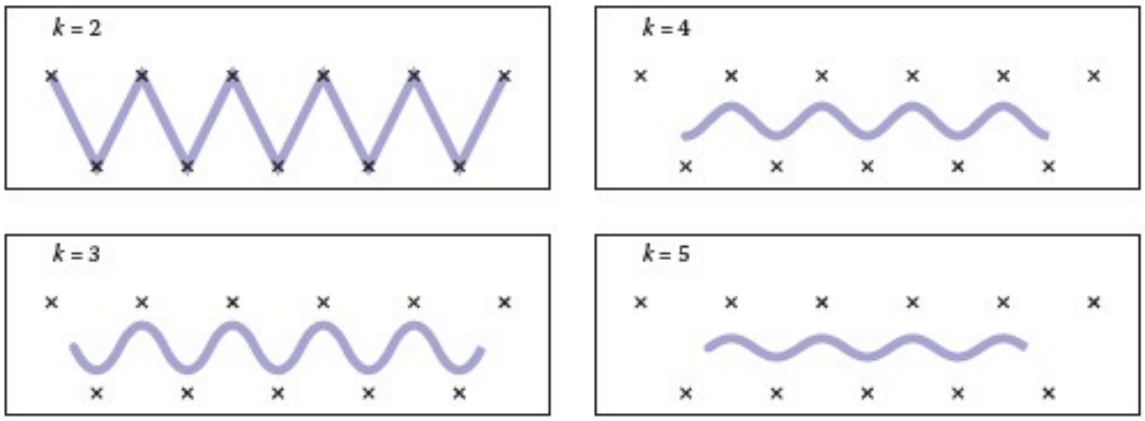
Berbeda dengan matriks yang diturunkan dari kendala di Bagian 15.5, matriks ini dibuat dari polinomial yang ditentukan oleh prosedur B-spline umum yang didefinisikan di bagian berikutnya.



Gambar 15.21. Kurva dibuat dari tujuh B-spline kuadrat (k=3), menggunakan tujuh titik kendali.



Gambar 15.22. Kubik (k = 4) B-spline dengan simpul seragam.



Gambar 15.23. Kurva B-spline menggunakan kumpulan simpul yang sama dan titik kontrol yang sama, untuk berbagai nilai k. Perhatikan bahwa saat k meningkat, rentang parameter yang valid untuk kurva menyusut.

B-Spline tidak seragam

Salah satu fitur bagus dari B-splines adalah mereka dapat didefinisikan untuk k > 1 apa pun. Jadi jika kita membutuhkan kurva yang lebih halus, kita cukup meningkatkan nilai k. Hal ini diilustrasikan pada Gambar 15.23.

Sejauh ini, kita telah mengatakan bahwa B-spline digeneralisasikan ke sembarang k > 1 dan sembarang n $\leq$ d. Ada generalisasi elastis untuk memperkenalkan sebelum kami menunjukkan bagaimana sebenarnya menghitung B-splines ini. B-splines didefinisikan untuk vektor simpul non-penurunan apa pun.

Untuk n dan k tertentu, himpunan B-spline (dan fungsi yang dibuat oleh kombinasi liniernya) memiliki n + k knot. Kita dapat menulis nilai simpul-simpul ini sebagai vektor, yang akan kita nyatakan sebagai t. Untuk B-spline seragam, vektor simpulnya adalah [1,2,3,...,n+ k]. Namun, B-spline dapat dibangkitkan untuk sembarang vektor simpul dengan panjang n + k, asalkan nilainya tidak menurun (misalnya, $t_{i+1} \geq t_i$).

Ada dua alasan utama mengapa jarak simpul yang tidak seragam berguna: memberikan kontrol atas rentang parameter apa dari keseluruhan fungsi yang dipengaruhi oleh setiap koefisien, dan ini memungkinkan kita untuk mengulang simpul (misalnya, membuat simpul tanpa spasi di antaranya) untuk membuat fungsi dengan properti berbeda di sekitar titik ini. Yang terakhir akan dipertimbangkan nanti di bagian ini.

Kemampuan untuk menentukan nilai simpul untuk B-spline sama dengan kemampuan untuk menentukan lokasi interpolasi untuk interpolasi kurva spline. Ini memungkinkan kita untuk mengaitkan fitur kurva dengan nilai parameter. Dengan menentukan vektor simpul yang tidak seragam, kami menentukan kisaran parameter yang dipengaruhi oleh masing-masing koefisien kurva B-spline. Ingat bahwa B-spline i tidak nol antara simpul dan simpul+k. Oleh karena itu, koefisien yang terkait dengannya hanya mempengaruhi kurva antara nilai parameter ini.

Satu tempat di mana kontrol atas nilai simpul sangat berguna adalah dalam memasukkan atau menghapus simpul di dekat awal urutan. Untuk mengilustrasikan hal ini, pertimbangkan kurva yang didefinisikan menggunakan B-splines linier (k = 2) seperti yang dibahas dalam Bagian 15.6.2

Untuk n =4 , vektor simpul seragam adalah [1, 2, 3, 4, 5, 6]. Kurva ini dikendalikan oleh satu set empat titik dan mencakup rentang parameter t =2sampai t =5 . "Akhir" kurva (t = 5) menginterpolasi titik kontrol terakhir. Jika kita memasukkan titik baru di tengah set titik, kita akan membutuhkan vektor simpul yang lebih panjang. Sifat lokalitas B-splines mencegah penyisipan ini mempengaruhi nilai kurva di ujungnya. Kurva yang lebih panjang masih akan menginterpolasi titik kontrol terakhirnya di ujungnya. Namun, jika kita memilih untuk menjaga jarak simpul seragam, vektor simpul baru akan menjadi [1, 2, 3, 4, 5, 6, 7]. Ujung kurva akan berdetak t =6 , dan nilai parameter di mana titik kontrol terakhir diinterpolasi akan menjadi nilai parameter yang berbeda dari sebelum penyisipan. Dengan jarak simpul tidak seragam, kita dapat menggunakan vektor simpul [1, 2, 3, 3, 5, 4, 5, 6] sehingga ujung-ujung kurva tidak terpengaruh oleh perubahan. Kemampuan untuk memiliki jarak simpul yang tidak seragam menjadikan sifat lokalitas B-splines sebagai sifat aljabar, dan juga sifat geometris.

Kami sekarang memperkenalkan metode umum untuk mendefinisikan B-splines. Diberikan nilai untuk jumlah koefisien n, parameter B-spline k, dan vektor simpul t (yang memiliki panjang n + k), persamaan rekursif berikut mendefinisikan B-splines:

Persamaan 15.19 dan 16.20

$$b_{i,1,\mathbf{t}}(t) = \begin{cases} 1 & \text{if } \mathbf{t}_i \le t < \mathbf{t}_{i+1}, \\ 0 & \text{otherwise.} \end{cases}$$

$$b_{i,k,\mathbf{t}}(t) = \frac{t-\mathbf{t}_i}{\mathbf{t}_{i+k-1}-\mathbf{t}_i} b_{i,k-1}(t) + \frac{\mathbf{t}_{i+k}-t}{\mathbf{t}_{i+k}-\mathbf{t}_{i+1}} b_{i+1,k-1}(t).$$

Persamaan ini diketahui adalah pengulangan Cox-deBoor. Ini mungkin digunakan untuk menghitung nilai spesifik untuk B-splines spesifik. Namun, ini lebih sering diterapkan secara aljabar untuk menurunkan persamaan seperti Persamaan 15.17 atau 15.18.

Sebagai contoh, pertimbangkan bagaimana kita akan menurunkan Persamaan 15.17. Menggunakan vektor simpul seragam [1,2,3,...], t_i = i, dan nilai k =3 pada Persamaan 15.20 menghasilkan

$$\begin{aligned} b_{i,3}(t) &= \frac{t-i}{(i+2)-i} b_{i,2} + \frac{(i+3)-t}{(i+3)-(i+1)} b_{i+1,2} \\ &= \frac{1}{2}(t-i) b_{i,2} + \frac{1}{2}(i+3-t) b_{i+1,2}. \end{aligned}$$

Untuk melihat bahwa ekspresi ini ekuivalen dengan Persamaan 15.17, kita perhatikan bahwa masing-masing (k = 1) B-splinesis seperti sakelar, hanya menyala untuk rentang parameter tertentu. Misalnya, bi,1 hanya tidak nol antara i dan i+1 . Jadi, jika i≤ t<i+1 , hanya B-spline pertama (k =1) dalam ekspresi yang tidak nol, jadi

$$B_{i,3}(t) = 1/2\ (t-1)^2\ jika\ i \leq t < i+1$$

Manipulasi serupa memberikan bagian lain dari Persamaan 15.17.

Knot Berulang dan Interpolasi B-Spline

Sementara B-splines memiliki banyak properti bagus, fungsi yang didefinisikan menggunakan mereka umumnya tidak menginterpolasi koefisien. Ini bisa merepotkan jika kita menggunakannya untuk mendefinisikan kurva yang ingin kita interpolasi pada titik tertentu. Kami memberikan gambaran singkat tentang cara menginterpolasi titik spesifik teh menggunakan B-spline di sini. Pembahasan lebih lengkap dapat dilihat pada buku-buku yang tercantum pada catatan bab.

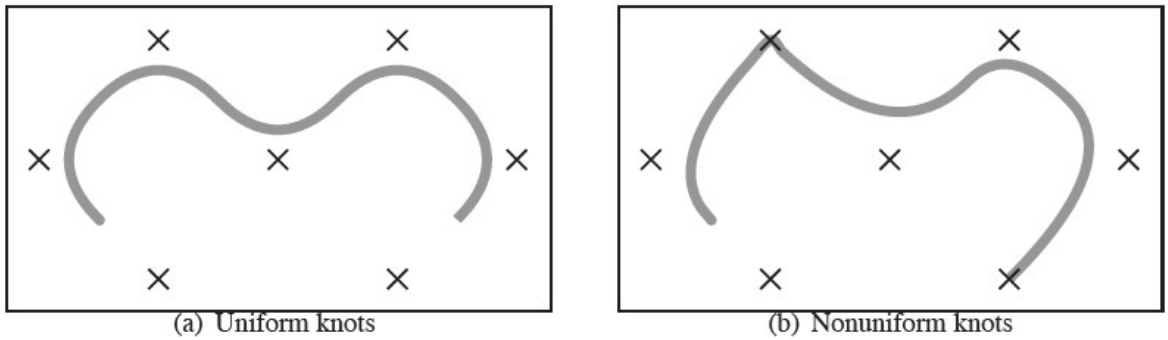
Salah satu cara untuk menyebabkan B-splines menginterpolasi koefisiennya adalah dengan repeatknots. Jika semua knot interior untuk B-spline tertentu memiliki nilai yang sama, maka fungsi keseluruhan akan menginterpolasi koefisien B-spline ini. Contohnya ditunjukkan pada Gambar 15.24.

Interpolasi dengan simpul berulang membutuhkan biaya tinggi: ini menghilangkan kelancaran B-spline dan fungsi keseluruhan yang dihasilkan serta kurva yang diwakili. Namun, pada awal dan akhir spline, di mana kontinuitas tidak menjadi masalah, pengulangan simpul berguna untuk membuat interpolasi titik akhir B-spline. Sementara nilai simpul pertama (atau terakhir) tidak penting untuk interpolasi, untuk kesederhanaan, kami membuat simpul k pertama (atau terakhir) memiliki nilai yang sama untuk mencapai interpolasi.

Interpolasi titik akhir B-spline kuadrat ditunjukkan pada Gambar 15.25. Dua yang pertama dan dua B-spline terakhir berbeda dari yang seragam. Ekspresi mereka dapat diturunkan melalui penggunaan pengulangan Cox-de Boor:

$$b_{1,3,[0,0,0,1,2]}(t) = \begin{cases} (1-t)^2\ jika\ 0 \leq t < 1 \\ 0\ jika\ lainnya \end{cases}$$

$$b_{2,3,[0,0,0,1,2]}(t) = \begin{cases} 2u - \frac{3}{1}u^2\ jika\ 0 \leq t < 1 u = t \\ \frac{1}{2}(1-u)\ jika\ 1 \leq t < 2 u = t-1 \\ 0\ jika\ tidak \end{cases}$$

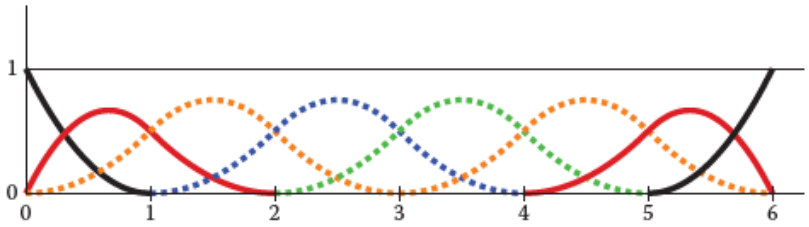


Gambar 15.24. Kurva yang diparameterisasi oleh B-splines kuadratik (k = 3) dengan tujuh titik kontrol.

Di sebelah kiri digunakan vektor simpul seragam [1,2,3,4,5,6,7,8,9,10]. Di sebelah kanan, digunakan jarak simpul tidak seragam [1,2,3,4,4,6,7,8,8,10]. Duplikasi simpul ke-4 dan ke-8 berarti bahwa semua simpul bagian dalam dari B-spline ke-3 dan ke-7 adalah sama, sehingga kurva menginterpolasi titik kontrol yang terkait dengan titik-titik tersebut.

15.14 NURBS

Terlepas dari semua generalitas yang disediakan oleh B-spline, ada beberapa fungsi yang tidak dapat direpresentasikan secara tepat dengan menggunakannya. Secara khusus, B-spline tidak dapat mewakili bagian kerucut. Untuk mewakili kurva tersebut, rasio dua polinomial digunakan. B-spline tidak seragam digunakan untuk mewakili pembilang dan penyebut. Bentuk paling umum dari ini adalah B-splines rasional yang tidak seragam, atau disingkat NURBS.



Gambar 15.25. Interpolasi titik akhir kuadratik (k = 3) B-splines, untuk n = 8. Vektor simpulnya adalah [0,0,0,1,2,3,4,5,6,6,6].

Dua B-spline pertama dan terakhir adalah aperiodik, sedangkan empat tengah (ditunjukkan sebagai garis putus-putus) periodik dan identik dengan yang ada di Gambar 15.20.

NURBS mengaitkan bobot skalar hi dengan setiap titik kontrol pi dan menggunakan B-spline yang sama untuk keduanya:

$$f(u) = \frac{\sum_{i=1}^{n} ¿1 h_i p_i b_i kt}{\sum_{i=1}^{n} h_i b_i k, t}$$

dimana bi,k,t adalah B-splines dengan parameter k dan simpul vektor t. NURBS sangat banyak digunakan untuk mewakili kurva dan permukaan dalam pemodelan geometris karena fleksibilitas yang luar biasa yang mereka berikan, selain sifat berguna dari B-splines.

15.15 RINGKASAN

Dalam bab ini, kita telah membahas sejumlah representasi untuk kurva bentuk bebas. Yang paling penting untuk grafis komputer adalah:

- Cardinalssplinesusesetofcubicpiecestointerpolatecontrolpoints. Mereka umumnya lebih disukai daripada interpolasi polinomial karena mereka lokal dan lebih mudah untuk dievaluasi.
- Kurva Be´zier mendekati titik kontrolnya dan memiliki banyak properti yang berguna dan algoritme terkait. Untuk alasan ini, mereka populer dalam aplikasi grafis.
- Kurva B-spline mewakili kurva sebagai kombinasi linier dari fungsi B-spline. Mereka bersifat umum dan memiliki banyak sifat yang berguna seperti dibatasi oleh lambung cembungnya dan variasi yang semakin berkurang. B-splines sering digunakan ketika kurva halus diinginkan.

15.16 CATATAN

Masalah merepresentasikan bentuk secara matematis adalah bidang tersendiri, umumnya dikenal sebagai pemodelan geometris. Mewakili kurva hanyalah permulaan dan umumnya merupakan pendahulu untuk pemodelan permukaan dan padatan. Diskusi kurva yang lebih menyeluruh dapat ditemukan di sebagian besar teks pemodelan geometris, lihat misalnya Pemodelan Geometris (Mortenson, 1985) untuk teks yang dapat diakses oleh siswa grafisa komputer. Banyak buku pemodelan geometris secara khusus berfokus pada kurva dan permukaan yang halus. Teks seperti Pengantar Splines untuk Digunakan di KomputerGraphics(Bartels,Beatty,&Barsky,1987),CurvesandSurfacesfor CAGD: A Practical Guide (Farin, 2002) dan Geometric Modeling with Splines: AnIntroduction(E.Cohen,Riesenfeld,&Elber,2001)memberikan detail yang cukup besar tentang kurva dan representasi permukaan. Buku-buku lain fokus pada matematika splines; Panduan Praktis untuk Splines (De Boor, 2001) adalah referensi standar.

Sejarah perkembangan representasi kurva dan permukaan sangat kompleks, lihat bab oleh Farin dalam Handbook of Computer Aided Geometric Design (Farin, Hoschek, & Kim, 2002) atau buku tentang An Introduction to NURBS: With Historical Perspective (DF Rogers, 2000) untuk diskusi. Banyak ide yang dikembangkan secara independen oleh beberapa kelompok yang mendekati masalah dari berbagai disiplin ilmu. Karena itu, mungkin sulit untuk mengaitkan ide dengan satu orang atau menunjuk pada sumber "asli". Hal ini juga menyebabkan keragaman notasi, terminologi, dan cara memperkenalkan konsep dalam literatur.

15.16 LATIHAN

Untuk Latihan 1-4, temukan matriks kendala, matriks basis, dan fungsi basis. Untuk membalikkan bilangan, Anda dapat menggunakan program seperti MATLAB atau OCTAVE (sistem seperti MATLAB gratis).

1. Segmen garis: diparameterisasi dengan p_0 terletak 25% dari jalan sepanjang segmen (u = 0,25), dan p1 terletak 75% dari jalan sepanjang segmen.
2. Kuadrat: diparameterisasi dengan p_0 sebagai posisi titik awal (u = 0), p_1, turunan pertama di titik awal, dan p_2, turunan kedua di titik awal.
3. Sebuah kubik: titik-titik kendalinya memiliki jarak yang sama (p_0 memiliki u = 0 , p_1 memiliki u = 1/3, p_2 memiliki u =2 /3, dan p_3 memiliki u = 1).
4. Aquintic: (derajat lima polinomial, sehingga bilangan 6×6) dengan p_0 adalah posisi awal, p_1 adalah turunan awal, p_2 adalah posisi tengah (u = 0,5), p_3 adalah turunan pertama di tengah, p_4 adalah posisi di akhir, dan p_5 adalah yang pertama turunan di akhir.
5. Bentuk Lagrange (Persamaan (15.12)) dapat digunakan untuk mewakili kubik interpolasi Latihan 3. Gunakan pada beberapa nilai parameter yang berbeda untuk memastikan bahwa ia menghasilkan hasil yang sama dengan fungsi dasar yang diturunkan pada Latihan 3.
6. Rancang parameter panjang busur untuk kurva yang ditunjukkan oleh fungsi parametrik

$$f(u) = (u, u^2).$$

7. Diberikan empat titik kendali dari segmen garis Hermite, hitunglah titik kendali dari segmen Be´zier yang setara.
8. Gunakan algoritma de Casteljau untuk mengevaluasi posisi kurva kubik Be´zier dengan titik kontrolnya pada (0, 0), (0, 1), (1, 1) dan (1, 0) untuk nilai parameter u = 0.5 dan u = 0.75. Menggambar sketsa akan membantu Anda melakukan ini.
9. Gunakan perulangan Cox-de Boor untuk menurunkan Persamaan (15.16).

BAB 16
ANIMASI KOMPUTER

Animasi berasal dari bahasa Latin anima dan berarti tindakan, proses, atau hasil dari memberikan kehidupan, minat, semangat, gerak, atau aktivitas. Motionisa mendefinisikan properti kehidupan dan sebagian besar seni animasi yang sebenarnya adalah tentang bagaimana menceritakan sebuah kisah, menunjukkan emosi, atau bahkan mengungkapkan detail halus karakter manusia melalui gerakan. Komputer adalah alat sekunder untuk mencapai tujuan ini—ini adalah alat yang dapat digunakan oleh animator yang terampil untuk membantu mendapatkan hasil yang diinginkannya dengan lebih cepat dan tanpa berkonsentrasi pada hal-hal teknis yang tidak diminatinya. Animasi tanpa komputer, yang sekarang sering disebut animasi "tradisional", memiliki sejarah panjang dan kaya yang terus menerus ditulis oleh ratusan orang yang masih aktif dalam seni ini. Seperti di bidang mapan mana pun, beberapa aturan yang telah teruji oleh waktu telah dikristalisasi yang memberikan panduan umum tingkat tinggi tentang bagaimana hal-hal tertentu harus dilakukan dan apa yang harus dihindari. Prinsip-prinsip animasi tradisional ini berlaku sama untuk animasi komputer, dan kita akan membahas beberapa di antaranya dalam bab ini.

Komputer, bagaimanapun, lebih dari sekedar alat. Selain membuat tugas utama animator tidak terlalu membosankan, komputer juga menambahkan beberapa kemampuan yang benar-benar unik yang sama sekali tidak tersedia atau sangat sulit diperoleh sebelumnya. Alat pemodelan modern memungkinkan pembuatan model tiga dimensi terperinci yang relatif mudah, algoritme rendering dapat menghasilkan rentang tampilan yang mengesankan, dari sepenuhnya fotorealistik hingga sangat bergaya, algoritme simulasi numerik yang kuat dapat membantu menghasilkan gerakan berbasis fisika yang diinginkan untuk objek yang sangat sulit dianimasikan, dan sistem penangkapan gerak memberikan kemampuan untuk merekam dan menggunakan gerakan kehidupan nyata. Perkembangan ini menyebabkan ledakan penggunaan teknik animasi komputer dalam film dan iklan, desain dan arsitektur otomotif, kedokteran dan penelitian ilmiah, di antara banyak bidang lainnya. Domain dan aplikasi yang benar-benar baru juga telah muncul termasuk film fitur animasi komputer, sistem virtual/augmented reality, dan, tentu saja, permainan komputer.

Bab lain dari buku ini mencakup banyak perkembangan yang disebutkan di atas (misalnya, pemodelan dan rendering geometris) secara lebih langsung. Disini kami akan memberikan gambaran saja tentang teknik dan algoritma yang langsung digunakan untuk membuat dan memanipulasi gerak. Secara khusus, kita akan membedakan dan menjelaskan secara singkat empat pendekatan utama animasi komputer:

- **Keyframing** memberikan kontrol paling langsung kepada animator yang menyediakan data yang diperlukan pada beberapa saat dan komputer mengisi sisanya.
- **Animasi prosedural** melibatkan fungsi dan prosedur matematika yang dirancang khusus, seringkali empiris, yang outputnya menyerupai gerakan tertentu.
- **Teknik berbasis fisika** memecahkan persamaan diferensial gerak.
- **Motion capture** menggunakan peralatan atau teknik khusus untuk merekam gerakan dunia nyata dan kemudian mentransfer gerakan ini ke dalam model komputer.

Kami tidak menyentuh sisi hati dari data lapangan di sini. Secara umum, di sini kita tidak mungkin melakukan lebih dari sekadar menggores permukaan subjek yang menarik dalam menciptakan gerakan dengan komputer. Kami berharap para pembaca yang benar-benar tertarik dengan topik ini akan melanjutkan perjalanan mereka jauh melampaui materi bab ini.

16.1 PRINSIP ANIMASI

Dalam makalah SIGGRAPH 1987 seminalnya (Lasseter, 1987), John Lasseter membawa prinsip-prinsip kunci yang dikembangkan pada awal tahun 1930-an oleh animator tradisional studio Walt Disney ke perhatian komunitas animasi komputer yang saat itu sedang berkembang. Dua belas prinsip disebutkan: squash and stretch, timing, antisipasi, tindak lanjut dan aksi tumpang tindih, slow-in dan slow-out, staging, arc, aksi sekunder, aksi lurus ke depan dan pose-to-pose, berlebihan, keterampilan menggambar yang solid, dan banding. Hampir dua dekade kemudian, aturan yang telah teruji waktu ini, yang dapat membuat perbedaan antara animasi yang alami dan menghibur dan yang tampak mekanistik dan membosankan, sama pentingnya seperti sebelumnya. Untuk animasi komputer, selain itu, sangat penting untuk menyeimbangkan kontrol dan fleksibilitas yang diberikan kepada animator dengan memanfaatkan sepenuhnya kemampuan komputer. Meskipun prinsip-prinsip ini diketahui secara luas, banyak faktor yang mempengaruhi seberapa banyak perhatian diberikan pada aturan-aturan ini dalam praktiknya. Sementara seorang animator karakter yang mengerjakan sebuah film fitur mungkin menghabiskan berjam-jam mencoba mengikuti beberapa saran ini (misalnya, menyesuaikan waktunya agar tepat), banyak perancang game cenderung percaya bahwa waktu mereka lebih baik dihabiskan di tempat lain.

16.2 *TIMING*

Pengaturan waktu, atau kecepatan tindakan, adalah inti dari setiap animasi. Seberapa cepat sesuatu terjadi memengaruhi makna tindakan, keadaan emosi, dan bahkan bobot yang dirasakan dari objek yang terlibat. Bergantung pada kecepatannya, tindakan yang sama, yaitu memutar kepala karakter dari kiri ke kanan, dapat berarti apa saja, mulai dari reaksi hingga dipukul oleh benda berat hingga mencari buku di rak buku secara perlahan atau meregangkan otot leher. Sangat penting untuk mengatur waktu yang tepat untuk tindakan spesifik yang ada. Tindakan harus memakan waktu yang cukup untuk diperhatikan sambil menghindari gerakan yang terlalu lambat dan berpotensi membosankan. Untuk proyek animasi komputer yang melibatkan suara yang direkam, jangkar waktu alami yang disediakan suara harus diikuti. Faktanya, di sebagian besar produksi, suara aktor direkam terlebih dahulu dan animasi lengkapnya kemudian disinkronkan ke rekaman ini. Karena benda besar dan berat cenderung bergerak lebih rendah daripada benda kecil dan ringan (dengan percepatan yang lebih kecil, lebih tepatnya), pengaturan waktu dapat digunakan untuk memberikan informasi yang signifikan tentang berat suatu benda.

16.3 TATA LETAK TINDAKAN

Pada setiap saat selama animasi, harus jelas bagi pemirsa ide (tindakan, suasana hati, ekspresi) apa yang sedang disajikan. Pementasan yang baik, atau perencanaan tindakan tingkat tinggi, harus mengarahkan pandangan pemirsa ke tempat tindakan penting saat ini terkonsentrasi, secara efektif mengatakan kepadanya "lihat ini, dan sekarang, lihat ini" tanpa menggunakan kata-kata. Beberapa keakraban dengan persepsi manusia dapat membantu kita dengan tugas yang sulit ini. Karena sistem visual manusia sebagian besar bereaksi terhadap perubahan relatif daripada nilai absolut dari rangsangan, gerakan tiba-tiba di lingkungan yang tenang atau kurangnya gerakan di beberapa bagian dari scene yang sibuk secara alami menarik perhatian. Tindakan yang sama ditampilkan sehingga siluet objek berubah sering kali lebih terlihat dibandingkan dengan pengaturan depan.

Pada tingkat yang sedikit lebih rendah, setiap tindakan dapat dibagi menjadi tiga bagian: antisipasi (persiapan tindakan), tindakan itu sendiri, dan tindak lanjut (penghentian tindakan). Dalam banyak kasus, tindakan itu sendiri adalah bagian terpendek dan, dalam beberapa hal, paling tidak menarik. Misalnya, menendang bola mungkin melibatkan persiapan ekstensif dari pihak penendang dan "pelacakan visual" yang panjang dari bola yang berangkat dengan banyak kesempatan untuk menunjukkan tekanan saat itu, keadaan emosi penendang,

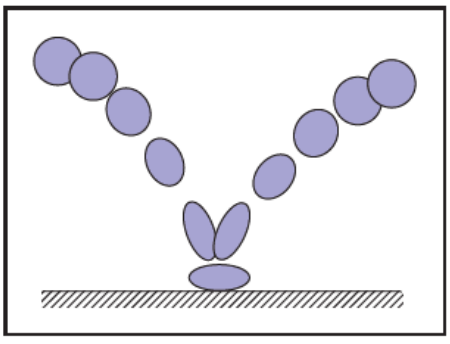
dan bahkan reaksi terhadap hasil yang diharapkan dari tindakan tersebut. Tindakan itu sendiri (gerakan kaki untuk menendang bola) cukup sederhana dan hanya membutuhkan sepersekian detik dalam kasus ini.

Tujuan dari antisipasi adalah untuk mempersiapkan penonton untuk apa yang akan terjadi. Ini menjadi sangat penting jika tindakan itu sendiri sangat cepat, sangat penting, atau sangat sulit. Menciptakan antisipasi yang lebih luas untuk tindakan tersebut berfungsi untuk menggarisbawahi sifat-sifat ini dan, jika terjadi peristiwa cepat, memastikan tindakan tersebut tidak akan terlewatkan.

Dalam kehidupan nyata, tindakan utama sering menyebabkan satu atau lebih tindakan tumpang tindih lainnya. Pelengkap yang berbeda atau bagian lepas dari objek biasanya menyeret di belakang bagian utama utama dan terus bergerak untuk sementara di bagian tindak lanjut dari tindakan utama. Selain itu, tindakan selanjutnya sering dimulai sebelum tindakan sebelumnya benar-benar berakhir. Seorang pemain mungkin mulai berlari saat dia masih melacak bola yang baru saja dia tentng. Mengabaikan aliran alami seperti itu biasanya dirasakan jika ada jeda antara tindakan dan dapat mengakibatkan gerakan mekanis seperti robot. Sementara tumpang tindih diperlukan untuk menjaga gerakan tetap alami, tindakan sekunder sering ditambahkan oleh animator untuk membuat gerakan lebih menarik dan mencapai kompleksitas animasi yang realistis. Penting untuk tidak membiarkan tindakan sekunder mendominasi tindakan utama.

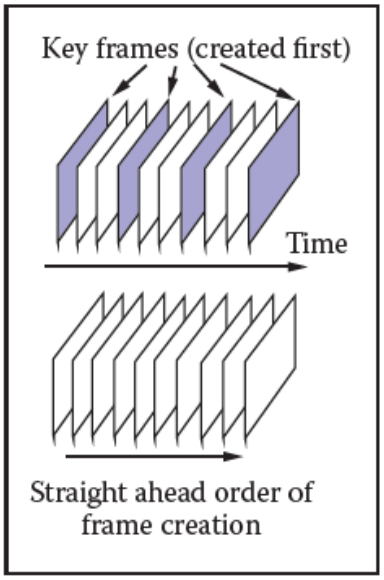
16.4 TEKNIK ANIMASI

Beberapa teknik khusus dapat digunakan untuk membuat gerakan terlihat lebih alami. Yang paling penting mungkin squash and stretch yang menyarankan untuk mengubah bentuk objek bergerak dengan cara tertentu saat bergerak. Seseorang umumnya akan meregangkan objek ke arah gerakan dan menekannya ketika gaya diterapkan padanya, seperti yang ditunjukkan pada Gambar 16.2 untuk animasi klasik dari bola yang memantul. Penting untuk mempertahankan volume total karena ini terjadi untuk menghindari ilusi bertambah atau menyusutnya objek. Semakin besar kecepatan gerak (atau gaya), semakin banyak peregangan (atau pemerasan) yang diterapkan. Deformasi semacam itu digunakan karena beberapa alasan. Untuk gerakan yang sangat cepat, sebuah objek dapat bergerak di antara dua frame berurutan dengan sangat cepat sehingga tidak ada tumpang tindih antara objek pada saat frame saat ini dan pada saat frame sebelumnya yang dapat menyebabkan strobing (varian aliasing). Memanjangkan objek dalam arah gerakan dapat memastikan tumpang tindih yang lebih baik dan membantu mata melawan efek yang tidak menyenangkan ini. Peregangan / pemampatan juga dapat digunakan untuk menunjukkan fleksibilitas objek dengan lebih banyak deformasi yang diterapkan untuk bahan yang lebih lentur. Jika benda itu dimaksudkan untuk tampak kaku, bentuknya sengaja dibiarkan sama ketika bergerak.



Gambar 16.2. Contoh klasik penerapan prinsip squash and stretch. Perhatikan bahwa volume bola yang memantul harus tetap sama sepanjang animasi.

Gerak alami jarang terjadi pada garis lurus, jadi ini biasanya harus dihindari dalam animasi dan busur harus digunakan sebagai gantinya. Demikian pula, tidak ada gerakan dunia nyata yang dapat secara instan mengubah kecepatannya—ini akan membutuhkan jumlah gaya yang tidak terbatas untuk diterapkan pada suatu objek. Hal ini diinginkan untuk menghindari situasi seperti itu dalam animasi juga. Secara khusus, gerakan harus dimulai dan diakhiri secara bertahap (lambat masuk dan keluar). Sementara animasi yang digambar tangan terkadang dilakukan melalui aksi langsung dengan animator mulai dari bingkai pertama dan menggambar satu bingkai demi satu dalam urutan hingga akhir, aksi pose-to-pose, juga dikenal sebagai keyframing, jauh lebih cocok untuk animasi komputer. Dalam teknik ini, animasi direncanakan dengan hati-hati melalui serangkaian bingkai kunci dengan jarak yang relatif jarang dengan sisa animasi (di antara bingkai) yang diisi hanya setelah kunci disetel (Gambar 16.3). Hal ini memungkinkan pengaturan waktu yang lebih tepat dan memungkinkan komputer untuk mengambil alih bagian yang paling membosankan dari proses—pembuatan bingkai di antara—menggunakan algoritme yang disajikan di bagian berikutnya.



Gambar 16.3. Keyframing (atas) mendorong perencanaan tindakan yang terperinci sementara tindakan langsung (bawah) mengarah pada hasil yang lebih spontan.

Hampir semua teknik yang diuraikan di atas dapat digunakan dengan jumlah berlebihan yang wajar untuk mencapai efek artistik yang lebih besar atau menggarisbawahi beberapa sifat khusus dari suatu tindakan atau karakter. Tujuan akhir untuk mencapai sesuatu yang ingin dilihat audiens, sesuatu yang menarik. Kompleksitas ekstrim atau terlalu banyak simetri dalam karakter atau tindakan cenderung kurang menarik. Untuk menciptakan hasil yang baik, seorang animator tradisional membutuhkan keterampilan menggambar yang solid. Analoginya, seorang animator komputer tentu harus memahami grafis komputer dan memiliki pengetahuan yang kuat tentang alat yang digunakannya.

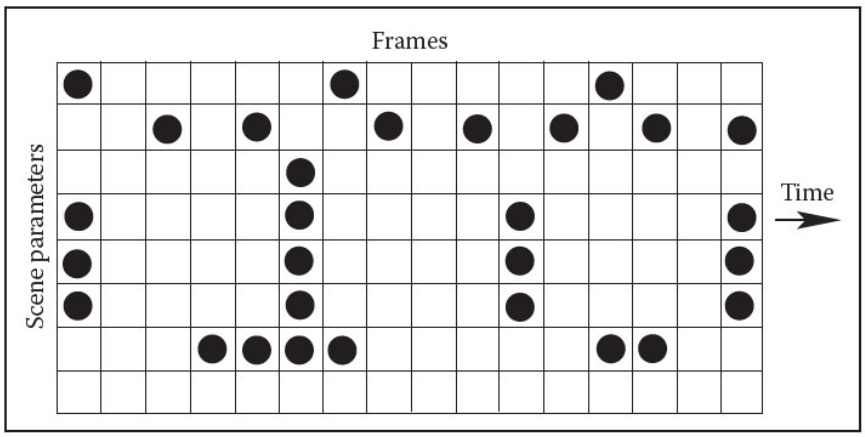
16.5 KONTROL ANIMATOR VS. METODE OTOMATIS

Dalam animasi tradisional, animator memiliki kendali penuh atas semua aspek proses produksi dan tidak ada yang mencegah produk akhir menjadi seperti yang direncanakan dalam setiap detailnya. Harga yang harus dibayar untuk fleksibilitas ini adalah bahwa setiap bingkai dibuat dengan tangan, yang mengarah ke perusahaan yang sangat memakan waktu dan tenaga. Dalam animasi komputer, ada tradeoff yang jelas antara, di satu sisi, memberi animator kontrol lebih langsung atas hasilnya, tetapi memintanya untuk berkontribusi lebih

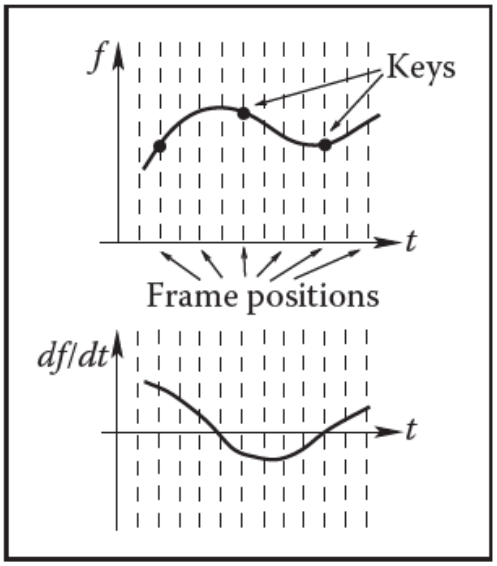
banyak pekerjaan dan, di sisi lain, mengandalkan lebih banyak teknik otomatis yang mungkin memerlukan pengaturan hanya beberapa parameter input tetapi menawarkan sedikit atau tidak ada kontrol atas beberapa properti hasil. Algoritme yang baik harus memberikan fleksibilitas yang cukup saat meminta animator hanya informasi yang intuitif, mudah diberikan, dan yang menurutnya perlu untuk mencapai efek yang diinginkan. Sementara kepatuhan sempurna dengan persyaratan ini tidak mungkin dalam praktek karena mungkin akan mengambil sesuatu yang dekat dengan mesin membaca pikiran, kami mendorong pembaca untuk mengevaluasi setiap teknik animasi komputer dari sudut pandang memberikan keseimbangan tersebut.

16.6 *KEYFRAMING*

Istilah keyframing dapat menyesatkan ketika diterapkan pada animasi komputer 3D karena tidak ada frame yang sebenarnya selesai (yaitu, gambar) yang biasanya terlibat. Pada saat tertentu, scene 3D yang sedang dianimasikan ditentukan oleh serangkaian angka: posisi pusat dari objek yang jatuh, warna RGB-nya, jumlah scalling yang diterapkan pada setiap objek di setiap sumbu, transformasi pemodelan antara bagian yang berbeda dari objek yang kompleks, posisi dan orientasi kamera, intensitas sumber cahaya, dll. Untuk menganimasikan sebuah scene, beberapa subset dari nilai-nilai ini harus berubah seiring waktu. Seseorang dapat, tentu saja, secara langsung menetapkan nilai-nilai ini di setiap bingkai, tetapi ini tidak akan terlalu efisien. Singkatnya, beberapa momen penting dalam waktu (bingkai kunci tk) dapat dipilih di sepanjang garis waktu animasi untuk setiap parameter dan nilai parameter ini (nilai kunci fk) ditetapkan hanya untuk bingkai yang dipilih ini. Kami akan menyebut kombinasi (tk,fk) dari bingkai kunci dan nilai kunci hanya sebuah kunci. Bingkai kunci tidak harus sama untuk parameter yang berbeda, tetapi seringkali logis untuk mengatur kunci setidaknya untuk beberapa di antaranya secara bersamaan. Misalnya, bingkai kunci yang dipilih untuk koordinat x-, y- dan z dari objek tertentu mungkin diatur pada bingkai yang sama persis dengan membentuk kunci vektor posisi tunggal (tk,pk). Bingkai kunci ini, bagaimanapun, mungkin sama sekali berbeda dari yang dipilih untuk orientasi atau warna objek. Bingkai kunci yang lebih dekat satu sama lain, semakin banyak kontrol yang dimiliki animator atas hasilnya; namun biaya melakukan lebih banyak pekerjaan pengaturan kunci harus dinilai. Oleh karena itu, biasanya memiliki jarak yang besar antara tombol di bagian animasi yang relatif sederhana, memusatkannya dalam interval di mana tindakan kompleks terjadi, seperti yang ditunjukkan pada Gambar 16.4.



Gambar 16.4. Pola tombol pengaturan yang berbeda (lingkaran hitam di atas) dapat digunakan secara bersamaan untuk scene yang sama. Diasumsikan bahwa ada lebih banyak bingkai sebelum, dan juga setelah, bagian ini.



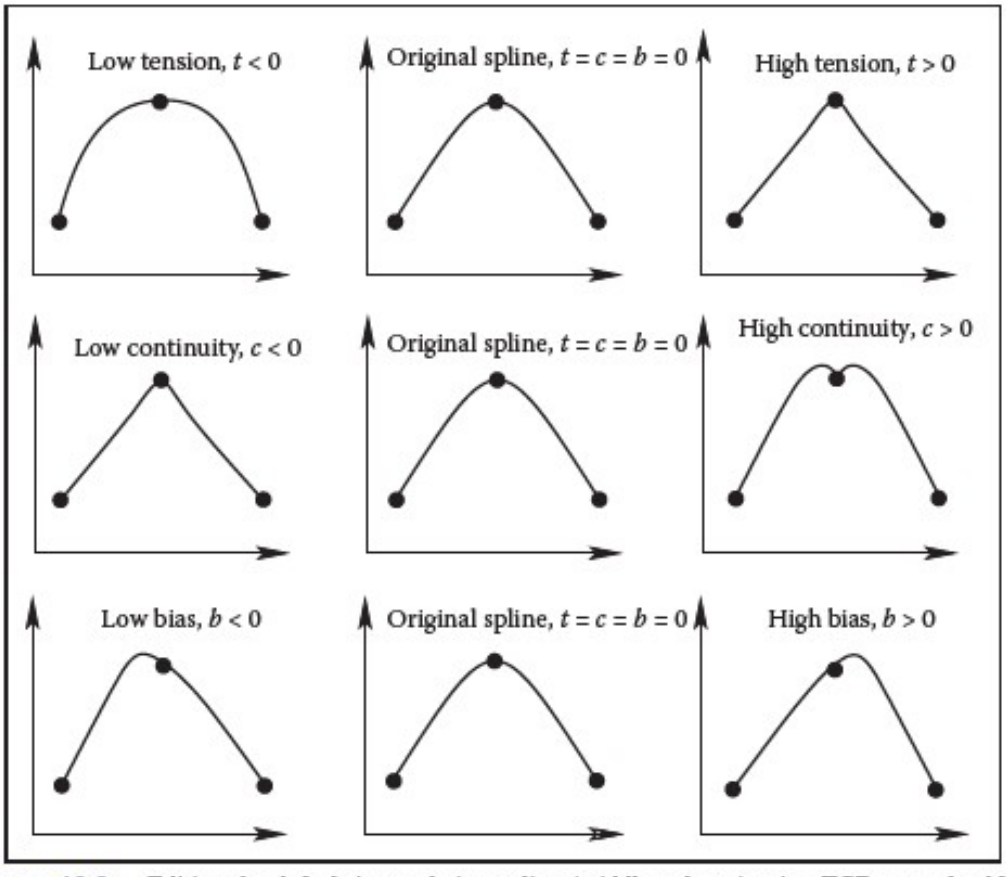
Gambar 16.5. Kurva kontinu f(t) masuk melalui kunci yang disediakan oleh animator meskipun hanya nilai pada posisi bingkai yang menarik. Turunan dari fungsi ini memberikan kecepatan perubahan parameter dan pertama-tama ditentukan secara otomatis oleh prosedur pemasangan.

Setelah animator menetapkan kunci (t_k, f_k), sistem harus menghitung nilai f untuk semua bingkai lainnya. Meskipun pada akhirnya hanya tertarik pada kumpulan nilai diskrit, akan lebih mudah untuk memperlakukan ini sebagai masalah interpolasi klasik yang sesuai dengan kurva animasi kontinu f(t) melalui kumpulan titik data yang disediakan (Gambar 16.5). Diskusi ekstensif tentang algoritma pencocokan kurva dapat ditemukan di Bab 15, dan kita tidak akan mengulanginya di sana. Karena animasi pada awalnya hanya menyediakan kunci dan bukan turunan (singgung), metode yang menghitung semua informasi yang diperlukan secara langsung dari kunci lebih disukai untuk animasi. Kecepatan perubahan parameter sepanjang kurva diberikan oleh turunan kurva terhadap waktu df/dt. Oleh karena itu, untuk menghindari lompatan kecepatan yang tiba-tiba, kontinuitas C^1 biasanya diperlukan. Tingkat kontinuitas yang lebih tinggi biasanya tidak diperlukan dari kurva animasi, karena turunan kedua, yang sesuai dengan percepatan atau gaya yang diterapkan, dapat mengalami perubahan yang sangat mendadak dalam situasi dunia nyata (bola membentur dinding padat), dan turunan yang lebih tinggi tidak secara langsung sesuai dengan parameter gerakan fisik apa pun. Pertimbangan ini menjadikan Catmull-Rom splines salah satu pilihan terbaik untuk pembuatan kurva animasi awal.

Kebanyakan sistem animasi memberi animator kemampuan untuk melakukan pengeditan halus interaktif dari kurva awal ini, termasuk memasukkan lebih banyak kunci, menyesuaikan kunci yang ada, atau memodifikasi garis singgung yang dihitung secara otomatis. Teknik lain yang berguna yang dapat membantu melemahkan bentuk kurva disebut kontrol TCB (TCB adalah singkatan dari tegangan, kontinuitas, dan bias). Idenya adalah untuk memperkenalkan tiga parameter baru yang dapat digunakan untuk mengubah bentuk kunci kurva melalui penyesuaian terkoordinasi dari garis singgung masuk dan keluar pada titik ini. Untuk kunci dengan jarak waktu yang seragam dengan jarak Δt di antara mereka, ekspresi Catmull-Rom standar untuk garis singgung T^{in}_i masuk dan tangen T^{out}_i keluar pada kunci internal (t_k, f_k).

Parameter tegangan mengontrol ketajaman kurva bumi dengan menskalakan garis singgung masuk dan keluar. Garis singgung yang lebih besar (tegangan yang lebih rendah) menyebabkan bentuk kurva yang lebih rata di dekat kunci. Bias b memungkinkan animator untuk secara selektif meningkatkan bobot tetangga kunci secara lokal menarik kurva lebih

dekat ke garis lurus yang menghubungkan kunci dengan kirinya (b dekat 1, "melampaui" aksi) atau kanan (b dekat-1, "mengurangi" tindakan) tetangga. Nilai kontinuitas c yang tidak nol membuat garis singgung masuk dan keluar berbeda yang memungkinkan animator membuat kekusutan pada kurva pada nilai kunci. Nilai parameter TCB yang berguna secara praktis biasanya terbatas pada interval[−1;1] dengan default t = c = b =0sesuai dengan spline Catmull-Rom asli. Contoh kemungkinan penyesuaian bentuk kurva ditunjukkan pada Gambar 16.6.

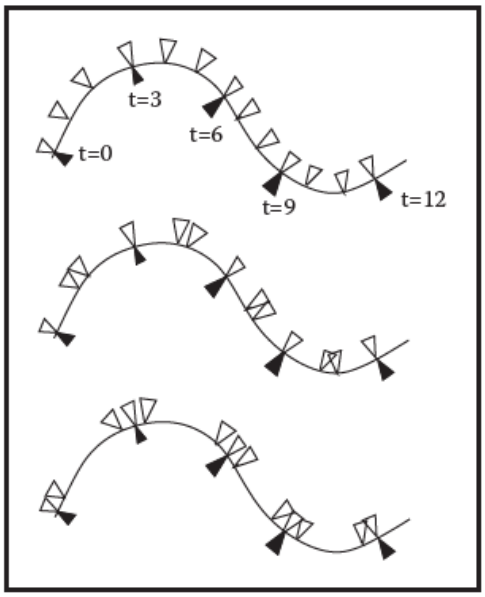


Gambar 16.6. Mengedit spline interpolasi default (kolom tengah) menggunakan kontrol TCB. Perhatikan bahwa semua kunci tetap pada posisi yang sama.

16.7 *MOTION CONTROL*

Sejauh ini, kami telah menjelaskan cara mengontrol bentuk kurva animasi melalui pemosisian tombol dan penyesuaian nilai tangen pada tombol. Ini, bagaimanapun, umumnya tidak cukup ketika seseorang ingin memiliki kontrol baik di mana objek bergerak, yaitu jalurnya, dan seberapa cepat ia bergerak di sepanjang jalur ini. Diberikan satu set posisi dalam tombol ruang, teknik penyesuaian kurva otomatis dapat membuat kurva melaluinya, tetapi gerakan yang dihasilkan hanya dibatasi dengan memaksa objek untuk sampai pada posisi kunci tertentu p_k pada bingkai kunci yang sesuai t_k, dan tidak ada yang secara langsung dikatakan tentang kecepatan gerak di antara tombol. Hal ini dapat menimbulkan masalah. Misalnya, jika sebuah benda bergerak sepanjang sumbu x dengan kecepatan 11 meter per detik selama 1 detik dan kemudian dengan 1 meter per detik selama 9 detik, ia akan tiba di posisi x = 20 setelah 10 detik sehingga memenuhi kunci animator (0,0) dan (10, 20). Agak tidak mungkin bahwa gerakan tersentak-sentak ini benar-benar diinginkan, dan gerakan seragam dengan kecepatan 2 meter/detik mungkin lebih dekat dengan apa yang diinginkan animator saat mengatur tombol-tombol ini. Meskipun biasanya tidak menampilkan perilaku ekstrem seperti itu, kurva polinomial yang dihasilkan dari prosedur pemasangan standar memang menunjukkan kecepatan gerak yang tidak seragam antara tombol seperti yang

ditunjukkan pada Gambar 16.7. Meskipun ini dapat ditoleransi (dalam batas) untuk beberapa parameter yang sistem visual manusianya tidak terlalu baik dalam menentukan ketidakseragaman dalam laju perubahan (seperti warna atau bahkan laju rotasi), kita harus melakukan yang lebih baik untuk posisi p dari objek di mana kecepatan secara langsung sesuai dengan pengalaman sehari-hari.



Gambar 16.7. Ketiga gerakan berada di sepanjang jalur 2D yang sama dan memenuhi set kunci di ujung segitiga hitam.

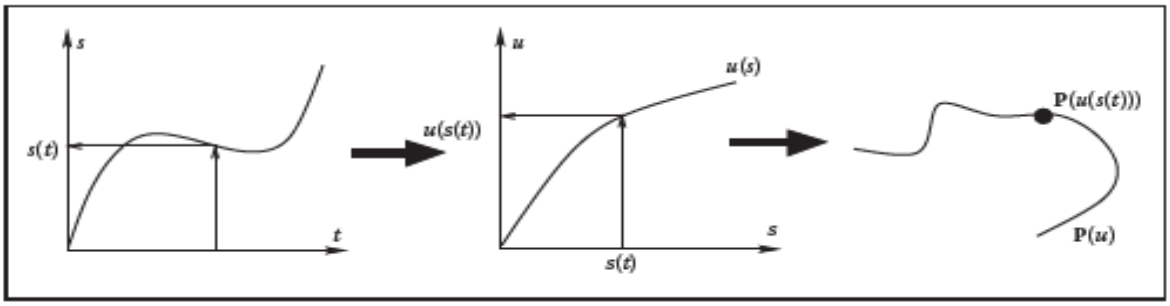
Ujung segitiga putih menunjukkan posisi benda pada interval t = 1. Kecepatan gerakan yang seragam antara tombol (atas) mungkin lebih dekat dengan apa yang diinginkan animator, tetapi prosedur pemasangan otomatis dapat menghasilkan salah satu dari dua gerakan lainnya.

Pertama-tama kita akan membedakan parameterisasi kurva yang digunakan selama prosedur pemasangan dari yang digunakan untuk animasi. Ketika kurva masuk melalui tombol posisi, kita akan menulis hasilnya sebagai fungsi p(u) dari beberapa parameter u. Ini akan menggambarkan geometri kurva dalam ruang. Panjang busur s adalah panjang fisik kurva. Cara alami bagi animator untuk mengontrol gerakan di sepanjang kurva yang ada sekarang adalah dengan menentukan fungsi tambahan s(t) yang sesuai dengan seberapa jauh objek seharusnya berada di sepanjang kurva pada waktu tertentu. Untuk mendapatkan posisi aktual dalam ruang, kita memerlukan satu lagi fungsi bantu u(s) yang menghitung nilai parameter u untuk panjang busur yang diberikan s. Proses lengkap menghitung posisi objek untuk waktu tertentu t kemudian diberikan dengan menyusun fungsi-fungsi ini (lihat Gambar 16.8):

$$\mathbf{p}(t) = \mathbf{p}(u(s(t))).$$

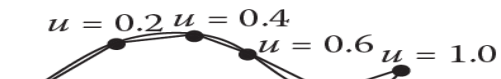
Beberapa fungsi standar dapat digunakan sebagai fungsi jarak-waktu s(t). Salah satu yang paling sederhana adalah fungsi linier yang berhubungan dengan kecepatan konstan: s(t)=vt dengan v = const. Contoh umum lainnya adalah gerak dengan percepatan konstan a (dan kecepatan awal v0) yang digambarkan oleh parabola $s(t) = v_0t + at^2/2$. Karena kecepatan berubah secara bertahap di sini, fungsi ini dapat membantu untuk memodelkan perilaku masuk dan keluar yang diinginkan. Lebih umum, kemiringan s(t) memberikan kecepatan gerak dengan kemiringan negatif yang sesuai dengan gerak mundur sepanjang kurva. Untuk mencapai sebagian besar fleksibilitas, kemampuan untuk mengedit s(t) secara interaktif biasanya diberikan kepada animator oleh sistem animasi. Fungsi jarak-waktu bukan satu-satunya cara untuk mengontrol gerakan. Dalam beberapa kasus mungkin lebih nyaman bagi pengguna untuk menentukan fungsi kecepatan-waktu v(t) atau bahkan fungsi percepatan-

waktu a(t). Karena ini adalah turunan pertama dan kedua dari s(t), untuk menggunakan jenis kontrol ini, sistem terlebih dahulu memulihkan fungsi jarak-waktu dengan mengintegrasikan input pengguna (dua kali dalam kasus a(t)).



Gambar 16.8. Untuk mendapatkan posisi dalam ruang pada waktu t tertentu, yang pertama menggunakan kontrol gerak yang ditentukan pengguna untuk mendapatkan jarak sepanjang kurva s(t) dan kemudian menghitung nilai parameter kurva yang sesuai u(s(t)). Kurva yang dipasang sebelumnya P(u) sekarang dapat digunakan untuk mencari posisi P(u(s(t))).

Hubungan antara parameter kurva dan panjang garis ditetapkan secara otomatis oleh sistem. Dalam praktiknya, sistem pertama-tama menentukan ketergantungan panjang busur pada parameter u (yaitu, fungsi invers s(u)). Dengan menggunakan fungsi ini, untuk setiap S yang diberikan, persamaan s(u)−S = 0 tanpa diketahui dapat diselesaikan dengan memperoleh u(S). Untuk sebagian besar kurva, fungsi s(u) tidak dapat dinyatakan dalam bentuk analitik tertutup dan integrasi numerik diperlukan (lihat Bab 14). Prosedur pencarian akar numerik standar (seperti metode Newton-Raphson, misalnya) kemudian dapat langsung digunakan untuk menyelesaikan persamaan s(u)−S = 0 untuk u.



u	s(u)
0.0	0.0
0.2	2.5
0.4	4.0
0.6	5.0
0.8	7.0
1.0	8.5

Gambar 16.9. Untuk membuat versi tabular dari s(u), kurva dapat didekati dengan sejumlah segmen garis yang menghubungkan titik-titik pada kurva yang diposisikan pada peningkatan parameter yang sama. Tabel dicari untuk menemukan u-interval untuk S tertentu. Untuk kurva di atas, misalnya, nilai u yang sesuai dengan posisi S=6,5 terletak antara u = 0,6 dan u = 0,8.

Teknik alternatif adalah dengan memperkirakan kurva itu sendiri sebagai himpunan segmen linier antara titik pi yang dihitung pada beberapa himpunan nilai parameter ui yang cukup rapat. Satu kemudian membuat tabel perkiraan panjang busur

$$s(u_i) = \sum_{j=1}^{i} \| p_j - p_{j-1} \| = s(u_{i-1}) + \| p_i - p_{i-1} \|$$

Karena s(u) adalah fungsi tak-turun dari u, maka kita dapat menemukan interval yang berisi nilai S dengan pencarian sederhana melalui tabel (lihat Gambar 16.9). Interpolasi linier dari nilai akhir u interval kemudian dilakukan untuk akhirnya menemukan u(S). Jika presisi yang

lebih besar diperlukan, beberapa langkah dari algoritma Newton-Raphson dengan nilai ini sebagai titik awal dapat diterapkan.

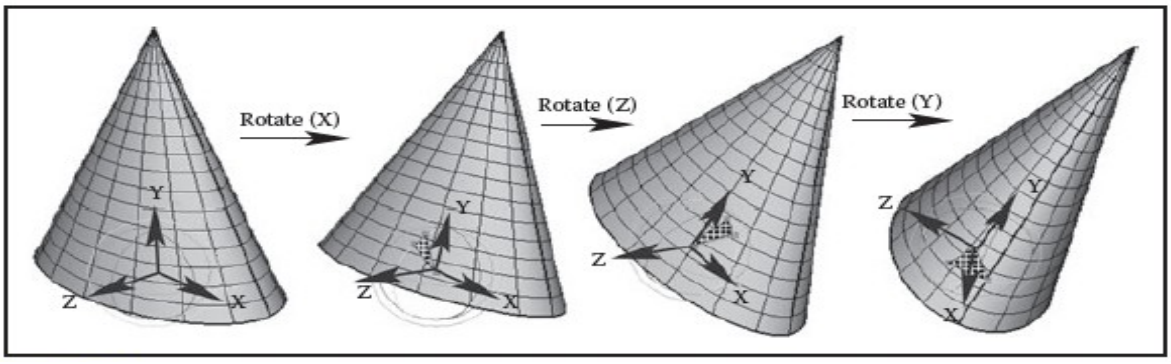
16.8 ROTASI INTERPOLASI

Teknik yang disajikan di atas dapat digunakan untuk menginterpolasi rangkaian kunci dari sebagian besar parameter yang menggambarkan scene. Rotasi tiga dimensi adalah salah satu gerakan penting yang metode interpolasi dan representasinya lebih khusus. Alasan untuk ini adalah bahwa menerapkan teknik standar pada rotasi 3D sering menimbulkan masalah praktis yang serius. Rotasi (perubahan orientasi suatu benda) adalah satu-satunya gerak selain translasi yang membuat bentuk benda tetap utuh. Oleh karena itu, ia memainkan peran khusus dalam menghidupkan objek kaku.

Ada beberapa cara untuk menentukan orientasi suatu objek. Pertama, matriks transformasi seperti yang dijelaskan dalam Bab 6 dapat digunakan. Sayangnya, interpolasi matriks rotasi naif (elemen demi elemen) tidak menghasilkan hasil yang benar. Misalnya, matriks "setengah" antara jam 2D dan rotasi 90 derajat berlawanan arah jarum jam adalah matriks nol:

$$\frac{1}{2}\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} + \frac{1}{2}\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Hasil yang benar, tentu saja, matriks satuan yang sesuai dengan tidak ada rotasi. Kedua, seseorang dapat menentukan orientasi arbitrer sebagai urutan tepat tiga rotasi di sekitar sumbu koordinat yang dipilih dalam urutan tertentu. Sumbu-sumbu ini dapat ditetapkan dalam ruang (representasi sudut tetap) atau tertanam ke dalam objek sehingga berubah setelah setiap rotasi (representasi sudut Euler seperti yang ditunjukkan pada Gambar 16.10). Ketiga sudut rotasi ini dapat dianimasikan secara langsung melalui keyframing standar, tetapi masalah halus yang dikenal sebagai kunci gimbal muncul. Kunci gimbal terjadi jika selama rotasi salah satu dari tiga sumbu rotasi secara tidak sengaja disejajarkan dengan yang lain, sehingga mengurangi satu jumlah derajat kebebasan yang tersedia seperti yang ditunjukkan pada Gambar 16.11 untuk perangkat fisik. Efek ini lebih umum daripada yang mungkin dipikirkan—satu putaran 90 derajat ke kanan (atau kiri) berpotensi menempatkan objek ke dalam gimballock. Akhirnya, setiap orientasi dapat ditentukan dengan memilih sumbu yang sesuai dalam ruang dan rotasi sudut di sekitar sumbu ini. Sementara animasi dalam representasi ini relatif mudah, menggabungkan dua rotasi, yaitu, menemukan sumbu dan sudut yang sesuai dengan urutan dua rotasi keduanya diwakili oleh sumbu dan sudut, adalah nontrivial. Sebuah peralatan matematika khusus, quaternions telah dikembangkan untuk membuat representasi ini cocok baik untuk menggabungkan beberapa rotasi menjadi satu dan untuk animasi.



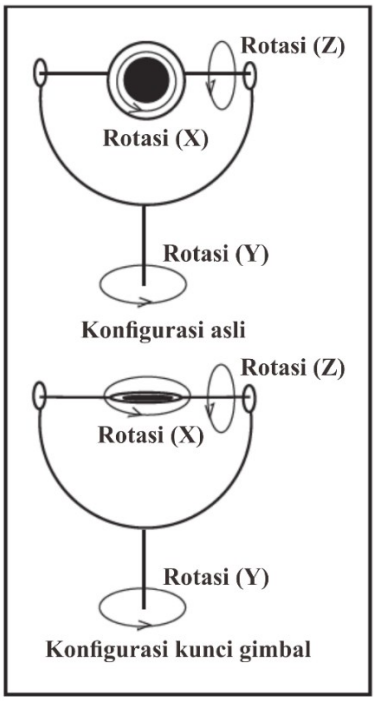
Gambar 16.10. Tiga sudut Euler dapat digunakan untuk menentukan orientasi objek sewenang-wenang melalui urutan tiga rotasi di sekitar sumbu koordinat yang tertanam ke dalam objek (sumbu Y selalu menunjuk ke ujung kerucut).

Perhatikan bahwa setiap rotasi diberikan dalam sistem koordinat baru. Representasi sudut tetap sangat mirip, tetapi sumbu koordinat yang digunakannya tetap dalam ruang dan tidak berputar dengan objek.

Diberikan vektor 3D v = (x, y, z) dan skalar s, sebuah quaternion q dibentuk dengan menggabungkan keduanya menjadi objek empat komponen: q = [w x y z] = [s; v]. Beberapa operasi barudidefinisikan untuk quaternions. Quaternionadditions hanya menjumlahkan bagian-bagian skalar dan vektor secara terpisah:

$$q_1 + q_2 \equiv [s_1 + s_2;\ \mathbf{v_1} + \mathbf{v_2}].$$

Perkalian dengan skalar a menghasilkan quaternion baru



Gambar 16.11. Dalam contoh ini, kunci gimbal terjadi ketika sumbu Z berputar 90 derajat. Rotasi X dan Y sekarang dilakukan di sekitar sumbu yang sama yang menyebabkan hilangnya satu derajat kebebasan.

$$aq \equiv [as;\ a\mathbf{v}].$$

Perkalian bilangan empat yang lebih kompleks didefinisikan sebagai:

$$q_1 \cdot q_2 \equiv [s_1 s_2 - \mathbf{v_1}\mathbf{v_2};\ s_1\mathbf{v_2} + s_2\mathbf{v_1} + \mathbf{v_1} \times \mathbf{v_2}],$$

di mana × menunjukkan perkalian silang vektor. Sangat mudah untuk melihat bahwa, mirip dengan matriks, perkalian quaternion adalah asosiatif, tetapi tidak komutatif. Kami akan lebih tertarik pada quaternion yang dinormalisasi—yang norma quaternionnya $|q| = \sqrt{s^2 + v^2}$sama dengan satu. Satu definisi akhir yang kita perlukan adalah definisi bilangan bulat terbalik:

$$q^{-1} = (1/|q|)[s;\ -\mathbf{v}].$$

Untuk menyatakan rotasi dengan sudut di sekitar sumbu yang melalui titik asal yang arahnya diberikan oleh vektor ternormalisasi n, sebuah quaternion ternormalisasi

$$q = [\cos(\varphi/2);\ \sin(\varphi/2)\mathbf{n}]$$

terbentuk. Untuk memutar titik p, satu mengubahnya menjadi quaternion qp = [0;p] dan menghitung hasil quaternion

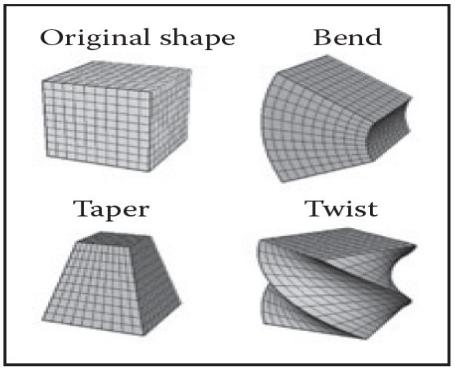
$$q'_p = q \cdot q_p \cdot q^{-1}$$

Gambar 16.12. Interpolasi quaternions harus dilakukan pada permukaan bola 3Dunit yang tertanam dalam ruang 4D. Namun, interpolasi yang lebih sederhana di sepanjang garis lurus 4D (lingkaran terbuka) diikuti dengan proyeksi ulang hasil ke bola (lingkaran hitam) seringkali cukup. Rotasi komposit diberikan hanya oleh produk dari quaternions yang mewakili setiap langkah rotasi terpisah. Untuk menganimasikan dengan quaternion, seseorang tidak dapat memperlakukannya sebagai titik dalam ruang empat dimensi dan menyetel kunci secara langsung di ruang ini. Untuk menjaga agar quaternions dinormalisasi, seseorang harus, secara tegas, membatasi prosedur interpolasi ke unit bola (objek 3D) dalam ruang 4D ini. Namun, versi bulat dari interpolasi linier (sering disebut slerp) sudah menghasilkan matematika yang agak tidak menyenangkan. Interpolasi linier 4D sederhana diikuti dengan proyeksi ke unit-unit yang ditunjukkan pada Gambar 16.12 praktik yang sangat sederhana dan sering kali cukup. Hasil yang lebih halus dapat diperoleh melalui aplikasi berulang prosedur interpolasi linier menggunakan algoritma de Casteljau.

16.9 DEFORMASI

Meskipun teknik untuk deformasi objek mungkin lebih tepat diperlakukan sebagai alat pemodelan, mereka secara tradisional dibahas bersama dengan metode animasi. Mungkin contoh paling sederhana dari operasi yang mengubah bentuk objek adalah scalling yang tidak seragam. Lebih umum, beberapa fungsi dapat diterapkan ke koordinat lokal dari semua titik yang menentukan objek (yaitu, simpul dari mesh segitiga atau poligon kontrol dari permukaan spline), memposisikan ulang titik-titik ini dan membuat bentuk baru: $p = f(p, \gamma)$ di mana adalah vektor parameter yang digunakan oleh fungsi deformasi. Memilih f yang berbeda (dan menggabungkannya dengan menerapkan satu demi satu) dapat membantu menciptakan deformasi yang sangat menarik. Contoh fungsi sederhana yang berguna termasuk menekuk, memutar, dan lancip yang ditunjukkan pada Gambar 16.13. Menganimasikan perubahan bentuk sangat mudah dalam hal ini dengan membingkai parameter fungsi deformasi. Kerugian dari teknik ini termasuk kesulitan memilih fungsi matematika untuk beberapa deformasi tidak standar dan fakta bahwa deformasi yang dihasilkan bersifat global dalam arti bahwa objek yang lengkap, dan bukan hanya sebagian darinya, dibentuk kembali.

Untuk mengubah bentuk objek secara lokal sambil memberikan kontrol lebih langsung atas hasilnya, seseorang dapat memilih satu simpul, memindahkannya ke lokasi baru dan menyesuaikan simpul dalam beberapa lingkungan untuk mengikuti simpul benih. Area yang dipengaruhi oleh deformasi dan jumlah perpindahan tertentu di berbagai bagian objek dikendalikan oleh fungsi redaman yang berkurang dengan jarak (biasanya dihitung di atas permukaan objek) ke titik awal. Seedvertexmotion dapat dibingkai kunci untuk menghasilkan animasi perubahan bentuk.

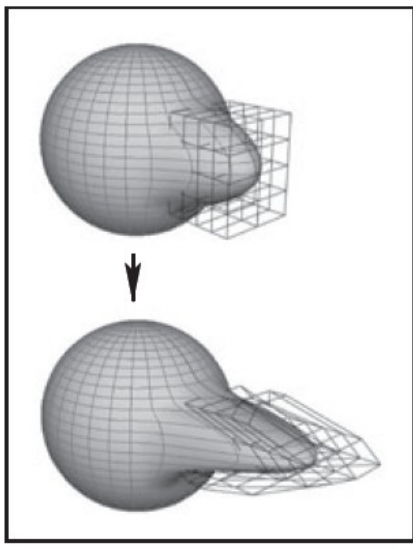


Gambar 16.13. Contoh populer deformasi global. Sudut membungkuk dan memutar, serta tingkat lancip, semuanya dapat dianimasikan untuk mencapai perubahan bentuk yang dinamis.

Teknik deformasi yang lebih umum disebut deformasi bentuk bebas/*free-form deformation* (FFD) (Sederberg & Parry, 1986). Sebuah grid koordinat lokal (dalam kebanyakan kasus bujursangkar) pertama kali dibuat untuk merangkum bagian dari objek yang akan dideformasi, dan koordinat (s, t, u) dari semua titik yang relevan dihitung sehubungan dengan grid ini. Pengguna kemudian dengan bebas membentuk kembali kisi titik kisi P_{ijk} menjadi kisi P'_{ijk} terdistorsi baru (Gambar 16.14). Objek direkonstruksi menggunakan koordinat yang dihitung di grid asli tidak terdistorsi di analog trivariat dari interpolan Be´zier (lihat Bab 15) dengan titik kisi terdistorsi P'_{ijk} melayani sebagai titik kontrol dalam ekspresi ini:

$$P(s,u,t) = \sum_{i=0}^{L} \binom{i}{L} (1-s)^{L-i} s^i \sum_{j=0}^{M} \binom{j}{M} (1-t)^{M-j} t^j \sum_{k=0}^{N} \binom{k}{N} (1-u)^{N-k} u^k \mathbf{P'_{ijk}},$$

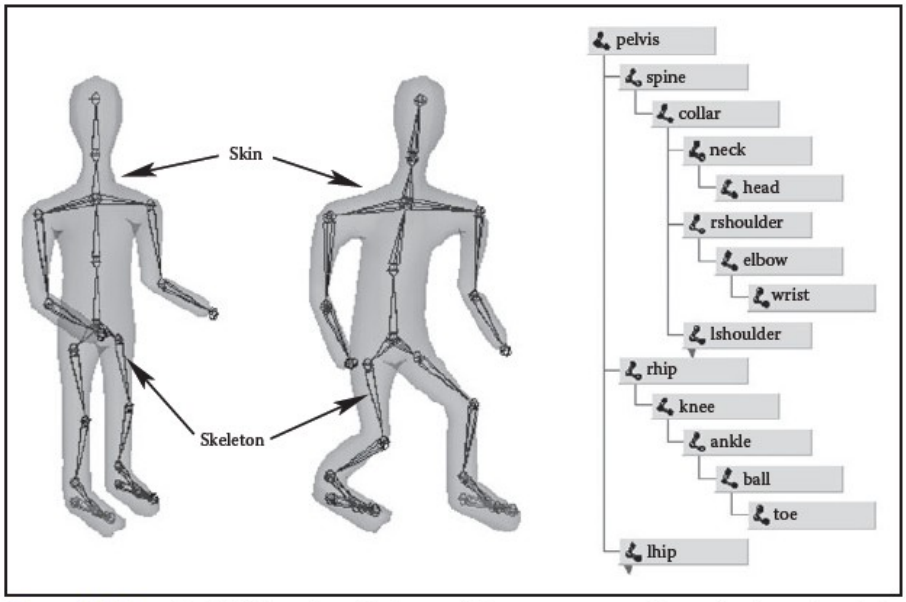
di mana L, M, N adalah indeks maksimum titik kisi di setiap dimensi. Akibatnya, kisi berfungsi sebagai versi objek dengan resolusi rendah untuk tujuan deformasi, memungkinkan perubahan bentuk yang mulus dari objek kompleks yang sewenang-wenang melalui sejumlah kecil penyesuaian intuitif. Kisi-kisi FFD sendiri dapat diperlakukan sebagai objek reguler oleh sistem dan dapat diubah, dianimasikan, dan bahkan dideformasi lebih lanjut jika perlu, yang mengarah pada perubahan yang sesuai pada objek yang dilekatkan kisi. Misalnya, memindahkan alat deformasi yang terdiri dari kisi asli dan kisi terdistorsi yang mewakili tonjolan melintasi suatu objek menghasilkan tonjolan yang bergerak melintasi objek.



Gambar 6.14. Menyesuaikan Kisi FFD Menghasilkan Deformasi Objek

16.10 ANIMASI KARAKTER

Animasi gambar artikulasi paling sering dilakukan melalui kombinasi keyframing dan teknik deformasi khusus. Model karakter yang dimaksudkan untuk animasi biasanya terdiri dari setidaknya dua lapisan utama seperti yang ditunjukkan pada Gambar 16.15. Gerakan permukaan yang sangat detail yang mewakili kulit terluar atau kulit karakter adalah apa yang pada akhirnya akan dilihat pemirsa dalam produk akhir. Kerangka di bawahnya adalah struktur hierarkis (pohon) sambungan yang menyediakan model kinematik dari gambar dan digunakan secara eksklusif untuk animasi. Dalam beberapa kasus, lapisan tengah tambahan yang kira-kira sesuai dengan otot dimasukkan di antara kerangka dan kulit.

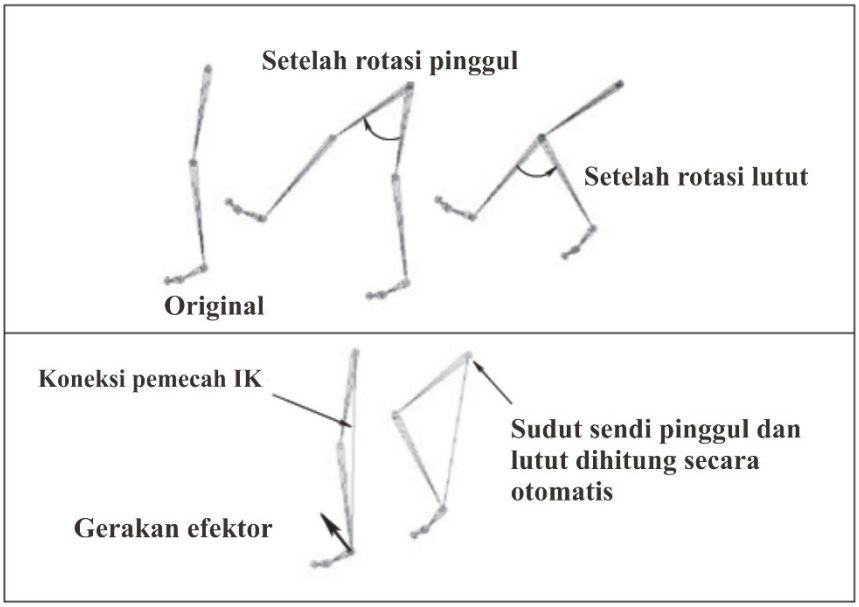


Gambar 16.15. (Kiri) Hirarki sendi, kerangka, berfungsi sebagai abstraksi kinematik dari karakter; (tengah) memposisikan ulang kerangka merusak objek kulit terpisah yang melekat padanya; (kanan) struktur data pohon digunakan untuk mewakili kerangka. Untuk kekompakan, struktur internal beberapa node disembunyikan (mereka identik dengan saudara kandung yang sesuai).

Setiap sendi kerangka bertindak sebagai induk untuk hierarki di bawahnya. Akar mewakili seluruh karakter dan diposisikan langsung di sistem koordinat dunia. Jika matriks transformasi lokal yang menghubungkan suatu sendi dengan induknya dalam hierarki tersedia, kita dapat memperoleh transformasi yang menghubungkan ruang lokal dari setiap sendi ke sistem dunia (yaitu, sistem akar) dengan hanya menggabungkan transformasi sepanjang jalur dari akar ke sendi. Untuk mengevaluasi seluruh kerangka (yaitu, menemukan posisi dan orientasi semua sambungan), dilakukan penelusuran kedalaman-pertama dari pohon sambungan lengkap. Tumpukan transformasi adalah struktur data alami untuk membantu tugas ini. Saat melintasi pohon, matriks komposit saat ini didorong pada tumpukan dan yang baru dibuat dengan mengalikan matriks saat ini dengan yang disimpan di sambungan. Saat menelusuri kembali ke induknya, transformasi ekstra ini harus dibatalkan sebelum cabang lain dikunjungi; ini mudah dilakukan hanya dengan memecahkan tumpukan. Meskipun teknik umum dan sederhana ini untuk menilai hierarki digunakan di seluruh grafis komputer, dalam animasi (dan robotika) diberi nama khusus — kinematika maju (FK). Sementara representasi umum untuk semua transformasi dapat digunakan, biasanya menggunakan set parameter khusus, seperti panjang tautan atau sudut sambungan, untuk menentukan kerangka. Untuk menganimasikan dengan kinematika maju, parameter rotasi dari semua sambungan dimanipulasi secara langsung. Teknik ini juga memungkinkan animator untuk mengubah jarak antara sambungan (panjang tautan), tetapi kita harus menyadari bahwa ini berhubungan dengan peregangan anggota badan dan sering kali terlihat agak tidak wajar.

Kinematika maju mengharuskan pengguna untuk mengatur parameter untuk semua sambungan yang terlibat dalam gerakan (Gambar 16.16 (atas)). Sebagian besar sendi ini, bagaimanapun, termasuk dalam simpul internal hierarki, dan gerakan mereka biasanya bukan sesuatu yang ingin dikhawatirkan oleh animator. Dalam kebanyakan situasi, animator hanya ingin mereka bergerak secara alami “sendirian”, dan seseorang lebih tertarik untuk menentukan perilaku titik akhir rantai sambungan, yang biasanya sesuai dengan sesuatu yang melakukan tindakan tertentu, seperti pergelangan kaki atau tangan. seujung jari. Animator lebih suka parameter semua sambungan internal ditentukan dari gerakan efektor akhir secara

otomatis oleh sistem. Kinematika terbalik (IK) memungkinkan kita melakukan hal itu (lihat Gambar 16.16(bawah)).



Gambar 16.16. Kinematika maju (atas) membutuhkan animator untuk menempatkan semua sendi ke posisi yang benar. Dalam kinematik terbalik (bawah), parameter beberapa sambungan internal dihitung berdasarkan gerakan efektor ujung yang diinginkan.

Misalkan x adalah posisi efektor akhir dan adalah vektor parameter yang diperlukan untuk menentukan semua sambungan internal sepanjang rantai dari akar ke sambungan akhir. Kadang-kadang orientasi dari sambungan akhir juga ditentukan secara langsung oleh animator, dalam hal ini kita mengasumsikan bahwa variabel yang bersesuaian termasuk dalam vektor x. Untuk kesederhanaan, bagaimanapun, kami akan menulis semua ekspresi khusus untuk vektor:

$$\mathbf{x} = (x_1, x_2, x_3)^T .$$

Karena setiap variabel dalam x adalah fungsi dari , maka dapat ditulis sebagai persamaan vektor x = F(α). Jika kita mengubah parameter sambungan internal dengan jumlah yang kecil, perubahan yang dihasilkan x pada posisi endeffector dapat ditulis secara kira-kira sebagai

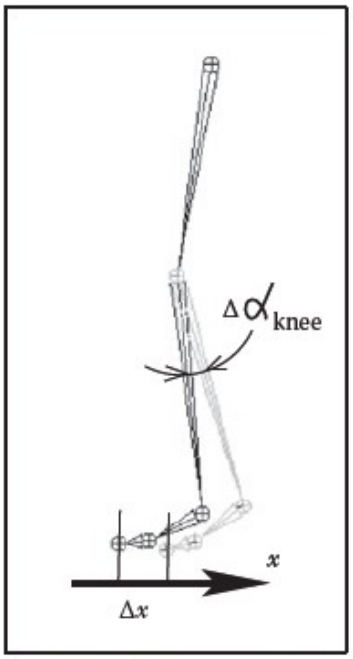
$$\delta\mathbf{x} = \frac{\partial \mathbf{F}}{\partial \alpha}\delta\alpha,$$

di mana F adalah matriks turunan parsial yang disebut Jacobian:

$$\frac{\partial \mathbf{F}}{\partial \alpha} = \begin{bmatrix} \frac{\partial f_1}{\partial \alpha_1} & \frac{\partial f_1}{\partial \alpha_2} & \cdots & \frac{\partial f_1}{\partial \alpha_n} \\ \frac{\partial f_2}{\partial \alpha_1} & \frac{\partial f_2}{\partial \alpha_2} & \cdots & \frac{\partial f_2}{\partial \alpha_n} \\ \frac{\partial f_3}{\partial \alpha_1} & \frac{\partial f_3}{\partial \alpha_2} & \cdots & \frac{\partial f_3}{\partial \alpha_n} \end{bmatrix}$$

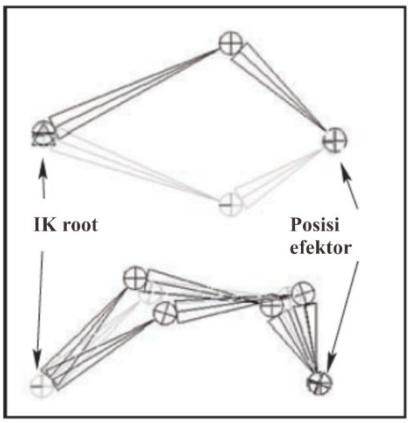
Pada setiap saat, kita mengetahui posisi yang diinginkan dari efektor akhir (ditetapkan oleh animator) dan, tentu saja, posisi efektor saat ini. Kurangi keduanya, kita akan mendapatkan penyesuaian yang diinginkan δx. Elemen dari matriks Jacobian berhubungan dengan perubahan koordinat dari efektor akhir ketika parameter internal tertentu diubah sementara yang lain tetap (lihat Gambar 16.17). Elemen-elemen ini dapat dihitung untuk setiap konfigurasi kerangka tertentu menggunakan hubungan geometris. Satu-satunya yang tidak diketahui dalam sistem persamaan (16.1) adalah perubahan dalam parameter internal α.

Setelah kami menyelesaikannya, kami memperbarui α = + δα yang memberikan semua informasi yang diperlukan untuk prosedur FK untuk memposisikan ulang kerangka.



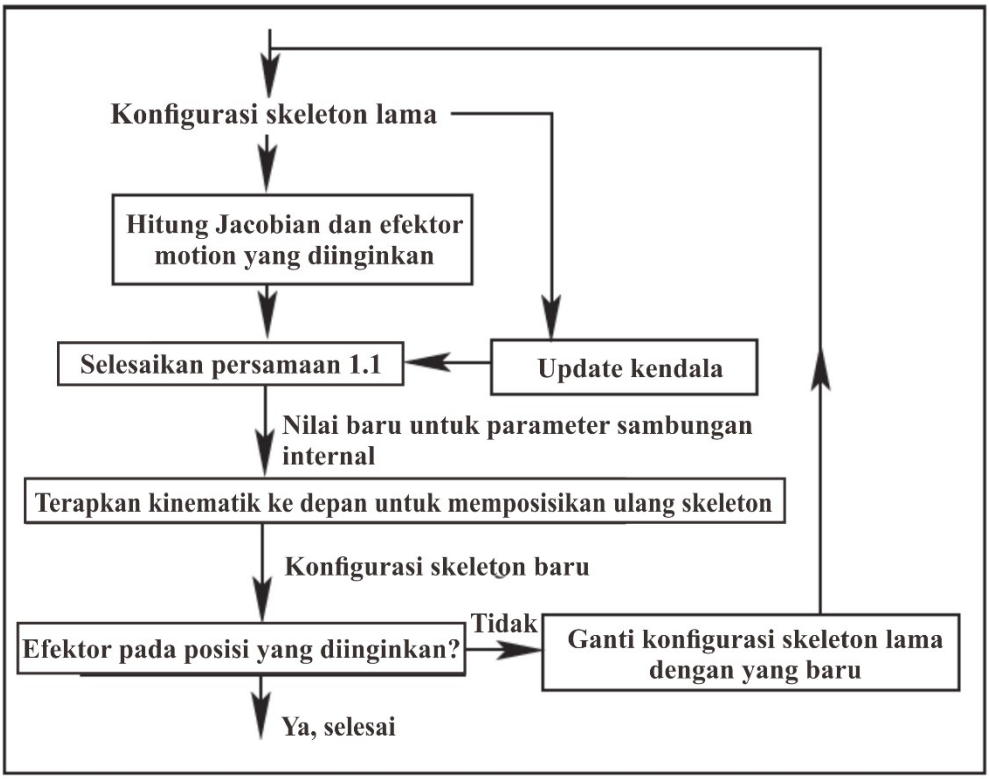
Gambar 16.17. Turunan parsial x/knee diberikan oleh limit x/knee. Perpindahan efektor dihitung sementara semua sendi, kecuali lutut, dijaga tetap.

Sayangnya, sistem (16.1) biasanya tidak dapat diselesaikan secara analitik dan, terlebih lagi, dalam kebanyakan kasus tidak dibatasi, yaitu, jumlah parameter sambungan internal yang tidak diketahui melebihi jumlah variabel dalam vektor x. Ini berarti bahwa gerakan kerangka yang berbeda dapat menghasilkan gerakan yang sama dari efektor akhir. Beberapa contoh ditunjukkan pada Gambar 16.18. Banyak cara untuk memperoleh solusi khusus untuk sistem seperti itu tersedia, termasuk yang mempertimbangkan kendala alami yang diperlukan untuk beberapa sendi kehidupan nyata (misalnya, menekuk lutut hanya ke satu arah). Kita juga harus ingat bahwa matriks Jacobian yang dihitung hanya valid untuk satu konfigurasi tertentu, dan harus diperbarui saat kerangka bergerak. Kerangka IK lengkap disajikan pada Gambar 16.19. Tentu saja, sambungan akar untuk IK tidak harus menjadi akar dari hierarki utama, dan beberapa pemecah IK dapat diterapkan ke bagian kerangka yang independen. Misalnya, seseorang dapat menggunakan pemecah terpisah untuk kaki kanan dan kiri dan satu lagi untuk membantu mencengkeram dengan tangan kanan, masing-masing dengan akarnya sendiri.

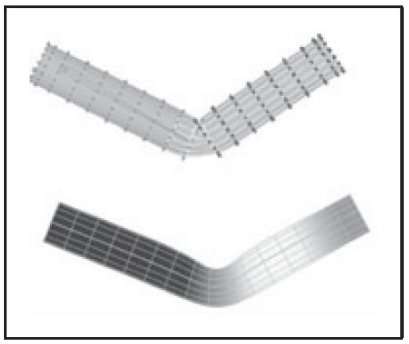


Gambar 16.18. Beberapa konfigurasi sendi internal dapat menghasilkan posisi efektor yang sama. (Atas) solusi "terbalik" terputus-putus; (bawah) rangkaian solusi.

Kombinasi pendekatan FK dan K biasanya digunakan untuk menghidupkan kerangka. Banyak gerakan umum (berjalan atau berlari, menggenggam, meraih, dll.) menunjukkan pola yang terkenal dari gerakan bersama yang memungkinkan untuk dengan cepat membuat gerakan yang tampak alami atau bahkan menggunakan perpustakaan "klip" semacam itu. Animator kemudian menyesuaikan hasil generik ini sesuai dengan parameter fisik karakter dan juga untuk memberikan lebih banyak individualitas.



Gambar 16.19. Diagram dari algoritma kinematika terbalik.



Gambar 16.20. Atas: Rigid skinning memberikan skin vertex pada sendi tertentu. Yang termasuk dalam sendi siku ditampilkan dalam warna hitam; Bawah: Menguliti lembut dapat memadukan pengaruh beberapa sendi. Bobot untuk sendi siku ditampilkan (lebih ringan = bobot lebih besar). Perhatikan deformasi kulit yang lebih halus dari bagian dalam kulit di dekat sendi.

Ketika kerangka mengubah posisinya, ia bertindak sebagai jenis deformer khusus yang diterapkan pada kulit karakter. Gerakan ditransfer ke permukaan ini dengan menetapkan setiap simpul kulit satu (menguliti kaku) atau lebih (menguliti halus) sendi sebagai penggerak (lihat Gambar 16.20). Dalam kasus pertama, sebuah simpul kulit hanya dibekukan ke dalam ruang lokal dari sambungan yang sesuai, yang dapat menjadi yang terdekat dalam ruang atau

yang dipilih langsung oleh pengguna. Titik tersebut kemudian mengulangi gerakan apa pun yang dialami bersama ini, dan posisinya dalam koordinat dunia ditentukan oleh prosedur standar FK. Meskipun sederhana, skinning yang kaku membuat sulit untuk mendapatkan deformasi kulit yang cukup halus di area dekat sendi atau juga untuk efek yang lebih halus seperti pernapasan atau otot. Deformer khusus tambahan yang disebut fleksor dapat digunakan untuk tujuan ini. Dalam smoothskinning, beberapa sendi dapat mempengaruhi verteks kulit menurut beberapa bobot yang diberikan oleh animator, memberikan kontrol yang lebih rinci atas hasil. Vektor perpindahan, di, disarankan oleh sendi yang berbeda yang mempengaruhi verteks kulit yang diberikan (masing-masing dihitung lagi dengan FK standar) dirata-ratakan menurut bobotnya dengan hitung perpindahan akhir dari simpul $d = \sum w_i d_i$. Bobot yang dinormalisasi ($\sum wi = 1$) adalah yang paling umum tetapi tidak diperlukan secara mendasar. Menetapkan bobot skinning halus untuk mencapai efek yang diinginkan tidaklah mudah dan membutuhkan keterampilan yang signifikan dari animator.

16.11 ANIMASI WAJAH

Kerangka sangat cocok untuk membuat sebagian besar gerakan tubuh karakter, tetapi kerangka tersebut sangat tidak nyaman untuk animasi wajah yang realistis. Alasannya adalah bahwa kulit wajah manusia digerakkan oleh otot-otot yang langsung melekat padanya, bertentangan dengan bagian tubuh lain di mana tujuan utama otot adalah untuk menggerakkan tulang-tulang kerangka dan setiap deformasi kulit adalah akibat sekunder. Hasil penataan anatomi wajah ini merupakan kumpulan ekspresi wajah dinamis yang sangat kaya yang digunakan manusia sebagai salah satu instrumen utama komunikasi. Kita semua sangat terlatih untuk mengenali variasi wajah seperti itu dan dapat dengan mudah memperhatikan penampilan alami yang tidak wajar. Ini tidak hanya memberikan tuntutan khusus pada animator tetapi juga membutuhkan model geometris resolusi tinggi dari wajah dan, jika fotorealisme diinginkan, sifat dan tekstur refleksi kulit yang akurat.

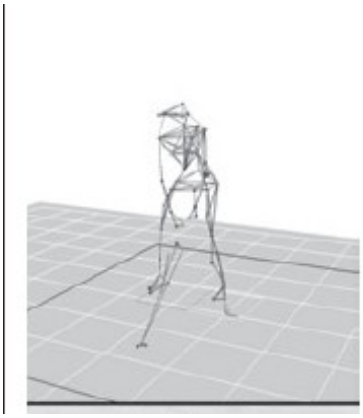
Meskipun dimungkinkan untuk mengatur pose kunci dari titik-demi-titik wajah dan interpolasi di antara mereka atau secara langsung mensimulasikan perilaku struktur otot yang mendasarinya menggunakan teknik berbasis fisika (lihat Bagian 16.5), pendekatan tingkat tinggi yang lebih khusus juga ada. Bentuk statis wajah tertentu dapat dicirikan oleh seperangkat parameter konformasi yang relatif kecil (skala keseluruhan, jarak dari mata ke dahi, panjang hidung, lebar rahang, dll.) yang digunakan untuk mengubah model wajah generik menjadi satu dengan fitur individual.

Seperangkat parameter ekspresif tambahan dapat digunakan untuk menggambarkan bentuk dinamis wajah untuk animasi. Contohnya termasuk rotasi kepala yang kaku, seberapa lebar mata terbuka, pergerakan beberapa titik fitur dari posisi statisnya, dll. Ini dipilih sehingga sebagian besar ekspresi yang menarik dapat diperoleh melalui beberapa kombinasi penyesuaian parameter, oleh karena itu, memungkinkan wajah untuk dianimasikan melalui standar bingkai kunci Untuk mencapai tingkat kontrol yang lebih tinggi, seseorang dapat menggunakan parameter ekspresif untuk membuat serangkaian ekspresi yang sesuai dengan emosi umum (netral, kesedihan, kebahagiaan, kemarahan, kejutan, dll.) dan kemudian memadukan pose kunci ini untuk mendapatkan "sedikit sedih" atau wajah "terkejut marah". Teknik serupa dapat digunakan untuk melakukan animasi lip-sync, tetapi pose kunci dalam kasus ini sesuai dengan fonem yang berbeda. Alih-alih menggunakan urutan ekspresi statis untuk menggambarkan ekspresi dinamis, Sistem Pengodean Tindakan Wajah/*Facial Action Coding System* (FACS) (Eckman & Friesen, 1978) menguraikan ekspresi wajah dinamis secara langsung menjadi sejumlah gerakan dasar yang disebut *action units* (AU). Himpunan AU didasarkan pada penelitian psikologis yang ekstensif dan mencakup gerakan-gerakan seperti mengangkat alis dalam, mengernyitkan, meregangkan bibir, dll. Menggabungkan AU dapat digunakan untuk mensintesis ekspresi yang diperlukan.

16.12 *MOTION CAPTURE*

Bahkan dengan bantuan teknik yang dijelaskan di atas, membuat animasi karakter yang tampak realistis dari awal tetap menjadi tugas yang menakutkan. Oleh karena itu wajar jika banyak perhatian diarahkan pada teknik yang merekam gerakan aktor di dunia nyata dan kemudian menerapkannya pada karakter yang dihasilkan komputer. Dua kelas utama teknik *motion capture* (MC) tersebut ada: elektromagnetik dan optik.

Dalam penangkapan gerak elektromagnetik, sensor elektromagnetik secara langsung mengukur posisinya (dan mungkin orientasinya) dalam 3D, seringkali memberikan hasil yang ditangkap secara real time. Kerugian dari teknik ini termasuk biaya peralatan yang signifikan, kemungkinan interferensi dari benda logam di dekatnya, dan ukuran sensor dan baterai yang mencolok yang dapat menjadi hambatan dalam melakukan gerakan amplitudo tinggi. Dalam MC optik, spidol berwarna kecil digunakan sebagai pengganti sensor aktif sehingga prosedurnya tidak terlalu mengganggu. Gambar 16.21 menunjukkan pengoperasian sistem tersebut. Dalam pengaturan paling dasar, gerakan direkam oleh dua kamera video yang dikalibrasi, dan triangulasi sederhana digunakan untuk mengekstrak posisi 3D penanda. Algoritma visi komputer yang lebih canggih yang digunakan untuk pelacakan akurat beberapa penanda dari video mahal secara komputasi, jadi, dalam banyak kasus, pemrosesan semacam itu dilakukan secara offline. Pelacakan optik umumnya kurang kuat daripada elektromagnetik. Oklusi penanda tertentu dalam beberapa bingkai, kemungkinan kesalahan identifikasi penanda, dan noise pada gambar hanyalah beberapa masalah umum yang harus diatasi. Memperkenalkan lebih banyak kamera untuk mengamati gerakan dari arah yang berbeda meningkatkan akurasi dan ketahanan, tetapi pendekatan ini lebih mahal dan membutuhkan waktu lebih lama untuk memproses data tersebut. MC optik menjadi lebih menarik karena daya komputasi yang tersedia meningkat dan algoritma visi komputer yang lebih baik dikembangkan. Karena sifat penanda yang berdampak rendah, metode optik cocok untuk menangkap gerakan wajah yang halus dan juga dapat digunakan dengan objek selain manusia —misalnya, hewan atau bahkan cabang pohon yang tertiup angin.



Gambar 16.21. Penangkapan gerak optik: penanda yang menempel pada tubuh pemain memungkinkan gerakan kerangka diekstraksi. ImagecourtesyofMotion Analysis Corp.

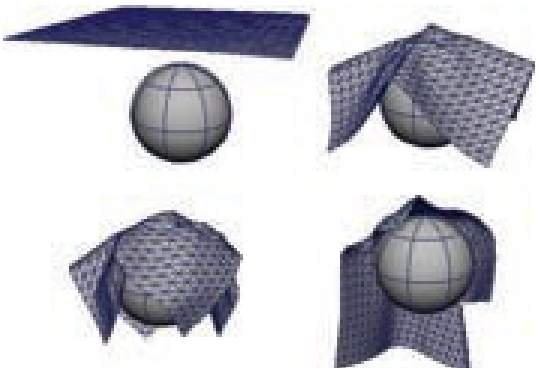
Dengan beberapa sensor atau penanda yang melekat pada tubuh pemain, satu set posisi 3D yang bergantung pada waktu dari beberapa kumpulan poin dapat direkam. Lokasi pelacakan ini biasanya merupakan persendian yang dipilih, tetapi, tentu saja, mereka masih berada di permukaan kulit dan bukan pada titik di mana tulang-tulang sebenarnya bertemu. Oleh karena itu, beberapa perawatan tambahan dan sedikit pemrosesan ekstra diperlukan untuk mengubah posisi yang direkam menjadi posisi sendi kerangka fisik. Misalnya, menempatkan dua penanda di sisi berlawanan dari siku atau pergelangan kaki memungkinkan sistem untuk mendapatkan posisi sambungan yang lebih baik dengan merata-ratakan lokasi dari dua penanda. Tanpa perawatan ekstra seperti itu, artefak yang sangat mencolok dapat

muncul karena posisi sambungan offset serta kebisingan yang melekat dan akurasi pengukuran yang tidak memadai. Karena ketidaktepatan fisik selama gerakan, misalnya, anggota badan karakter dapat kehilangan kontak dengan objek yang seharusnya mereka sentuh saat berjalan atau menggenggam, masalah seperti kaki-geser (skating) dari kerangka dapat terjadi. Sebagian besar masalah ini dapat diperbaiki dengan menggunakan teknik kinematika terbalik yang secara eksplisit dapat memaksa perilaku yang diperlukan dari ujung ekstremitas.

Posisi sendi yang dipulihkan sekarang dapat langsung diterapkan ke kerangka karakter yang dihasilkan komputer. Prosedur ini mengasumsikan bahwa dimensi fisik dari karakter tersebut identik dengan pelaku. Menargetkan ulang gerakan yang direkam ke karakter yang berbeda dan, lebih umum, mengedit data MC, memerlukan perawatan yang signifikan untuk memenuhi batasan yang diperlukan (seperti mempertahankan kaki di tanah atau tidak membiarkan siku menekuk ke belakang) dan mempertahankan tampilan alami keseluruhan dari gerakan yang dimodifikasi. Umumnya, semakin besar perubahan yang diinginkan dari aslinya, semakin kecil kemungkinan untuk mempertahankan kualitas hasilnya. Pendekatan yang menarik untuk masalah ini adalah merekam banyak koleksi gerakan dan menyatukan klip pendek dari perpustakaan ini untuk mendapatkan gerakan yang diinginkan. Meskipun topik ini saat ini merupakan bidang penelitian yang sangat aktif, keterbatasan kemampuan untuk menyesuaikan gerakan yang direkam dengan kebutuhan animator tetap menjadi salah satu kelemahan utama teknik penangkapan gerak.

16.13 ANIMASI BERBASIS FISIKA

Dunia di sekitar kita diatur oleh hukum fisika, banyak di antaranya dapat diformalkan sebagai himpunan parsial atau, dalam beberapa kasus yang lebih sederhana, persamaan diferensial biasa. Salah satu aplikasi asli komputer adalah (dan tetap) memecahkan persamaan tersebut. Oleh karena itu wajar untuk mencoba menggunakan teknik numerik yang dikembangkan selama beberapa dekade terakhir untuk mendapatkan gerakan realistis untuk animasi komputer.



Gambar 16.22. Simulasi kain yang realistis sering dilakukan dengan metode berbasis fisika. Dalam contoh ini, gaya-gaya menyebabkan tumbukan dan gravitasi.

Karena kompleksitas relatif dan biaya yang signifikan, animasi berbasis fisika paling sering digunakan dalam situasi ketika teknik lain tidak tersedia atau tidak menghasilkan hasil yang cukup realistis. Contoh utama termasuk animasi fluida (yang mencakup banyak fenomena fase gas yang dijelaskan oleh persamaan yang sama—asap, awan, api, dll.), simulasi kain (contoh ditunjukkan pada Gambar 16.22), gerakan benda tegar, dan deformasi akurat dari elastisitas objek. Persamaan yang mengatur dan detail dari pendekatan numerik yang umum digunakan berbeda dalam setiap kasus ini, tetapi banyak ide dan kesulitan mendasar tetap dapat diterapkan di seluruh aplikasi. Ada banyak metode untuk memecahkan ODE dan PDE secara numerik, tetapi membahasnya secara rinci jauh di luar cakupan buku

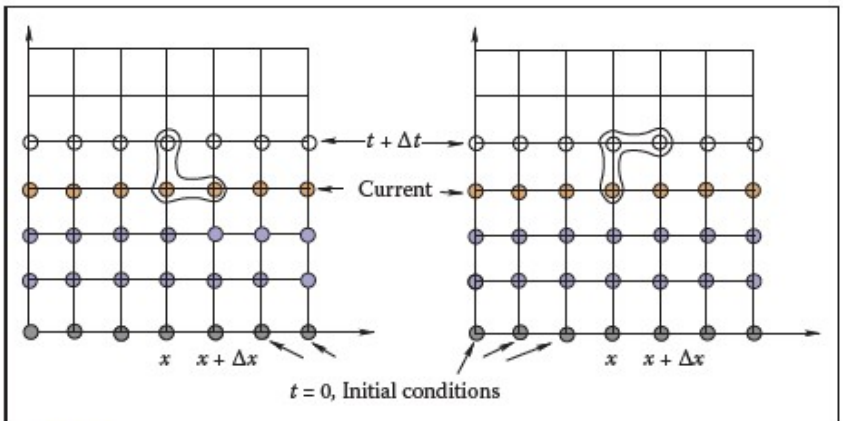
ini. Untuk memberi pembaca rasa teknik berbasis fisika dan beberapa masalah yang terlibat, kami akan menyebutkan secara singkat di sini hanya pendekatan perbedaan hingga — salah satu keluarga algoritma yang paling sederhana dan paling populer secara konseptual yang telah diterapkan pada sebagian besar, jika tidak semua , persamaan diferensial yang ditemui dalam animasi.

Ide kunci dari pendekatan ini adalah mengganti persamaan diferensial dengan analog diskritnya—persamaan selisih. Untuk melakukan ini, domain minat yang berkelanjutan diwakili oleh satu set titik yang terbatas di mana solusi akan dihitung. Dalam kasus yang paling sederhana, kisi-kisi persegi panjang seragam yang ditunjukkan pada Gambar 16.23. Setiap turunan yang ada dalam ODE atau PDE asli kemudian diganti dengan aproksimasinya melalui nilai fungsi pada titik-titik grid. Salah satu cara untuk melakukannya adalah dengan mengurangi nilai fungsi pada titik tertentu dari nilai fungsi untuk titik tetangganya pada kisi:

Persamaan 16.2

$$\frac{df(t)}{dt} \approx \frac{\Delta f}{\Delta t} = \frac{f(t+\Delta t) - f(t)}{\Delta t} \text{ or } \frac{\partial f(x,t)}{\partial x} \approx \frac{\Delta f}{\Delta x} = \frac{f(x+\Delta x, t) - f(x,t)}{\Delta x}$$

Ekspresi ini, tentu saja, bukan satu-satunya cara. Seseorang dapat, misalnya, menggunakan f(t−Δt) sebagai ganti f(t) di atas dan membaginya dengan 2Δt. Untuk persamaan yang mengandung turunan waktu, sekarang dimungkinkan untuk menyebarkan nilai fungsi yang tidak diketahui ke depan dalam urutan langkah ukuran t dengan memecahkan sistem persamaan perbedaan (satu di setiap lokasi spasial) untuk f(t +Δ t yang tidak diketahui). Beberapa kondisi awal, yaitu nilai dari fungsi yang tidak diketahui att = 0, diperlukan untuk memulai proses. Informasi lain, seperti nilai pada batas domain, mungkin juga diperlukan tergantung pada masalah spesifik.



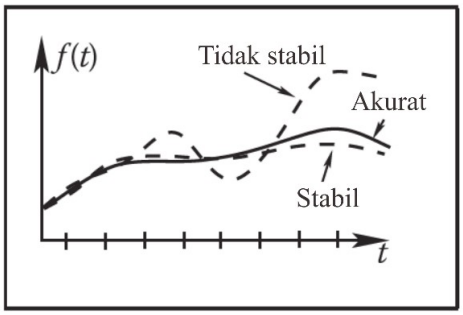
Gambar 16.23. Dua skema perbedaan yang mungkin untuk persamaan yang melibatkan turunan f/∂x dan f/∂t.

(Kiri) Skema Aneksplisit mengungkapkan nilai yang tidak diketahui (lingkaran terbuka) hanya melalui nilai yang diketahui pada saat ini (lingkaran oranye) dan mungkin masa lalu (lingkaran biru); (Kanan) Skema implisit mencampur nilai yang diketahui dan tidak diketahui dalam satu persamaan sehingga perlu untuk menyelesaikan semua persamaan tersebut sebagai suatu sistem. Untuk kedua skema, informasi tentang nilai pada batas kanan diperlukan untuk menutup proses

Perhitungan (t+Δt) dapat dilakukan dengan mudah untuk apa yang disebut skema eksplisit ketika semua nilai lain yang ada diambil pada saat ini dan satu-satunya yang tidak diketahui dalam persamaan perbedaan yang sesuai f(t +Δ t) dinyatakan melalui nilai-nilai yang diketahui ini. Skema implisit mencampur nilai-nilai saat ini dan masa depan dan mungkin menggunakan, misalnya,

$$\frac{f(x+\Delta x, t+\Delta t) - f(x, t+\Delta t)}{\Delta x}$$

sebagai aproksimasi dari $\frac{\partial f}{\partial x}$. Dalam hal ini kita harus memecahkan sistem persamaan aljabar pada setiap langkah.



Gambar 16.24. Solusi yang tidak stabil mungkin mengikuti solusi yang tepat pada awalnya, tetapi dapat menyimpang jauh dari waktu ke waktu. Akurasi solusi yang stabil mungkin masih tidak cukup untuk aplikasi tertentu.

Pilihan skema perbedaan dapat secara dramatis mempengaruhi semua aspek dari algoritma. Yang paling jelas di antara mereka adalah akurasi. Dalam limit $\Delta t \rightarrow 0$ atau $\Delta x \rightarrow 0$, ekspresi dari tipe dalam Persamaan (16.2) adalah eksak, tetapi untuk ukuran langkah hingga beberapa skema memungkinkan aproksimasi turunan yang lebih baik daripada yang lain. Stabilitas skema perbedaan terkait dengan seberapa cepat kesalahan numerik, yang selalu ada dalam praktik, dapat tumbuh seiring waktu. Untuk skema yang stabil, pertumbuhan ini dibatasi, sedangkan untuk skema yang tidak stabil bersifat eksponensial dan dapat dengan cepat mengatasi solusi yang dicari (lihat Gambar 16.24). Penting untuk disadari bahwa sementara beberapa ketidaktepatan dalam solusi yang dapat ditoleransi (dan, pada kenyataannya, akurasi yang dibutuhkan dalam fisika dan teknik jarang diperlukan untuk animasi), hasil yang tidak stabil sama sekali tidak berarti, dan orang harus menghindari penggunaan skema yang tidak stabil. Umumnya, skema eksplisit tidak stabil atau bisa menjadi tidak stabil pada ukuran langkah yang lebih besar sementara skema implisit stabil tanpa syarat. Skema implisit memungkinkan ukuran langkah yang lebih besar (dan, oleh karena itu, langkah lebih sedikit) itulah sebabnya mengapa mereka populer meskipun kebutuhan untuk memecahkan sistem persamaan aljabar pada setiap langkah. Skema eksplisit menarik karena kesederhanaannya jika kondisi stabilitasnya dapat dipenuhi. Mengembangkan skema perbedaan yang baik dan algoritma yang sesuai untuk masalah tertentu tidak mudah, dan untuk kebanyakan situasi standar disarankan untuk menggunakan metode yang ada. Banyak literatur yang membahas rincian teknik ini tersedia.

Seseorang harus ingat bahwa, dalam banyak kasus, menghitung semua suku yang diperlukan dalam persamaan saja merupakan tugas yang sulit dan memakan waktu. Dalam simulasi benda kaku atau kain, misalnya, sebagian besar gaya yang bekerja pada sistem disebabkan oleh tumbukan antar benda. Oleh karena itu, pada setiap langkah selama animasi, seseorang harus memecahkan masalah deteksi tabrakan yang murni geometris, tetapi sangat nontrivial. Dalam kondisi seperti itu, skema yang memerlukan evaluasi lebih sedikit dari kekuatan tersebut mungkin memberikan penghematan komputasi yang signifikan.

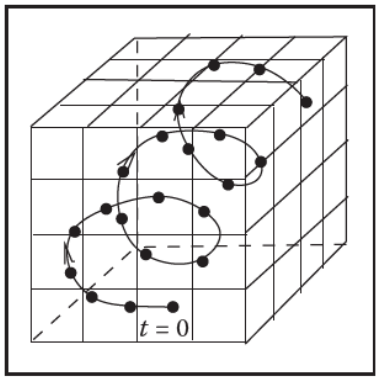
Meskipun hasil penyelesaian persamaan bergantung waktu yang tepat memberikan gerak yang sangat realistis, pendekatan ini memiliki keterbatasan. Pertama-tama, sangat sulit untuk mengontrol hasil animasi berbasis fisika. Sifat-sifat matematis dasar dari persamaan-persamaan tersebut menyatakan bahwa setelah kondisi awal disetel ulang, solusinya didefinisikan secara unik. Ini tidak meninggalkan banyak ruang untuk input animasi dan, jika

hasilnya tidak memuaskan karena suatu alasan, seseorang hanya memiliki beberapa pilihan. Mereka sebagian besar terbatas pada menyesuaikan kondisi awal yang digunakan, mengubah sifat fisik sistem, atau bahkan memodifikasi persamaan itu sendiri dengan memperkenalkan istilah buatan yang dimaksudkan untuk "menggerakkan" solusi ke arah yang diinginkan animator. Membuat perubahan seperti itu membutuhkan keterampilan yang signifikan serta pemahaman tentang fisika yang mendasarinya dan, idealnya, metode numerik. Tanpa pengetahuan ini, realisme yang disediakan oleh animasi berbasis fisika dapat dihancurkan atau masalah numerik yang parah mungkin muncul.

16.14 TEKNIK PROSEDURAL

Bayangkan bahwa seseorang dapat menulis (dan mengimplementasikan komputer) fungsi matematika yang menghasilkan gerakan yang diinginkan dengan tepat dengan beberapa panduan animator. Teknik berbasis fisika yang diuraikan di atas dapat diperlakukan sebagai kasus khusus dari pendekatan seperti itu ketika "fungsi" yang terlibat adalah prosedur untuk menyelesaikan persamaan diferensial tertentu dan "panduan" adalah himpunan kondisi awal dan batas, suku persamaan ekstra, dll.

Namun, jika kita hanya mementingkan hasil akhir, kita tidak harus mengikuti pendekatan berbasis fisika. Misalnya, gelombang amplitudo konstan sederhana di permukaan danau dapat langsung dibuat dengan menerapkan fungsi $f(x, t) = A\cos(\omega t - kx + \emptyset)$ dengan frekuensi konstan ω, vektor gelombang k dan fase untuk mendapatkan perpindahan pada titik 2D x pada waktu t. Kumpulan gelombang seperti itu dengan fase acak dan sesuai dengan amplitudo terpilih, frekuensi, dan vektor gelombang dapat menghasilkan animasi permukaan air yang sangat realistis tanpa secara eksplisit menyelesaikan persamaan dinamika fluida. Ternyata fungsi matematika lain yang agak sederhana juga dapat membuat pola atau objek yang sangat menarik. Beberapa fungsi tersebut, sebagian besar didasarkan pada suara kisi, telah dijelaskan dalam Bagian 11.5. Menambahkan ketergantungan waktu pada fungsi-fungsi ini memungkinkan untuk menghidupkan fenomena kompleks tertentu yang jauh lebih murah dan lebih murah daripada dengan teknik berbasis fisika sambil mempertahankan kualitas visual yang sangat tinggi dari hasil. Jika noise(x) adalah fungsi pembangkit pola yang mendasari, seseorang dapat membuat varian yang bergantung waktu dengan memindahkan posisi argumen melalui kisi. Kasus paling sederhana adalah gerakan dengan kecepatan konstan: *timenoise(x, t) = noise(x + vt)*, *tetapi* gerakan yang lebih kompleks melalui kisi, tentu saja, juga mungkin dan, pada kenyataannya, lebih umum. Salah satu jalur tersebut, spiral, ditunjukkan pada Gambar 16.25. Pendekatan lain adalah dengan menganimasikan parameter yang digunakan untuk menghasilkan fungsi noise. Hal ini sangat tepat jika penampakannya berubah secara signifikan seiring waktu—awan menjadi lebih bergejolak, misalnya. Dengan cara ini seseorang dapat menghidupkan proses dinamis pembentukan awan menggunakan fungsi yang menghasilkan yang statis.

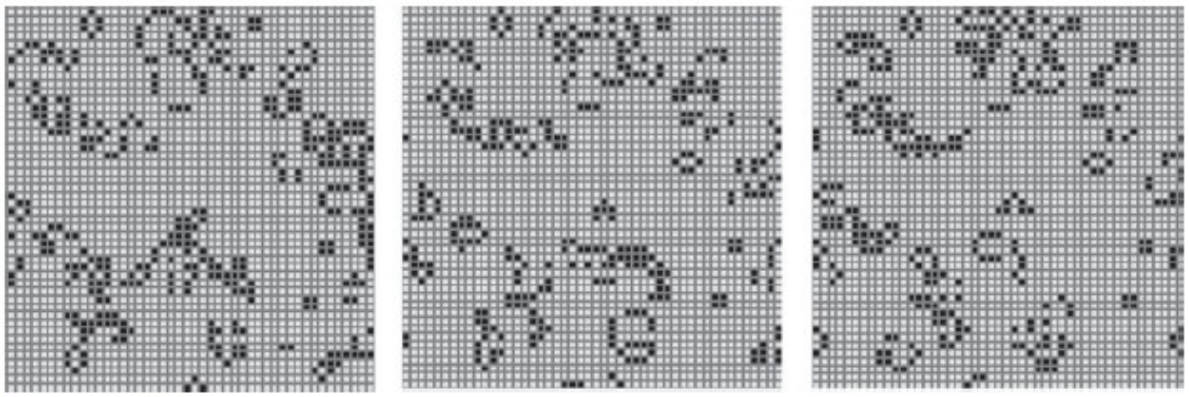


Gambar 16.25. Sebuah jalan melalui kebisingan prosedural mendefinisikan kubus dilalui untuk menghidupkan pola yang dihasilkan.

Untuk beberapa teknik prosedural, ketergantungan waktu adalah komponen yang lebih integral. Automata seluler paling sederhana beroperasi pada kotak persegi panjang 2D di mana nilai biner disimpan di setiap lokasi (sel). Untuk membuat pola waktu yang bervariasi, beberapa aturan yang disediakan pengguna untuk memodifikasi nilai-nilai ini diterapkan berulang kali. Aturan biasanya melibatkan beberapa rangkaian kondisi pada nilai saat ini dan nilai tetangga sel. Misalnya, aturan otomatis seluler Game of Life 2D yang populer yang ditemukan pada tahun 1970 oleh matematikawan Inggris John Conway adalah sebagai berikut:

1. Sel mati (yaitu, nilai biner pada lokasi tertentu adalah 0) dengan tepat tiga tetangga hidup menjadi sel hidup (yaitu, nilainya disetel ke 1).
2. Sel hidup dengan dua atau tiga tetangga hidup tetap hidup.
3. Dalam semua kasus lain, sel mati atau tetap mati.

Setelah aturan diterapkan ke semua lokasi grid, pola baru dibuat dan siklus evolusi baru dapat dimulai. Tiga cuplikan sampel dari distribusi sel hidup pada waktu yang berbeda ditunjukkan pada Gambar 16.26. Lebih canggih secara otomatis beroperasi pada beberapa grid 3D dari kemungkinan nilai titik mengambang dan dapat digunakan untuk pemodelan dinamika awan dan fenomena gas lainnya atau sistem biologis yang peralatan ini awalnya diciptakan (perhatikan terminologi). Kompleksitas pola yang mengejutkan dapat muncul dari hanya beberapa aturan yang dipilih dengan baik, tetapi cara menulis aturan tersebut untuk menciptakan perilaku yang diinginkan seringkali tidak jelas. Ini adalah masalah umum dengan teknik prosedural: hanya terbatas, jika ada, panduan tentang cara membuat prosedur baru atau bahkan menyesuaikan parameter yang sudah ada. Oleh karena itu, banyak penyesuaian dan pembelajaran dengan coba-coba (“berdasarkan pengalaman”) biasanya diperlukan untuk membuka potensi penuh metode prosedural.



Gambar 16.26. Beberapa tahap (tidak berurutan) dalam evolusi robot Game of Life.

Hidup sel ditampilkan dalam warna hitam. Objek stabil, osilator, pola perjalanan, dan banyak lagi yang menarik konstruksi dapat dihasilkan dari penerapan aturan yang sangat sederhana. Gambar dibuat menggunakan program oleh

16.15 ALAN HENSEL

Pendekatan menarik lainnya yang juga awalnya dikembangkan untuk menggambarkan objek biologis adalah teknik yang disebut sistem-L (menurut nama penemu aslinya, Astrid Lindenmayer). Pendekatan ini didasarkan pada kumpulan tata bahasa dari aturan rekursif untuk menulis ulang string simbol. Ada dua jenis simbol: simbol terminal berdiri untuk elemen dari sesuatu yang ingin kita wakili dengan tata bahasa. Bergantung pada artinya, tata bahasa dapat menggambarkan struktur pohon dan semak, bangunan dan seluruh kota, atau pemrograman dan bahasa alami. Animasi, L-sistem paling populer untuk mewakili tanaman

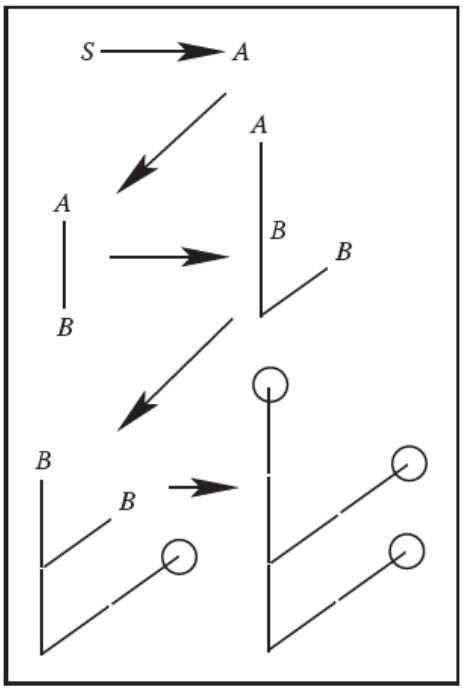
dan terminal yang sesuai adalah instruksi untuk sistem pemodelan geometris: letakkan daun (atau cabang) pada posisi saat ini—kita akan menggunakan simbol @ dan hanya menggambar lingkaran, memindahkan posisi saat ini ke depan dengan beberapa jumlah unit (simbol f), putar arah arus 60 derajat mengelilingi sumbu Z dunia (simbol +), pop (simbol [) atau push (simbol]) posisi/orientasi saat ini, dll. Simbol nonterminal tambahan (dilambangkan dengan huruf kapital) hanya memiliki semantik daripada arti langsung apa pun. Mereka dimaksudkan untuk akhirnya ditulis ulang melalui terminal. Kami mulai dari simbol awal nonterminal khusus S dan terus menerapkan aturan tata bahasa ke string saat ini secara paralel, yaitu, mengganti semua nonterminal yang saat ini ada untuk mendapatkan string baru, sampai kami berakhir dengan string yang hanya berisi terminal dan karena itu tidak ada lagi substitusi yang mungkin. . Rangkaian instruksi pemodelan ini kemudian digunakan untuk menampilkan geometri yang sebenarnya. Misalnya, seperangkat aturan (produksi)

$$S \rightarrow A$$
$$A \rightarrow [+B]fA$$
$$A \rightarrow B$$
$$B \rightarrow fB$$
$$B \rightarrow f@$$

mungkin menghasilkan urutan langkah penulisan ulang berikut yang ditunjukkan pada Gambar 16.27:

$$S \_\!\!\rightarrow A \_\!\!\rightarrow [+B]fA \_\!\!\rightarrow [+fB]f[+B]fA \_\!\!\rightarrow$$
$$[+ff@]f[+fB]fB \_\!\!\rightarrow [+ff@]f[+ff@]ff@.$$

Seperti yang ditunjukkan di atas, biasanya ada banyak produksi berbeda untuk terminal yang sama yang memungkinkan pembuatan banyak objek berbeda dengan tata bahasa yang sama. Pilihan aturan mana yang akan diterapkan dapat bergantung pada simbol mana yang terletak di sebelah simbol yang diganti (sensitivitas konteks) atau dapat dilakukan secara acak dengan beberapa probabilitas yang ditetapkan untuk setiap aturan (sistem L stokastik). Aturan yang lebih kompleks dapat memodelkan interaksi dengan lingkungan, seperti pemangkasan ke bentuk tertentu, dan parameter dapat dikaitkan dengan simbol untuk mengontrol perintah geometris yang dikeluarkan.



Gambar 16.27. Langkah-langkah derivasi berturut-turut menggunakan sistem-L sederhana. Huruf kapital menunjukkan nonterminal dan menggambarkan posisi di mana nonterminal yang sesuai akan diperluas. Mereka bukan bagian dari output yang sebenarnya.

L-sistem sudah menangkap perubahan topologi tanaman dengan waktu: setiap string perantara yang diperoleh dalam proses penulisan ulang dapat ditafsirkan sebagai versi "lebih muda" dari tanaman (lihat Gambar 16.27). Untuk perubahan yang lebih signifikan, produksi yang berbeda dapat berlaku pada waktu yang berbeda yang memungkinkan struktur tanaman berubah secara signifikan seiring pertumbuhannya. Pohon muda, misalnya, menghasilkan banyak cabang baru, sedangkan pohon yang lebih tua hanya bercabang sedikit.

Model tanaman yang sangat realistis telah dibuat dengan sistem-L. Namun, seperti kebanyakan teknik prosedural, seseorang memerlukan pengalaman untuk menerapkan sistem-L yang ada secara bermakna, dan menulis tata bahasa baru untuk menangkap beberapa efek yang diinginkan tentu tidak mudah.

16.16 GRUP OBJEK

Untuk menganimasikan banyak objek, tentu saja, cukup menerapkan teknik standar yang dijelaskan dalam bab sejauh ini untuk masing-masing objek tersebut. Ini bekerja cukup baik untuk sejumlah moderat objek independen yang gerakan yang diinginkan diketahui sebelumnya. Namun, dalam banyak kasus, beberapa jenis tindakan terkoordinasi dalam lingkungan yang dinamis diperlukan. Jika hanya beberapa objek yang terlibat, animator dapat menggunakan sistem berbasis kecerdasan buatan (AI) untuk secara otomatis menentukan tugas segera untuk setiap objek berdasarkan beberapa tujuan tingkat tinggi, merencanakan gerakan yang diperlukan, dan melaksanakan rencana tersebut. Banyak game modern menggunakan objek otonom semacam itu untuk membuat monster pintar atau kolaborator pemain.

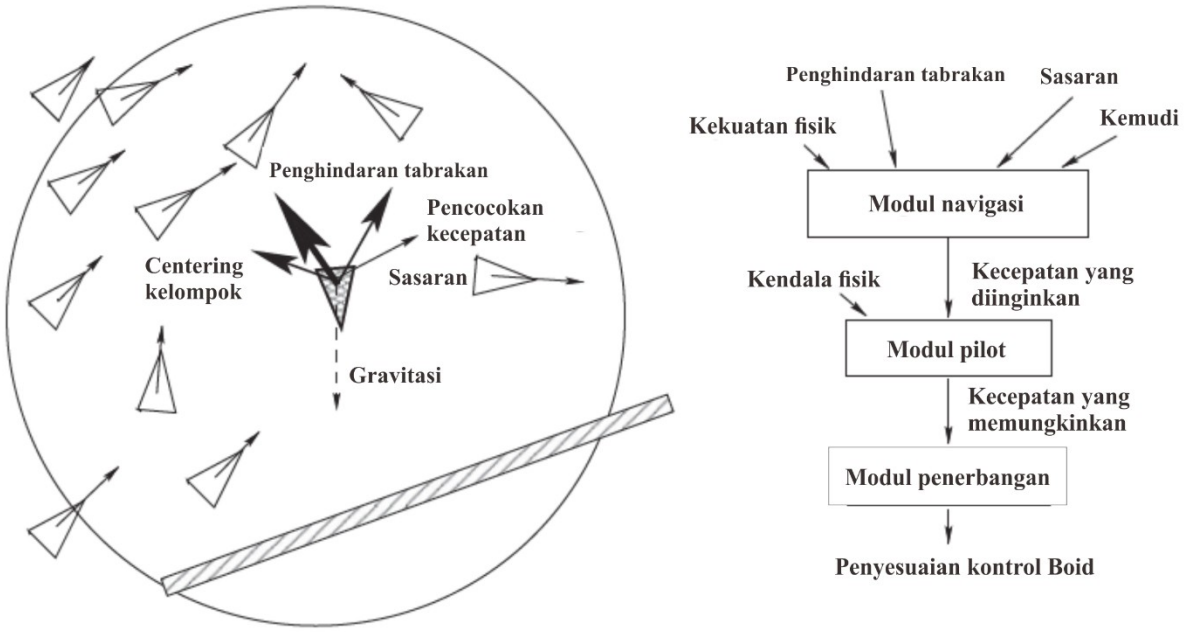
Menariknya, karena jumlah objek dalam suatu kelompok bertambah dari hanya beberapa menjadi beberapa lusin, ratusan, dan ribuan, anggota individu dari suatu kelompok harus memiliki “kecerdasan” yang sangat terbatas agar kelompok secara keseluruhan dapat menunjukkan apa yang tampak seperti gerakan yang digerakkan oleh tujuan yang terkoordinasi. Ternyata pengelompokan ini adalah perilaku yang muncul yang dapat muncul sebagai akibat dari interaksi terbatas anggota kelompok dengan hanya beberapa tetangga terdekat mereka (Reynolds, 1987). Berkelompok harus akrab bagi siapa saja yang telah mengamati gerakan sinkron yang menakjubkan dari kawanan burung atau sekawanan ikan. Teknik ini juga dapat digunakan untuk mengendalikan sekelompok hewan yang bergerak di atas medan atau bahkan kerumunan manusia.

Pada saat tertentu, gerakan anggota kelompok, sering disebut boid bila diterapkan pada flok, adalah hasil dari penyeimbangan beberapa kecenderungan yang sering bertentangan, yang masing-masing menunjukkan vektor kecepatannya sendiri (lihat Gambar 16.28). Pertama, ada gaya fisik eksternal F yang bekerja pada balok, seperti gravitasi atau angin. Kecepatan baru akibat gaya tersebut dapat dihitung secara langsung melalui hukum Newton sebagai

$$\mathbf{v}^{physics}_{new} = \mathbf{v}_{old} + \mathbf{F}\Delta t/m.$$

Kedua, seorang boid harus bereaksi terhadap lingkungan global dan perilaku anggota kelompok lainnya. Penghindaran tabrakan adalah salah satu hasil utama dari interaksi tersebut. Sangat penting untuk mengelompokkan bahwa setiap anggota kelompok hanya memiliki bidang pandang yang terbatas, dan karena itu hanya menyadari hal-hal yang terjadi dalam beberapa lingkungan dari posisinya saat ini. Untuk menghindari objek dalam lingkungan, strategi yang paling sederhana dan tidak sempurna adalah dengan mengatur medan gaya tolak yang terbatas di sekitar setiap objek tersebut. Ini akan membuat vektor kecepatan kedua yang diinginkan $v^{col\_avoid}_{new}$, juga diberikan oleh hukum Newton. Interaksi dengan anggota kelompok lain dapat dimodelkan dengan secara simultan menerapkan perilaku kemudi yang berbeda yang menghasilkan beberapa vektor kecepatan tambahan yang diinginkan v^{steer}_{new}. Menjauh dari tetangga untuk menghindari kerumunan, mengarahkan ke

arah kawanan untuk memastikan kohesi, dan menyesuaikan kecepatan agar sejajar dengan arah depan tetangga adalah hal yang paling umum. Akhirnya, beberapa vektor kecepatan tambahan yang diinginkan v^{goal}_{new} biasanya diterapkan untuk mencapai tujuan global yang dibutuhkan. Ini dapat berupa vektor-vektor di sepanjang suatu jalur di ruang angkasa, mengikuti beberapa pemimpin kawanan yang ditentukan secara spesifik, atau sekadar mewakili dorongan migrasi dari anggota kawanan.

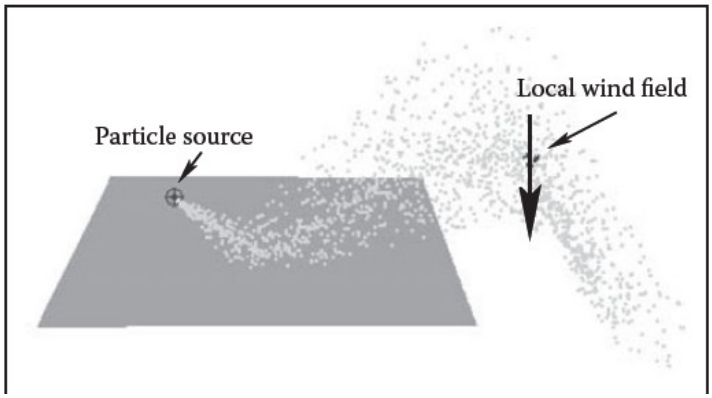


Gambar 16.28. (Kiri) Anggota kelompok individu (boid) dapat mengalami beberapa desakan dengan kepentingan yang berbeda (ditunjukkan oleh ketebalan garis) yang harus dinegosiasikan menjadi satu vektor kecepatan. Aboid hanya menyadari lingkungannya yang terbatas (lingkaran). (Kanan) Kontrol Boid biasanya diimplementasikan sebagai tiga modul terpisah.

Setelah semua v baru ditentukan, vektor akhir yang diinginkan dinegosiasikan berdasarkan prioritas di antara mereka. Penghindaran tabrakan dan pencocokan kecepatan biasanya memiliki prioritas yang lebih tinggi. Alih-alih rata-rata sederhana dari vektor kecepatan yang diinginkan yang dapat menyebabkan pembatalan dorongan dan perilaku "bergerak ke mana-mana" yang tidak wajar, strategi alokasi percepatan digunakan. Beberapa jumlah tetap percepatan disediakan untuk sebuah boid dan sebagian kecil dari itu diberikan kepada setiap dorongan dalam urutan prioritas. Jika total akselerasi yang tersedia habis, beberapa dorongan prioritas yang lebih rendah akan memiliki efek yang lebih kecil pada gerakan atau diabaikan sama sekali. Harapannya adalah bahwa begitu tugas yang paling penting saat ini (penghindaran tabrakan di sebagian besar situasi) diselesaikan, tugas-tugas lain dapat diselesaikan dalam waktu dekat. Penting juga untuk menghormati beberapa batasan fisik objek nyata, misalnya, menjepit akselerasi atau kecepatan yang terlalu tinggi ke beberapa nilai realistis. Bergantung pada kerumitan internal anggota kawanan, tahap akhir animasi mungkin adalah mengubah vektor kecepatan yang dinegosiasikan menjadi seperangkat parameter tertentu (posisi ayunan burung, orientasi model bidang dalam ruang, konfigurasi tulang kerangka kaki) yang digunakan untuk mengontrol gerakan bola. Diagram implementasi sistem ocking ditunjukkan pada Gambar 16.28 (kanan).

Jauh lebih sederhana, tetapi masih sangat berguna, versi kontrol grup diimplementasikan oleh sistem partikel (Reeves, 1983). Jumlah partikel dalam suatu sistem biasanya jauh lebih besar daripada jumlah partikel dalam satu kawanan dan bisa mencapai puluhan atau ratusan ribu, atau bahkan lebih. Selain itu, jumlah partikel yang tepat dapat berfluktuasi selama animasi dengan partikel baru lahir dan beberapa partikel lama

dihancurkan pada setiap langkah. Partikel biasanya benar-benar independen satu sama lain, mengabaikan tetangganya dan berinteraksi dengan lingkungan hanya dengan mengalami gaya eksternal dan tabrakan dengan objek, bukan melalui penghindaran tabrakan seperti yang terjadi pada kawanan. Pada setiap langkah selama animasi, sistem pertama-tama menciptakan partikel baru dengan beberapa parameter awal, mengakhiri yang lama, dan kemudian menghitung gaya yang diperlukan dan memperbarui kecepatan dan posisi partikel yang tersisa menurut hukum Newton.



Gambar 16.29. Setelah dipancarkan oleh sumber terarah, partikel bertabrakan dengan objek dan kemudian ditiup ke bawah oleh medan angin lokal setelah mereka melewati penghalang.

Semua parameter sistem partikel (jumlah partikel, rentang hidup partikel, kecepatan awal, dan lokasi partikel, dll.) biasanya di bawah kendali langsung animator. Aplikasi utama dari sistem partikel meliputi pemodelan kembang api, ledakan, penyemprotan cairan, asap dan api, atau objek dan fenomena kabur lainnya tanpa batas yang tajam. Untuk mencapai tampilan yang realistis, penting untuk memperkenalkan beberapa keacakan ke semua parameter, misalnya, memiliki jumlah partikel yang lahir (dan dihancurkan) secara acak di setiap langkah dengan kecepatan yang dihasilkan menurut beberapa distribusi. Selain menetapkan parameter awal yang sesuai, mengontrol gerakan sistem partikel biasanya dilakukan dengan membuat pola gaya tertentu di ruang angkasa—meniup partikel ke arah baru setelah mencapai beberapa lokasi tertentu atau menambahkan pusat tarik-menarik, misalnya. Kita harus ingat bahwa dengan segala kelebihannya, kesederhanaan implementasi dan kemudahan kontrol menjadi yang utama, sistem partikel biasanya tidak memberikan tingkat karakteristik realisme simulasi berbasis fisika sebenarnya dari fenomena yang sama.

16.17 CATATAN

Dalam bab ini kita telah berkonsentrasi pada teknik yang digunakan dalam animasi 3D. Ada juga serangkaian algoritme yang kaya untuk membantu produksi Animasi 2d dan pemrosesan pasca gambar yang dibuat oleh sistem rendering grafis komputer. Ini termasuk teknik untuk membersihkan gambar artis yang dipindai, ekstraksi fitur, antara 2Din otomatis, pewarnaan, warping gambar, peningkatan dan pengomposisian, dan banyak lainnya.

Salah satu perkembangan paling signifikan di bidang animasi komputer adalah meningkatnya kekuatan dan ketersediaan sistem animasi yang canggih. Meskipun berbeda dalam rangkaian fitur, struktur internal, detail antarmuka pengguna, dan harga, sebagian besar sistem tersebut menyertakan dukungan ekstensif tidak hanya untuk animasi, tetapi juga untuk pemodelan dan rendering, mengubahnya menjadi platform produksi yang lengkap. Juga umum untuk menggunakan sistem ini untuk membuat gambar diam. Misalnya, banyak gambar untuk gambar di bagian ini diproduksi menggunakan perangkat lunak Maya yang banyak disumbangkan oleh Alias.

Produksi animasi skala besar adalah proses yang sangat kompleks yang biasanya melibatkan upaya gabungan oleh lusinan orang dengan latar belakang berbeda yang tersebar di banyak departemen atau bahkan perusahaan. Untuk mengoordinasikan aktivitas ini dengan lebih baik, jalur produksi tertentu dibuat yang dimulai dengan sketsa cerita dan karakter, dilanjutkan dengan merekam suara yang diperlukan, membangun model, dan karakter rig untuk animasi. Setelah animasi yang sebenarnya dimulai, biasanya untuk kembali dan merevisi desain, model, dan rig asli untuk memperbaiki masalah gerak dan penampilan yang ditemukan. Menyiapkan pencahayaan dan properti material kemudian diperlukan, setelah itu dimungkinkan untuk memulai rendering. Dalam sebagian besar proyek yang cukup kompleks, tahap pascapemrosesan dan pengomposisian yang ekstensif menyatukan gambar dari sumber yang berbeda dan menyelesaikan produk.

Kami menyimpulkan bab ini dengan mengingatkan pembaca bahwa di bidang animasi komputer, setiap kecanggihan teknis adalah hal sekunder dari cerita yang bagus, karakter ekspresif, dan faktor artistik lainnya, yang sebagian besar sulit atau tidak mungkin diukur. Aman untuk mengatakan bahwa Putri Salju dan tujuh kurcacinya akan selalu berbagi layar dengan raksasa hijau dan keledai, dan sebagian besar penonton akan jauh lebih tertarik pada karakter dan cerita daripada di mana, jika ada, komputer (dan dengan cara apa) membantu menciptakannya.

BAB 17
MENGGUNAKAN HARDWARE GRAFIS

17.1 GAMBARAN UMUM *HARDWARE*

Sebagian besar fokus buku ini adalah materi tentang dasar-dasar yang mendasari grafis komputer daripada pada setiap spesifik yang berkaitan dengan API atau perangkat keras tempat algoritme dapat diimplementasikan. Bab ini mengambil rute yang sedikit berbeda dan memadukan rincian penggunaan perangkat keras grafis dengan beberapa masalah praktis yang terkait dengan pemrograman perangkat keras itu. Bab ini dirancang untuk menjadi panduan pengantar untuk perangkat keras grafis dan dapat digunakan sebagai dasar untuk serangkaian lab mingguan yang menyelidiki perangkat keras grafis.

17.2 APA YANG DIMAKSUD DENGAN HARDWARE GRAFIS?

Perangkat keras grafis menjelaskan komponen perangkat keras yang diperlukan untuk dengan cepat merender3 Objek sebagai piksel pada layar komputer Anda menggunakan arsitektur perangkat keras berbasis rasterisasi khusus (dan dalam beberapa kasus, berbasis ray-tracer). Penggunaan istilah perangkat keras grafis dimaksudkan untuk memperoleh pengertian tentang komponen fisik yang diperlukan untuk melakukan berbagai komputasi grafis. Dengan kata lain, perangkat keras adalah seperangkat chipset, transistor, bus, prosesor, dan inti komputasi yang ditemukan pada kartu video saat ini. Seperti yang akan Anda pelajari dalam bab ini, dan akhirnya mengalami sendiri, perangkat keras grafis saat ini sangat baik dalam memproses deskripsi 3 Objek dan mengubah representasi tersebut menjadi piksel berwarna yang memenuhi monitor Anda.

Catatan

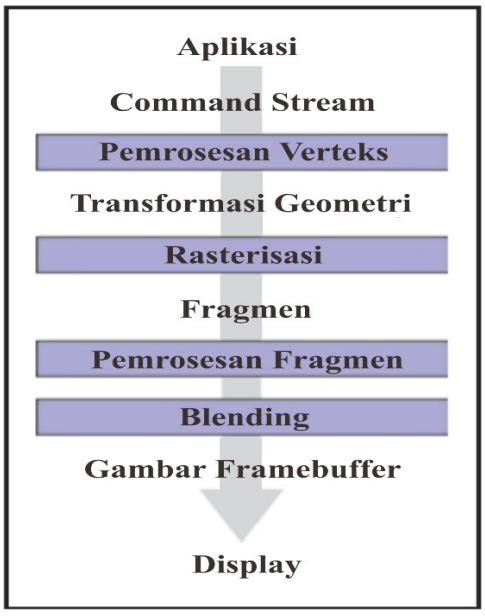
Grafis Real-Time: Dengan grafis waktu nyata, secara umum yang kami maksudkan adalah bahwa komputasi terkait grafis dilakukan cukup cepat sehingga hasilnya dapat segera dilihat. Mampu melakukan operasi pada 60Hz atau lebih tinggi dianggap waktu nyata. Setelah waktu untuk menyegarkan tampilan (kecepatan bingkai) turun di bawah 15Hz, kecepatannya dianggap lebih interaktif daripada waktu nyata, tetapi perbedaan ini tidak penting. Karena perhitungan harus cepat, persamaan yang digunakan untuk membuat grafis sering kali merupakan perkiraan untuk apa yang dapat dilakukan jika tersedia lebih banyak waktu.

Fragmen: Fragmen adalah istilah yang menggambarkan informasi yang terkait dengan piksel sebelum diproses pada tahap akhir dari jalur grafis. Definisi ini mencakup banyak data yang mungkin digunakan untuk menghitung warna piksel, seperti kedalaman scene piksel, koordinat tekstur, atau informasi stensil.

Perangkat keras grafis telah berubah sangat cepat selama dekade terakhir. Perangkat keras grafis yang lebih baru menyediakan lebih banyak kemampuan pemrosesan paralel, serta dukungan yang lebih baik untuk rendering khusus. Salah satu penjelasan untuk langkah cepat adalah industri video game dan momentum ekonominya. Intinya adalah bahwa setiap kartu grafis baru memberikan kinerja dan kemampuan pemrosesan yang lebih baik. Hasilnya, video game tampak lebih realistis secara visual. Prosesor pada perangkat keras grafis, sering disebut GPU, atau Unit Pemrosesan Grafis, sangat paralel dan menghasilkan ribuan utas eksekusi secara bersamaan. Perangkat keras dirancang untuk throughput yang memungkinkan jumlah piksel dan simpul yang lebih besar untuk diproses dalam waktu yang lebih singkat. Semua paralelisme ini bagus untuk algoritme grafis, tetapi pekerjaan lain telah diuntungkan dari perangkat keras paralel. Selain video game, GPU digunakan untuk mempercepat komputasi

fisika, mengembangkan kode ray tracing real-time, memecahkan persamaan terkait Navier-Stokes untuk simulasi aliran fluida, dan mengembangkan kode lebih cepat untuk memahami iklim (Purcell, Buck, Mark, & Hanrahan, 2002; SG Parker et al. , 2010; Haris, 2004). Beberapa API dan SDK telah dikembangkan yang memungkinkan komputasi tujuan umum langsung, seperti OpenCL dan CUDA NVIDIA. API penelusuran sinar yang dipercepat perangkat keras juga ada untuk mempercepat persimpangan objek sinar (S. G. Parker et al., 2010). Demikian pula, API standar yang digunakan untuk memprogram komponen grafis videogame, seperti OpenGL dan DirectX, juga memungkinkan mekanisme untuk memanfaatkan kemampuan paralel perangkat keras grafis. Banyak dari API ini berubah saat perangkat keras baru dikembangkan untuk mendukung komputasi yang lebih canggih.

Perangkat keras grafis dapat diprogram. Sebagai pengembang, Anda memiliki kendali atas banyak komputasi yang terkait dengan pemrosesan geometri, simpul, dan fragmen yang akhirnya menjadi piksel. Perubahan perangkat keras terbaru serta pembaruan berkelanjutan pada API, seperti OpenGL atau DirectX, mendukung saluran yang sepenuhnya dapat diprogram. Perubahan ini memberi pengembang lisensi kreatif untuk mengeksploitasi komputasi yang tersedia di GPU. Sebelum ini, pipa rasterisasi fungsi tetap memaksa perhitungan ke gaya transformasi verteks, pencahayaan, dan pemrosesan fragmen tertentu. Fungsi tetap dari pipeline memastikan bahwa pewarnaan dasar, pencahayaan, dan tekstur dapat terjadi dengan sangat cepat. Baik itu antarmuka yang dapat diprogram, atau komputasi fungsi tetap, komputasi dasar dari pipeline rasterisasi serupa, dan ikuti ilustrasi pada Gambar 17.1. Dalam pipa rasterisasi, simpul ditransformasikan dari ruang lokal ke ruang global, dan akhirnya menjadi koordinat layar, setelah ditransformasikan oleh matriks transformasi tampilan dan proyeksi. Himpunan koordinat layar yang terkait dengan simpul geometri dirasterisasi menjadi fragmen. Tahap akhir dari pipeline memproses fragmen menjadi piksel dan dapat menerapkan pencarian tekstur per-fragmen, pencahayaan, dan pencampuran yang diperlukan. Secara umum, pipeline cocok untuk eksekusi paralel dan inti GPU dapat digunakan untuk memproses simpul dan fragmen secara bersamaan. Rincian tambahan tentang pipa rasterisasi dapat ditemukan di Bab 8.



Gambar 17.1. Pipa perangkat keras grafis dasar terdiri dari tahapan yang mengubah data 3D menjadi objek layar 2D yang siap untuk rasterisasi dan pewarnaan dengan tahapan pemrosesan piksel.

17.3 MULTIPROSESOR HETEROGEN

Saat menggunakan perangkat keras grafis, akan lebih mudah untuk membedakan antara CPU dan GPU sebagai entitas komputasi yang terpisah. Dalam konteks ini, istilah host digunakan untuk merujuk ke CPU termasuk utas dan memori yang tersedia untuknya. Istilah perangkat digunakan untuk merujuk ke GPU, atau unit pemrosesan grafis, dan utas serta memori yang terkait dengannya. Ini masuk akal karena sebagian besar perangkat keras grafis terdiri dari perangkat keras eksternal yang terhubung ke mesin melalui bus PCI. Perangkat keras juga dapat disolder ke mesin sebagai chipset terpisah. Dalam pengertian ini, perangkat keras grafis mewakili co-prosesor khusus karena CPU (dan intinya) dapat diprogram, seperti halnya GPU dan intinya. Semua program yang menggunakan perangkat keras grafis harus terlebih dahulu membuat mapping antara CPU dan memori GPU. Ini adalah detail tingkat rendah yang diperlukan agar driver perangkat keras grafis yang berada di dalam sistem operasi dapat menghubungkan antara perangkat keras dan sistem operasi dan perangkat lunak sistem windowing. Ingatlah bahwa karena host (CPU) dan perangkat (GPU) terpisah, data harus dikomunikasikan antara kedua sistem. Secara lebih formal, mapping antara sistem operasi, driver perangkat keras, perangkat keras, dan sistem windowing ini dikenal sebagai konteks grafis. Konteks biasanya dibuat melalui panggilan API ke sistem windowing. Detail tentang membangun konteks berada di luar cakupan bab ini, tetapi banyak perpustakaan pengembangan sistem windowing memiliki cara untuk menanyakan perangkat keras grafis untuk berbagai kemampuan dan menetapkan konteks grafis berdasarkan persyaratan tersebut. Karena pengaturan konteks bergantung pada sistem windowing, itu juga berarti bahwa kode tersebut tidak mungkin menjadi kode lintas platform. Namun, dalam praktiknya, atau setidaknya saat memulai, sangat tidak mungkin bahwa kode pengaturan konteks tingkat rendah seperti itu akan diperlukan karena banyak API tingkat tinggi yang ada untuk membantu orang mengembangkan aplikasi interaktif portabel.

Catatan

Host: Dalam program perangkat keras grafis, host mengacu pada komponen CPU dari aplikasi.

Perangkat: Sisi GPU dari aplikasi grafis, termasuk data dan komputasi yang disimpan dan dijalankan pada GPU

Banyak kerangka kerja untuk mengembangkan aplikasi interaktif yang mendukung permintaan perangkat input seperti keyboard atau mouse. Beberapa kerangka kerja menyediakan akses ke jaringan, sistem audio, dan sumber daya sistem tingkat tinggi lainnya. Dalam hal ini, banyak dari API ini adalah cara yang lebih disukai untuk mengembangkan grafis, dan bahkan aplikasi game.

Percepatan perangkat keras lintas-platformdicapai dengan OpenGLAPI. OpenGL adalah API grafis standar industri terbuka yang mendukung akselerasi perangkat keras pada banyak jenis perangkat keras grafis. OpenGL mewakili salah satu API paling umum untuk memprogram perangkat keras grafis bersama dengan API seperti DirectX. Sementara OpenGL tersedia di banyak sistem operasi dan arsitektur perangkat keras, DirectX khusus untuk sistem berbasis Microsoft. Untuk tujuan bab ini, konsep dan contoh pemrograman perangkat keras akan disajikan dengan OpenGL.

17.4 PEMROGRAMAN DENGAN OPENGL

Saat Anda memprogram dengan OpenGL API, Anda menulis kode untuk setidaknya dua prosesor: CPU dan GPU. OpenGLisimplementedinaC-styleAPI dan semua fungsi diawali dengan "gl" untuk menunjukkan penyertaannya dengan OpenGL. Panggilan fungsi OpenGL mengubah keadaan perangkat keras grafis dan dapat digunakan untuk mendeklarasikan dan mendefinisikan geometri, memuat vertex dan fragment shader, dan menentukan bagaimana komputasi akan terjadi saat data melewati perangkat keras.

Varian OpenGL yang disajikan bab ini adalah versi OpenGL 3.3 Core Profile. Meskipun bukan versi OpenGL terbaru, OpenGL versi 3.3 sejalan dengan arah pemrograman OpenGL di masa depan. Versi-versi ini difokuskan pada peningkatan efisiensi sementara juga sepenuhnya menempatkan pemrograman saluran pipa di tangan pengembang. Banyak dari panggilan fungsi yang ada di versi OpenGL sebelumnya tidak ada di API yang lebih baru ini. Misalnya, rendering mode langsung tidak digunakan lagi. Render mode segera digunakan untuk mengirim data dari memori CPU ke memori kartu grafis sesuai kebutuhan setiap frame dan seringkali sangat tidak efisien, terutama untuk model yang lebih besar dan scene yang kompleks. API saat ini berfokus pada penyimpanan data pada kartu grafis sebelum dibutuhkan dan membuat instance pada waktu render. Sebagai contoh lain, tumpukan matriks OpenGL telah ditinggalkan juga, membiarkan pengembang menggunakan perpustakaan matriks pihak ketiga (seperti GLM) atau kelas mereka sendiri untuk membuat matriks yang diperlukan untuk melihat, proyeksi, dan transformasi, seperti yang disajikan dalam Bab 7 Akibatnya, bahasa shader OpenGL (GLSL) telah mengambil peran yang lebih besar juga, melakukan transformasi matriks yang diperlukan bersama dengan pencahayaan dan bayangan di dalam shader. Karena pipeline fungsi tetap yang melakukan transformasi per-vertex dan pencahayaan tidak lagi ada, programmer harus mengembangkan sendiri semua shader. Contoh bayangan yang disajikan dalam bab ini akan menggunakan spesifikasi shader versi GLSL 3.3 Core Profile. Pembaca di masa mendatang,bab ini ingin menjelajahi spesifikasi OpenGLandOpenGL Shading Language saat ini untuk detail tambahan tentang apa yang dapat didukung oleh API dan bahasa ini.

17.5 PEMROGRAMAN HARDWARE GRAFIS : *BUFFER, STATE* DAN *SHADER*

Tiga konsep akan membantu untuk memahami pemrograman perangkat keras grafis kontemporer. Yang pertama adalah gagasan buffer data, yang cukup sederhana, alokasi linear memori pada perangkat yang dapat menyimpan berbagai data di mana GPU akan beroperasi. Yang kedua adalah gagasan bahwa kartu grafis mempertahankan status komputasi yang menentukan bagaimana komputasi yang terkait dengan data scene dan shader akan terjadi pada perangkat keras grafis. Selain itu, status dapat dikomunikasikan dari host ke perangkat dan bahkan di dalam perangkat antara shader. Shader mewakili mekanisme komputasi yang terjadi pada GPU terkait dengan pemrosesan per-vertex atau per-fragment. Bab ini akan fokus pada vertex dan fragment shader, tetapi geometri khusus dan shader komputasi juga ada di versi OpenGL saat ini. Shader memainkan peran yang sangat penting dalam bagaimana perangkat keras grafis modern berfungsi.

Buffer

Buffer adalah struktur utama untuk menyimpan data pada perangkat keras grafis. Mereka mewakili memori internal perangkat keras grafis yang terkait dengan segala sesuatu mulai dari geometri, tekstur, dan data bidang gambar. Sehubungan dengan pipa asterisasi yang dijelaskan dalam Bab 8, komputasi yang terkait dengan rasterisasi yang dipercepat perangkat keras membaca dan menulis berbagai buffer pada GPU. Dari sudut pandang pemrograman, aplikasi harus menginisialisasi buffer pada GPU yang diperlukan untuk aplikasi tersebut. Ini sama dengan operasi penyalinan host ke perangkat. Pada akhir berbagai tahap eksekusi, salinan perangkat ke host dapat dilakukan juga untuk menarik data dari GPU ke memori CPU. Selain itu, mekanisme memang ada di API OpenGL yang memungkinkan memori perangkat dipetakan ke memori host sehingga program aplikasi dapat menulis langsung ke buffer di kartu grafis.

Display Buffer

Dalam saluran grafis, rangkaian warna piksel terakhir dapat dihubungkan ke tampilan, atau dapat ditulis ke disk sebagai gambar PNG. Data yang terkait dengan piksel ini umumnya berupa larik nilai warna 2D. Data secara inheren 2D, tetapi secara efisien direpresentasikan

pada GPU sebagai array memori linier 1D. Array ini mengimplementasikan buffer tampilan, yang akhirnya dipetakan ke windows. Rendering gambar melibatkan komunikasi perubahan ke buffer tampilan pada perangkat keras grafis melalui API grafis. Di akhir jalur rasterisasi, tahap pemrosesan fragmen dan pencampuran menulis data ke memori buffer tampilan output. Sementara itu, sistem windowing membaca isi buffer tampilan untuk menghasilkan gambar raster pada windows monitor.

Siklus Refresh

Sebagian besar aplikasi lebih menyukai keadaan tampilan buffer ganda. Artinya, ada dua buffer yang terkait dengan windows grafis: buffer depan dan buffer belakang. Tujuan dari sistem buffer ganda adalah bahwa aplikasi dapat mengkomunikasikan perubahan ke buffer belakang (dan dengan demikian, menulis perubahan ke buffer itu) sementara memori buffer depan digunakan untuk menggerakkan warna piksel pada windows.

Di akhir loop rendering, buffer ditukar melalui pertukaran pointer. Pointer buffer depan menunjuk ke buffer belakang dan pointer buffer belakang kemudian ditetapkan ke buffer depan sebelumnya. Dengan cara ini, sistem windowing akan me-refresh isi window dengan buffer terbaru. Jika swap pointer buffer disinkronkan dengan refresh sistem windowing dari seluruh tampilan, rendering akan tampak mulus. Jika tidak, pengguna dapat mengamati robekan geometri pada tampilan sebenarnya saat perubahan geometri dan fragmen scene diproses (dan dengan demikian ditulis ke buffer tampilan) lebih cepat daripada layar di-refresh.

Ketika tampilan dianggap sebagai buffer memori, salah satu operasi paling sederhana pada tampilan pada dasarnya adalah operasi pengaturan memori (atau penyalinan) yang menghilangkan, atau mengosongkan memori ke status default. Untuk program grafis, ini mungkin berarti membersihkan latar belakang windows ke warna tertentu. Untuk menghapus warna latar belakang (menjadi hitam) pada aplikasi OpenGL, dapat digunakan kode berikut:

```
glClearColor( 0.0f, 0.0f, 0.0f, 1.0f );
glClear( GL_COLOR_BUFFER_BIT );
```

Tiga argumen pertama untuk fungsi `ClearColor` merepresentasikan komponen warna hijau, dan biru, yang ditentukan dalam rentang [0,1]. Argumen keempat mewakili opacity, atau nilai alfa, mulai dari 0,0 yang benar-benar transparan hingga 1,0 yang sepenuhnya buram. Nilai alfa digunakan untuk menentukan transparansi melalui berbagai operasi pencampuran fragmen di tahap akhir jalur pipa.

Operasi ini hanya menghapus buffer warna. Selain buffer warna, yang ditentukan oleh GL `COLOR BUFFER BIT`, yang dibersihkan menjadi hitam dalam hal ini, perangkat keras grafis juga menggunakan buffer kedalaman untuk mewakili jarak fragmen relatif terhadap kamera (Anda mungkin ingat diskusi tentang algoritma z-buffer di Bab 8). Membersihkan buffer kedalaman diperlukan untuk memastikan pengoperasian algoritma zbuffer dan memungkinkan penghilangan permukaan tersembunyi yang benar terjadi. Membersihkan buffer kedalaman dapat dicapai dengan atau menggabungkan dua nilai bidang bit, sebagai berikut:

```
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
```

Dalam aplikasi grafis interaktif dasar, langkah pembersihan ini biasanya merupakan operasi pertama yang dilakukan sebelum geometri atau fragmen apa pun diproses.

Mesin State

Dengan mengilustrasikan operasi buffer-clearing untuk buffer warna dan kedalaman tampilan, gagasan tentang status perangkat keras grafis juga diperkenalkan. Fungsi `glClearColor` menyetel nilai warna default yang ditulis ke semua piksel dalam buffer

warna saat glClear dipanggil. Panggilan clear menginisialisasi komponen warna buffer tampilan dan juga dapat mengatur ulang nilai buffer kedalaman. Jika warna bening tidak berubah dalam suatu aplikasi, warna bening hanya perlu disetel sekali, dan sering kali ini dilakukan dalam inisialisasi program `OpenGL`. Setiap kali `glClear` dipanggil, ia menggunakan status warna bening yang telah ditetapkan sebelumnya.

Perhatikan juga bahwa status algoritme buffer-z dapat diaktifkan dan dinonaktifkan sesuai kebutuhan. Algoritma z-buffer juga dikenal di OpenGL sebagai uji kedalaman. Dengan mengaktifkannya, nilai kedalaman fragmen akan dibandingkan dengan nilai kedalaman yang saat ini disimpan di buffer kedalaman sebelum menulis warna fragmen apa pun ke buffer warna. Terkadang, uji kedalaman tidak diperlukan dan berpotensi memperlambat aplikasi. Menonaktifkan uji kedalaman akan mencegah komputasi buffer-z dan mengubah perilaku yang dapat dieksekusi. Mengaktifkan z-buffertest dengan OpenGL dilakukan sebagai berikut:

```
glEnable(GL_DEPTH_TEST);
glDepthFunc(GL_LESS);
```

Panggilan glEnable mengaktifkan pengujian kedalaman sementara panggilan `glDepthFunc` mengatur mekanisme bagaimana perbandingan kedalaman dilakukan. Dalam hal ini, fungsi kedalaman diatur ke nilai default `GL_LESS` untuk menunjukkan bahwa variabel keadaan lain ada dan dapat dimodifikasi. Kebalikan dari panggilan `glEnable` adalah panggilan `glDisable.`

Ide negara dalam OpenGL meniru penggunaan variabel statis di kelas berorientasi objek. Jika diperlukan, pemrogram mengaktifkan, menonaktifkan, dan/atau mengatur status variabel OpenGL yang berada di kartu grafis. Status ini kemudian memengaruhi setiap komputasi yang berhasil pada perangkat keras. Secara umum, program OpenGL yang efisien berusaha meminimalkan perubahan status, mengaktifkan status yang diperlukan, sementara menonaktifkan status yang tidak diperlukan untuk rendering.

17.6 TATA LETAK APLIKASI OPENGL DASAR

Aplikasi OpenGL yang sederhana dan mendasar memiliki, pada intinya, loop tampilan yang dipanggil secepat mungkin, atau pada kecepatan yang bertepatan dengan kecepatan refresh monitor atau perangkat tampilan. Contoh loop di bawah ini menggunakan library GLFW, yang mendukung pengkodean OpenGL di berbagai platform.

```
while (!glfwWindowShouldClose(window)) {
{
      // OpenGL code is called here,
      // each time this loop is executed.
      glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
      // Swap front and back buffers
      glfwSwapBuffers(window);
      // Poll for events
      glfwPollEvents();
      if (glfwGetKey( window, GLFW_KEY_ESCAPE ) == GLFW_PRESS)
      glfwSetWindowShouldClose(window, 1);
}
```

Loop dibatasi ketat untuk beroperasi hanya saat windows terbuka. Contoh inimengatur ulang nilai buffer warna dan juga menyetel ulang nilai kedalaman buffer z dalam memori perangkat keras grafis berdasarkan nilai yang ditetapkan sebelumnya (atau default). Perangkat input, seperti keyboard, mouse, jaringan, atau beberapa mekanisme interaksi lainnya diproses

pada akhir loop untuk mengubah keadaan struktur data yang terkait dengan program. Panggilan ke `glfwSwapBuffers` menyinkronkan konteks grafis dengan penyegaran tampilan, melakukan pertukaran penunjuk antara buffer depan dan belakangsehingga status grafis yang diperbarui ditampilkan di layar pengguna. Panggilan untuk menukar buffer terjadi setelah semua panggilan grafis dikeluarkan.

Sementara secara konseptual terpisah, buffer kedalaman dan warna sering disebut framebuffer. Dengan mengosongkan konten framebuffer, aplikasi dapat melanjutkan dengan panggilan OpenGL tambahan untuk mendorong geometri dan fragmen melalui pipa grafis. Frame buffer berhubungan langsung dengan ukuran windows yang telah dibuka untuk menampung konteks grafis. Dimensi windows, atau viewport, diperlukan oleh OpenGL untuk menyusun matriks M_{vp} (dari Bab7) di dalam perangkat keras. Ini diselesaikan melalui kode berikut, didemonstrasikan lagi dengan toolkit GLFW, yang menyediakan fungsi untuk menanyakan dimensi windows (atau framebuffer) yang diminta:

```
int nx, ny;
glfwGetFramebufferSize(window, &nx, &ny);
glViewport(0, 0, nx, ny);
```

Dalam contoh ini, `glViewport` menyetel status OpenGL untuk dimensi windows menggunakan nx dan ny untuk lebar dan tinggi windows dan area pandang yang ditentukan untuk memulai dari titik asal.

Secara teknis, OpenGL menulis ke memori framebuffer sebagai hasil dari operasi yang meraster geometri, dan memproses fragmen. Penulisan ini terjadi sebelum piksel ditampilkan di monitor pengguna.

17.7 GEOMETRI

Mirip dengan ide buffer tampilan, geometri juga ditentukan menggunakan array untuk menyimpan data titik dan atribut titik lainnya, seperti warna titik, normal, atau koordinat tekstur yang diperlukan untuk bayangan. Konsep buffer akan digunakan untuk mengalokasikan penyimpanan pada perangkat keras grafis, mentransfer data dari host ke perangkat.

Menjelaskan Geometri untuk Perangkat Keras

Salah satu tantangan dengan pemrograman perangkat keras grafis adalah pengelolaan data 3D dan transfernya ke dan dari memori perangkat keras grafis. Sebagian besar perangkat keras grafis bekerja dengan perangkat primitif geometris tertentu. Jenis primitif yang berbeda memanfaatkan kompleksitas primitif untuk kecepatan pemrosesan pada perangkat keras grafis. Primitif yang lebih sederhana terkadang dapat diproses dengan sangat cepat. Peringatannya adalah bahwa tipe primitif perlu tujuan umum untuk memodelkan berbagai geometri dari yang sangat sederhana hingga yang sangat kompleks. Pada perangkat keras grafis tipikal, tipe primitif terbatas pada satu atau lebih hal berikut:

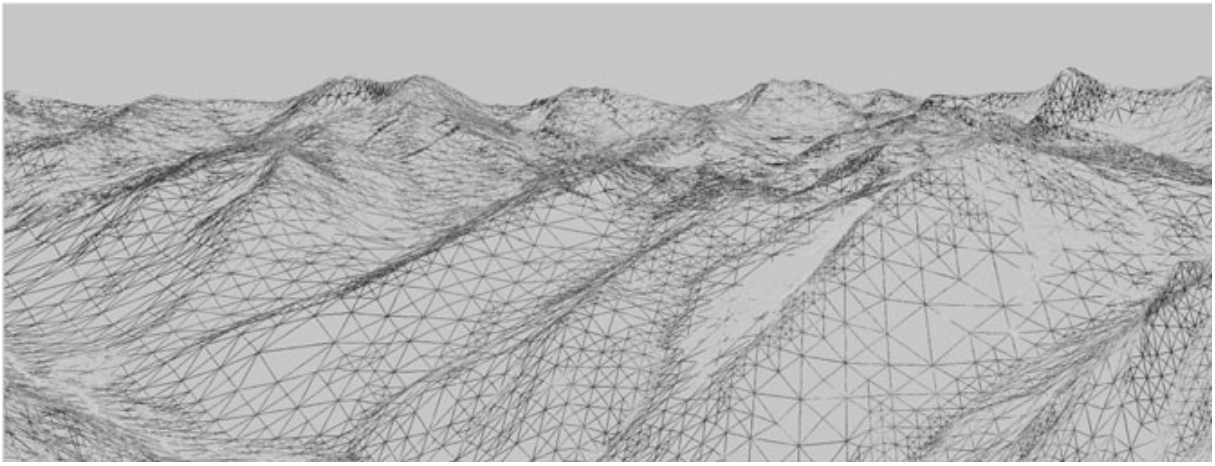
- **titik**—singleverces tunggal yang digunakan untuk merepresentasikan titik atau sistem partikel;
- **garis**—pasangan simpul yang digunakan untuk mewakili garis, siluet, atau penyorotan tepi;
- **segitiga**—segitiga, garis segitiga, segitiga berindeks, garis segitiga berindeks, segi empat, atausegitiga yang mendekatipermukaan geometris.

Ketiga tipe primitif ini membentuk blok bangunan dasar untuk sebagian besar geometri yang dapat didefinisikan. Contoh mesh segitiga yang dirender dengan OpenGL ditunjukkan pada Gambar 17.2.

Catatan

Primitif: Tiga primitif (titik, garis, segitiga, dan paha depan) benar-benar satu-satunya primitif yang tersedia! Bahkan saat membuat permukaan berbasis spline, seperti NURBS, permukaan tersebut diuji menjadi segitiga primitif oleh perangkat keras grafis.

Rendering Titik: Titik dan garis primitif mungkin awalnya tampak terbatas dalam penggunaan, tetapi peneliti telah menggunakan titik untuk membuat geometri yang sangat kompleks (Rusinkiewicz & Levoy, 2000; Dachsbacher, Vogelgsang, & Stamminger, 2003).



Gambar 17.2. Bagaimana geometri Anda diatur akan memengaruhi kinerja aplikasi Anda. Penggambaran gambar rangka dari kumpulan data medan Little Cottonwood Canyon ini menunjukkan puluhan ribu segitiga yang diatur sebagai jaring segitiga yang berjalan pada kecepatan waktu nyata. *Gambar dirender menggunakan sistem VTerrain Project terrainmilik Ben Discoe*.

17.8 PANDANGAN PERTAMA PADA *SHADER*

Versi modern OpenGL mengharuskan shader digunakan untuk memproses simpul dan fragmen. Dengan demikian, tidak ada primitif yang dapat dirender tanpa setidaknya satu shader simpul untuk memproses simpul primitif yang masuk dan shader lain untuk memproses fragmen raster. Jenis shader tingkat lanjut ada dalam OpenGL dan OpenGL Shading Language: shader geometri dan shader komputasi. Shader geometri dirancang untuk memproses primitif, berpotensi membuat primitif tambahan, dan dapat mendukung operasi instance geometris. Compute shader dirancang untuk melakukan komputasi umum pada GPU, dan dapat dihubungkan ke set shader yang diperlukan untuk aplikasi tertentu. Untuk informasi lebih lanjut tentang geometri dan shader komputasi, pembaca dirujuk ke dokumen spesifikasi OpenGL dan sumber daya lainnya.

Contoh Shader Vertex

Shader vertex memberikan kontrol atas bagaimana verteks ditransformasikan dan sering membantu menyiapkan data untuk digunakan dalam shader fragmen. Selain transformasi standar dan potensi operasi pencahayaan per titik, shader titik dapat digunakan untuk melakukan komputasi umum pada GPU. Misalnya, jika simpul mewakili partikel dan gerakan partikel dapat (hanya) dimodelkan dalam komputasi shader vertex, CPU sebagian besar dapat dihapus dari melakukan komputasi tersebut. Kemampuan untuk melakukan perhitungan pada simpul yang sudah disimpan dalam memori perangkat keras grafis adalah potensi peningkatan kinerja. Meskipun pendekatan ini berguna dalam beberapa situasi, komputasi umum tingkat lanjut mungkin lebih tepat dikodekan dengan computeshaders.

Dalam Bab 7, matriks viewport M_{vp} diperkenalkan. Ini mengubah koordinat volume tampilan kanonik menjadi koordinat layar. Dalam volume tampilan kanonik, koordinat ada dalam kisaran [−1,1]. Apa pun di luar rentang ini terpotong. Jika kita membuat asumsi awal bahwa geometri ada dalam rentang ini dan nilai z diabaikan, kita dapat membuat shader simpul yang sangat sederhana. Shader vertex ini melewati posisi vertex ke tahap rasterisasi, di mana transformasi viewport terakhir akan terjadi. Perhatikan bahwa karena penyederhanaan ini, tidak ada proyeksi, tampilan, atau transformasi model yang akan diterapkan ke simpul masuk. Ini awalnya rumit untuk membuat apa pun kecuali scene yang sangat sederhana, tetapi akan membantu memperkenalkan konsep shader dan memungkinkan Anda untuk membuat segitiga awal ke layar. Shader vertex passthrough berikut:

```
#version 330 core
layout(location=0) in vec3 in_Position;
void main(void)
{
gl_Position = vec4(in_Position, 1.0);
}
```

Vertex shader ini hanya melakukan satu hal. Ini melewati posisi verteks masuk sebagai `gl_Position` yang digunakan OpenGL untuk rasterisasi fragmen. Perhatikan bahwa `gl_Position` adalah variabel cadangan bawaan yang menandakan salah satu output utama yang diperlukan dari shader simpul. Perhatikan juga versi string di baris pertama. Dalam hal ini, string menginstruksikan kompiler GLSL bahwa versi 3.3 dari profil GLSL Core akan digunakan untuk mengkompilasi bahasa bayangan.

Vertex dan fragmen shader adalah operasi SIMD yang masing-masing beroperasi pada semua simpul atau fragmen yang sedang diproses di dalam pipa. Data tambahan dapat dikomunikasikan dari host ke shader yang dieksekusi pada perangkat dengan menggunakan variabel input, output, atau seragam. Data yang dilewatkan ke dalam shader diawali dengan kata kunci in. Lokasi data tersebut yang terkait dengan atribut verteks tertentu atau indeks output fragmen juga ditentukan secara langsung di shader. Jadi,

```
layout(location=0) in vec3 in_Position;
```

menentukan bahwa `in Position` adalah variabel input yang bertipe `vec3`. Sumber data tersebut adalah indeks atribut 0 yang diasosiasikan dengan geometri. Nama variabel ini ditentukan oleh programmer, dan hubungan antara geometri yang masuk dan shader terjadi saat menyiapkan data vertex pada perangkat. GLSL berisi berbagai jenis bagus yang berguna untuk program grafis, termasuk `vec2, vec3, vec4, mat2, mat3,` dan `mat4` untuk beberapa nama. Tipe standar seperti int atau float juga ada. Dalam pemrograman shader, vektor, seperti `vec4` menampung 4 komponen yang sesuai dengan komponen x, y, z, dan w dari koordinat homogen, atau komponen r, g, b, dan dari tupel RGBA. Label untuk tipe dapat dipertukarkan sesuai kebutuhan (dan bahkan diulang) dalam apa yang disebut swizzling (misalnya, di Position.zyxa). Selain itu, label komponen-bijaksana kelebihan beban dan dapat digunakan dengan tepat untuk menyediakan konteks.

Semua shader harus memiliki fungsi utama yang melakukan komputasi utama di semua input. Dalam contoh ini, fungsi utama hanya menyalin posisi simpul input (`in Position`), yang bertipe vec3 ke dalam variabel output shader simpul built-in, yang bertipe `vec4`. Perhatikan bahwa banyak tipe bawaan memiliki konstruktor yang berguna untuk konversi seperti yang disajikan di sini untuk mengonversi tipe vec3 posisi verteks masuk menjadi tipe `gl Position`'s `vec4`. Koordinat homogen digunakan dengan OpenGL, jadi 1.0 ditetapkan sebagai koordinat keempat untuk menunjukkan bahwa vektor adalah posisi.

Contoh Shader Fragmen

Jika shader simpul paling sederhana hanya melewatkan koordinat klip, shader fragmen paling sederhana menetapkan warna fragmen ke nilai konstan.

```
#version 330 core
layout(location=0) out vec4 out_FragmentColor;
void main(void)
{
out_FragmentColor = vec4(0.49, 0.87, 0.59, 1.0);
}
```

Dalam contoh ini, semua fragmen akan diatur ke warna hijau muda. Salah satu perbedaan utama adalah penggunaan kata kunci out. Secara umum, kata kunci masuk dan keluar dalam program shader menunjukkan aliran data masuk dan keluar dari shader. Sementara shader simpul menerima simpul masuk dan mengeluarkannya ke variabel bawaan, shader fragmen mendeklarasikan nilai keluarnya yang ditulis ke buffer warna:

```
layout(location=0) out vec4 out_FragmentColor;
```

Variabel output dari `out FragmentColor` sekali lagi ditentukan oleh pengguna. Lokasi output adalah indeks buffer warna 0. Fragment shader dapat menghasilkan beberapa buffer, tetapi ini adalah topik lanjutan yang diserahkan kepada pembaca yang akan dibutuhkan jika objek framebuffer OpenGL diselidiki. Penggunaan kata kunci `layout` dan `location` membuat hubungan eksplisit antara data geometris aplikasi di vertex shader dan buffer warna output di fragment shader.

17.9 MEMUAT, MENGKOMPILASI, DAN MENGGUNAKAN *SHADER*

Program shader ditransfer ke perangkat keras grafis dalam bentuk string karakter. Mereka kemudian harus dikompilasi dan dihubungkan. Selanjutnya, shader digabungkan bersama ke dalam program shader sehingga pemrosesan verteks dan fragmen terjadi secara konsisten. Pengembang dapat mengaktifkan shader yang telah berhasil dikompilasi dan ditautkan ke program shader sesuai kebutuhan, sekaligus menonaktifkan shader jika tidak diperlukan. Sementara proses rinci membuat, memuat, mengkompilasi, dan menautkan program shader tidak disediakan dalam bab ini, fungsi OpenGL berikut akan membantu dalam membuat shader:

- **glCreateShadermembuat** pegangan ke shader pada perangkat keras.
- **glShaderSourcememuat** string karakter ke dalam memori perangkat keras grafis.
- **glCompileShader** melakukan kompilasi sebenarnya dari shader di dalam perangkat keras.

Fungsi-fungsi di atas perlu dipanggil untuk setiap shader. Jadi, untuk passthrough shader sederhana, masing-masing fungsi tersebut akan dipanggil untuk kode shader vertex dan kode shader fragmen yang disediakan. Di akhir fase kompilasi, status kompilasi dan kesalahan apa pun dapat ditanyakan menggunakan perintah OpenGL tambahan.

Setelah kedua kode shader dimuat dan dikompilasi, mereka dapat dihubungkan ke program shader. Program shader adalah apa yang digunakan untuk mempengaruhi rendering geometri.

- **glCreateProgram** membuat objek program yang akan berisi shader yang telah dikompilasi sebelumnya.
- **glAttachShader** melampirkan shader ke objek program shader. Dalam contoh sederhana, fungsi ini akan dipanggil untuk objek vertexshader yang dikompilasi dan objek shader fragmen yang dikompilasi.

- **glLinkProgram** menautkan shader secara internal setelah semua shader dipasang ke objek program.
- **glUseProgram** mengikat program shader untuk digunakan pada perangkat keras grafis. Karena shader diperlukan, pegangan program terikat menggunakan fungsi ini. Ketika tidak ada shader yang dibutuhkan, mereka dapat dilepaskan dengan menggunakan program shader menangani 0 sebagai argumen untuk fungsi ini.

17.10 OBJEK *VERTEX BUFFER*

Simpul disimpan pada perangkat keras grafis menggunakan buffer, yang dikenal sebagai objek buffer verteks. Selain simpul, atribut simpul tambahan apa pun, seperti warna, vektor normal, atau koordinat tekstur, juga akan ditentukan menggunakan objek buffer simpul.

Pertama, mari kita fokus pada menentukan geometri primitif itu sendiri. Ini dimulai dengan mengalokasikan simpul yang terkait dengan primitif dalam memori host aplikasi. Cara paling umum untuk melakukan ini adalah dengan mendefinisikan sebuah array pada host untuk memuat simpul-simpul yang diperlukan untuk primitif. Misalnya, sebuah segitiga tunggal, yang terisi penuh dalam volume kanonik, dapat didefinisikan secara statis pada host sebagai berikut:

```
GLfloat vertices[] = {-0.5f, -0.5f, 0.0f, 0.5f, -0.5f, 0.0f,
0.0f, 0.5f, 0.0f};
```

Jika shader passthrough sederhana digunakan untuk segitiga ini, maka semua simpul akan di-ender. Meskipun segitiga ditempatkan pada bidang z = 0, koordinat z untuk contoh ini tidak terlalu penting karena pada dasarnya diturunkan dalam transformasi akhir menjadi koordinat layar. Hal lain yang perlu diperhatikan adalah penggunaan tipe OpenGL memiliki tipe terkait yang umumnya dapat bercampur dengan baik dengan tipe standar (seperti oat). Untuk ketepatan, tipe OpenGL akan digunakan saat diperlukan.

Catatan : Sistem Koordinat OpenGL: Sistem koordinat yang digunakan oleh OpenGL identik dengan yang disajikan dalam buku ini. Ini adalah sistem koordinat tangan kanan dengan +x ke kanan, +y ke atas, dan +z menjauh dari layar (atau windows). Jadi, –z menunjuk ke monitor.

Sebelum simpul dapat diproses, buffer simpul terlebih dahulu dibuat pada perangkat untuk menyimpan simpul. Simpul pada host kemudian ditransfer ke perangkat. Setelah ini, buffer simpul dapat menjadi referensi yang diperlukan untuk menggambar array simpul yang disimpan dalam buffer. Selain itu, setelah transfer awal data vertex, tidak ada penyalinan data tambahan di seluruh bus host ke perangkat, terutama jika geometri tetap statis di seluruh pembaruan loop rendering. Memori host apa pun juga dapat dihapus jika dialokasikan secara dinamis.

Objek buffer verteks, sering disebut VBO, mewakili mekanisme utama dengan OpenGL modern untuk menyimpan verteks dan atribut verteks dalam memori grafis. Untuk tujuan efisiensi, pengaturan awal VBO dan transfer data terkait verteks sebagian besar terjadi sebelum memasuki loop tampilan. Sebagai contoh, untuk membuat VBO untuk segitiga ini, kode berikut dapat digunakan:

```
GLuint triangleVBO[1];
glGenBuffers(1, triangleVBO);
glBindBuffer(GL_ARRAY_BUFFER, triangleVBO[0]);
glBufferData(GL_ARRAY_BUFFER, 9 * sizeof(GLfloat), vertices,
GL_STATIC_DRAW);
glBindBuffer(GL_ARRAY_BUFFER, 0);
```

Tiga panggilan OpenGL diperlukan untuk membuat dan mengalokasikan objek buffer vertex. Yang pertama, `glGenBuffers` membuat pegangan yang dapat digunakan untuk merujuk ke VBO setelah disimpan di perangkat. Beberapa pegangan ke VBO (disimpan dalam array) dapat dibuat dalam satu panggilan `glGenBuffers`, seperti yang diilustrasikan tetapi tidak digunakan di sini. Perhatikan bahwa ketika objek buffer dibuat, alokasi ruang sebenarnya pada perangkat belum dilakukan.

Dengan OpenGL, objek, seperti objek buffer vertex, adalah target utama untuk komputasi dan pemrosesan. Objek harus terikat ke status OpenGL yang diketahui saat digunakan dan tidak terikat saat tidak digunakan. Contoh penggunaan objek OpenGL termasuk objek buffer vertex, objek framebuffer, objek tekstur, dan program shader, tonameafew. Dalam contoh saat ini, GL ARRAY BUFFER keadaan Open GL terikat pada pegangan VBO segitiga yang telah dibuat sebelumnya. Ini pada dasarnya membuat VBO sebagai objek buffer simpul aktif. Setiap operasi yang mempengaruhi buffer vertex yang mengikuti perintah `glBindBuffer(GL ARRAY BUFFER, triangleVBO[0])` akan menggunakan data segitiga di VBO baik dengan membaca data atau menulisnya.

Data simpul disalin dari host (`vertices` array) ke perangkat (saat ini terikat GL ARRAY BUFFER) menggunakan panggilan. Argumen mewakili tipe target, ukuran dalam byte buffer yang akan disalin, penunjuk ke buffer host, dan tipe enumerasi yang menunjukkan bagaimana buffer akan digunakan. Dalam contoh saat ini,targetnya adalah `GL STATIC BUFFER`, ukuran data adalah `9* sizeof(Glfloat)`, dan argumen terakhir adalah `GL STATIC DRAW` yang menunjukkan kepada OpenGL bahwa simpul tidak akan berubah selama rendering. Akhirnya, ketika VBO tidak lagi membutuhkan target aktif untuk membaca atau menulis, panggilan itu tidak terikat dengan `glBindBuffer(GL ARRAY BUFFER, 0)` Secara umum, mengikat objek atau buffer OpenGL mana pun ke handle0, unbinds, atau menonaktifkan buffer yang memengaruhi fungsionalitas selanjutnya.

17.11 OBJEK *VERTEX ARRAY*

Sementara objek buffer simpul adalah wadah penyimpanan untuk simpul (dan atribut simpul), objek larik simpul mewakili mekanisme OpenGL untuk menggabungkan buffer simpul ke dalam keadaan simpul yang konsisten yang dapat dikomunikasikan dan ditautkan dengan shader di perangkat keras grafis. Ingat bahwa pipa fungsi tetap di masa lalu tidak ada lagi dan oleh karena itu, status per-vertex, seperti warna normal atau bahkan vertex, harus disimpan dalam buffer perangkat keras dan kemudian direferensikan dalam shader, menggunakan variabel input (mis., in).

Seperti halnya objek buffer vertex, objek array vertex, atau VAO, harus dibuat dan dialokasikan dengan status apa pun yang diperlukan ditetapkan saat objek array vertex terikat. Misalnya, kode berikut menunjukkan cara membuat VAO untuk memuat segitiga VBO yang telah ditentukan sebelumnya:

```
GLuint VAO;
glGenVertexArrays(1, &VAO);
glBindVertexArray(VAO);
glEnableVertexAttribArray(0);
glBindBuffer(GL_ARRAY_BUFFER, triangleVBO[0]);
glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 3 *
sizeof(GLfloat), 0);
glBindVertexArray(0);
```

Saat mendefinisikan objek array vertex, objek buffer vertex tertentu dapat diikat ke atribut vertex tertentu (atau input) dalam kode shader. Ingat penggunaan

```
layout(location=0) in vec3 in_Position
```

inthepassthroughvertexshader. Sintaks ini menunjukkan bahwa variabel shader akan menerima datanya dari indeks atribut 0 dalam objek array simpul terikat. Dalam kode host, mapping dibuat menggunakan panggilan.

```
glEnableVertexAttribArray(0);
glBindBuffer(GL_ARRAY_BUFFER, triangleVBO[0]);
glVertexAttribPointer(0,    3,    GL_FLOAT,    GL_FALSE,    3    *
sizeof(GLfloat), 0);
```

Panggilan pertama mengaktifkan indeks atribut vertex (dalam hal ini, 0). Dua panggilan berikutnya menghubungkan objek buffer simpul yang telah ditentukan sebelumnya yang menyimpan simpul ke atribut simpul itu sendiri. Karena `lVertexAttribPointer` menggunakan VBO saat ini terikat, penting bahwa `glBindBuffer` dikeluarkan sebelum menetapkan penunjuk atribut vertex. Pemanggilan fungsi ini membuat mapping yang mengikat simpul di buffer simpul kita ke variabel `in Position` di dalam shader simpul. Panggilan `glVertexAttribPointer` tampaknya rumit tetapi pada dasarnya menetapkan indeks atribut 0 untuk menampung tiga komponen (mis., x, y, z) dari `GLfloats` (argumen ke-2 dan ke-3) yang tidak dinormalisasi (argumen keempat). Argumen kelima menginstruksikan OpenGL bahwa tiga nilai float memisahkan awal dari setiap himpunan simpul. Dengan kata lain, simpul-simpul itu terkemas rapat dalam memori, satu demi satu. Argumen terakhir adalah pointer ke data, tetapi karena buffer vertex telah diikat sebelum panggilan ini, data akan diasosiasikan dengan vertexbuffer.

Langkah-langkah sebelumnya yang menginisialisasi dan membangun objek array vertex, objek buffer vertex, dan shader semuanya harus dieksekusi sebelum memasuki loop tampilan. Semua memori dari buffer vertex akan ditransfer ke GPU dan objek array vertex akan membuat koneksi antara data dan indeks variabel input shader. Dalam loop tampilan, panggilan berikut akan memicu pemrosesan objek array vertex:

```
glBindVertexArray(VAO);
glDrawArrays(GL_TRIANGLES, 0, 3);
glBindVertexArray(0);
```

Perhatikan lagi, bahwa panggilan bind membuat objek array vertex aktif. Panggilan ke `glDrawArrays` memulai pipeline untuk geometri ini, menjelaskan bahwa geometri harus ditafsirkan sebagai serangkaian primitif segitiga yang dimulai dari offset 0 dan hanya menampilkan tiga indeks. Dalam contoh ini, hanya ada tiga elemen dalam array dan primitifnya adalah segitiga, jadi segitiga tunggal akan dirender.

Menggabungkan semua langkah ini, kode yang dirakit untuk segitiga akan menyerupai berikut ini, dengan asumsi bahwa pemuatan data bayangan dan simpul terkandung dalam fungsi eksternal:

```
// Set the vieport once
int nx, ny;
glfwGetFramebufferSize(window, &nx, &ny);
glViewport(0, 0, nx, ny);
// Set clear color state
glClearColor( 0.0f, 0.0f, 0.0f, 1.0f );
// Create the Shader programs, VBO, and VAO
GLuint shaderID = loadPassthroughShader();
```

```
GLuint VAO = loadVertexData();
while (!glfwWindowShouldClose(window)) {
{
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
glUseProgram( shaderID );
glBindVertexArray(VAO);
glDrawArrays(GL_TRIANGLES, 0, 3);
glBindVertexArray(0);
glUseProgram( 0 );
// Swap front and back buffers
glfwSwapBuffers(window);
// Poll for events
glfwPollEvents();
if (glfwGetKey( window, GLFW_KEY_ESCAPE ) == GLFW_PRESS)
glfwSetWindowShouldClose(window, 1);
}
```

Gambar 17.3 menunjukkan hasil penggunaan shader dan status vertex untuk merender segitiga volume tampilan kanonik.

Gambar 17.3. Segitiga kanonik dirender menggunakan vertex sederhana dan shader fragmen.

17.12 MATRIKS TRANSFORMASI

Versi OpenGL saat ini telah menghapus tumpukan matriks yang pernah digunakan untuk merujuk matriks proyeksi dan tampilan model dari perangkat keras. Karena tumpukan matriks ini tidak ada lagi, programmer harus menulis kode matriks yang dapat ditransfer ke vertex shader tempat transformasi akan terjadi. Itu awalnya mungkin tampak menantang. Namun, beberapa library dan toolkit telah dikembangkan untuk membantu pengembangan kode OpenGL lintas platform. Salah satu perpustakaan ini, GLM, atau Matematika OpenGL, telah dikembangkan untuk melacak spesifikasi OpenGL dan GLSL secara dekat sehingga interoperasi antara GLM dan perangkat keras akan bekerja dengan mulus.

17.13 GLM

GLM menyediakan beberapa tipe matematika dasar yang berguna untuk grafis komputer. Untuk tujuan kami, kami akan fokus hanya pada beberapa jenis dan beberapa fungsi yang memanfaatkan transformasi matriks dalam shader dengan mudah. Beberapa jenis yang akan digunakan antara lain sebagai berikut:

- **glm::vec3**—compactarray dari 3 float yang dapat diakses menggunakan akses komponen-bijaksana yang sama yang ditemukan di shader;
- **glm::vec4**—compactarray dari 4 float yang dapat diakses menggunakan akses komponen-bijaksana yang sama yang ditemukan di shader;
- **glm::mat4**—matriks penyimpanan 4×4 yang direpresentasikan sebagai 16 float. Matriks disimpan dalam format kolom-mayor.

Demikian pula, GLM menyediakan fungsi untuk membuat matriks proyeksi, Morth dan Mp, serta fungsi untuk menghasilkan matriks tampilan, Mcam:

- **glm::orto** membuat matriks proyeksi ortografis 4x4.
- **glm::perspective** membuat matriks perspektif 4x4.
- **glm::lookAt** menciptakan transformasi homogen 4 × 4 yang menerjemahkan dan mengarahkan kamera.

Menggunakan Proyeksi Ortografi

Perpanjangan sederhana untuk contoh sebelumnya akan menempatkan tuas segitiga ke dalam sistem koordinat yang lebih fleksibel dan merender scene menggunakan proyeksi ortografis. Simpul dalam contoh sebelumnya bisa menjadi:

```
GLfloat vertices[] = {-3.0f, -3.0f, 0.0f, 3.0f, -3.0f, 0.0f,
0.0f, 3.0f, 0.0f};
```

Menggunakan GLM, proyeksi ortografis dapat dibuat dengan mudah di host. Misalnya,

```
glm::mat4 projMatrix = glm::ortho(-5.0f, 5.0f, -5.0, 5.0, -
10.0f, 10.0f);
```

Matriks proyeksi kemudian dapat diterapkan ke setiap simpul yang mengubahnya menjadi koordinat klip. Shader vertex akan dimodifikasi untuk melakukan operasi ini:

$$\mathbf{v}_{canon} = \mathbf{M}_{orth}\mathbf{v}.$$

Perhitungan ini akan terjadi pada shader vertex yang dimodifikasi yang menggunakan variabel seragam untuk mengkomunikasikan data dari host ke perangkat. Variabel seragam mewakili data statis yang invarian di seluruh eksekusi program shader. Datanya sama untuk semua elemen dan tetap statis. Namun, variabel seragam dapat dimodifikasi oleh aplikasi antara eksekusi shader. Ini adalah mekanisme utama bahwa data dalam aplikasi host dapat mengkomunikasikan perubahan pada komputasi shader. Data seragam sering mewakili keadaan grafis yang terkait dengan aplikasi. Misalnya, proyeksi, tampilan, atau matriks model dapat diatur dan diakses melalui variabel seragam. Informasi tentang sumber cahaya dalam scene juga dapat diperoleh melalui variabel seragam.

Memodifikasi vertex shader memerlukan penambahan variabel seragam untuk menampung matriks proyeksi. Kita dapat menggunakan tipe mat4 GLSL untuk menyimpan data ini. Matriks proyeksi kemudian dapat digunakan secara alami untuk mengubah simpul masuk ke dalam sistem koordinat kanonik:

```
#version 330 core
layout(location=0) in vec3 in_Position;
uniform mat4 projMatrix;
void main(void)
{
gl_Position = projMatrix * vec4(in_Position, 1.0);
}
```

Kode aplikasi hanya perlu mentransfer variabel seragam dari memori host (matt GLM4) ke program shader perangkat (mat4 GLSL). Ini cukup mudah, tetapi mengharuskan sisi host aplikasi memperoleh pegangan ke variabel seragam setelah program shader telah ditautkan. Misalnya, untuk mendapatkan pegangan ke variabel `projMatrix`, panggilan berikut akan dikeluarkan satu kali, setelah penautan program shader selesai:

```
GLint pMatID = glGetUniformLocation(shaderProgram,
"projMatrix");
```

Argumen pertama adalah pegangan objek program shader dan argumen kedua adalah string karakter dari nama variabel di shader. Id kemudian dapat digunakan dengan berbagai panggilan fungsi OpenGL `glUniform` untuk mentransfer memori pada host ke perangkat. Namun, program shader harus terlebih dahulu diikat sebelum menetapkan nilai yang terkait dengan variabel seragam. Juga, karena GLM digunakan untuk menyimpan matriks proyeksi pada host, fungsi pembantu GLM akan digunakan untuk mendapatkan penunjuk ke matriks yang mendasarinya, dan memungkinkan penyalinan untuk dilanjutkan.

```
glUseProgram( shaderID );
glUniformMatrix4fv(pMatID, 1, GL_FALSE,
glm::value_ptr(projMatrix));
glBindVertexArray(VAO);
glDrawArrays(GL_TRIANGLES, 0, 3);
glBindVertexArray(0);
glUseProgram( 0 );
```

Perhatikan bentuk yang digunakan `glUniform`. Nama fungsi diakhiri dengan karakter yang membantu menentukan cara penggunaannya. Dalam kasus ini, satu matriks float 4 × 4 sedang dipindahkan ke dalam variabel seragam. v menunjukkan bahwa array berisi data, bukan lewat nilai. Argumen ketiga memungkinkan OpenGL mengetahui apakah matriks harus ditransposisikan (fitur yang berpotensi berguna), dan argumen terakhir adalah penunjuk ke memori tempat matriks berada.

Dengan bagian bab ini, Anda harus memiliki pemahaman bahwa shader dan data verteks bermain dalam merender objek dengan OpenGL. Shader, khususnya, membentuk peran yang sangat penting dalam OpenGL modern. Bagian selanjutnya akan mengeksplorasi lebih lanjut peran shader dalam rendering scene, mencoba untuk membangun peran yang dimainkan shader dalam gaya rendering lain yang disajikan dalam buku ini.

Shading dengan Atribut Per-Vertex

Contoh sebelumnya menentukan segitiga tunggal tanpa data tambahan. Atribut simpul, seperti vektor normal, koordinat tekstur, atau bahkan warna, dapat disisipkan dengan data simpul dalam buffer simpul. Tata letak memori sangat mudah. Di bawah ini, warna setiap simpul ada setelah setiap simpul di dalam sinar. Tiga komponen digunakan untuk mewakili saluran merah, hijau, dan biru. Mengalokasikan buffer vertex identik dengan pengecualian bahwa ukuran array sekarang adalah 18 `GLfloats`, bukan 9.

```
GLfloat vertexData[] = {0.0f, 3.0f, 0.0f, 1.0f, 1.0f, 0.0f, -
3.0f,
-3.0f, 0.0f, 0.0f, 1.0f, 1.0f, 3.0f, -3.0f, 0.0f, 1.0f, 0.0f,
1.0f};
```

Spesifikasi objek array vertex berbeda. Karena data warna disisipkan di antara simpul, penunjuk atribut simpul harus melintasi data dengan tepat. Indeks atribut vertex kedua juga harus diaktifkan. Membangun dari contoh sebelumnya, kami membangun VAO baru sebagai berikut:

```
glBindBuffer(GL_ARRAY_BUFFER, m_triangleVBO[0]);
glEnableVertexAttribArray(0);
glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 6 *
sizeof(GLfloat),
0);
glEnableVertexAttribArray(1);
glVertexAttribPointer(1, 3, GL_FLOAT, GL_FALSE, 6 *
sizeof(GLfloat),
(const GLvoid *)12);
```

VBO tunggal digunakan dan diikat sebelum mengatur atribut karena data verteks dan warna terkandung di dalam VBO. Atribut simpul pertama diaktifkan pada indeks0, yang akan mewakili simpul dalam shader. Perhatikan bahwa langkah (5 argumen) berbeda karena simpul dipisahkan oleh enam oat (misalnya, thex, y, z dari simpul yang diikuti oleh lainnya, g, b dari warna). Indeks atribut verteks kedua diaktifkan dan akan mewakili atribut warna simpul di shader di lokasi 1. Ini memiliki langkah yang sama, tetapi argumen terakhir sekarang mewakili offset penunjuk untuk awal nilai warna pertama. Sementara 12 digunakan dalam contoh di atas, ini identik dengan menyatakan `3 * sizeof(GLfloat)`. Dengan kata lain, kita perlu melompati tiga float yang mewakili nilai vertex, y, z untuk menemukan atribut warna pertama dalam larik.

Shader untuk contoh ini hanya sedikit dimodifikasi. Perbedaan utama dalam vertex shader (ditunjukkan di bawah) adalah (1) atribut kedua, warna, berada di lokasi 1 dan (2) `vColor` adalah variabel output yang diatur di badan utama vertex shader.

```
#version 330 core
layout(location=0) in vec3 in_Position;
layout(location=1) in vec3 in_Color;
out vec3 vColor;
uniform mat4 projMatrix;
void main(void)
{
vColor = in_Color;
gl_Position = projMatrix * vec4(in_Position, 1.0);
}
```

Ingat bahwa kata kunci masuk dan keluar mengacu pada aliran data antara shader. Data yang mengalir keluar dari vertex shader menjadi data input di shader fragmen yang terhubung, asalkan nama variabelnya cocok. Selain itu, variabel keluar yang dilewatkan ke shader fragmen diinterpolasi di seluruh fragmen menggunakan interpolasi barycentric. Beberapa modifikasi interpolasi dapat dilakukan dengan kata kunci tambahan, tetapi detail ini akan diserahkan kepada pembaca. Dalam contoh ini, tiga simpul ditentukan, masing-masing dengan nilai warna tertentu. Di dalam shader fragmen, warna akan diinterpolasi di seluruh permukaan segitiga.

Perubahan shader fragmen sederhana. Variabel `vColor` yang diset dan dilewatkan dari vertex shader sekarang menjadi variabel in. Saat fragmen diproses, `vColor vec3` akan berisi nilai interpolasi yang benar berdasarkan lokasi fragmen di dalam segitiga.

```
#version 330 core
layout(location=0) out vec4 fragmentColor;
in vec3 vColor;
void main(void)
{
```

```
fragmentColor = vec4(vColor, 1.0);
}
```

Gambar yang dihasilkan dari menjalankan shader ini dengan data segitiga ditunjukkan pada Gambar 17.4.

17.14 STRUKTUR DATA VERTEX

Contoh sebelumnya menggambarkan interleaving data dalam array. Buffer vertex dapat digunakan dalam berbagai cara, termasuk buffer vertex terpisah untuk atribut model yang berbeda. Menyisipkan data memiliki keuntungan karena atribut yang terkait dengan simpul berada di dekat simpul dalam memori dan kemungkinan dapat memanfaatkan lokalitas memori saat beroperasi di shader. Sementara penggunaan array interleaved ini mudah, dapat menjadi rumit untuk mengelola model besar dengan cara ini, terutama karena struktur data digunakan untuk membangun infrastruktur perangkat lunak yang kuat (dan berkelanjutan) untuk grafis (lihat Bab 12). Agak mudah untuk menyimpan data simpul sebagai vektor dari struktur yang berisi simpul dan atribut terkait. Ketika dilakukan dengan cara ini, struktur hanya perlu dipetakan ke dalam buffer vertex. Misalnya, struktur berikut berisi posisi vertex dan warna vertex, menggunakan tipe `vec3` GLM:

```
struct vertexData
{
glm::vec3 pos;
glm::vec3 color;
};
std::vector< vertexData > modelData;
```

Vektor STL akan menampung semua simpul yang terkait dengan semua segitiga dalam model. Kami akan terus menggunakan tata letak yang sama untuk segitiga seperti pada contoh sebelumnya, yang merupakan strip segitiga dasar. Setiap tiga simpul mewakilisatriangle dalam daftar. Ada organisasi data lain yang dapat digunakan dengan OpenGL, dan Bab 12 menyajikan opsi lain untuk mengatur data secara lebih efisien.

Gambar 17.4. Mengatur warna setiap simpul di shader simpul dan meneruskan data ke shader fragmen menghasilkan interpolasi barycentric dari warna.

Setelah data dimuat ke dalam vektor, panggilan yang sama digunakan sebelum memuat data ke objek buffer vertex:

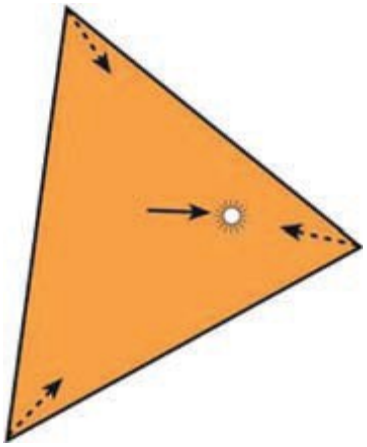
```
int numBytes = modelData.size() * sizeof(vertexData);

glBufferData(GL_ARRAY_BUFFER, numBytes, modelData.data(),
GL_STATIC_DRAW);
glBindBuffer(GL_ARRAY_BUFFER, 0);
```

Vektor STL menyimpan data secara berurutan. Struk `vertexData` yang digunakan di atas diwakili oleh tata letak memori yang datar (tidak mengandung pointer ke elemen data lain) dan bersebelahan. Namun, vektor STL adalah abstraksi dan penunjuk yang mereferensikan memori yang mendasarinya harus ditanyakan menggunakan anggota `data()`. Pointer itu disediakan untuk panggilan ke `glBufferData`.. Penetapan atribut pada objek array vertex identik karena lokalitas atribut vertex tetap sama.

Shading pada Fragment Processor

Bab pipa grafis (Bab 8) dan bayangan permukaan bab (Bab 10) melakukan pekerjaan yang bagus untuk menggambarkan dan menggambarkan efek per-verteks dan per fragmen bayangan karena mereka berhubungan dengan asterisasi dan bayangan secara umum. Dengan perangkat keras grafis modern, penerapan algoritme bayangan dalam prosesor fragmen menghasilkan hasil visual yang lebih baik dan perkiraan pencahayaan yang lebih akurat. Bayangan yang dihitung pada basis titik puncak melunak menjadi subjek artefak visual yang terkait dengan penjualan tes geometri yang mendasarinya. Khususnya, naungan sepuluh titik berbasis per titik gagal mendekati intensitas yang sesuai di seluruh permukaan segitiga karena pencahayaan hanya dihitung pada setiap titik. Misalnya, jika jarak ke sumber cahaya kecil, dibandingkan dengan ukuran wajah yang diarsir, pencahayaan pada wajah akan salah. Gambar 17.5 mengilustrasikan situasi ini. Bagian tengah segitiga tidak akan diterangi dengan terang, meskipun sangat dekat dengan sumber cahaya, karena pencahayaan pada verteks yang jauh dari sumber cahaya digunakan untuk menyisipkan bayangan di seluruh wajah. Tentu saja, meningkatkan tesselasi geometri dapat meningkatkan visual. Namun, solusi ini tidak banyak digunakan dalam grafis waktu nyata karena geometri tambahan yang diperlukan untuk penerangan yang lebih akurat dapat menghasilkan rendering yang lebih lambat.



Gambar 17.5. Jarak ke sumber cahaya relatif kecil terhadap ukuran segitiga.

Shader fragmen beroperasi pada fragmen yang muncul dari rasterisasi setelah simpul ditransformasi dan dipotong. Secara umum, fragment shader harus mengeluarkan nilai yang ditulis ke framebuffer. Sering kali, ini adalah warna piksel. Jika uji kedalaman diaktifkan, nilai kedalaman fragmen akan digunakan untuk mengontrol apakah warna dan kedalamannya

ditulis ke memori framebuffer. Data yang digunakan oleh Fragmen Shader untuk perhitungan berasal dari berbagai sumber:

- **Built-in OpenGL variables /Variabel OpenGL bawaan.** Variabel ini disediakan oleh sistem. Contoh variabel shader fragmen termasuk `gl FragCoord` atau `gl FrontFacing`.. Variabel ini dapat berubah berdasarkan revisi OpenGL dan GLSL, jadi Anda disarankan untuk memeriksa spesifikasi versi OpenGL dan GLSL yang Anda targetkan.
- **Uniform Variable/Variabel seragam**. Variabel seragam ditransfer dari host ke perangkat dan dapat berubah sesuai kebutuhan berdasarkan input pengguna atau perubahan status simulasi dalam aplikasi. Variabel-variabel ini dideklarasikan dan didefinisikan oleh programmer untuk digunakan baik di dalam vertex maupun fragmentshader. Matriks proyeksi dalam contoh shader vertex sebelumnya dikomunikasikan ke shader melalui variabel seragam. Jika diperlukan, nama variabel seragam yang sama dapat digunakan di dalam vertex dan fragmentshader.
- **Variabel Input**. Variabel input ditentukan dalam shader fragmen dengan kata kunci awalan masuk. Ingat bahwa data dapat mengalir masuk dan keluar dari shader. Vertex shader dapat menampilkan data ke tahap shader berikutnya menggunakan kata kunci out (mis., `out vec3 vColor`,, dalam contoh sebelumnya). Output ditautkan ke input ketika tahap berikutnya menggunakan kata kunci in diikuti dengan kualifikasi jenis dan nama yang sama (misalnya, dalam `vec3 vColor` pada contoh sebelumnya'scorrespondingshadershader).

Setiap data yang diteruskan ke shader fragmen melalui mekanisme tautan masuk-keluar akan bervariasi pada basis per-fragmen menggunakan interpolasi barycentric. Interpolasi dihitung di luar shader oleh perangkat keras grafis. Dalam infrastruktur ini, pecahan shader dapat digunakan untuk melakukan algoritma pengarsir per-fragmen yang mengevaluasi persamaan tertentu di seluruh muka segitiga. Shader simpul memberikan komputasi pendukung, mengubah simpul, dan menetapkan nilai per simpul menengah yang akan diinterpolasi untuk kode fragmen.

Kode program shader berikut mengimplementasikan per-fragment, shading Blinn-Phong. Ini menyatukan banyak dari apa yang telah disajikan dalam bab ini sejauh ini dan mengikatnya ke deskripsi shader dari Bab 4. Buffer simpul interleaved digunakan untuk memuat posisi simpul dan vektor normal. Nilai-nilai ini bermanifestasi dalam vertex shader sebagai atribut array vertex untuk indeks 0 dan indeks 1. Perhitungan bayangan yang terjadi dalam kode shader fragmen dilakukan dalam koordinat kamera (kadang-kadang disebut sebagai eye-space).

Program Blinn-Phong Shader: Vertex Shader

Program empat tahap shader vertex digunakan untuk mentransformasikan simpul masuk menggunakan matriks Mmodel dan Mcam menjadi koordinat kamera. Ini juga menggunakan matriks normal, $(M^{-1})^T$, untuk mengubah atribut vektor normal yang masuk dengan tepat. Shader vertex mengeluarkan tiga variabel ke tahap fragmen:

- **Normal**. Vektor normal simpul yang ditransformasikan ke dalam sistem koordinat kamera.
- **h**. Setengah-vektor dibutuhkan untuk Blinn-Phongshading.
- **l**. Arah cahaya diubah menjadi sistem koordinat kamera.

Masing-masing variabel ini kemudian akan tersedia untuk komputasi fragmen, setelah menerapkan interpolasi barycentric di tiga simpul dalam segitiga. Lampu titik tunggal digunakan dengan program shader ini. Posisi dan intensitas cahaya dikomunikasikan ke vertex dan fragment shader menggunakan variabel seragam. Data ringan dideklarasikan menggunakan qualifer struct GLSL, yang memungkinkan variabel untuk dikelompokkan

bersama dengan cara yang berarti. Meskipun tidak disajikan di sini, GLSL mendukung array dan struktur kontrol for-loop, sehingga lampu tambahan dapat dengan mudah ditambahkan ke contoh ini.

Semua matriks juga disediakan ke vertex shader menggunakan variabel seragam. Untuk saat ini, kita akan membayangkan bahwa matriks model (atau transformasi lokal) akan diatur ke matriks indentitas. Pada bagian berikut, detail lebih lanjut akan diberikan untuk memperluas bagaimana modelmatrix dapat ditentukan pada host menggunakan GLM.

```
#version 330 core
//
// Blinn-Phong Vertex Shader
//
layout(location=0) in vec3 in_Position;
layout(location=1) in vec3 in_Normal;
out vec4 normal;
out vec3 half;
out vec3 lightdir;
struct LightData {
vec3 position;
vec3 intensity;
};
uniform LightData light;
uniform mat4 projMatrix;
uniform mat4 viewMatrix;
uniform mat4 modelMatrix;
uniform mat4 normalMatrix;

void main(void)
{
// Calculate lighting in eye space: transform the local
// position to world and then camera coordinates.
vec4 pos = viewMatrix * modelMatrix * vec4(in_Position, 1.0);
vec4 lightPos = viewMatrix * vec4(light.position, 1.0);
normal = normalMatrix * vec4(in_Normal, 0.0);
vec3 v = normalize( -pos.xyz );
lightdir = normalize( lightPos.xyz - pos.xyz );
half = normalize( v + lightdir );
gl_Position = projMatrix * pos;
}
```

Fungsi utama vertex shader pertama-tama mengubah posisi dan posisi cahaya menjadi koordinat kamera menggunakan tipe `vec4`. agar sesuai dengan matriks 4x4 dari `mat4`. GLSL. Kami kemudian mengubah vektor normal dan menyimpannya dalam variabel normal vec4 keluar. Vektor pandangan (atau mata) dan vektor arah cahaya kemudian dihitung, yang mengarah ke perhitungan setengah vektor yang dibutuhkan untuk Blinn-Phongshading. Perhitungan akhir menyelesaikan perhitungan

$$\mathbf{v}_{\text{canon}} = \mathbf{M}_{\text{proj}}\mathbf{M}_{\text{cam}}\mathbf{M}_{\text{model}}\mathbf{v}$$

dengan menerapkan matriks proyeksi. Ini kemudian menetapkan koordinat kanonik dari vertex ke GLSL vertex shader output variabel `gl Position`. Setelah ini, simpul berada dalam koordinat klip dan siap untuk rasterisasi.

Program Shader Blinn-Phong: Shader Fragmen

Fragmen shader menghitung model bayangan Blinn-Phong. Ini menerima nilai interpolasi barycentric untuk titik normal, setengah vektor, dan arah cahaya. Perhatikan bahwa variabel-variabel ini ditentukan menggunakan kata kunci in saat mereka masuk dari tahap pemrosesan vertex. Data cahaya juga dibagi dengan shader fragmen menggunakan spesifikasi seragam yang sama yang digunakan dalam shader vertex. Matriks tidak diperlukan sehingga tidak ada variabel matriks seragam yang dideklarasikan. Sifat material untuk model geometri dikomunikasikan melalui variabel seragam untuk menentukan k_a, k_d, k_s, I_a, dan p. Bersama-sama, data memungkinkan shader fragmen untuk menghitung Persamaan 4.3:

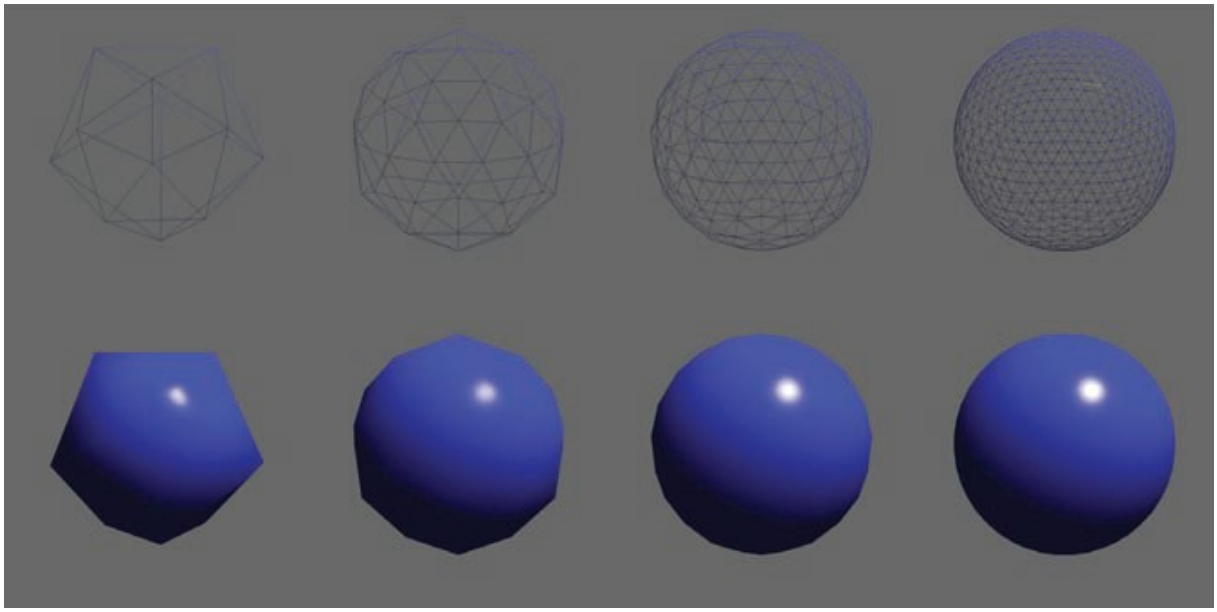
$$L = k_a I_a + k_d I max(0, \mathbf{n} \cdot \mathbf{l}) + k_s I max(0, \mathbf{n} \cdot \mathbf{h})^p$$

pada setiap fragmen.

```
#version 330 core
//
// Blinn-Phong Fragment Shader
//
in vec4 normal;
in vec3 half;
in vec3 lightdir;
layout(location=0) out vec4 fragmentColor;
struct LightData {
vec3 position;
vec3 intensity;
};
uniform LightData light;
uniform vec3 Ia;
uniform vec3 ka, kd, ks;
uniform float phongExp;
void main(void)
{
vec3 n = normalize(normal.xyz);
vec3 h = normalize(half);
vec3 l = normalize(lightdir);
vec3 intensity = ka * Ia
+ kd * light.intensity * max( 0.0, dot(n, l) )
+ ks * light.intensity
* pow( max( 0.0, dot(n, h) ), phongExp );
fragmentColor = vec4( intensity, 1.0 );
}
```

Shader fragmen menulis intensitas yang dihitung ke buffer output warna fragmen. Gambar 17.6 mengilustrasikan beberapa contoh yang menunjukkan efek bayangan perfragmen di berbagai tingkat tessellation pada model geometris. Fragmen shader ini memperkenalkan penggunaan struktur untuk menahan variabel seragam. Perlu dicatat bahwa mereka adalah struktur yang ditentukan pengguna, dan dalam contoh ini, tipe LightData hanya memegang posisi cahaya dan intensitasnya. Dalam kode host, variabel seragam dalam struktur direferensikan menggunakan nama variabel yang sepenuhnya memenuhi syarat ketika meminta pegangan ke variabel seragam, seperti dalam:

```
lightPosID = shader.createUniform( "light.position" );
lightIntensityID = shader.createUniform( "light.intensity" );
```



Gambar 17.6. Bayangan per-fragmen diterapkan di seluruh peningkatan tessellation dari bola subdivisi. Sorotan specular terlihat dengan tessellations yang lebih rendah.

17.15 SHADER NORMAL

Setelah Anda memiliki program shader yang berfungsi, sepertiBlinn-Phong yang disajikan di sini, mudah untuk mengembangkan ide Anda dan mengembangkan newshader. Mungkin juga membantu untuk mengembangkan satu set shader yang sangat spesifik untuk bugging. Salah satu shader tersebut adalah program shader biasa. Bayangan normal sering membantu untuk memahami apakah geometri yang masuk diatur dengan benar atau apakah perhitungannya benar. Dalam contoh ini, vertex shader tetap sama. Hanya shader fragmen yang berubah:

```
#version 330 core
in vec4 normal;
layout(location=0) out vec4 fragmentColor;
void main(void)
{
// Notice the use of swizzling here to access
// only the xyz values to convert the normal vec4
// into a vec3 type!
vec3 intensity = normalize(normal.xyz) * 0.5 + 0.5;
fragmentColor = vec4( intensity, 1.0 );
}
```

Shader mana pun yang Anda mulai buat, pastikan untuk mengomentarinya! Spesifikasi GLSL memungkinkan komentar untuk disertakan dalam kode shader, jadi tinggalkan beberapa detail yang dapat memandu Anda nanti.

Gambar 17.7. Gambar dijelaskan dari kiri ke kanan. Orientasi lokal default naga, berbaring miring. Setelah rotasi –90 derajat terhadap X, naga itu tegak tetapi masih terpusat di sekitar titik asal. Akhirnya, setelah menerapkan terjemahan 1.0 di Y , naga siap untuk membuat instance.

17.16 MESH DAN INSTANCING

Setelah shader dasar berfungsi, menarik untuk mulai membuat scene yang lebih kompleks. Beberapa file model 3D mudah dimuat dan yang lain membutuhkan lebih banyak usaha. Salah satu representasi file 3Dobject sederhana adalah format OBJ. OBJ adalah format yang banyak digunakan dan beberapa kode tersedia untuk memuat jenis file ini. Susunan mekanisme struct yang disajikan sebelumnya berfungsi dengan baik untuk memuat data OBJ pada host. Kemudian dapat dengan mudah ditransfer ke dalam objek array VBO dan vertex.

Banyak model 3D didefinisikan dalam sistem koordinat lokalnya sendiri dan memerlukan berbagai transformasi untuk menyelaraskannya dengan sistem koordinat OpenGL. Misalnya, ketika file OBJStanfordDragon's dimuat ke dalam sistem koordinat OpenGL, ia tampak berbaring miring di titik asal. Menggunakan GLM, kita dapat membuat transformasi model untuk menempatkan objek dalam scene kita. Untuk model naga, ini berarti memutar -90 derajat terhadap X, dan kemudian menerjemahkan ke atas dalam Y . Transformasi model yang efektif menjadi

$$\mathbf{M}_{\text{model}} = \mathbf{M}_{\text{translate}} \mathbf{M}_{\text{rotX}},$$

dan naga disajikan tegak dan di atas bidang tanah, seperti yang ditunjukkan pada Gambar 17.7. Untuk melakukan ini, kami menggunakan beberapa fungsi dari GLM untuk menghasilkan transformasi model lokal:

- **glm::translate** membuat matriks terjemahan.
- **glm::rotate** membuat matriks rotasi, yang ditentukan dalam derajat atau radian tentang sumbu tertentu.
- **glm::scale** membuat matriks skala.

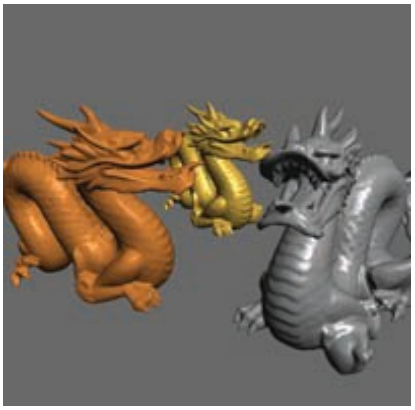
Kita dapat menerapkan fungsi-fungsi ini untuk membuat transformasi model dan meneruskan matriks model ke shader menggunakan variabel seragam. Blinn-Phong vertex shader berisi instruksi yang menerapkan transformasi lokal ke vertex yang masuk. Kode berikut menunjukkan bagaimana model naga dirender:

```
glUseProgram( BlinnPhongShaderID );
// Describe the Local Transform Matrix
glm::mat4 modelMatrix = glm::mat4(1.0); // Identity Matrix
modelMatrix = glm::translate(modelMatrix, glm::vec3(0.0f,
1.0f, ←_
0.0f));
```

```
float rot = (-90.0f / 180.0f) * M_PI;
modelMatrix = glm::rotate(modelMatrix, rot, glm::vec3(1, 0,
0));
// Set the Normal Matrix
glm::mat4 normalMatrix =
glm::transpose( glm::inverse( viewMatrix←_
* modelMatrix ) );
// Pass the matrices to the GPU memory
glUniformMatrix4fv(nMatID, 1, GL_FALSE, glm::value_ptr(←_
normalMatrix));
glUniformMatrix4fv(pMatID, 1, GL_FALSE,
glm::value_ptr(projMatrix←_
));
glUniformMatrix4fv(vMatID, 1, GL_FALSE,
glm::value_ptr(viewMatrix←_
));
glUniformMatrix4fv(mMatID, 1, GL_FALSE, glm::value_ptr(←_
modelMatrix));
// Set material for this object
glm::vec3 kd( 0.2, 0.2, 1.0 );
glm::vec3 ka = kd * 0.15f;
glm::vec3 ks( 1.0, 1.0, 1.0 );
float phongExp = 32.0;
glUniform3fv(kaID, 1, glm::value_ptr(ka));
glUniform3fv(kdID, 1, glm::value_ptr(kd));
glUniform3fv(ksID, 1, glm::value_ptr(ks));
glUniform1f(phongExpID, phongExp);
// Process the object and note that modelData.size() holds
// the number of vertices, not the number of triangles!
glBindVertexArray(VAO);
glDrawArrays(GL_TRIANGLES, 0, modelData.size());
glBindVertexArray(0);
glUseProgram( 0 );
```

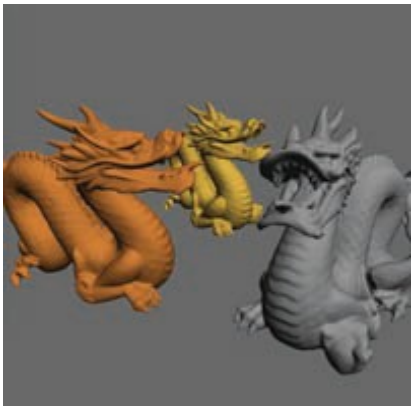
17.17 *MODEL INSTANCE*

Instance dengan OpenGL diimplementasikan secara berbeda dari instance dengan ray tracer. Dengan ray tracer, sinar ditransformasikan secara terbalik ke dalam ruang lokal objek menggunakan matriks transformasi model. Dengan OpenGL, pembuatan instance dilakukan dengan memuat satu salinan objek sebagai objek array vertex (dengan objek buffer vertex terkait), dan kemudian menggunakan kembali geometri sesuai kebutuhan. Seperti pelacak sinar, hanya satu objek yang dimuat ke dalam memori, tetapi banyak yang dapat direnderer.



Gambar 17.8. Hasil dari menjalankan program shader Blinn-Phong pada tiga naga menggunakan variabel uniform untuk menentukan sifat material dan transformasi.

ModernOpenGL dengan baik mendukung gaya instancing ini karenavertexshaders dapat (dan harus) menghitung transformasi yang diperlukan untuk mengubah simpul menjadi koordinat klip. Dengan menulis shader umum yang menyematkan transformasi ini, seperti yang disajikan dengan shader verteks Blinn-Phong, model dapat dirender dengan geometri lokal dasar yang sama. Jenis material dan transformasi yang berbeda dapat ditanyakan dari struktur kelas tingkat yang lebih tinggi untuk mengisi variabel seragam yang diteruskan dari host ke perangkat setiap frame. Animasi dan kontrol interaktif juga mudah dibuat karena transformasi model dapat berubah seiring waktu di seluruh iterasi loop tampilan. Gambar 17.8 dan 17.9 menggunakan jejak memori satu naga, namun menampilkan tiga model naga yang berbeda ke layar.



Gambar 17.9. Mengatur variabel uniform ks = (0 , 0, 0) pada program shader BlinnPhong menghasilkan shading Lambertian.

17.18 OBJEK TEKSTUR

Tekstur efektif berarti memanipulasi efek visual dengan OpenGLshaders. Mereka digunakan secara ekstensif dengan banyak algoritme grafis berbasis perangkat keras dan OpenGL mendukungnya secara native dengan objek Tekstur. Seperti konsep OpenGL sebelumnya, objek tekstur harus dialokasikan dan diinisialisasi dengan menyalin data di host ke memori GPU dan menyetel status OpenGL. Koordinat tekstur sering diintegrasikan ke dalam objek buffer titik dan diteruskan sebagai atribut titik ke program shader. Fragment shader biasanya melakukan fungsi pencarian tekstur menggunakan koordinat tekstur interpolasi yang diteruskan dari vertex shader.

Tekstur agak sederhana untuk ditambahkan ke kode Anda jika Anda sudah memiliki objek array shader dan vertex yang berfungsi. Teknik OpenGL standar untuk membuat objek

pada perangkat keras digunakan dengan tekstur. Namun, sumber data tekstur harus ditentukan terlebih dahulu. Data dapat dimuat dari file (misalnya, PNG, JPG, EXR, atau format file gambar HDR) atau dibuat secara prosedural di host (dan bahkan di GPU). Setelah data dimuat ke memori host, data disalin ke memori GPU, dan secara opsional, status OpenGL yang terkait dengan tekstur dapat diatur. Data tekstur OpenGL dimuat sebagai buffer linier memori yang berisi data yang digunakan untuk tekstur. Pencarian tekstur pada perangkat keras dapat berupa kueri 1D, 2D, atau 3D. Terlepas dari kueri dimensi tekstur, data dimuat ke memori dengan cara yang sama, menggunakan memori yang dialokasikan secara linier pada host. Dalam contoh berikut, proses pemuatan data dari file gambar (atau pembuatannya secara prosedural) diserahkan kepada pembaca, tetapi nama variabel disediakan yang cocok dengan apa yang mungkin ada jika gambar dimuat (mis., imgData, imgWidth, imgHeight).

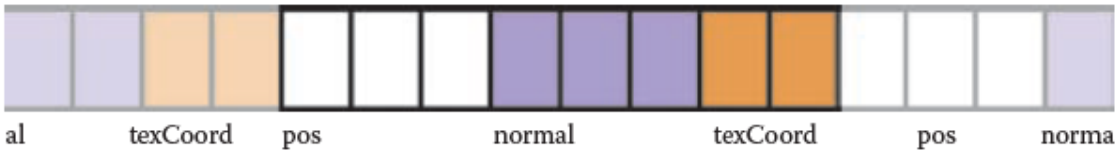
```
float *imgData = new float[ imgHeight * imgWidth * 3 ];
...
GLuint texID;
glGenTextures(1, &texID);
glBindTexture(GL_TEXTURE_2D, texID);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER,
GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
GL_LINEAR);
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, imgWidth, imgHeight, 0,
GL_RGB, GL_FLOAT, imgData);
glBindTexture(GL_TEXTURE_2D, 0);
delete [] imgData;
```

Contoh yang disajikan di sini menyoroti untuk menyiapkan dan menggunakan tekstur dasar 2DOpenGL dengan program shader. Proses untuk membuat objek OpenGL seharusnya sudah familiar sekarang. Pegangan (atau ID) harus dibuat pada perangkat untuk merujuk ke objek tekstur (misalnya, dalam hal ini, texID). Id kemudian diikat untuk memungkinkan operasi status tekstur berikutnya memengaruhi status tekstur. Ada sekumpulan status tekstur dan parameter OpenGL yang cukup luas yang memengaruhi interpretasi koordinat tekstur dan pemfilteran pencarian tekstur. Berbagai target tekstur ada dengan perangkat keras grafis. Dalam hal ini, target tekstur ditetapkan sebagai GL TEXTURE 2D dan akan muncul sebagai argumen pertama dalam fungsi yang berhubungan dengan tekstur. Untuk OpenGL target tekstur khusus ini menyiratkan bahwa koordinat tekstur akan ditentukan dalam cara yang dinormalisasi perangkat (yaitu, dalam kisaran [0,1]). Selain itu, data tekstur harus dialokasikan sehingga dimensi lebar dan tinggi adalah pangkat dua (mis., 512×512, 1024×512, dll.). Parameter tekstur disetel untuk tekstur yang terikat saat ini dengan memanggil glTexParameter. Tanda tangan untuk fungsi ini mengambil berbagai bentuk tergantung pada jenis data yang ditetapkan. Dalam hal ini, koordinat tekstur akan dijepit oleh perangkat keras ke kisaran eksplisit [0,1]. Filter pembesar dan pembesar objek tekstur OpenGL diatur untuk menggunakan pemfilteran linier (bukan tetangga terdekat - GL TERDEKAT) secara otomatis saat melakukan pencarian tekstur. Bab 11 memberikan detail substansial tentang tekstur, termasuk detail tentang pemfilteran yang dapat terjadi dengan pencarian tekstur. Perangkat keras grafis dapat melakukan banyak operasi ini secara otomatis dengan menyetel status tekstur terkait.

Terakhir, panggilan ke glTexImage2D melakukan salinan host ke perangkat untuk tekstur. Ada beberapa argumen untuk fungsi ini, tetapi operasi keseluruhannya adalah mengalokasikan ruang pada kartu grafis (misalnya, lebar gambar Ximg Tinggi) dari tiga titik (argumen ke-7 dan ke-8: GL RGB dan GL FLOAT) dan menyalin data tekstur linier ke perangkat keras (misalnya, penunjuk imgData). Argumen yang tersisa berhubungan dengan

pengaturan tingkat detail mipmap (argumen ke-2), menentukan format internal (misalnya, GL RGB argumen ke-3) dan apakah teksturnya memiliki batas atau tidak (argumen ke-6).

Saat mempelajaritekstur OpenGL aman untuk menjaga laut, defaultnya tercantum di sini. Namun, pembaca disarankan untuk mempelajari lebih lanjut tentang peta mip dan potensi format internal tekstur karena diperlukan pemrosesan grafis yang lebih maju.



Gambar 17.10. Tata letak data setelah menambahkan koordinat tekstur ke buffer vertex. Setiap blok mewakili GLfloat, yaitu 4 byte. Posisi dikodekan sebagai blok putih, normal sebagai ungu, dan koordinat tekstur sebagai oranye.

Alokasi dan inisialisasi objek tekstur terjadi dengan kode di atas. Modifikasi tambahan harus dilakukan pada buffer vertex dan objek array vertex untuk menghubungkan koordinat tekstur yang benar dengan deskripsi geometrik. Mengikuti contoh sebelumnya, koordinat penyimpanan untuk tekstur merupakan modifikasi langsung ke struktur data vertex:

```
struct vertexData
{
glm::vec3 pos;
glm::vec3 normal;
glm::vec2 texCoord;
};
```

Akibatnya, objek buffer verteks akan bertambah besar dan penyisipan koordinat tekstur akan membutuhkan perubahan langkah dalam spesifikasi atribut verteks untuk objek vertexarray. Gambar 17.10 mengilustrasikan interleaving dasar data dalam buffer vertex.

```
glBindBuffer(GL_ARRAY_BUFFER, m_triangleVBO[0]);
glEnableVertexAttribArray(0);
glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 8 *
sizeof(GLfloat), 0);
glEnableVertexAttribArray(1);
glVertexAttribPointer(1, 3, GL_FLOAT, GL_FALSE, 8 *
sizeof(GLfloat), (const GLvoid *)12);
glEnableVertexAttribArray(2);
glVertexAttribPointer(2, 2, GL_FLOAT, GL_FALSE, 8 *
sizeof(GLfloat), (const GLvoid *)24);
glBindVertexArray(0);
```

Dengan potongan kode di atas, koordinat tekstur ditempatkan pada lokasi atribut verteks 2. Perhatikan perubahan ukuran dari ukuran koordinat tekstur (misalnya, argumen ke-2 dari `glVertexAttribPointer` 2 untuk koordinat tekstur yang bertepatan dengan jenis vec2 dalam struktur). Pada titik ini, semua inisialisasi akan selesai untuk objek tekstur.

Objek tekstur harus diaktifkan (atau diikat) sebelum merender objek array vertex dengan shader Anda. Secara umum, perangkat keras grafis memungkinkan penggunaan beberapa objek tekstur saat menjalankan program shader. Dengan cara ini, program shader dapat menerapkan tekstur dan efek visual yang canggih. Jadi, untuk mengikat tekstur untuk digunakan dengan shader, itu harus dikaitkan dengan salah satu dari banyak unit tekstur yang berpotensi. Unit tekstur mewakili mekanisme di mana shader dapat menggunakan banyak

tekstur. Dalam contoh di bawah ini, hanya satu tekstur yang digunakan sehingga unit tekstur 0 akan dibuat aktif dan terikat pada tekstur kita.

Fungsi yang mengaktifkan unit tekstur adalah `glActiveTexture`.. Satu-satunya argumennya adalah unit tekstur untuk membuat aktif. Ini diatur ke `GL_TEXTURE0` di bawah, tetapi bisa juga `GL_TEXTURE1` atau `GL_TEXTURE2`, misalnya, jika beberapa tekstur diperlukan di shader. Setelah kesatuan tekstur menjadi aktif, sebuah objek tekstur dapat diikat dengan menggunakan panggilan `glBindTexture`.

```
glUseProgram(shaderID);
glActiveTexture(GL_TEXTURE0);
glBindTexture(GL_TEXTURE_2D, texID);
glUniform1i(texUnitID, 0);
glBindVertexArray(VAO);
glDrawArrays(GL_TRIANGLES, 0, 3);
glBindVertexArray(0);
glBindTexture(GL_TEXTURE_2D, 0);
glUseProgram(0);
```

Sebagian besar kode di atas seharusnya merupakan ekstensi logis dari apa yang telah Anda kembangkan sejauh ini. Perhatikan panggilan ke `glUniform` sebelum merender objek array vertex. Dalam pemrograman perangkat keras grafis modern, shader melakukan pekerjaan pencarian tekstur dan pencampuran, dan oleh karena itu, harus memiliki data tentang unit tekstur mana yang menyimpan tekstur yang digunakan dalam shader. Unit tekstur aktif dipasok ke shader menggunakan variabel seragam. Dalam hal ini, 0 diatur untuk menunjukkan bahwa pencarian tekstur akan berasal dari unit tekstur 0. Ini akan diperluas di bagian berikut.

Pencarian Tekstur di Shader

Program shader melakukan pencarian dan pencampuran apa pun yang mungkin diperlukan. Sebagian besar dari perhitungan itu biasanya masuk ke dalam shader fragmen, tetapi shader vertex sering melakukan perhitungan fragmen dengan melewatkan koordinat tekstur ke shader fragmen. Dengan cara ini, koordinat tekstur akan diinterpolasi dan memberikan pencarian per-fragmen dari data tekstur. Perubahan sederhana diperlukan untuk menggunakan program tekstur data di shader. Menggunakan shader Blinn-Phongvertex yang disediakan sebelumnya, hanya diperlukan tiga perubahan:

1. Koordinat tekstur adalah atribut per-vertex yang disimpan di dalam objek array vertex. Mereka terkait dengan indeks atribut vertex 2 (atau lokasi 2).

```
layout(location=2) in vec2 in_TexCoord;
```

2. Fragmen shader akan melakukan pencarian tekstur dan akan membutuhkan koordinat tekstur yang diinterpolasi. Variabel ini akan ditambahkan sebagai variabel output yang diteruskan ke shader fragmen.

```
out vec2 tCoord;
```

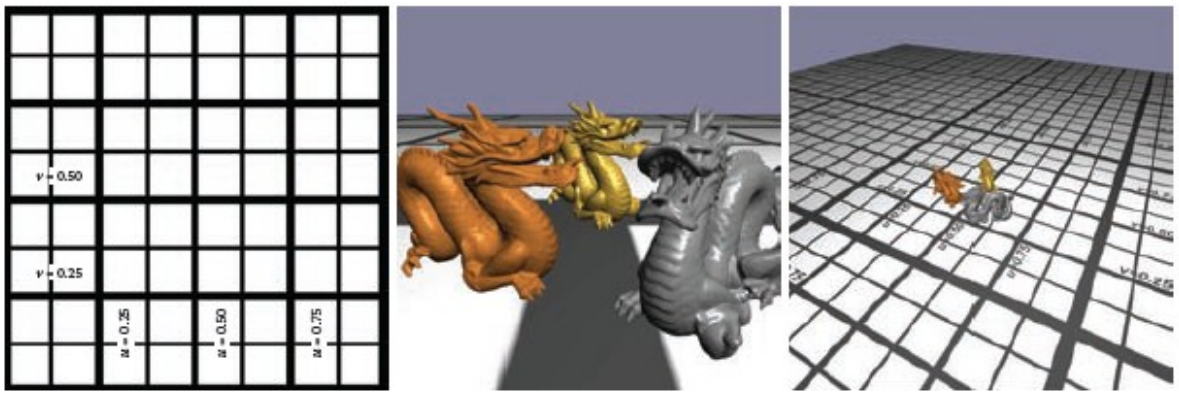
3. Salin atribut titik masuk ke variabel output di fungsi utama.

```
// Pass the texture coordinate to the fragment
shader
tCoord = in_TexCoord;
```

Fragmen shader juga membutuhkan perubahan sederhana. Pertama, koordinat tekstur interpolasi masuk yang dilewatkan dari vertexshader harus dideklarasikan. Juga ingat bahwa

variabel seragam harus menyimpan unit tekstur yang teksturnya terikat. Ini harus dikomunikasikan ke shader sebagai tipe sampler. Sampler adalah jenis bahasa bayangan yang memungkinkan pencarian data dari objek tekstur tunggal. Dalam contoh ini, hanya satu sampler yang diperlukan, tetapi dalam shader di mana beberapa pencarian tekstur digunakan, beberapa variabel sampler akan digunakan. Ada juga beberapa jenis sampler tergantung pada jenis objek tekstur. Dalam contoh yang disajikan di sini, tipe `GL_TEXTURE 2D` digunakan untuk membuat status tekstur. Sampler terkait dalam shader fragmen adalah dari jenis `sampler2D`. Dua deklarasi variabel berikut harus ditambahkan ke fragmentshader:

```
in vec2 tCoord;
uniform sampler2D textureUnit;
```



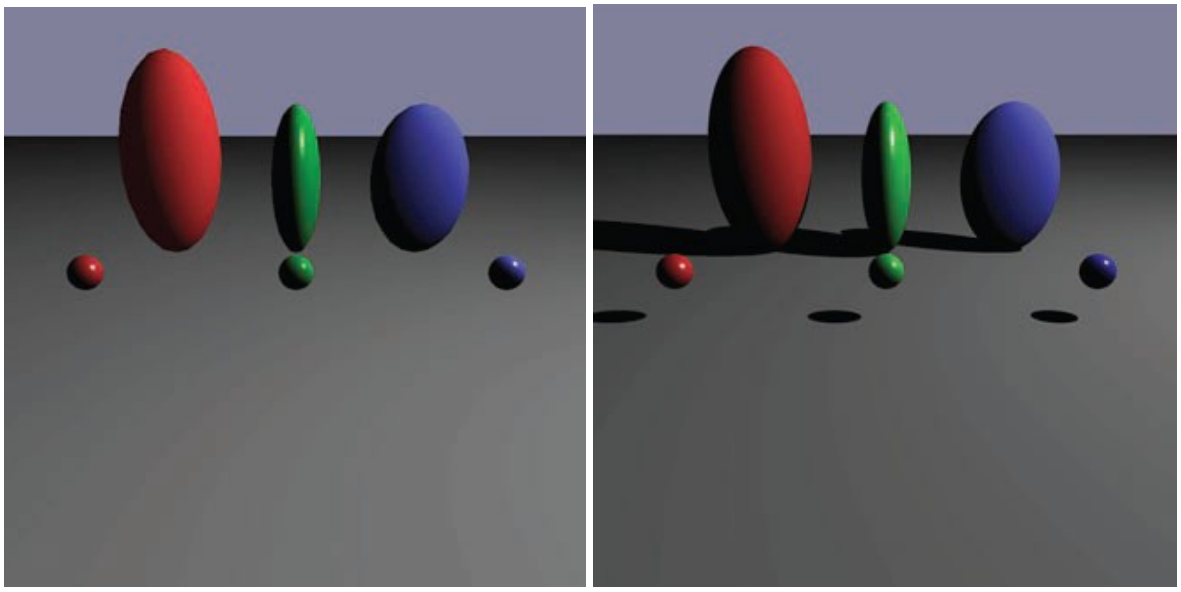
Gambar 17.11. Gambar paling kiri menunjukkan tekstur, gambar 1024 × 1024 piksel. Gambar tengah menunjukkan scene dengan tekstur yang diterapkan menggunakan koordinat tekstur dalam kisaran [0,1] sehingga hanya satu gambar yang ditempatkan pada bidang dasar. Gambar paling kanan memodifikasi parameter tekstur sehingga `GL_REPEAT` digunakan untuk `GL_TEXTURE_WRAP_S` dan `GL_TEXTURE_WRAP_T` dan koordinat tekstur berkisar dari [0,5]. Hasilnya adalah tekstur tiling berulang lima kali di kedua dimensi tekstur.

Modifikasi terakhir masuk ke fungsi utama kode shader fragmen. Tekstur disampel menggunakan fungsi pencarian tekstur GLSL dan (dalam hal ini), menggantikan koefisien difus dari geometri. Argumen pertama untuk tekstur mengambil jenis sampel yang menampung unit tekstur tempat tekstur terikat. Argumen kedua adalah koordinat tekstur. Fungsi mengembalikan tipe `vec4`. Dalam cuplikan kode di bawah ini, tidak ada nilai alfa yang digunakan dalam perhitungan akhir sehingga nilai pencarian tekstur yang dihasilkan dipilih berdasarkan komponen hanya untuk komponen RGB. Koefisien difus dari pencarian tekstur diatur ke tipe vec3 yang digunakan dalam persamaan iluminasi.

```
vec3 kdTexel = texture(textureUnit, tCoord).rgb;
vec3 intensity = ka * Ia + kdTexel * light.intensity
* max( 0.0, dot(n, l) ) + ks * light.intensity
* pow( max( 0.0, dot(n, h) ), phongExp );
```

Gambar 17.11 mengilustrasikan hasil dari penggunaan modifikasi shader ini. Gambar paling kanan dalam gambar memperluas kode contoh dengan mengaktifkan tiling tekstur dengan status OpenGL. Perhatikan bahwa perubahan ini hanya dilakukan dalam kode host dan shader tidak berubah. Untuk mengaktifkan tiling ini, yang memungkinkan koordinat tekstur di luar rentang normal perangkat, parameter tekstur untuk `GL_TEXTURE_WRAP_S` dan `GL_TEXTURE_WRAP_T` diubah dari `GL_CLAMP_to_GL_REPEAT`. Selain itu, kode host yang mengatur koordinat tekstur sekarang berkisar dari [0,5].

Sebagai catatan tambahan, target tekstur lain yang mungkin berguna untuk berbagai aplikasi adalah `GL_TEXTURE_RECTANGLE`. Persegi panjang tekstur adalah objek tekstur unik yang tidak dibatasi dengan persyaratan gambar kekuatan dua lebar dan tinggi dan menggunakan koordinat tekstur yang tidak dinormalisasi. Selain itu, mereka tidak mengizinkan tiling berulang. Jika tekstur persegi panjang digunakan, shader harus mereferensikannya menggunakan tipe sampler khusus: `sampler2DRect`..



Gambar 17.12. Di sebelah kiri, satu bola tesselated dibuat enam kali menggunakan transformasi model yang berbeda untuk membuat scene ini menggunakan program shader per-fragmen. Gambar di sebelah kanan dirender menggunakan pelacak sinar Whitted basi. Perhatikan efek bayangan pada persepsi scene. Bayangan per-fragmen memungkinkan sorotan specular menjadi serupa di kedua gaya rendering.

17.19 DESAIN BERORIENTASI OBJEK UNTUK PEMROGRAMAN *HARDWARE* GRAFIS

Seiring dengan meningkatnya keakraban Anda dengan OpenGL, menjadi bijaksana untuk merangkum sebagian besar dari apa yang dijelaskan dalam bab ini ke dalam struktur kelas yang dapat berisi data spesifik model dan memungkinkan rendering berbagai objek dalam scene. Misalnya, pada Gambar 17.12, satu bola dibuat enam kali untuk membuat tiga ellipsoid dan tiga bola. Setiap model menggunakan geometri dasar yang sama namun memiliki sifat material dan transformasi model yang berbeda. Jika Anda telah mengikuti seluruh buku dan menerapkan ray tracer, seperti yang dijelaskan dalam Bab 4, maka kemungkinan implementasi Anda didasarkan pada desain berorientasi objek yang solid. Desain tersebut dapat dimanfaatkan untuk membuat pengembangan program perangkat keras grafis dengan OpenGL lebih mudah. Arsitektur perangkat lunak ray tracer yang khas akan mencakup beberapa kelas yang memetakan langsung ke perangkat keras grafis serta aplikasi rasterisasi perangkat lunak. Kelas dasar abstrak dalam ray tracer yang mewakili permukaan, material, lampu, shader, dan kamera dapat disesuaikan untuk menginisialisasi status perangkat keras grafis, memperbarui status tersebut, dan jika perlu merender data kelas ke framebuffer. Antarmuka untuk fungsi virtual ini mungkin perlu disesuaikan dengan implementasi spesifik Anda, tetapi lintasan pertama yang memperluas desain kelas permukaan mungkin menyerupai berikut ini:

```
class surface
virtual bool initializeOpenGL( )
virtual bool renderOpenGL( glm::mat4&Mp, glm::mat4&Mcam)
```

Melewati matriks proyeksi dan tampilan untuk menghasilkan fungsi yang mampu dan bukan arahan tentang bagaimana matriks ini dikelola. Matriks ini berasal dari kelas kamera yang dapat dimanipulasi dengan menginterpretasikan input keyboard, mouse, atau joystick. Fungsi inisialisasi (setidaknya untuk turunan permukaan) akan berisi objek buffer vertex dan alokasi objek array vertex dan kode inisialisasi. Selain memulai drawarrays untuk setiap objek vertexarray, fungsi render juga perlu mengaktifkan program shader dan meneruskan matriks yang diperlukan ke shader, seperti yang diilustrasikan sebelumnya dalam contoh model naga. Saat Anda bekerja untuk mengintegrasikan algoritma urutan gambar dan urutan objek (perangkat keras dan perangkat lunak) ke dalam kerangka data dasar yang sama, beberapa tantangan desain perangkat lunak akan muncul, sebagian besar terkait dengan akses data dan organisasi. Namun, ini adalah latihan yang sangat berguna untuk menjadi rekayasa perangkat lunak yang mahir untuk pemrograman grafis dan akhirnya mendapatkan pengalaman yang solid dalam menggabungkan algoritma rendering Anda.

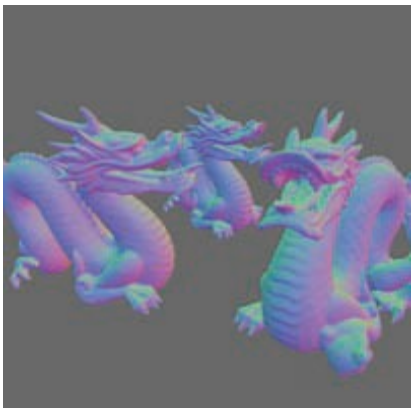
Pembelajaran Lebih Lanjut

Bab ini dirancang untuk memberikan pengantar sekilas ke dalam pemrograman perangkat keras grafis, yang dipengaruhi oleh OpenGL API. Ada banyak arah yang dapat ditempuh oleh pembelajaran berkelanjutan Anda. Banyak topik, seperti objek framebuffer, tekstur renderto, mapping lingkungan, shader geometri, computershader, dan shader iluminasi tingkat lanjut tidak dibahas. Area ini mewakili tahap selanjutnya dalam mempelajari perangkat keras grafis, tetapi bahkan di dalam area yang tercakup, ada banyak arah yang dapat ditempuh untuk mengembangkan pemahaman perangkat keras grafis yang lebih kuat. Pemrograman perangkat keras grafis akan terus berkembang dan berubah. Pembaca yang tertarik harus mengharapkan perubahan ini dan melihat ke dokumen spesifikasi untuk OpenGL dan OpenGL Shading Language untuk lebih banyak detail tentang apa yang mampu dilakukan OpenGL dan bagaimana perangkat keras berhubungan dengan komputasi tersebut.

Pertanyaan yang Sering Diajukan

- Bagaimana cara men-debug program shader?

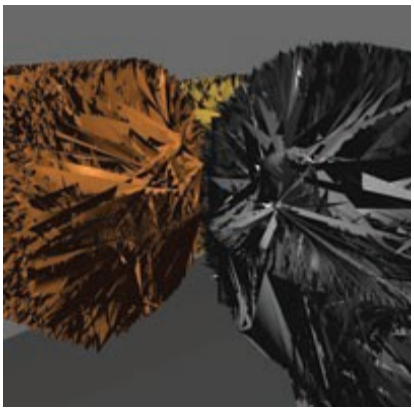
Pada sebagian besar platform, men-debug vertex shader dan fragment shader tidak mudah. Namun, semakin banyak dukungan yang tersedia melalui berbagai driver, ekstensi sistem operasi, dan IDE untuk memberikan informasi terkait kepada pengembang. Ini masih bisa menantang, jadi gunakan shader untuk men-debug kode Anda secara visual. Jika tidak ada yang muncul di layar, coba render vektor normal, setengah vektor, atau apa pun yang memberi Anda gambaran tentang di mana kesalahannya (atau tidak). Gambar 17.13 mengilustrasikan normalshader yang sedang beroperasi. Jika gambar muncul di windows Anda, pastikan gambar tersebut sesuai dengan yang Anda harapkan (lihat Gambar 17.14)!



Gambar 17.13. Menerapkan shader normal ke model kompleks untuk keperluan debugging.

17.20 CATATAN

Ada banyak sumber daya bagus yang tersedia untuk mempelajari lebih lanjut tentang detail teknis yang terkait dengan pemrograman perangkat keras grafis. Titik awal yang baik mungkin adalah dokumen spesifikasi OpenGL dan GLSL. Mereka tersedia secara online gratis di situs webtheopengl.org. Dokumen-dokumen ini akan memberikan rincian lengkap untuk semua versi OpenGL yang berbeda dan sedang berkembang.



Gambar 17.14. Debugging visual penting! Dapatkah Anda menemukan apa yang salah dari gambar atau dari mana memulai debugging? Ketika langkah yang salah diterapkan ke objek array vertex, rendering menjadi serba salah.

17.21 LATIHAN

Bagian dari bab ini secara kasar diatur untuk memandu siswa melalui proses pembuatan aplikasi OpenGL modern. Beberapa upaya ekstra akan diperlukan untuk memahami detail yang berkaitan dengan pengaturan windows dan konteks OpenGL. Namun, harus dimungkinkan untuk mengikuti bagian untuk satu set lab satu jam mingguan:

1. **Lab 1: Penyiapan kode dasar untuk aplikasi OpenGL**. Ini termasuk menginstal driver yang diperlukan dan perangkat lunak terkait seperti GLM dan GLFW. Siswa kemudian dapat menulis kode untuk membuka windows dan menghapus buffer warna.
2. **Lab 2: Membuat shader.** Karena shader dasar diperlukan untuk memvisualisasikan output dalam OpenGL modern, dimulai dengan upaya untuk membuat shader yang sangat mendasar akan sangat membantu. Di lab ini, atau lab, siswa dapat membuat (atau menggunakan) kelas untuk memuat, mengkompilasi, dan menautkan shader ke program shader
3. **Lab3: Membuat klip koordinat segitiga dan bayangan**. Menggunakan kelas shader dari lab sebelumnya, siswa akan menambahkan shader passthrough dan membuat geometri sederhana untuk direnderer.
4. **Lab4: Memperkenalkan GLM.**Memulai GLMuntuk menghasilkan matriks proyeksi dan matriks tampilan untuk melihat scene yang lebih umum, namun sederhana.
5. **Lab 5: Gunakan GLM untuk transformasi lokal**. Siswa dapat memperluas program shader kerja mereka untuk menggunakan transformasi lokal, mungkin menerapkan animasi berdasarkan perubahan transformasi.
6. **Lab6: Pengembangan Shader.** KembangkanLambertian atauBlinn-Phongshaders.
7. **Lab 7: Bekerja dengan bahan.** Siswa dapat menjelajahi properti material tambahan dan gaya rendering dengan program shader yang berbeda.
8. **Lab 8: Memuat model 3D**. Dengan menggunakan kode untuk memuat file OBJ, siswa dapat mengeksplorasi lebih lanjut kemampuan perangkat keras grafis mereka termasuk batas pemrosesan perangkat keras untuk aplikasi waktu nyata.
9. **Lab 9: Tekstur**. Menggunakan PNG (atau format lain), siswa dapat memuat gambar ke perangkat keras dan mempraktikkan berbagai strategi mapping tekstur.
10. **Lab 10: Integrasi dengan kode rendering**. Jika file scene digunakan untuk menggambarkan scene untuk pelacak (atau rasterizer), kode OpenGL siswa dapat

diintegrasikan ke dalam kerangka kerja rendering lengkap menggunakan struktur dan kelas umum untuk membangun sistem yang lengkap.

Daftar ini hanya panduan. Di laboratorium untuk kursus grafis komputer saya, siswa diberikan materi untuk memulai ide minggu ini. Setelah ide dasar mereka bekerja, lab selesai setelah mereka menambahkan putaran atau eksplorasi kreatif ide ke kode mereka. Saat siswa terbiasa dengan pemrograman perangkat keras grafis, mereka dapat menjelajahi bidang minat tambahan, seperti tekstur, render ke tekstur, atau shader dan algoritma grafis yang lebih canggih.

BAB 18

CAHAYA

Dalam bab ini, kita membahas masalah praktis pengukuran cahaya, biasanya disebut radiometri. Istilah yang muncul dalam radiometri mayat pertama kali tampak aneh dan memiliki terminologi dan notasi yang mungkin sulit untuk diluruskan. Namun, karena radiometri sangat mendasar bagi komputergrafis, maka radiometri layak dipelajari sampai meresap. Bab ini juga mencakup fotometri, yang mengambil besaran radiometrik dan menskalakannya untuk memperkirakan seberapa banyak cahaya "berguna" yang ada. Misalnya, lampu hijau mungkin tampak dua kali lebih terang daripada cahaya biru dengan kekuatan yang sama karena mata lebih sensitif terhadap cahaya hijau. Fotometri mencoba untuk mengukur perbedaan tersebut.

18.1 RADIOMETRI

Meskipun kita dapat mendefinisikan satuan radiometrik dalam banyak sistem, kita menggunakan satuan SI (Satuan Sistem Internasional). Satuan SI yang familier mencakup satuan metrik meter (m) dan gram (g). Cahaya pada dasarnya adalah bentuk energi yang merambat, sehingga berguna untuk menentukan satuan energi SI, yaitu joule (J).

18.2 FOTON

Untuk membantu intuisi kami, kami akan menjelaskan radiometri dalam hal koleksi sejumlah besar foton, dan bagian ini menetapkan apa yang dimaksud dengan foto dalam konteks ini. Untuk keperluan bab ini, foton adalah kuantum cahaya yang memiliki posisi, arah rambat, dan panjang gelombang . Agak aneh, satuan SI yang digunakan untuk panjang gelombang adalah nanometer (nm). Ini terutama karena alasan historis, dan 1nm = 10^{-9} m. Satuan lain, angstrom, kadang-kadang digunakan, dan satu nanometer adalah sepuluh angstrom. Sebuah foton juga memiliki kecepatan c yang hanya bergantung pada indeks bias n medium yang dilaluinya. Kadang-kadang frekuensi f = c/λ juga digunakan untuk cahaya. Hal ini sesuai karena tidak seperti λ dan c, f tidak berubah ketika foton dibiaskan menjadi medium dengan indeks bias baru. Ukuran invarian lainnya adalah jumlah energi q yang dibawa oleh foton, yang diberikan oleh hubungan berikut:

Rumus 18.1

$$q = hf = \frac{hc}{\lambda}$$

di mana h = 6 .63 × 10^{-34} Js adalah konstanta Plank. Meskipun besaran ini dapat diukur dalam sistem satuan apa pun, kita akan menggunakan satuan SI bila memungkinkan.

Energi Spektral

Jika kita memiliki banyak koleksi foton, energi totalnya Q dapat dihitung dengan menjumlahkan energi qi dari setiap foton. Pertanyaan yang masuk akal untuk ditanyakan adalah "Bagaimana energi didistribusikan melintasi panjang gelombang?" Cara mudah untuk menjawabnya adalah dengan mempartisi foton ke dalam bin, pada dasarnya membuat histogramnya. Kami kemudian memiliki energi yang terkait dengan interval. Misalnya, kita dapat menghitung semua energi antara λ = 500 nm dan λ = 600 nm dan menjadikannya 10,2 J, dan ini dapat dilambangkan q[500.600] = 10.2. Jika kita membagi interval panjang

gelombang menjadi dua interval 50 nm, kita mungkin menemukan bahwa q[500.550] = 5.2 dan q[550.600] = 5.0. Ini memberitahu kita bahwa ada sedikit lebih banyak energi dalam setengah panjang gelombang pendek dari interval [500.600]. Jika kita membagi menjadi 25 nmb, kita dapat menemukan q[500.525] = 2.5, dan seterusnya. Hal yang menyenangkan tentang sistem ini adalah mudah. Hal buruk tentang itu adalah bahwa pilihan ukuran interval menentukan nomornya.

Sistem yang lebih umum digunakan adalah membagi energi dengan ukuran interval. Jadi, alih-alih q[500.600] = 10,2 kita akan memiliki

$$Q_\lambda[500,600]=\frac{10.2}{100}=0.12\,J\,(nm)^{-1}$$

Pendekatan ini bagus, karena ukuran interval memiliki dampak yang jauh lebih kecil pada ukuran keseluruhan angka. Ide langsungnya adalah mendorong ukuran interval ke nol. Ini bisa menjadi canggung, karena untuk yang cukup kecil, Qλ akan menjadi nol atau besar tergantung pada apakah ada satu foton atau tidak ada foton dalam interval. Ada dua aliran pemikiran untuk memecahkan dilema itu. Pertama-tama mengasumsikan bahwa kecil, tetapi tidak terlalu kecil sehingga sifat kuantum cahaya ikut bermain. Yang kedua adalah mengasumsikan bahwa cahaya adalah kontinum daripada foton individu, jadi turunan sejati dQ/dλ adalah tepat. Kedua cara berpikir tentang itu tepat dan mengarah ke mesin komputasi yang sama. Dalam praktiknya, tampaknya sebagian besar orang yang mengukur cahaya lebih menyukai interval yang kecil, tetapi terbatas, karena itulah yang dapat mereka ukur di laboratorium. Kebanyakan orang yang melakukan teori atau komputasi lebih menyukai interval tak terhingga, karena itu membuat mesin kalkulus tersedia.

Besaran Qλ disebut energi spektral, dan merupakan besaran intensif yang berlawanan dengan besaran luas seperti energi, panjang, atau massa. Besaran intensif dapat dianggap sebagai fungsi densitas yang menyatakan densitas dari besaran ekstensif pada titik yang tidak terhingga. Misalnya, energi Q pada panjang gelombang tertentu mungkin nol, tetapi energi spektral (kerapatan energi) Qλ adalah besaran yang berarti. Contoh yang mungkin lebih familiar adalah bahwa populasi suatu negara mungkin 25 juta, tetapi populasi pada suatu titik di negara itu tidak ada artinya. Namun, kepadatan penduduk yang diukur dalam orang per meter persegi adalah berarti, asalkan diukur pada area yang cukup besar. Sama seperti foton, kepadatan populasi bekerja paling baik jika kita berpura-pura bahwa kita dapat melihat populasi sebagai kontinum di mana kepadatan populasi tidak pernah menjadi granular bahkan ketika areanya kecil.

Kami akan mengikuti konvensi grafis di mana energi spektral hampir selalu digunakan, dan energi jarang digunakan. Ini menghasilkan proliferasi subskrip jika notasi "benar" digunakan. Sebagai gantinya, kami akan menghapus subskrip dan menggunakan Q untuk menunjukkan energi spektral. Hal ini dapat menyebabkan kebingungan ketika orang-orang di luar grafis membaca makalah grafis, jadi waspadalah terhadap masalah standar ini. Intuisi Anda tentang energi spektral mungkin terbantu dengan membayangkan alat pengukur dengan sensor yang mengukur energi cahaya Δq. Jika Anda menempatkan filter berwarna di depan sensor yang hanya memungkinkan cahaya dalam interval $[\lambda-\Delta\lambda/2,\lambda+\Delta\lambda/2]$, maka energi spektral pada λ adalah $Q=\Delta q/\Delta\lambda$.

18.3 KEKUATAN

Hal ini berguna untuk memperkirakan tingkat produksi energi untuk sumber cahaya. Laju ini disebut daya, dan diukur dalam watt, W, yang merupakan nama lain untuk joule per detik. Ini paling mudah dipahami dalam keadaan tunak, tetapi karena daya adalah kuantitas intensif (kepadatan dari waktu ke waktu), ia didefinisikan dengan baik bahkan ketika produksi energi bervariasi dari waktu ke waktu. Satuan daya mungkin lebih familiar, misalnya bola

lampu 100 watt. Bola lampu tersebut menarik sekitar 100J energi setiap detik. Kekuatan cahaya yang dihasilkan sebenarnya akan kurang dari 100 W karena kehilangan panas, dll, tetapi kita masih dapat menggunakan contoh ini untuk membantu memahami lebih banyak tentang foton. Misalnya, kita bisa merasakan berapa banyak foton yang dihasilkan dalam satu detik dengan cahaya 100 W. Misalkan rata-rata foton yang dihasilkan memiliki energi λ = 500 nm foton. Frekuensi fotonis seperti itu

$$f = \frac{c}{\lambda} = \frac{3\,x\,10^{8} ms^{-1}}{500\,x\,10^{-9} m} = 6\,x\,10^{14}\,s^{-1}$$

Energi foton itu adalah hf ≈ 4 × 10^{-19} J. Itu berarti 10^{20} foton yang mengejutkan dihasilkan setiap detik, bahkan jika bola lampu tidak terlalu efisien. Ini menjelaskan mengapa mensimulasikan kamera dengan kecepatan rana yang cepat dan foton yang disimulasikan secara langsung merupakan pilihan yang tidak efisien untuk menghasilkan gambar.

Seperti halnya energi, kami sangat tertarik dengan daya spektral yang diukur dalam $W(nm)^{-1}$. Sekali lagi, meskipun simbol standar formal untuk kekuatan spektral adalah , kami akan menggunakan tanpa subskrip untuk kenyamanan dan konsistensi dengan sebagian besar literatur grafis. Satu hal yang perlu diperhatikan adalah bahwa daya spektral untuk sumber cahaya biasanya lebih kecil dari daya. Misalnya, jika sebuah cahaya memancarkan daya 100 W yang terdistribusi secara merata pada panjang gelombang 400 nm hingga 800 nm, maka daya spektralnya adalah 100 W/400 nm = $0,25W(nm)^{-1}$. Ini adalah sesuatu yang perlu diingat jika Anda mengatur kekuatan spektral sumber cahaya dengan tangan untuk keperluan debugging.

Alat pengukur energi spektral pada bagian terakhir dapat dimodifikasi dengan mengambil daerah dengan ashutter yang terbuka untuk selang waktu t yang berpusat pada waktu t. Kekuatan spektral kemudian akan menjadi =Δq/(ΔtΔλ).

Penyinaran

Kuantitas radiasi muncul secara alami jika Anda mengajukan pertanyaan "Berapa banyak cahaya yang mengenai titik ini?" Tentu saja jawabannya adalah “tidak ada”, dan sekali lagi kita harus menggunakan fungsi kepadatan. Jika titik tersebut berada pada suatu permukaan, wajar jika menggunakan luas untuk mendefinisikan fungsi kerapatan kita. Kami memodifikasi perangkat dari bagian terakhir untuk memiliki sensor area A terbatas yang lebih kecil dari medan cahaya yang diukur. Penyinaran spektral H hanyalah daya per satuan luas Δ∅/ΔA. Sepenuhnya diperluas ini adalah

$$H = \frac{\Delta q}{\Delta A \Delta t \Delta \lambda}$$

Jadi, satuan radiasi penuh adalah $Jm^{-2}s^{-1}(nm)^{-1}$. Perhatikan bahwa satuan SI untuk pancaran meliputi invers-meter-squared untuk luas dan inverse-nanometer untuk panjang gelombang. Inkonsistensi yang tampak ini (menggunakan nanometer dan meter) muncul karena satuan alami untuk luas dan panjang gelombang cahaya tampak.

Ketika cahaya keluar dari permukaan, misalnya ketika dipantulkan, kuantitas yang sama dengan irradiansi disebut eksitasi radiasi, E. Adalah berguna untuk memiliki kata yang berbeda untuk cahaya datang dan cahaya keluar, karena titik yang sama berpotensi memiliki irradiansi dan eksitasi yang berbeda.

Cahaya

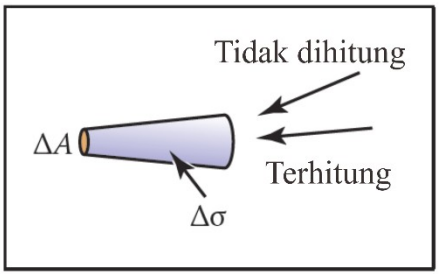
Meskipun penyinaran memberitahu kita berapa banyak cahaya yang tiba di suatu titik, penyinaran memberitahu kita sedikit tentang arah datangnya cahaya. Untuk mengukur sesuatu yang analog dengan apa yang kita lihat dengan mata kita, kita harus mampu mengasosiasikan "seberapa banyak cahaya" dengan arah tertentu. Kita dapat membayangkan alat sederhana

untuk mengukur besaran seperti itu (Gambar 18.1). Kami menggunakan pengukur radiasi kecil dan menambahkan "baffler" berbentuk kerucut yang membatasi cahaya yang mengenai penghitung ke berbagai sudut dengan sudut padat . Respon detektor adalah sebagai berikut:

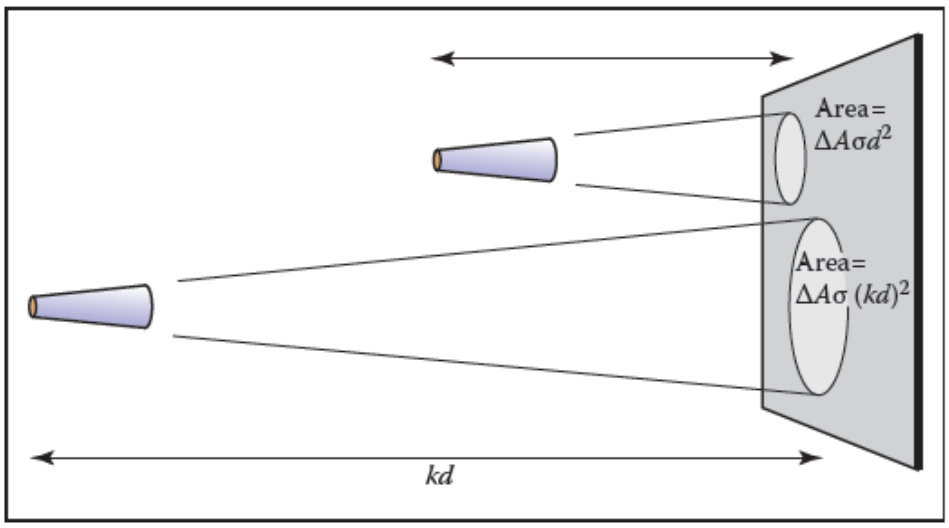
$$respon = \frac{\Delta H}{\Delta \sigma}$$

$$¿ \frac{\Delta q}{\Delta A \Delta \sigma \Delta t \Delta \lambda}$$

Ini adalah pancaran spektral cahaya yang bepergian di ruang angkasa. Sekali lagi, kita akan menghilangkan "spektral" dalam diskusi kita dan menganggapnya implisit.

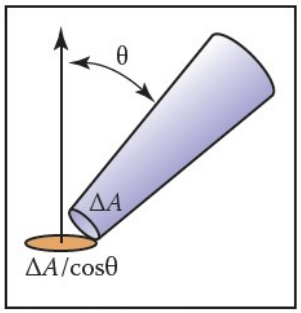


Gambar 18.1. Dengan menambahkan penutup mata yang hanya menunjukkan sudut $\Delta\sigma$ padat kecil ke detektor penyinaran, kita mengukur pancaran.



Gambar 18.2. Sinyal yang diterima detektor pancaran tidak bergantung pada jarak ke permukaan yang diukur. Gambar ini mengasumsikan bahwa detektor menunjuk ke area di permukaan yang memancarkan cahaya dengan cara yang sama.

Radiance adalah apa yang biasanya kita komputasi dalam program grafis. Sifat pancaran yang luar biasa adalah bahwa ia tidak berubah sepanjang garis dalam ruang. Untuk melihat mengapa hal ini benar, periksa dua detektor pancaran yang keduanya melihat ke permukaan seperti yang ditunjukkan pada Gambar 18.2. Asumsikan bahwa garis-garis yang diamati detektor cukup berdekatan sehingga permukaan memancarkan/memantulkan cahaya “sama” di kedua area yang diukur. Karena luas permukaan yang dijadikan sampel sebanding dengan kuadrat jarak, dan karena cahaya yang mencapai detektor berbanding terbalik dengan kuadrat jarak, kedua detektor harus memiliki pembacaan yang sama.



Gambar 18.3. Penyinaran pada permukaan yang ditunjuk oleh kerucut lebih kecil daripada yang diukur pada detektor dengan faktor kosinus.

Hal ini berguna untuk mengukur pancaran yang mengenai permukaan. Kita dapat memikirkan menempatkan penyekat kerucut dari detektor pancaran pada suatu titik di permukaan dan mengukur pancaran H pada permukaan yang berasal dari arah di dalam kerucut (Gambar 18.3). Perhatikan bahwa "detektor" permukaan tidak sejajar dengan kerucut. Untuk alasan ini, kita perlu menambahkan suku koreksi kosinus pada definisi pancaran kita:

$$respon = \frac{\Delta H}{\Delta \sigma \cos\theta}$$

$$¿ \frac{\Delta q}{\Delta A \cos\theta \Delta \sigma \Delta t \Delta \lambda}$$

Seperti halnya penyinaran dan pancaran pancaran, akan berguna untuk membedakan antara pancaran pancaran pada suatu titik di permukaan dan pancaran dari titik tersebut. Istilah untuk konsep-konsep ini kadang-kadang digunakan dalam literatur grafis adalah pancaran permukaan Ls untuk pancaran (meninggalkan) permukaan, dan pancaran medan Lf untuk insiden pancaran pada permukaan. Keduanya membutuhkan suku cosinus, karena keduanya sesuai dengan konfigurasi pada Gambar 18.3:

$$L_s = \frac{\Delta E}{\Delta \sigma \cos\theta}$$

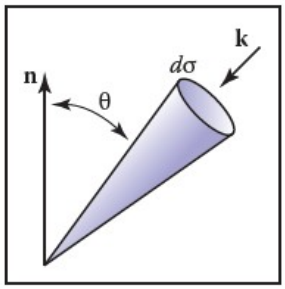
$$L_f = \frac{\Delta H}{\Delta \sigma \cos\theta}$$

Radiasi dan Kuantitas Radiometrik Lainnya

Jika kita memiliki permukaan yang pancaran medannya adalah L_f, maka kita dapat menurunkan semua besaran radiometrik darinya. Inilah salah satu alasan mengapa pancaran dianggap sebagai besaran radiometrik yang "mendasar". Misalnya, pancaran sinarnya dapat dinyatakan sebagai

$$H = \int_{semua\ k} L_f(k) \cos\theta \, d\sigma$$

Rumus ini memiliki beberapa konvensi notasi yang umum dalam grafis yang membuat rumus tersebut tidak jelas bagi pembaca yang tidak terbiasa dengan rumus tersebut (Gambar 18.4). Pertama, k adalah arah datang dan dapat dianggap sebagai vektor satuan, arah, atau pasangan (θ,ϕ) dalam koordinat bola terhadap normal permukaan. Arah memiliki sudut solid diferensial $d\sigma$ yang terkait dengannya. Pancaran medan berpotensi berbeda untuk setiap arah, jadi kita menulisnya sebagai fungsi L(k).



Gambar 18.4. Arah k memiliki sudut solid diferensial dσ yang terkait dengannya.

Sebagai contoh, kita dapat menghitung iradiansi H pada permukaan yang memiliki pancaran medan konstan L_f ke segala arah. Untuk mengintegrasikan, kami menggunakan sistem koordinat bola klasik dan mengingat bahwa sudut solid diferensial adalah

$$d\sigma \equiv \sin\theta \, d\theta \, d\varphi,$$

Ini menunjukkan kepada kita kemunculan pertama dari konstanta yang berpotensi mengejutkan. Faktor-faktor ini sering terjadi dalam radiometri dan merupakan artefak tentang bagaimana kami memilih untuk mengukur sudut padat, yaitu, luas unit bola adalah kelipatan daripada kelipatan satu.

Demikian pula, kita dapat menemukan kekuatan yang menghantam permukaan dengan mengintegrasikan pancaran mereka di seluruh area permukaan:

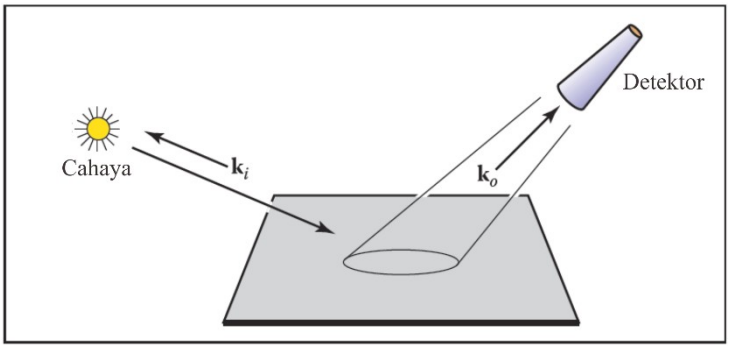
$$\Phi = \int_{semua\ x} H(x)\, dA$$

di mana x adalah titik di permukaan, dan dA adalah area diferensial yang terkait dengan titik itu. Perhatikan bahwa kami tidak memiliki istilah atau simbol khusus untuk daya masuk versus daya keluar. Perbedaan itu tampaknya tidak cukup untuk mendorong perbedaan itu.

18.4 BRDF

Karena kami tertarik pada tampilan permukaan, kami ingin mengkarakterisasi bagaimana permukaan memantulkan cahaya. Pada tingkat intuitif, untuk setiap cahaya datang yang datang dari arah ki, ada beberapa fraksi yang tersebar di sudut padat kecil di dekat arah keluar k_o. Ada banyak cara untuk memformalkan konsep seperti itu, dan tidak mengherankan, cara standar untuk melakukannya terinspirasi dengan membangun perangkat pengukuran sederhana. Perangkat seperti itu ditunjukkan pada Gambar 18.5, di mana sumber cahaya kecil diposisikan dalam arahki dilihat dari permukaan titik, dan detektor ditempatkan di arah k_o. Untuk setiap pasangan arah (k_i, k_o),wetakeberjalan dengan detektor.

Sekarang kita hanya perlu memutuskan bagaimana mengukur kekuatan sumber cahaya dan membuat fungsi pantulan kita tidak bergantung pada kekuatan ini. Misalnya, jika kita mengganti cahaya dengan cahaya yang lebih terang, kita tidak ingin menganggap permukaan memantulkan cahaya secara berbeda. Kita bisa menempatkan radiance meter pada titik yang diterangi untuk mengukur cahaya. Namun, untuk mendapatkan pembacaan yang akurat yang tidak bergantung pada detektor $\Delta\sigma$, kita memerlukan cahaya untuk subtent sudut padat yang lebih besar dari $\Delta\sigma$. Sayangnya, pengukuran yang dilakukan oleh detektor pancaran keliling kami dalam arah k_0 juga akan menghitung cahaya yang berasal dari titik-titik di luar scone detektor yang baru. Jadi ini sepertinya bukan solusi praktis.



Gambar 18.5. Perangkat pengukuran sederhana untuk pemantulan arah. Posisi cahaya dan detektor dipindahkan ke setiap pasangan arah yang mungkin. Perhatikan bahwa baik ki dan ko menunjuk menjauh dari permukaan untuk memungkinkan timbal balik.

Atau, kita dapat menempatkan pengukur radiasi pada titik di permukaan yang diukur. Ini akan membutuhkan area yang tidak terlalu bergantung pada kehalusan geometri sumber cahaya. Hal ini menunjukkan karakterisasi refleksi sebagai rasio:

$$\frac{\rho L_s}{H}$$

di mana fraksi ρ ini akan bervariasi dengan arah datang dan keluar k_i dan k_o, H adalah radiasi untuk posisi cahaya ki, dan Ls adalah pancaran permukaan yang diukur dalam arah ko. Jika kita mengambil pengukuran seperti itu untuk semua pasangan arah, kita akan mendapatkan fungsi 4D $\rho(k_i, k_o)$. Fungsi ini disebut fungsi distribusi refleksi dua arah/*bidirectional reflectance distribution function* (BRDF). BRDF adalah semua yang perlu kita ketahui untuk mengkarakterisasi sifat arah dari bagaimana permukaan memantulkan cahaya.

Refleksi Hemispherical Directional

Mengingat BRDF, mudah untuk bertanya, "Berapa fraksi cahaya datang yang dipantulkan?" Namun, jawabannya tidak begitu mudah; fraksi yang dipantulkan tergantung pada distribusi arah cahaya yang masuk. Untuk alasan ini, kami biasanya hanya menetapkan fraksi yang direfleksikan untuk arah kejadian tetap k_i. Fraksi ini disebut pemantulan hemisferis terarah. Fraksi ini, $R(k_i)$ didefinisikan oleh

$$R(k¿¿i)=\frac{daya\,ke\,semua\,arah\,keluar\,k_0}{daya\,dalam\,sinar\,dari\,arah\,k_i}¿$$

Perhatikan bahwa jumlah ini antara nol dan selesai untuk alasan konservasi energi. Jika kita membiarkan daya datang i mengenai area kecil A, maka radiasinya adalah Φi/ΔA. Juga, rasio daya yang masuk hanyalah rasio pancaran radiasi terhadap radiasi:

$$R(k¿¿i)=\frac{E}{H}¿$$

Pancaran dalam arah tertentu yang dihasilkan dari kekuatan ini adalah dengan definisi BRDF:

$$L(k¿¿o)=H\rho(k¿¿i,k_0)¿¿$$

$$¿\frac{\Phi_i}{\Delta A}$$

Dan dari definisi pancaran, kita juga memiliki

$$L(k¿¿o)=\frac{\Delta E}{\Delta\sigma_0\cos\theta_0}¿$$

di mana E adalah pancaran pancaran patch kecil dalam arah ko. Dengan menggunakan dua definisi ini untuk pancaran, kita peroleh

$$H\rho(k¿¿i,k_o)=\frac{\Delta E}{\Delta\sigma_0\cos\theta_0}¿$$

Menata ulang istilah, kita dapatkan

$$\frac{\Delta E}{H}=\rho(k¿¿i,¿k_o)\Delta\sigma_0\cos\theta_0¿$$

Ini hanyalah kontribusi kecil untuk E/H yang direfleksikan di dekat ko tertentu. Untuk mencari total R(k_i), kita jumlahkan semua k_o yang keluar. Dalam bentuk integral ini adalah

$$R(k¿¿i)=\int_{semua\ k_o}\rho(k¿¿i,¿k_o)\cos\theta_0 d\sigma_0¿¿$$

BRDF Difusi Ideal

Sebuah permukaan difus yang ideal disebut Lambertian. Permukaan seperti itu tidak mungkin di alam karena alasan termodinamika, tetapi secara matematis mereka menghemat energi. BRDF Lambertian memiliki ρ sama dengan konstanta untuk semua sudut. Ini berarti permukaan akan memiliki pancaran yang sama untuk semua sudut pandang, dan pancaran ini akan sebanding dengan pancaran.

Jadi, untuk permukaan Lambertian yang memantulkan sempurna (R = 1), kita memiliki =1/π, dan untuk permukaan Lambertian di mana R(k_i) = r, kita memiliki

$$\rho(k¿¿i,k_o)=\frac{r}{\pi}¿$$

Ini adalah contoh lain di mana penggunaan asteradian untuk sudut padat menentukan konstanta normalisasi dan dengan demikian memperkenalkan faktor .

Ekuasi Transport

Dengan definisi BRDF, kita dapat menggambarkan pancaran permukaan dalam bentuk pancaran datang dari semua arah yang berbeda. Karena dalam grafis komputer kita dapat menggunakan matematika ideal yang mungkin tidak praktis untuk digunakan di lab, kita juga dapat menulis BRDF dalam bentuk pancaran saja. Jika kita mengambil sebagian kecil dari cahaya dengan sudut padat i dengan pancaran L_i dan “mengukur” pancaran pancaran dalam arah k_o karena bagian kecil dari cahaya ini, kita dapat menghitung BRDF (Gambar 18.6). Penyinaran akibat potongan kecil cahaya adalah $H = L_i \cos\theta_i \Delta\sigma_i$. Jadi BRDFnya adalah

$$\rho=\frac{L_o}{L_i\cos\theta_i\Delta\sigma_i}$$

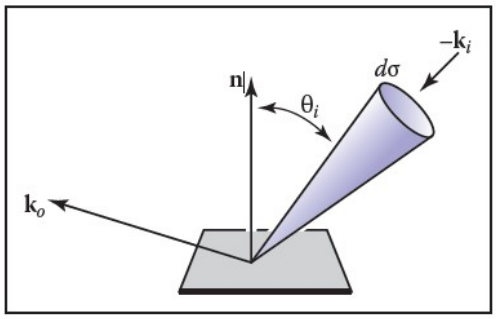
Formulir ini dapat berguna dalam beberapa situasi. Dengan mengatur ulang istilah, kita dapat menulis bagian pancaran yang disebabkan oleh cahaya yang datang dari arah k_i:

$$\Delta L_o L_i\cos\theta_i\Delta\sigma_i$$

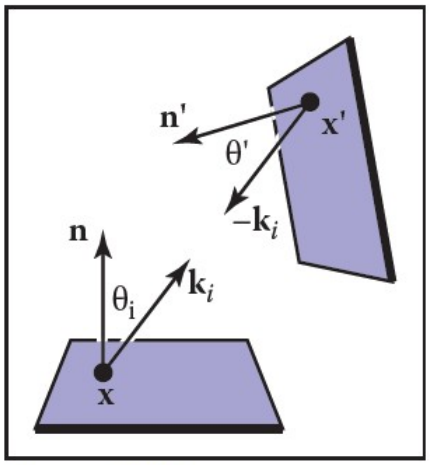
Jika ada cahaya yang datang dari berbagai arah Li(ki), kita dapat menjumlahkan semuanya. Dalam bentuk integral, dengan notasi untuk pancaran permukaan dan medan, ini adalah

$$L_s(k_o)=\int_{semua k_i}\rho(k¿¿i,k_o)L_f\cos\theta_i\sigma_i¿$$

Ini sering disebut persamaan rendering dalam grafis komputer (Immel, Cohen, & Greenberg, 1986).



Gambar 18.6. Geometri untuk persamaan transportasi dalam bentuk arahnya.



Gambar 18.7. Cahaya yang datang ke satu titik datang dari titik lain.

Kadang-kadang berguna untuk menulis persamaan transpor dalam bentuk pancaran permukaan saja (Kajiya, 1986). Perhatikan, bahwa dalam lingkungan tertutup, pancaran medan $L_f(k_i)$ berasal dari suatu permukaan dengan pancaran permukaan $L_s(-k_i) = L_f(k_i)$ (Gambar 18.7). Sudut padat yang diwakili oleh titik x pada gambar diberikan oleh

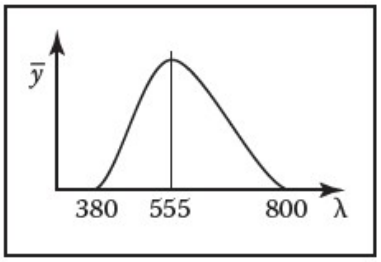
$$\Delta \sigma_i = \frac{\Delta A' \cos \theta'}{¿ \left| x - x' \right| \vee ¿^2} ¿$$

di mana $\Delta A'$ area yang kita kaitkan dengan x-x'. Perhatikan bahwa kita menggunakan vektor tak ternormalisasi x−x untuk menunjukkan arah dari x' ke x. Perhatikan juga bahwa kita menulis L_s sebagai fungsi posisi dan arah.

18.5 PHOTOMETRI

Untuk setiap kuantitas radiometrik spektral ada kuantitas fotometrik terkait yang mengukur seberapa banyak kuantitas itu "berguna" bagi pengamat manusia. Diberikan besaran radiometrik spektral $f_r(\lambda)$, besaran fotometrik terkait f_p di mana y adalah fungsi efisiensi cahaya dari sistem visual manusia. Fungsi ini nol di luar batas integrasi di atas, sehingga batasnya bisa 0 dan dan fp tidak akan berubah. Fungsi efisiensi luminous akan dibahas lebih rinci dalam Bab 19, tetapi kita membahas sifat umumnya di sini. Konstanta terdepan adalah membuat definisi tersebut konsisten dengan besaran fotometrik absolut historis.

Fungsi efisiensi cahaya tidak sama sensitifnya terhadap semua panjang gelombang (Gambar 18.8). Untuk panjang gelombang di bawah 380 nm (*rentang ultraviolet*), cahaya tidak terlihat oleh manusia dan dengan demikian memiliki nilai y nol. Dari 380 nm secara bertahap meningkat hingga λ = 555 nm di mana puncaknya. Ini adalah lampu hijau murni. Kemudian, secara bertahap menurun hingga mencapai batas wilayah inframerah pada 800nm.



Gambar 18.8. Fungsi efisiensi cahaya versus panjang gelombang (nm).

Simbol Y untuk luminance berasal dari kolorimetri. Sebagian besar bidang lain menggunakan simbol L; kami tidak akan mengikuti konvensi itu karena terlalu membingungkan untuk menggunakan L baik untuk luminansi maupun pancaran spektral. Luminance memberi seseorang gambaran umum tentang bagaimana "terang" sesuatu tidak tergantung pada adaptasi pemirsa. Perhatikan bahwa kertas hitam di bawah siang hari tidak secara subyektif lebih gelap daripada kertas putih dengan luminansi yang lebih rendah di bawah sinar bulan; membaca terlalu banyak tentang luminansi adalah berbahaya, tetapi ini adalah jumlah yang sangat berguna untuk mendapatkan nuansa kuantitatif untuk output cahaya relatif yang dapat dilihat. Satuan lm adalah singkatan dari lumens. Perhatikan bahwa sebagian besar bola lampu dinilai dalam hal daya yang mereka konsumsi dalam watt, dan cahaya berguna yang dihasilkannya dalam lumen. Bohlam yang lebih efisien menghasilkan lebih banyak cahaya di mana y besar dan dengan demikian menghasilkan lebih banyak lumen per watt. Sebuah cahaya "sempurna" akan mengubah semua daya menjadi cahaya 555 nm dan akan menghasilkan 683 lumen per watt. Satuan luminans adalah (lm/W)(W/(m^2sr)) = lm/(m^2sr). Kuantitas satu lumen per teradian didefinisikan sebagai satu candela (cd), jadi luminansi biasanya digambarkan dalam satuan cd/m2.

Pertanyaan yang Sering Diajukan

- ***Apa itu "intensitas"?***

Istilah intensitas digunakan dalam berbagai konteks dan penggunaannya bervariasi menurut zaman dan disiplin ilmu. Dalam praktiknya, ini tidak lagi bermakna sebagai besaran radiometrik tertentu, tetapi berguna untuk diskusi intuitif. Kebanyakan kertas yang menggunakannya melakukannya di tempat pancaran.

- ***Apa itu "radiositas"?***

Istilah radiositas digunakan sebagai pengganti eksitasi radiasi di beberapa bidang. Hal ini juga kadang-kadang digunakan untuk menggambarkan dunia-spacelight transportalgorithms.

18.6 CATATAN

Besaran radiometrik umum yang tidak dijelaskan dalam bab ini adalah intensitas pancaran (I), yang merupakan daya spektral per steradian yang dipancarkan dari sumber titik tak terhingga. Biasanya harus dihindari dalam program grafis karena sumber titik menyebabkan masalah implementasi. Perlakuan radiometri yang lebih ketat dapat ditemukan di Metode Analitik untuk Transportasi Cahaya Simulasi (Arvo, 1995). Istilah radiometrik dan fotometrik dalam bab ini berasal dari standar Illumination Engineering Society yang semakin banyak digunakan oleh semua bidang sains dan teknik (American National Standard Institute,

1986). Diskusi yang lebih luas tentang standar radiometrik dan penampilan dapat ditemukan di Prinsip Sintesis Gambar Digital (Glassner, 1995).

18.7 LATIHAN

1. Untuk permukaan yang menyebar tanpa pancaran L, berapakah pancaran sinarnya?
2. Berapa daya total yang keluar dari permukaan difus dengan luas 4m 2 dan pancaran L?
3. Jika lampu neon dan lampu pijar sama-sama mengkonsumsi daya 20 watt, mengapa lampu neon biasanya lebih disukai?

BAB 19

WARNA

Foton adalah pembawa informasi optik. Mereka menyebar melalui media yang mengambil sifat-sifat yang terkait dengan gelombang. Pada batas permukaan mereka berinteraksi dengan materi, berperilaku lebih sebagai partikel. Mereka juga dapat diserap oleh retina, di mana informasi yang mereka bawa ditranskode menjadi sinyal listrik yang selanjutnya diproses oleh otak. Hanya di sanalah sensasi warna dihasilkan.

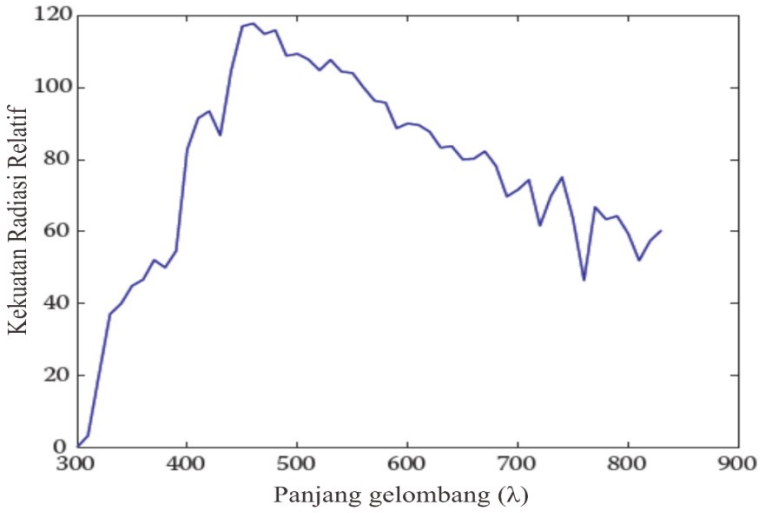
Akibatnya, studi warna dalam semua samarannya menyentuh beberapa bidang yang berbeda: fisika untuk perambatan cahaya melalui ruang, kimia untuk interaksinya dengan materi, dan ilmu saraf dan psikologi untuk aspek yang berkaitan dengan persepsi dan kognisi warna (Reinhard et al. 2008).

Dalam grafis komputer, kami secara tradisional mengambil pandangan sederhana tentang bagaimana cahaya merambat melalui ruang. Fotonstravel sepanjang jalur lurus sampai mereka mencapai batas permukaan dan kemudian dipantulkan sesuai dengan fungsi refleksi dari beberapa macam. Sebuah foton tunggal akan membawa sejumlah energi, yang diwakili oleh panjang gelombangnya. Jadi, foton hanya akan memiliki satu panjang gelombang. Hubungan antara panjang gelombang λ dan jumlah energi yang dibawanya (ΔE) diberikan oleh

$$\lambda \Delta E = 1239.9$$

dimana ΔE diukur dalam elektron volt (eV).

Dalam grafis komputer, tidak terlalu efisien untuk mensimulasikan foton tunggal; bukan koleksi besar dari mereka disimulasikan pada waktu yang sama. Jika kita mengambil sejumlah besar foton, masing-masing membawa jumlah energi yang mungkin berbeda, maka bersama-sama mereka mewakili spektrum. Spektrum dapat dianggap sebagai grafis di mana jumlah foton diplot terhadap panjang gelombang. Karena dua foton dengan panjang gelombang yang sama membawa energi dua kali lebih banyak daripada foton tunggal pada panjang gelombang itu, grafis ini juga dapat dilihat sebagai plot energi terhadap panjang gelombang. Contoh spektrum ditunjukkan pada Gambar 19.1. Kisaran panjang gelombang yang sensitif bagi manusia kira-kira antara 380 dan 800 nanometer (nm).



Gambar 19.1. Spektrum menggambarkan berapa banyak energi yang tersedia pada setiap panjang gelombang , di sini diukur sebagai daya pancaran relatif. Spektrum khusus ini mewakili siang hari rata-rata.

Saat mensimulasikan cahaya, maka dimungkinkan untuk melacak sinar yang masing-masing membawa spektrum. Penyaji yang menyelesaikan ini biasanya disebut penyaji spektral. Dari bab-bab sebelumnya, harus jelas bahwa kita biasanya tidak mengeluarkan biaya untuk membangun penyaji spektral. Sebagai gantinya, kami mengganti spektrum dengan representasi yang biasanya menggunakan komponen merah, hijau, dan biru. Alasan mengapa hal ini mungkin ada hubungannya dengan penglihatan manusia dan akan dibahas nanti dalam bab ini.

Mensimulasikan cahaya dengan menelusuri sinar menangani fisika cahaya, meskipun perlu dicatat bahwa beberapa sifat cahaya, termasuk, misalnya, polarisasi, difraksi, dan interferensi, tidak dimodelkan dengan cara ini.

Pada batas permukaan, kita biasanya memodelkan apa yang terjadi dengan cahaya dengan menggunakan fungsi refleksi. Fungsi-fungsi ini dapat diukur secara langsung dengan menggunakan gonioreflectometers, yang menghasilkan sejumlah besar data tabel, yang dapat diwakili secara lebih ringkas oleh berbagai fungsi yang berbeda. Meskipun demikian, fungsi pantulan ini bersifat empiris, yaitu, menghilangkan kimia yang terjadi ketika foton diserap dan dipancarkan kembali oleh elektron. Jadi, fungsi pemantulan berguna untuk pemodelan dalam grafis komputer, tetapi tidak memberikan penjelasan tentang bagaimana panjang gelombang tertentu dari area cahaya yang diserap dan yang lainnya dipantulkan. Oleh karena itu, kita tidak dapat menggunakan fungsi pemantulan untuk menjelaskan mengapa cahaya memantulkan komposisi spektral pisang yang tampak kuning. Untuk itu, kita harus mempelajari teori orbital molekul, topik di luar cakupan buku ini.

Akhirnya, ketika cahaya mencapai retina, itu ditranskode menjadi sinyal listrik yang disebarkan ke otak. Sebagian besar otak dikhususkan untuk memproses sinyal visual, yang sebagian menimbulkan sensasi warna. Jadi, bahkan jika kita mengetahui spektrum cahaya yang dipantulkan dari pisang, kita belum tahu mengapa manusia mengaitkan istilah "kuning" dengannya. Selain itu, seperti yang akan kita temukan di sisa bab ini, persepsi kita tentang warna jauh lebih rumit daripada yang terlihat pada pandangan pertama. Ini berubah dengan iluminasi, bervariasi antara pengamat, dan bervariasi dalam waktu pengamat dari waktu ke waktu.

Dengan kata lain, spektrum cahaya yang datang dari pisang dirasakan dalam konteks lingkungan. Untuk memprediksi bagaimana pengamat mempersepsikan “spektrum pisang” membutuhkan pengetahuan tentang lingkungan yang mengandung pisang serta lingkungan pengamat. Dalam banyak kasus, kedua lingkungan ini sama. Namun, ketika kita menampilkan foto pisang di monitor, maka kedua lingkungan ini akan berbeda. Karena persepsi visual manusia bergantung pada lingkungan tempat pengamat berada, ia mungkin mempersepsikan pisang dalam foto secara berbeda dari bagaimana pengamat yang melihat pisang secara langsung akan melihatnya. Ini memiliki dampak yang signifikan pada bagaimana kita harus menangani warna dan menggambarkan kompleksitas yang terkait dengan warna.

Untuk menekankan peran penting yang dimainkan oleh penglihatan manusia, kita hanya perlu melihat definisi warna: “Warna adalah aspek persepsi visual yang dengannya pengamat dapat membedakan perbedaan antara dua bidang pandang bebas-struktur dengan ukuran dan bentuk yang sama, seperti mungkin disebabkan oleh perbedaan komposisi spektral energi radiasi yang bersangkutan dalam pengamatan” (Wyszecki & Stiles, 2000). Intinya, tanpa pengamat manusia tidak ada warna.

Untungnya, banyak dari apa yang kita ketahui tentang warna dapat diukur, sehingga kita dapat melakukan perhitungan untuk mengoreksi keanehan penglihatan manusia dan dengan demikian menampilkan gambar yang akan muncul untuk mengamati cara perancang

gambar tersebut dimaksudkan. Bab ini berisi teori dan matematika yang diperlukan untuk melakukannya.

19.1 *COLORIMETRY*

Kolorimetri adalah ilmu pengukuran dan deskripsi warna. Karena warna pada akhirnya merupakan respons manusia, pengukuran warna harus dimulai dengan pengamatan manusia. Fotodetektor di retina manusia terdiri dari batang dan kerucut. Batang sangat sensitif dan ikut bermain dalam kondisi cahaya rendah. Dalam kondisi pencahayaan normal, kerucut beroperasi, memediasi penglihatan manusia. Ada tiga jenis kerucut dan bersama-sama mereka terutama bertanggung jawab untuk penglihatan warna.

Meskipun dimungkinkan untuk secara langsung merekam output listrik kerucut saat beberapa stimulus visual disajikan, prosedur seperti itu akan menjadi invasif, sementara pada saat yang sama mengabaikan perbedaan yang kadang-kadang substansial antara pengamat. Selain itu, sebagian besar pengukuran warna telah dikembangkan jauh sebelum teknik perekaman langsung seperti itu tersedia.

Alternatifnya adalah mengukur warna dengan cara mengukur respons manusia terhadap bercak warna. Ini mengarah pada eksperimen pencocokan warna, yang akan dijelaskan nanti di bagian ini. Melaksanakan eksperimen ini telah menghasilkan beberapa pengamat standar, yang dapat dianggap sebagai perkiraan statistik dari pengamat manusia yang sebenarnya. Namun, pertama-tama, kita perlu menjelaskan beberapa asumsi yang mendasari kemungkinan pencocokan warna, yang diringkas oleh hukum Grassmann.

19.2 HUKUM GRASSMANN

Mengingat bahwa manusia memiliki tiga jenis kerucut yang berbeda, hukum eksperimental pencocokan warna dapat diringkas sebagai generalisasi trikromatik (Wyszecki & Stiles, 2000), yang menyatakan bahwa setiap stimulus warna dapat dicocokkan sepenuhnya dengan campuran aditif dari tiga warna yang dimodulasi dengan tepat. sumber. Fitur warna ini sering digunakan dalam praktik, misalnya oleh televisi dan monitor yang mereproduksi banyak warna berbeda dengan menambahkan campuran cahaya merah, hijau, dan biru untuk setiap piksel. Itu juga alasan mengapa penyaji dapat dibangun hanya menggunakan tiga nilai untuk menggambarkan setiap warna.

Generalisasi trikromatik memungkinkan untuk membuat kecocokan warna antara stimulus yang diberikan dan campuran aditif dari tiga stimulus warna lainnya. Hermann Grassmann adalah orang pertama yang menjelaskan aturan aljabar yang dianut oleh pencocokan warna. Mereka dikenal sebagai hukum Grassmann tentang pencocokan warna aditif (Grassmann, 1853) dan adalah sebagai berikut:

- **hukum simetri**. Jika rangsangan warna A cocok dengan rangsangan warna B, maka B cocok dengan A.
- **hukum transitif**. Jika A cocok dengan B dan B cocok dengan C, maka A cocok dengan C.
- **Hukum proporsionalitas**. Jika A cocok dengan B, maka A cocok dengan B, di mana adalah faktor skala positif.
- **hukum aditif.** Jika A cocok dengan B, C cocok dengan D, dan A + C cocok dengan B + D, maka A + D cocok dengan B + C.

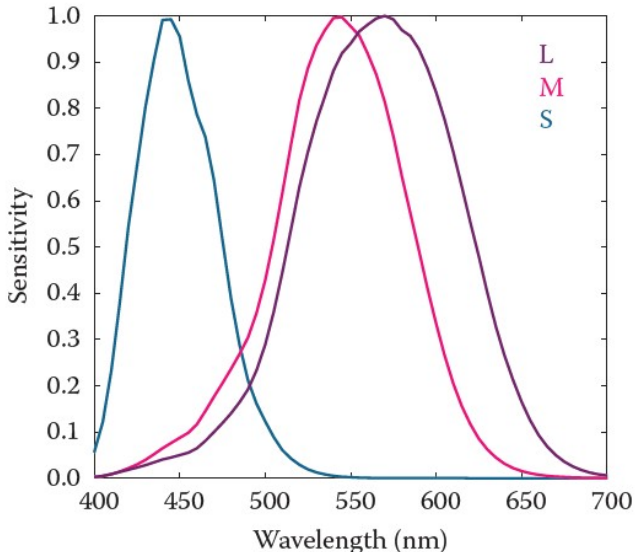
19.3 RESPON KERUCUT

Setiap jenis kerucut sensitif terhadap rentang panjang gelombang, yang mencakup sebagian besar rentang yang terlihat. Namun, sensitivitas terhadap panjang gelombang tidak terdistribusi secara merata, tetapi mengandung panjang gelombang puncak di mana sensitivitasnya paling besar. Letak panjang gelombang puncak ini berbeda untuk setiap jenis kerucut. Tiga jenis kerucut diklasifikasikan sebagai kerucut S, M, dan L, di mana huruf-hurufnya mewakili pendek, sedang, dan panjang, menunjukkan di mana dalam spektrum yang terlihat sensitivitas puncak berada.

Respon dari kerucut yang diberikan kemudian besarnya sinyal listrik yang dikeluarkannya, sebagai fungsi dari spektrum panjang gelombang yang terjadi pada kerucut. Fungsi tanggapan kerucut untuk setiap jenis kerucut sebagai fungsi panjang gelombang diberikan oleh L(λ), M(λ), dan S(λ). Mereka diplot pada Gambar 19.2. Respon aktual terhadap stimulus dengan komposisi spektral tertentu Φ(λ) kemudian diberikan untuk setiap jenis kerucut oleh

$$L = \int_{\lambda} \Phi(\lambda)\, L(\lambda)\, d\lambda,$$
$$M = \int_{\lambda} \Phi(\lambda)\, M(\lambda)\, d\lambda,$$
$$S = \int_{\lambda} \Phi(\lambda)\, S(\lambda)\, d\lambda.$$

Ketiga tanggapan terpadu ini dikenal sebagai nilai tristimulus.



Gambar 19.2. Fungsi respons kerucut untuk kerucut L, M, dan S.

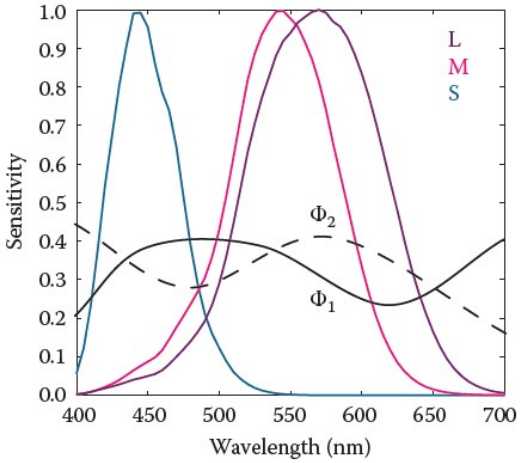
19.4 EKSPERIMEN PENCOCOKAN WARNA

Mengingat bahwa nilai tristimulus dibuat dengan mengintegrasikan produk dua fungsi pada rentang yang terlihat, segera jelas bahwa sistem visual manusia tidak bertindak sebagai detektor panjang gelombang sederhana. Sebaliknya, fotoreseptor kami bertindak sebagai integrator linier. Akibatnya, dimungkinkan untuk menemukan dua komposisi spektral yang berbeda, misalnya $\Phi_1(\lambda)$ dan $\Phi_2(\lambda)$, yang setelah integrasi menghasilkan respons yang sama (L,M,S). Fenomena ini dikenal sebagai metamerisme, contohnya ditunjukkan pada Gambar 19.3.

Metamerismemisahkan fitur utama penglihatan manusia yang memungkinkan konstruksi perangkat reproduksi warna, termasuk gambar berwarna dalam buku ini dan apa pun yang direproduksi pada printer, televisi, dan monitor.

Eksperimen pencocokan warna juga mengandalkan prinsip metamerisme. Misalkan kita memiliki tiga sumber cahaya dengan warna berbeda, masing-masing dengan dial untuk mengubah intensitasnya. Kami menyebut ketiga sumber cahaya ini sebagai primer. Kita sekarang harus dapat menyesuaikan intensitas masing-masing sedemikian rupa sehingga ketika dicampur bersama secara aditif, spektrum yang dihasilkan terintegrasi ke nilai tristimulus yang cocok dengan warna yang dirasakan dari sumber cahaya keempat yang tidak diketahui. Ketika kami melakukan percobaan seperti itu, kami pada dasarnya telah mencocokkan warna primer kami dengan warna yang tidak diketahui. Posisi dari tiga dial kami kemudian merupakan representasi dari warna sumber cahaya keempat.

Dalam percobaan seperti itu, kami telah menggunakan hukum Grassmann untuk menambahkan tiga spektrum primer kami. Kami juga telah menggunakan metamerisme, karena spektrum gabungan dari tiga primer kami hampir pasti berbeda dari spektrum sumber cahaya keempat. Namun, nilai tristimulus yang dihitung dari dua spektrum ini akan identik, setelah menghasilkan kecocokan warna.



Gambar 19.3. Dua rangsangan 1(λ) dan 2(λ) mengarah ke nilai tristimulus yang sama setelah integrasi.

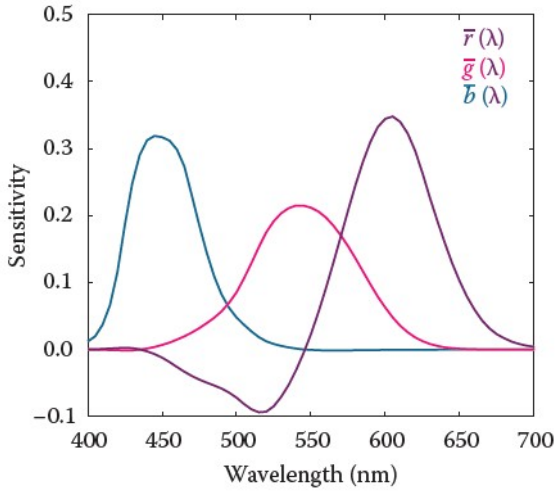
Perhatikan bahwa kita tidak benar-benar mengetahui fungsi respons kerucut untuk melakukan eksperimen semacam itu. Selama kita menggunakan pengamat yang sama dalam kondisi yang sama, kita dapat mencocokkan warna dan merekam posisi dial kita untuk setiap warna. Namun, tidak nyaman untuk melakukan eksperimen seperti itu setiap kali kita ingin mengukur warna. Untuk alasan ini, kami ingin mengetahui fungsi respons spektral dan rata-rata untuk serangkaian pengamat yang berbeda untuk menghilangkan variabilitas antarpengamat.

19.5 PENGAMAT STANDAR

Jika kita melakukan eksperimen pencocokan warna untuk rentang warna yang besar, yang dilakukan oleh sekumpulan pengamat yang berbeda, dimungkinkan untuk menghasilkan kumpulan data pencocokan warna rata-rata. Jika kita secara khusus menggunakan sumber cahaya monokromatik untuk dicocokkan dengan cahaya primer kita, kita dapat mengulangi percobaan ini untuk semua panjang gelombang tampak. Nilai tristimulus yang dihasilkan kemudian disebut nilai tristimulus spektral, dan dapat diplot terhadap panjang gelombang λ, ditunjukkan pada Gambar 19.4.

Dengan menggunakan seperangkat sumber cahaya primer yang terdefinisi dengan baik, nilai tristimulus spektral menghasilkan tiga fungsi pencocokan warna. *Commission*

Internationale d'Eclairage (CIE) telah mendefinisikan tiga primer tersebut sebagai sumber cahaya monokromatik masing-masing 435,8, 546,1, dan 700 nm. Dengan ketiga sumber cahaya monokromatik ini, semua panjang gelombang tampak lainnya dapat dicocokkan dengan menambahkan jumlah masing-masing yang berbeda. Jumlah masing-masing yang diperlukan untuk mencocokkan panjang gelombang tertentu dikodekan dalam fungsi pencocokan warna, diberikan oleh r(λ), g(λ), dan b(λ) dan diplot pada Gambar 19.4. Nilai tristimulus yang terkait dengan fungsi pencocokan warna ini disebut R, G, dan B.



Gambar 19.4. Nilai tristimulus spektral dirata-ratakan pada banyak pengamat. Primer dimana sumber cahaya monokromatik dengan panjang gelombang 435,8, 546,1, dan 700 nm.

Mengingat bahwa kita menambahkan cahaya, dan cahaya tidak boleh negatif, Anda mungkin telah melihat anomali pada Gambar 19.4: untuk membuat kecocokan untuk beberapa panjang gelombang, cahaya perlu dikurangi. Meskipun tidak ada yang namanya cahaya negatif, kita dapat menggunakan hukum Grassmann sekali lagi, dan alih-alih mengurangi cahaya dari campuran primer, kita dapat menambahkan jumlah cahaya yang sama ke warna yang dicocokkan.

Fungsi pencocokan warna CIE r(λ), g(λ), dan b(λ) memungkinkan kita untuk menentukan apakah distribusi spektral 1 cocok dengan distribusi spektral kedua 2 hanya dengan membandingkan trinilai stimulus yang dihasilkan yang diperoleh dengan mengintegrasikan dengan fungsi pencocokan warna berikut:

$$\int_\lambda \Phi_1(\lambda)\,\bar{r}(\lambda) = \int_\lambda \Phi_2(\lambda)\,\bar{r}(\lambda),$$
$$\int_\lambda \Phi_1(\lambda)\,\bar{g}(\lambda) = \int_\lambda \Phi_2(\lambda)\,\bar{g}(\lambda),$$
$$\int_\lambda \Phi_1(\lambda)\,\bar{b}(\lambda) = \int_\lambda \Phi_2(\lambda)\,\bar{b}(\lambda).$$

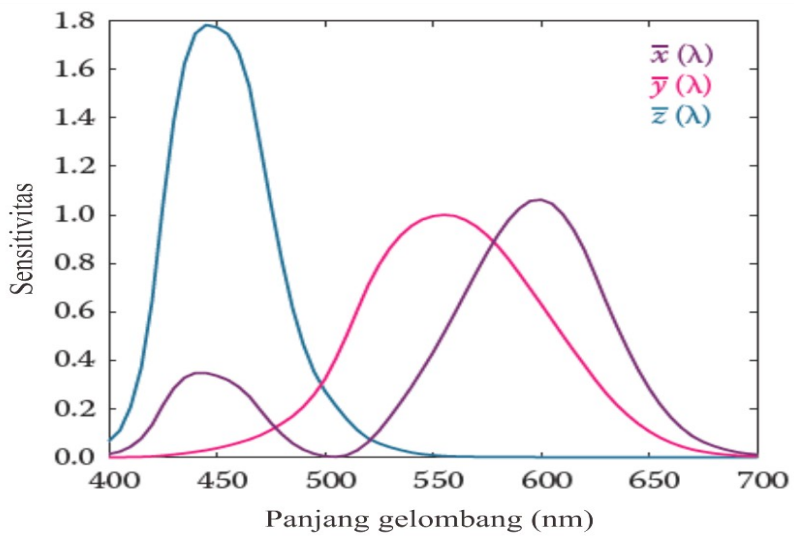
Tentu saja, kecocokan warna hanya dijamin jika ketiga nilai tristimulus cocok.

Pentingnya fungsi pencocokan warna ini terletak pada kenyataan bahwa kita sekarang dapat mengomunikasikan dan menggambarkan warna secara kompak melalui nilai tristimulus. Untuk fungsi spektral tertentu, fungsi pencocokan warna CIE menyediakan cara yang tepat untuk menghitung nilai tristimulus. Selama semua orang menggunakan fungsi pencocokan warna yang sama, itu harus selalu memungkinkan untuk menghasilkan kecocokan.

Jika fungsi pencocokan warna yang sama tidak tersedia, maka dimungkinkan untuk mengubah satu set nilai tristimulus menjadi satu set nilai tristimulus yang berbeda yang sesuai

untuk set primer yang sesuai. CIE telah mendefinisikan salah satu transformasi tersebut karena dua alasan khusus. Pertama, pada tahun 1930-an integrasi numerik sulit dilakukan, dan terlebih lagi untuk fungsi yang bisa positif dan negatif. Kedua, CIE telah mengembangkan fungsi respon fotopicluminance, CIE V (λ). Menjadi diinginkan untuk memiliki tiga fungsi integrasi, di mana V (λ) adalah satu dan ketiganya positif pada rentang yang terlihat.

Untuk membuat satu set fungsi pencocokan warna positif, perlu untuk mendefinisikan primer imajiner. Dengan kata lain, untuk mereproduksi warna apa pun dalam spektrum yang terlihat, kita membutuhkan sumber cahaya yang tidak dapat diwujudkan secara fisik. Fungsi pencocokan warna yang diselesaikan oleh CIE diberi nama x(λ), y(λ), dan z(λ) dan ditunjukkan pada Gambar 19.5. Perhatikan bahwa y(λ) sama dengan fungsi respons pencahayaan fotopik V (λ) dan masing-masing fungsi ini memang positif. Mereka dikenal sebagai pengamat standar CIE 1931.



Gambar 19.5 Supaya mereproduksi warna apapun dalam spektrum yang terlihat membutuhkan sumber cahaya yang tidak dapat diwujudkan secara fisik

Nilai tristimulus yang sesuai disebut X, Y, dan Z, untuk menghindari kebingungan dengan nilai tristimulus R, G, dan B yang biasanya terkait dengan primer yang dapat direalisasikan. Konversi dari nilai tristimulus (R, G, B) ke nilai tristimulus (X, Y, Z) didefinisikan dengan transformasi 3×3 sederhana:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \frac{1}{0.17697} \begin{bmatrix} 0.4900 & 0.3100 & 0.2000 \\ 0.17697 & 0.81240 & 0.01063 \\ 0.0000 & 0.0100 & 0.9900 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix}.$$

Untuk menghitung tiga nilai stimulus, kita biasanya langsung mengintegrasikan fungsi pencocokan warna pengamat standar dengan spektrum yang diinginkan Φ (λ), daripada melalui CIE r(λ), g(λ), dan fungsi pencocokan warna $\bar{b}(\lambda)$ terlebih dahulu, diikuti oleh transformasi di atas. Ini memungkinkan kami menghitung pengukuran warna yang konsisten dan juga menentukan kapan dua warna cocok satu sama lain.

19.6 KOORDINAT KROMATISITAS

Setiap warna dapat diwakili oleh satu set tiga nilai tristimulus (X,Y,Z). Kita dapat mendefinisikan sistem koordinat ortogonal dengan sumbu X, Y, dan Z dan memplot setiap warna dalam ruang 3D yang dihasilkan. Ini disebut ruang warna. Tingkat spasial dari volume di mana warna terletak kemudian disebut gamut warna.

Memvisualisasikan warna dalam ruang warna 3D cukup sulit. Selain itu, nilai Y dari setiap warna sesuai dengan luminansinya, berdasarkan fakta bahwa y(λ) sama dengan V (λ). Oleh karena itu, kami dapat memproyeksikan nilai tristimulus ke ruang 2D yang mendekati informasi kromatik, yaitu, informasi yang tidak bergantung pada pencahayaan. Proyeksi ini disebut diagram kromatisitas dan diperoleh dengan normalisasi sambil pada saat yang sama menghilangkan informasi luminansi:

$$x = \frac{X}{X+Y+Z}$$

$$y = \frac{Y}{X+Y+Z}$$

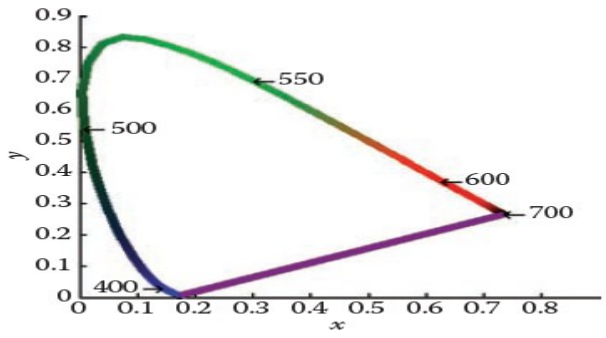
$$z = \frac{Z}{X+Y+Z}$$

Mengingat bahwa x + y + z sama dengan 1, nilai z berlebihan, memungkinkan kita untuk memplot kromatisitas x dan y satu sama lain dalam diagram kromatisitas. Meskipun x dan y sendiri tidak cukup untuk menggambarkan warna sepenuhnya, kita dapat menggunakan dua koordinat kromatisitas ini dan satu dari tiga nilai tristimulus, secara tradisional Y , untuk memulihkan dua nilai tristimulus lainnya:

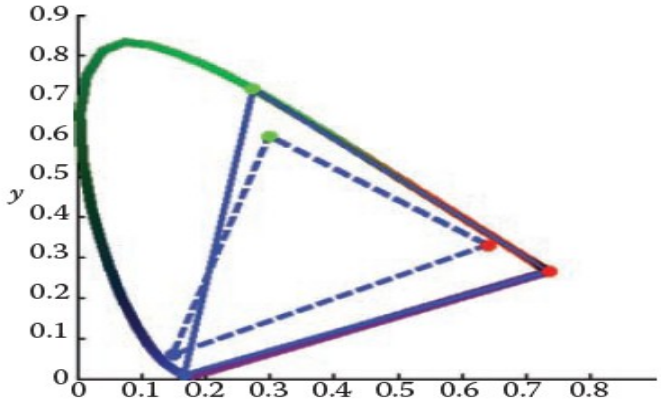
$$X = \frac{x}{y} Y$$

$$Z = \frac{1-x-y}{y} Y$$

Dengan memplot semua warna monokromatik (spektral) dalam diagram kromatisitas, kita memperoleh kurva berbentuk tapal kuda. Titik pada kurva ini disebut lokus spektrum. Semua warna lain akan menghasilkan titik yang terletak di dalam kurva ini. Lokus spektrum untuk pengamat standar 1931 ditunjukkan pada Gambar 19.6. Garis ungu antara kedua ujung tapal kuda tidak mewakili warna monokromatik, melainkan kombinasi dari rangsangan panjang gelombang pendek dan panjang.



Gambar 19.6. Lokus spektrum untuk pengamat standar CIE 1931.



Gambar 19.7. Batas kromatisitas CIE RGB primer pada 435,8, 546,1, dan 700 nm (padat) dan HDTV tipikal (putus-putus).

Primer (non-monokromatik) dapat diintegrasikan pada semua panjang gelombang tampak, yang mengarah ke nilai tristimulus (X,Y,Z), dan selanjutnya ke koordinat kromatisitas (x,y), yaitu titik pada diagram kromatisitas. Mengulangi ini untuk dua atau lebih hasil primer satu set titik-titik pada diagram kromatisitas yang dapat dihubungkan dengan garis lurus. Volume yang terbentang dengan cara ini menunjukkan kisaran warna yang dapat direproduksi oleh campuran aditif dari primer ini. Contoh sistem tiga primer ditunjukkan pada Gambar 19.7.

Diagram kromatisitas memberikan wawasan tentang campuran warna aditif. Namun, mereka harus digunakan dengan hati-hati. Pertama, bagian dalam tapal kuda tidak boleh diwarnai, karena setiap sistem reproduksi warna akan memiliki warna primernya sendiri dan hanya dapat mereproduksi beberapa bagian dari diagram kromatisitas. Kedua, karena fungsi pencocokan warna CIE tidak mewakili kepekaan kerucut manusia, jarak antara dua titik pada diagram kromatisitas bukan merupakan indikator yang baik untuk seberapa berbeda warna-warna ini akan dirasakan.

Diagram kromatisitas yang lebih seragam dikembangkan untuk setidaknya sebagian mengatasi masalah kedua ini. Diagram kromatisitas CIE u v memberikan jarak persepsi yang lebih seragam dan oleh karena itu umumnya lebih disukai daripada diagram kromatisitas (x,y). Dihitung dari (X,Y,Z) nilai tristimulus dengan menerapkan normalisasi yang berbeda,

$$u' = \frac{4X}{X + 15Y + 3Z}$$

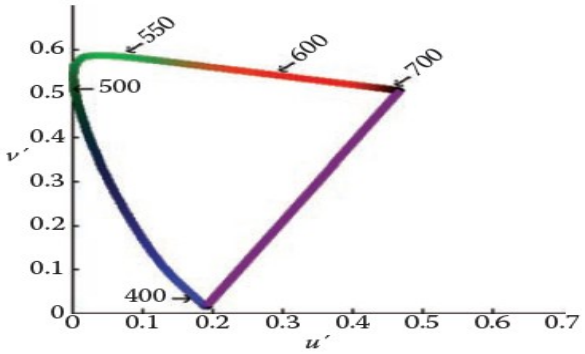
$$v' = \frac{9Y}{X + 15Y + 3Z}$$

dan alternatifnya dapat dihitung secara langsung dari (x,y) koordinat kromatisitas:

$$u' = \frac{4x}{-2x + 12y + 3}$$

$$v' = \frac{9x}{-2x + 12y + 3}$$

Diagram kromatisitas CIEu’v’ditunjukkan pada Gambar 19.8.



Gambar 19.8. Diagram kromatisitas CIE u'v'.

19.7 *COLOR SPACE*

Seperti dijelaskan di atas, setiap warna dapat diwakili oleh tiga angka, misalnya ditentukan oleh nilai tristimulus (X, Y, Z). Namun, primernya bersifat imajiner, artinya tidak mungkin membuat perangkat yang memiliki tiga sumber cahaya (semuanya positif) yang dapat mereproduksi semua warna dalam spektrum yang terlihat.

Untuk alasan yang sama, pengkodean gambar dan perhitungan pada gambar mungkin tidak praktis. Ada, misalnya, sejumlah besar kemungkinan nilai XYZ yang tidak sesuai dengan warna fisik apa pun. Hal ini akan menyebabkan penggunaan bit yang tersedia untuk penyimpanan tidak efisien dan persyaratan yang lebih tinggi untuk kedalaman bit untuk menjaga integritas visual setelah pemrosesan gambar. Meskipun dimungkinkan untuk membuat perangkat penangkap yang memiliki primer yang dekat dengan fungsi pencocokan warna CIE XYZ, biaya perangkat keras dan pemrosesan gambar menjadikan opsi ini tidak menarik. Tidak mungkin membuat tampilan yang sesuai dengan CIE XYZ. Untuk alasan ini, perlu untuk merancang ruang warna lain: realisasi fisik, pengkodean yang efisien, keseragaman persepsi, dan spesifikasi warna intuitif.

Ruang warna CIE XYZ masih aktif digunakan, sebagian besar untuk konversi antara ruang warna lainnya. Ini dapat dilihat sebagai ruang warna yang tidak bergantung pada perangkat. Ruang warna lain kemudian dapat didefinisikan dalam hubungan dengan CIEXYZ, yang sering ditentukan oleh transformasi tertentu. Misalnya, perangkat tampilan trikromatik linier dan aditif dapat diubah ke dan dari CIE XYZ melalui matriks 3 × 3 sederhana. Beberapa transformasi nonlinier juga dapat ditentukan, misalnya untuk meminimalkan kesalahan persepsi ketika data disimpan dengan kedalaman bit yang terbatas, atau untuk mengaktifkan tampilan langsung pada perangkat yang memiliki hubungan nonlinier antara sinyal input dan jumlah cahaya yang dipancarkan.

19.8 KONSTRUKSI TRANSFOM MATRIK

Untuk perangkat tampilan dengan tiga primer, katakanlah merah, hijau, dan biru, kita dapat mengukur komposisi spektral dari cahaya yang dipancarkan dengan mengirimkan vektor warna (1, 0, 0), (0, 1, 0), dan (0 , 0, 1). Vektor-vektor ini mewakili tiga kasus yaitu di mana salah satu primadona penuh, dan dua lainnya lepas. Dari output spektral terukur, kita kemudian dapat menghitung koordinat kromatisitas yang sesuai (x_R, y_R), (x_G, y_G), dan (x_B, y_B).

Tampilan titik putih didefinisikan sebagai spektrum yang dipancarkan ketika vektor warna (1, 1, 1) dikirim ke tampilan. Koordinat kromatisitasnya adalah (x_W, y_W). Tiga primer dan titik putih mencirikan tampilan dan masing-masing diperlukan untuk membangun matriks transformasi antara ruang warna tampilan dan CIE *XYZ*.

Keempat koordinat kromatisitas ini dapat diperluas menjadi triplet kromatisitas yang merekonstruksi koordinatz dari z = 1−x−y, mengarah ke kembar tiga(x_R, y_R, z_R), (x_G, y_G, z_G), (x_B, y_B, z_B), dan (x_W, y_W, z_W). Jika kita mengetahui luminansi maksimum titik putih, kita dapat menghitung nilai tristimulus yang sesuai (X_W, Y_W, Z_W) dan kemudian menyelesaikan himpunan persamaan berikut untuk skalar rasio luminansi S_R, S_G, dan S_B:

$$X_W = x_R S_R + x_G S_G + x_B S_B,$$
$$Y_W = y_R S_R + y_G S_G + y_B S_B,$$
$$Z_W = z_R S_R + z_G S_G + z_B S_B.$$

Konversi antara RGB dan XYZ kemudian diberikan oleh

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} XR\,SR & XGSG & XBSB \\ YRSR & YGSG & YBSB \\ ZRSR & ZGSG & ZBSB \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Luminansi warna tertentu dapat dihitung dengan mengevaluasi baris tengah matriks yang dibangun dengan cara ini:

$$Y = y_R S_R R + y_G S_G G + y_B S_B B.$$

Untuk mengkonversi antara XYZ dan RGB dari perangkat yang diberikan, matriks di atas dapat dengan mudah dibalik.

	R	*G*	*B*	White
x	0.6400	0.3000	0.1500	0.3127
y	0.3300	0.6000	0.0600	0.3290

Tabel 19.1. Koordinat kromatisitas (x, y) untuk primer dan titik putih yang ditentukan oleh ITU-R BT.709. Standar RGB juga menggunakan primer dan titik putih ini.

Jika sebuah gambar direpresentasikan dalam ruang warna RGB yang titik primer dan putihnya tidak diketahui, maka hal terbaik berikutnya adalah mengasumsikan bahwa gambar tersebut dikodekan dalam ruang warna RGB standar. Sebuah pilihan yang masuk akal kemudian mengasumsikan bahwa gambar itu ditentukan menurut ITU-R BT.709, yang merupakan spesifikasi yang digunakan untuk pengkodean dan penyiaran HDTV. Primer dan titik putihnya ditentukan dalam Tabel 19.1. Perhatikan bahwa titik primer dan putih yang sama digunakan untuk mendefinisikan ruang warna sRGB yang terkenal. Transformasi antara ruang warna RGB dan CIE XYZ ini dan sebaliknya diberikan oleh

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.4124 & 0.3576 & 0.1805 \\ 0.2126 & 0.7152 & 0.0722 \\ 0.0193 & 0.1192 & 0.9505 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 3.2405 & -1.5371 & -0.4985 \\ -0.9693 & 1.8706 & 0.0416 \\ 0.0556 & -0.2040 & 1.0572 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Dengan mengganti nilai RGB maksimum perangkat, kita dapat menghitung titik putih. Untuk ITU-R BT.709, nilai maksimumnya adalah (R_W,G_W,B_W)= (100.100.100), mengarah ke titik putih (X_W,Y_W,Z_W) = (95 .05.100.00, 108.90).

	R	*G*	*B*	White
x	0.6400	0.3000	0.1500	0.3127
y	0.3300	0.6000	0.0600	0.3290

Selain transformasi linier, ruang warna sRGB dicirikan oleh transformasi nonlinier berikutnya. Pengkodean nonlinier diberikan oleh

$$R_{\text{sRGB}} = \begin{cases} 1.055\,R^{1/2.4} - 0.055 & R > 0.0031308, \\ 12.92\,R & R \le 0.0031308; \end{cases}$$
$$G_{\text{sRGB}} = \begin{cases} 1.055\,G^{1/2.4} - 0.055 & G > 0.0031308, \\ 12.92\,G & G \le 0.0031308; \end{cases}$$
$$B_{\text{sRGB}} = \begin{cases} 1.055\,B^{1/2.4} - 0.055 & B > 0.0031308, \\ 12.92\,B & B < 0.0031308. \end{cases}$$

Pengkodean nonlinier ini membantu meminimalkan kesalahan persepsi karena kesalahan kuantisasi dalam aplikasi digital.

19.9 RUANG RGB TERGANTUNG PERANGKAT

Karena setiap perangkat biasanya memiliki kumpulan primer dan titik putihnya sendiri, kami menyebut ruang warna RGB terkait yang bergantung pada perangkat. Perlu dicatat bahwa bahkan jika semua perangkat ini beroperasi dalam ruang RGB, mereka mungkin memiliki warna primer dan titik putih yang sangat berbeda. Oleh karena itu, jika kita memiliki gambar yang ditentukan dalam beberapa ruang RGB, itu mungkin tampak sangat berbeda bagi kita, tergantung pada perangkat mana kita menampilkannya.

Ini jelas merupakan situasi yang tidak diinginkan, akibat kurangnya manajemen warna. Namun, jika gambar ditentukan dalam ruang warna RGB yang diketahui, pertama-tama dapat dikonversi ke XYZ, yang merupakan perangkat independen, dan kemudian dapat dikonversi ke ruang RGB perangkat yang akan ditampilkan.

Ada beberapa ruang warna RGB lain yang terdefinisi dengan baik. Mereka masing-masing terdiri dari transformasi armatrik linier diikuti oleh transformasi nonlinier, mirip dengan ruang warna RGB tersebut. Maka transformasi nonlinier dapat diparameterisasi sebagai berikut:

$$R_{\text{nonlinear}} = \begin{cases} (1+f)\,R^{\gamma} - f & t < R \le 1, \\ s\,R & 0 \le R \le t; \end{cases}$$
$$G_{\text{nonlinear}} = \begin{cases} (1+f)\,G^{\gamma} - f & t < G \le 1, \\ s\,G & 0 \le G \le t; \end{cases}$$
$$B_{\text{nonlinear}} = \begin{cases} (1+f)\,B^{\gamma} - f & t < B \le 1, \\ s\,B & 0 \le B \le t. \end{cases}$$

Parameter s, f, t dan γ, bersama dengan primer dan titik putih, menentukan kelas ruang warna RGB yang digunakan di berbagai industri. Beberapa transformasi umum tercantum dalam Tabel 19.2.

19.10 *CONE SPACE* LMS

Sinyal kerucut yang disebutkan di atas dapat dinyatakan dalam ruang warna CIEXYZ. Transformasi matriks untuk menghitung sinyal LMS dari XYZ dan sebaliknya diberikan oleh:

$$\begin{bmatrix} L \\ M \\ S \end{bmatrix} = \begin{bmatrix} 0.38971 & 0.68898 & -0.07868 \\ -0.22981 & 1.18340 & 0.04641 \\ 0.00000 & 0.00000 & 1.00000 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 1.91019 & -1.11214 & 0.20195 \\ 0.37095 & 0.62905 & 0.00000 \\ 0.00000 & 0.00000 & 1.00000 \end{bmatrix} \begin{bmatrix} L \\ M \\ S \end{bmatrix}$$

Tabel 19.2. Transformasi untuk ruang warna RGB standar (setelah (Pascale, 2003)).

Color space	XYZ to RGB matrix	RGB to XYZ matrix	Nonlinear transform
sRGB	$\begin{bmatrix} 3.2405 & -1.5371 & -0.4985 \\ -0.9693 & 1.8760 & 0.0416 \\ 0.0556 & -0.2040 & 1.0572 \end{bmatrix}$	$\begin{bmatrix} 0.4124 & 0.3576 & 0.1805 \\ 0.2126 & 0.7152 & 0.0722 \\ 0.0193 & 0.1192 & 0.9505 \end{bmatrix}$	$\gamma = 1/2.4 \approx 0.42$ $f = 0.055$ $s = 12.92$ $t = 0.0031308$
Adobe RGB (1998)	$\begin{bmatrix} 2.0414 & -0.5649 & -0.3447 \\ -0.9693 & 1.8760 & 0.0416 \\ 0.0134 & -0.1184 & 1.0154 \end{bmatrix}$	$\begin{bmatrix} 0.5767 & 0.1856 & 0.1882 \\ 0.2974 & 0.6273 & 0.0753 \\ 0.0270 & 0.0707 & 0.9911 \end{bmatrix}$	$\gamma = \frac{1}{2\frac{51}{256}} \approx \frac{1}{2.2}$ $f = $ N.A. $s = $ N.A. $t = $ N.A.
HDTV (HD-CIF)	$\begin{bmatrix} 3.2405 & -1.5371 & -0.4985 \\ -0.9693 & 1.8760 & 0.0416 \\ 0.0556 & -0.2040 & 1.0572 \end{bmatrix}$	$\begin{bmatrix} 0.4124 & 0.3576 & 0.1805 \\ 0.2126 & 0.7152 & 0.0722 \\ 0.0193 & 0.1192 & 0.9505 \end{bmatrix}$	$\gamma = 0.45$ $f = 0.099$ $s = 4.5$ $t = 0.018$
NTSC (1953)/ ITU-R BT.601-4	$\begin{bmatrix} 1.9100 & -0.5325 & -0.2882 \\ -0.9847 & 1.9992 & -0.0283 \\ 0.0583 & -0.1184 & 0.8976 \end{bmatrix}$	$\begin{bmatrix} 0.6069 & 0.1735 & 0.2003 \\ 0.2989 & 0.5866 & 0.1145 \\ 0.0000 & 0.0661 & 1.1162 \end{bmatrix}$	$\gamma = 0.45$ $f = 0.099$ $s = 4.5$ $t = 0.018$
PAL/SECAM	$\begin{bmatrix} 3.0629 & -1.3932 & -0.4758 \\ -0.9693 & 1.8760 & 0.0416 \\ 0.0679 & -0.2289 & 1.0694 \end{bmatrix}$	$\begin{bmatrix} 0.4306 & 0.3415 & 0.1783 \\ 0.2220 & 0.7066 & 0.0713 \\ 0.0202 & 0.1296 & 0.9391 \end{bmatrix}$	$\gamma = 0.45$ $f = 0.099$ $s = 4.5$ $t = 0.018$
SMPTE-C	$\begin{bmatrix} 3.5054 & -1.7395 & -0.5440 \\ -1.0691 & 1.9778 & 0.0352 \\ 0.0563 & -0.1970 & 1.0502 \end{bmatrix}$	$\begin{bmatrix} 0.3936 & 0.3652 & 0.1916 \\ 0.2124 & 0.7010 & 0.0865 \\ 0.0187 & 0.1119 & 0.9582 \end{bmatrix}$	$\gamma = 0.45$ $f = 0.099$ $s = 4.5$ $t = 0.018$
SMPTE-240M	$\begin{bmatrix} 2.042 & -0.565 & -0.345 \\ -0.894 & 1.815 & 0.032 \\ 0.064 & -0.129 & 0.912 \end{bmatrix}$	$\begin{bmatrix} 0.567 & 0.190 & 0.193 \\ 0.279 & 0.643 & 0.077 \\ 0.000 & 0.073 & 1.016 \end{bmatrix}$	$\gamma = 0.45$ $f = 0.1115$ $s = 4.0$ $t = 0.0228$
Wide Gamut	$\begin{bmatrix} 1.4625 & -0.1845 & -0.2734 \\ -0.5228 & 1.4479 & 0.0681 \\ 0.0346 & -0.0958 & 1.2875 \end{bmatrix}$	$\begin{bmatrix} 0.7164 & 0.1010 & 0.1468 \\ 0.2587 & 0.7247 & 0.0166 \\ 0.0000 & 0.0512 & 0.7740 \end{bmatrix}$	$\gamma = $ N.A. $f = $ N.A. $s = $ N.A. $t = $ N.A.

Transformasi ini dikenal sebagai transformasi Hunt-Pointer-Estevez (Hunt, 2004) dan digunakan dalam transformasi adaptasi kromatik serta dalam pemodelan tampilan warna.

19.11 CIE 1976 LA∗B∗

Ruang lawan warna dicirikan oleh saluran yang mewakili saluran akromatik (luminance), serta dua saluran yang mengkodekan oposisi warna. Ini sering merupakan saluran merah-hijau dan kuning-biru. Saluran lawan warna ini dengan demikian mengkodekan dua kromatisitas sepanjang satu sumbu, yang dapat memiliki nilai positif dan negatif. Misalnya, saluran merah-hijau mengkodekan merah untuk nilai positif dan hijau untuk nilai negatif. Nilai nol mengkodekan kasus khusus: netral yang bukan merah atau hijau. Saluran kuning-biru bekerja dengan cara yang hampir sama.

Karena setidaknya dua warna dikodekan pada masing-masing dari dua sumbu kromatik, tidak mungkin untuk mengkodekan campuran merah dan hijau. Juga tidak mungkin untuk mengkodekan kuning dan biru secara bersamaan. Meskipun ini mungkin tampak merugikan, diketahui bahwa sistem visual manusia menghitung atribut serupa di awal jalur visual. Akibatnya, manusia tidak dapat melihat warna yang secara bersamaan merah dan hijau, atau kuning dan biru. Kami tidak melihat sesuatu yang menyerupai hijau kemerahan, atau biru kekuning-kuningan. Kami, bagaimanapun, dapat melihat campuran warna seperti merah kekuningan (oranye) atau biru kehijauan, karena ini dikodekan di saluran kromatik.

Sistem lawan warna yang paling relevan untuk grafis komputer adalah model warna CIE 1976 L∗a∗b∗. Ini adalah ruang warna yang kurang lebih seragam, berguna, antara lain, untuk perhitungan perbedaan warna. Hal ini juga dikenal sebagai CIELAB.

Masukan ke CIELAB adalah nilai tristimulus stimulus (X,Y,Z) serta nilai tristimulus permukaan pencerminan putih difus yang diterangi oleh iluminan, (X_n, Y_n, Z_n). Oleh karena itu, CIELAB melampaui ruang warna biasa, karena ia memperhitungkan sepetak warna dalam konteks iluminasi yang diketahui. Dengan demikian dapat dilihat sebagai ruang penampilan warna yang belum sempurna.

Tiga saluran yang didefinisikan dalam CIELAB adalah L∗, a∗, dan b∗. Saluran L∗ mengkodekan kecerahan warna, yaitu, pantulan yang dirasakan dari tambalan dengan nilai tristimulus (X,Y,Z). Thea∗ dan b∗ adalah saluran chromaticopponent. Transformasi antara XYZ dan CIELAB diberikan oleh

$$\begin{bmatrix} L^* \\ a^* \\ b^* \end{bmatrix} = \begin{bmatrix} 0 & 116 & 0 & -16 \\ 500 & -500 & 0 & 0 \\ 0 & 200 & -200 & 0 \end{bmatrix} \begin{bmatrix} f\,(X/X_n) \\ f\,(Y/Y_n) \\ f\,(Z/Z_n) \\ 1 \end{bmatrix}.$$

Fungsi f didefinisikan sebagai

$$f(r) = \begin{cases} \sqrt[3]{r} & \text{for } r > 0.008856, \\ 7.787\,r + \dfrac{16}{116} & \text{for } r \le 0.008856. \end{cases}$$

Seperti yang dapat dilihat dari formulasi ini, saluran kromatik bergantung pada luminansi Y . Meskipun persepsi ini akurat, itu berarti bahwa kita tidak dapat memplot nilai a∗ dan b∗ dalam diagram kromatisitas. KecerahanL∗ dinormalisasi antara 0 dan 100 untuk hitam dan putih. Meskipun saluran a∗ dan b∗ tidak dibatasi secara eksplisit, mereka biasanya dalam kisaran [−128, 128].

19.12 ADAPTASI CHROMATIC

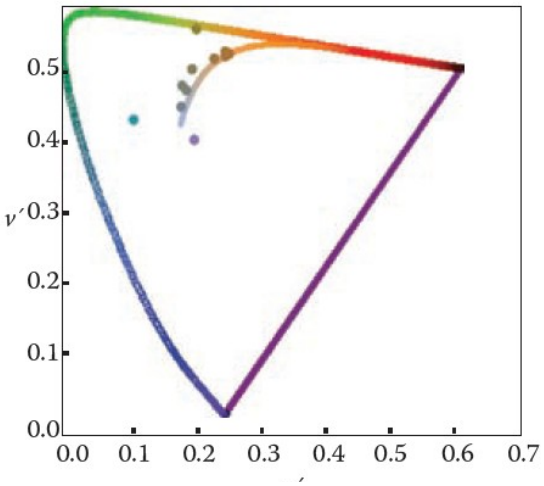
Ruang warna CIELAB yang baru saja dijelaskan mengambil nilai tristimulus dari stimulus dan nilai tristimulus cahaya yang dipantulkan dari patch difus putih sebagai input. Dengan demikian, ia membentuk awal dari sebuah sistem di mana lingkungan pandang diperhitungkan.

Lingkungan tempat kita mengamati objek dan gambar memiliki pengaruh besar pada cara kita memandang objek tersebut. Rentang lingkungan menonton yang kita temui dalam kehidupan sehari-hari sangat luas, dari cahaya matahari hingga cahaya bintang dan dari cahaya lilin hingga cahaya neon. Kondisi pencahayaan tidak hanya merupakan kisaran yang sangat besar dalam jumlah cahaya yang ada, tetapi juga sangat bervariasi dalam warna cahaya yang dipancarkan.

Sistem visual manusia mengakomodasi perubahan lingkungan ini melalui proses yang disebut adaptasi. Ada tiga jenis adaptasi yang dapat dibedakan, yaitu adaptasi terang, adaptasi gelap, dan adaptasi kromatik. Adaptasi cahaya mengacu pada perubahan yang terjadi ketika kita berpindah dari lingkungan yang sangat gelap ke lingkungan yang sangat terang. Ketika ini terjadi, pada awalnya kita terpesona oleh cahaya, tetapi segera kita beradaptasi dengan situasi baru dan mulai membedakan objek di lingkungan kita. Adaptasi gelap mengacu pada kebalikannya—ketika kita beralih dari lingkungan terang ke lingkungan gelap. Pada awalnya, kita melihat sangat sedikit, tetapi setelah jangka waktu tertentu, detail akan mulai muncul. Waktu yang dibutuhkan untuk beradaptasi dengan kegelapan umumnya jauh lebih lama daripada untuk adaptasi cahaya.

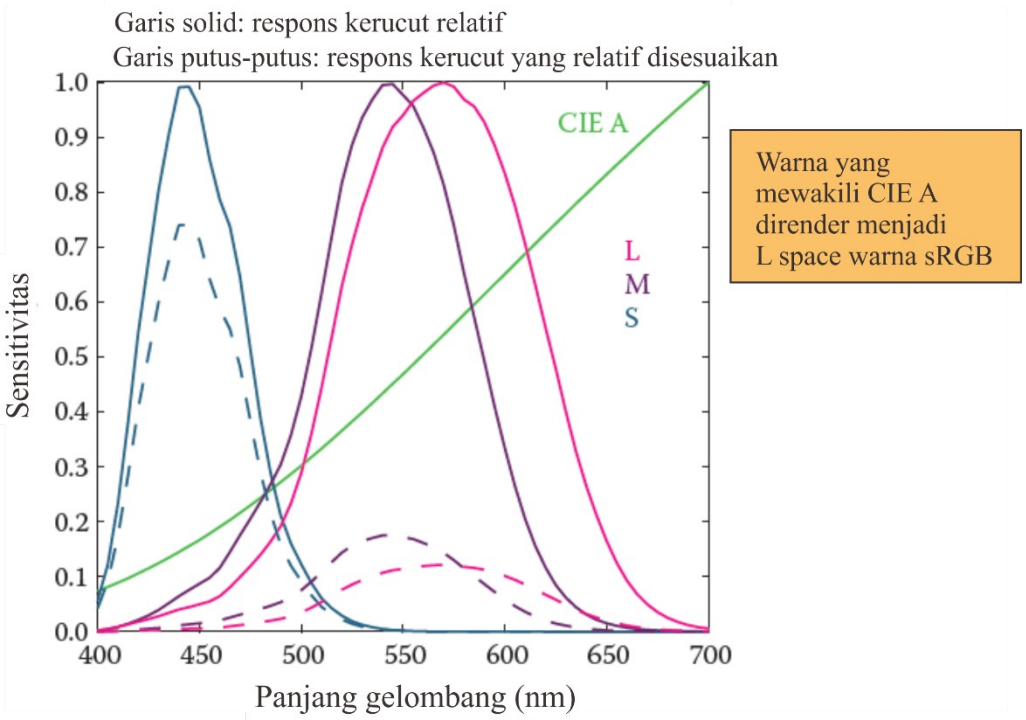
Adaptasi kromatik mengacu pada kemampuan kita untuk beradaptasi, dan sebagian besar mengabaikan, variasi warna iluminasi. Adaptasi kromatik, pada dasarnya, setara biologis dari operasi white balancing yang tersedia di sebagian besar kamera modern. Sistem visual manusia secara efektif menormalkan kondisi tampilan untuk menghadirkan pengalaman visual yang cukup konsisten. Dengan demikian, kami menunjukkan sejumlah keteguhan warna: pantulan objek tampak relatif konstan meskipun ada variasi dalam iluminasi.

Meskipun kami dapat mengabaikan sebagian besar perubahan dalam lingkungan tampilan, kami tidak dapat melakukannya sepenuhnya. Misalnya, warna tampak jauh lebih berwarna pada hari yang cerah daripada pada hari berawan. Meskipun penampilan telah berubah, kami tidak menganggap bahwa objek yang mencerminkan diri mereka sendiri sebenarnya telah mengubah sifat fisik mereka. Dengan demikian, kami memahami bahwa kondisi pencahayaan telah memengaruhi tampilan warna secara keseluruhan.



Gambar 19.9. Serangkaian sumber cahaya diplot dalam diagram kromatisitas CIE u'v'. Selembar kertas putih yang disinari oleh salah satu sumber cahaya ini mempertahankan tampilan warna putih.

Meskipun demikian, keteguhan warna berlaku untuk konten berwarna. Adaptasi kromatik memungkinkan objek putih tampak putih untuk sejumlah besar kondisi pencahayaan, seperti yang ditunjukkan pada Gambar 19.9.



Gambar 19.10. Contoh kontrol penguatan fotoreseptor independen gaya von Kries. Respon kerucut relatif (garis padat) dan respons kerucut relatif yang disesuaikan dengan iluminan CIE A (putus-putus) ditampilkan. Bagian warna yang terpisah mewakili iluminan CIE Dialihkan ke dalam ruang warna RGB

Model komputasi adaptasi kromatik cenderung berfokus pada mekanisme kontrol penguatan di kerucut. Salah satu model paling sederhana mengasumsikan bahwa setiap kerucut beradaptasi secara independen terhadap energi yang diserapnya. Ini berarti bahwa jenis kerucut yang berbeda beradaptasi secara berbeda tergantung pada spektrum cahaya yang diserap. Adaptasi tersebut kemudian dapat dimodelkan sebagai scalling sinyal kerucut yang adaptif dan independen:

$$L_a = \alpha\, L,$$
$$M_a = \beta\, M,$$
$$S_a = \gamma\, S,$$

di mana (L_a, M_a, S_a) adalah sinyal kerucut yang disesuaikan secara kromatis, dan α, β,dan γ adalah kontrol penguatan independen yang ditentukan oleh lingkungan tampilan. Jenis adaptasi independen ini juga dikenal sebagai adaptasi von-Kries. Contohnya ditunjukkan pada Gambar 19.10.

Pencahayaan yang beradaptasi dapat diukur dari permukaan putih di tempat kejadian. Dalam kasus yang ideal, ini akan menjadi permukaan Lambertian. Dalam gambar digital, iluminasi penyesuaian kepala juga dapat didekati sebagai nilai tristimulus maksimum dari scene. Pengukur cahaya atau perhitungan dengan cara ini, headaptingwhite, diberikan oleh (L_w, M_w, S_w). Von Kries adaptation kemudian secara sederhana melakukan scalling dengan kebalikan dari putih yang mengadaptasi.

Dalam banyak kasus, kami tertarik pada stimulus apa yang harus dihasilkan di bawah satu iluminasi untuk mencocokkan warna tertentu di bawah iluminasi yang berbeda. Misalnya, jika kita memiliki tambalan berwarna yang diterangi oleh cahaya siang hari, kita mungkin bertanya pada diri sendiri nilai tristimulus apa yang harus dihasilkan untuk membuat tambalan warna yang cocok yang akan diterangi oleh cahaya pijar.

Oleh karena itu, kami tertarik untuk menghitung warna yang sesuai, yang dapat dicapai dengan mengalirkan dua perhitungan adaptasi kromatik. Intinya, transformasi von Kries yang disebutkan sebelumnya membagi iluminan yang beradaptasi—dalam contoh kami, iluminasi siang hari. Jika kita kemudian mengalikan dalam lampu pijar, kita telah menghitung warna yang sesuai.

Ada beberapa transformasi adaptasi kromatik yang lebih rumit dan lebih akurat yang ada (Reinhard et al., 2008). Namun, model von Kries yang sederhana tetap sangat efektif dalam memodelkan adaptasi kromatik dan dengan demikian dapat digunakan untuk mencapai keseimbangan putih dalam gambar digital.

Pentingnya adaptasi kromatik dalam konteks rendering, adalah bahwa kita telah selangkah lebih dekat dengan memperhitungkan lingkungan pandang pengamat, tanpa harus mengoreksinya dengan menyesuaikan scene dan merender ulang gambar kita. Sebagai gantinya, kita dapat memodelkan dan merender scene kita, dan kemudian, sebagai pascaproses gambar, mengoreksi iluminasi lingkungan tampilan. Untuk memastikan bahwa whitebalancing tidak memperkenalkan artefak, namun, penting untuk memastikan bahwa gambar dirender ke format titik mengambang. Jika dirender ke format gambar 8bit tradisional, transformasi adaptasi kromatik dapat memperbesar kesalahan kuantisasi.

19.13 TAMPILAN WARNA

Sementara kolorimetri memungkinkan kita untuk secara akurat menentukan dan mengkomunikasikan warna dengan cara yang tidak bergantung pada perangkat, dan adaptasi kromatik memungkinkan kita untuk memprediksi kecocokan warna di seluruh perubahan dalam iluminasi, alat ini masih cukup untuk menggambarkan seperti apa warna sebenarnya.

Untuk memprediksi persepsi sebenarnya dari suatu objek, kita perlu mengetahui lebih banyak informasi tentang lingkungan dan memperhitungkan informasi itu. Sistem visual manusia selalu beradaptasi dengan lingkungannya, yang berarti bahwa persepsi warna akan sangat dipengaruhi oleh perubahan tersebut. Model penampilan warna memperhitungkan ukuran rangsangan itu sendiri, serta lingkungan pandang. Ini berarti bahwa deskripsi warna yang dihasilkan tidak tergantung pada kondisi tampilan.

Pentingnya pemodelan penampilan warna dapat dilihat pada contoh berikut. Pertimbangkan gambar yang ditampilkan pada layar LCD. Saat mencetak gambar yang sama dan melihatnya dalam konteks yang berbeda, seringkali gambar akan terlihat sangat berbeda. Model tampilan warna dapat digunakan untuk memprediksi perubahan yang diperlukan untuk menghasilkan reproduksi warna lintas media yang akurat (Fairchild, 2005).

Meskipun pemodelan tampilan warna menawarkan alat penting untuk reproduksi warna, implementasi sebenarnya cenderung relatif rumit dan tidak praktis dalam penggunaan praktis. Dapat diantisipasi bahwa situasi ini dapat berubah seiring waktu. Namun, sampai saat itu, kami meninggalkan deskripsi mereka ke buku teks yang lebih khusus (Fairchild, 2005).

19.14 CATATAN

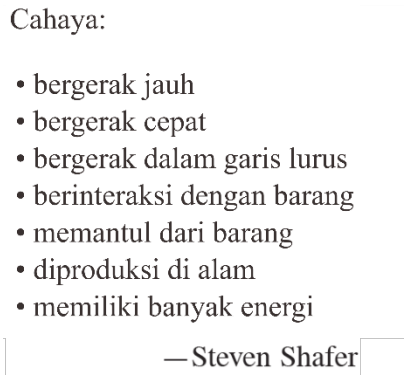
Dari semua buku tentang teori warna, karya Reinhard et al. (Reinhard et al., 2008) paling langsung diarahkan ke disiplin ilmu teknik, termasuk grafis komputer, visi komputer, dan pemrosesan gambar. Pengantar umum lain untuk teori warna diberikan oleh Berns (Berns, 2000) dan Stone (Stone, 2003). Wyszecki dan Stiles telah menghasilkan volume data dan formula yang komprehensif, membentuk karya referensi yang sangat diperlukan (Wyszecki & Stiles, 2000). Untuk reproduksi warna, kami merekomendasikan buku Hunt (Hunt, 2004). Model penampilan warna secara komprehensif dijelaskan dalam buku Fairchild (Fairchild, 2005). Untuk masalah warna yang terkait dengan video dan HDTV, buku Poynton sangat penting (Poynton, 2003).

BAB 20
PERSEPSI VISUAL

Tujuan akhir dari komputer grafis adalah untuk menghasilkan gambar untuk dilihat oleh orang-orang. Jadi, keberhasilan sistem grafis komputer tergantung pada seberapa baik sistem itu menyampaikan informasi yang relevan kepada manusia pengamat. Kompleksitas intrinsik dari dunia fisik dan keterbatasan perangkat layar memungkinkan pemirsa untuk menampilkan pola cahaya yang identik yang akan terjadi ketika melihat lingkungan alami. Ketika tujuan sistem grafis komputer adalah realisme fisik, hal terbaik yang dapat kita harapkan adalah sistem tersebut efektif secara persepsi: gambar yang ditampilkan harus "terlihat" sebagaimana dimaksud. Untuk aplikasi seperti ilustrasi teknis, seringkali diinginkan untuk menyoroti informasi yang relevan secara visual dan efektivitas persepsi menjadi persyaratan eksplisit.

Seniman dan ilustrator telah mengembangkan secara empiris berbagai alat dan teknik untuk menyampaikan informasi visual secara efektif. Salah satu pendekatan untuk meningkatkan efektivitas persepsi grafis komputer adalah dengan memanfaatkan metode ini dalam sistem otomatis kami. Pendekatan kedua dibangun secara langsung di atas pengetahuan tentang sistem penglihatan manusia dengan menggunakan efektivitas persepsi sebagai kriteria optimasi dalam desain sistem grafis komputer. Kedua pendekatan ini tidak sepenuhnya berbeda. Memang, salah satu pemeriksaan sistematis pertama dari persepsi visual ditemukan dalam buku catatan Leonardoda Vinci.

Sisa bab ini memberikan gambaran sebagian tentang apa yang diketahui tentang persepsi visual pada orang. Penekanannya adalah pada aspek penglihatan manusia yang paling relevan dengan grafis komputer. Sistem visual manusia sangat kompleks baik dalam operasi maupun arsitekturnya. Bab seperti ini paling-paling dapat memberikan ringkasan poin-poin penting, dan penting untuk menghindari generalisasi yang berlebihan dari apa yang disajikan di sini. Perlakuan lebih mendalam dari persepsi visual dapat ditemukan di Wandell (1995) dan Palmer (1999); Gregory (1997) dan Yantis (2000) memberikan informasi tambahan yang berguna. Referensi visi komputer yang baik seperti Forsyth dan Ponce (2002) juga membantu. Penting untuk dicatat bahwa meskipun lebih dari 150 tahun penelitian intensif, pengetahuan kita tentang banyak aspek penglihatan masih sangat terbatas dan tidak sempurna.



Gambar 20.1. Sifat cahaya membuat penglihatan menjadi indra yang kuat.

20.1 ILMU VISI

Penglihatan umumnya disepakati sebagai indra yang paling kuat pada manusia. Penglihatan menghasilkan informasi yang lebih berguna tentang dunia daripada mendengar sentuhan, penciuman, atau rasa. Ini adalah konsekuensi langsung dari fisika cahaya (Gambar 20.1). Penerangan menyebar, terutama pada siang hari tetapi juga pada malam hari karena cahaya bulan, cahaya bintang, dan sumber buatan. Permukaan mencerminkan sebagian besar iluminasi insiden dan melakukannya dengan cara yang istimewa untuk bahan tertentu dan yang bergantung pada bentuk permukaan. Fakta bahwa cahaya (kebanyakan) bergerak dalam garis lurus melalui udara memungkinkan penglihatan untuk memperoleh informasi dari lokasi yang jauh.

Studi tentang visi memiliki sejarah yang panjang dan kaya. Banyak dari apa yang kita ketahui tentang mata menelusuri kembali karya para filsuf dan fisikawan di tahun 1600-an. Dimulai pada pertengahan 1800-an, ada ledakan pekerjaan oleh psikolog perseptual yang mengeksplorasi fenomenologi penglihatan dan mengusulkan model bagaimana penglihatan dapat bekerja. Pertengahan 1900-an melihat awal ilmu saraf modern, yang menyelidiki baik kerja skala halus neuron individu maupun organisasi arsitektur otak dan sistem saraf berskala besar. Sebagian besar penelitian ilmu saraf berfokus pada penglihatan. Baru-baru ini, ilmu komputer telah berkontribusi pada pemahaman persepsi visual dengan menyediakan alat untuk secara tepat menggambarkan model hipotesis komputasi visual dan dengan memungkinkan pemeriksaan empiris program visi komputer. Istilah ilmu penglihatan diciptakan untuk merujuk pada studi multidisiplin tentang persepsi visual yang melibatkan psikologi perseptual, ilmu saraf, dan analisis komputasi.

Ilmu penglihatan memandang tujuan penglihatan sebagai menghasilkan informasi tentang objek, lokasi, dan peristiwa di dunia dari pola cahaya yang dicitrakan yang mencapai pemirsa. Psikolog menggunakan istilah stimulus distal untuk merujuk pada dunia fisik yang sedang diamati dan stimulus proksimal untuk merujuk pada gambar retina.[4] Menggunakan terminologi ini, fungsi penglihatan adalah untuk menghasilkan deskripsi aspek stimulus distal yang diberikan stimulus proksimal. Persepsi visual dikatakan veridikal apabila deskripsi yang dihasilkan secara akurat mencerminkan dunia nyata. Dalam praktiknya, tidak masuk akal untuk memikirkan deskripsi objek, lokasi, dan peristiwa ini secara terpisah. Sebaliknya, penglihatan lebih dipahami dalam konteks fungsi motorik dan kognitif yang dilayaninya.

20.2 SENSITIVITAS VISUAL

Sistem penglihatan membuat deskripsi lingkungan visual berdasarkan sifat iluminasi insiden. Akibatnya, penting untuk memahami sifat iluminasi insiden yang dapat dideteksi oleh sistem penglihatan manusia. Satu pengamatan kritis tentang sistem penglihatan manusia adalah bahwa ia terutama sensitif terhadap pola cahaya daripada sensitif terhadap besarnya energi cahaya yang mutlak. Mata tidak beroperasi sebagai fotometer. Sebaliknya, ia mendeteksi pola spasial, temporal, dan spektral dalam cahaya yang dicitrakan di retina dan informasi tentang pola cahaya ini membentuk dasar untuk semua persepsi visual.

Ada utilitas ekologis untuk sensitivitas sistem penglihatan terhadap variasi iluminasi ruang dan waktu. Mampu untuk secara akurat merasakan perubahan di lingkungan sangat penting untuk kelangsungan hidup kita.[5] Sebuah sistem yang mengukur perubahan energi cahaya daripada besarnya energi itu sendiri juga masuk akal secara rekayasa, karena membuatnya lebih mudah untuk mendeteksi pola cahaya pada rentang intensitas cahaya yang

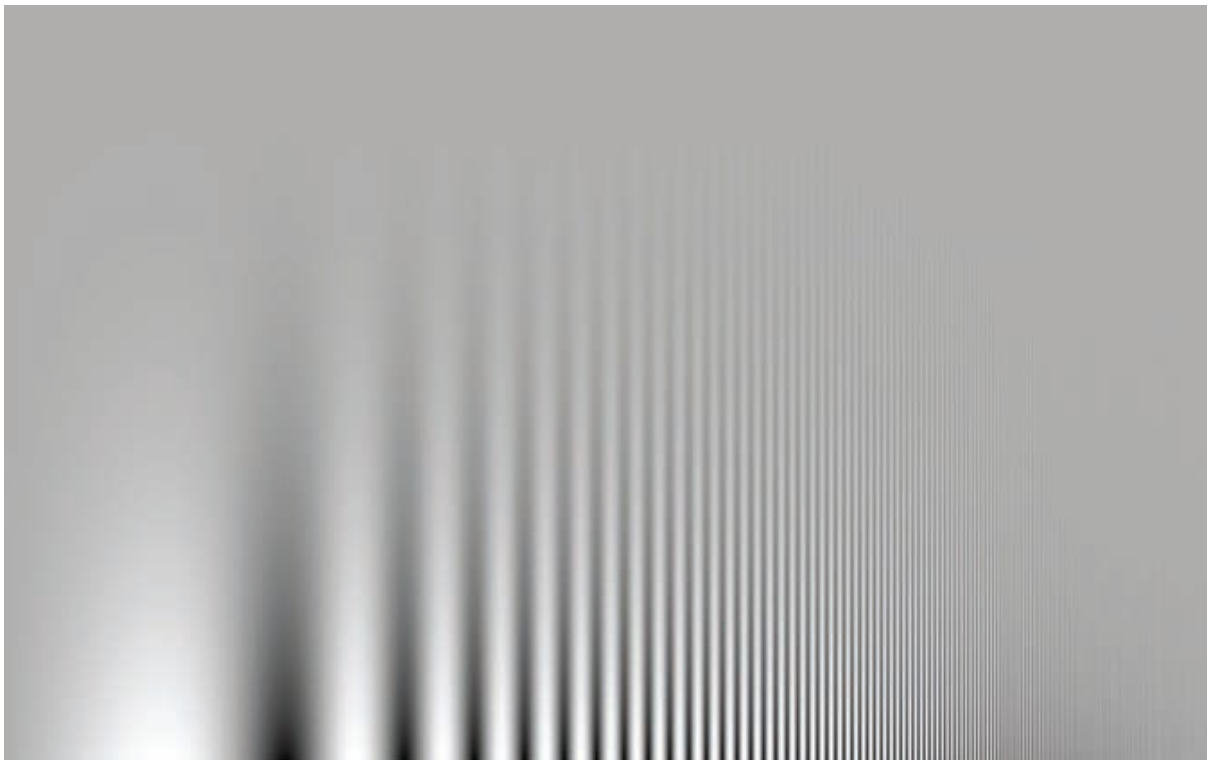
[4] Dalam visi komputer, istilah scene sering digunakan untuk merujuk ke dunia luar, sedangkan istilah gambar digunakan untuk merujuk pada proyeksi scene ke bidang penginderaan.

[5] Kadang-kadang dikatakan bahwa tujuan utama dari penglihatan adalah untuk mendukung makan, menghindari dimakan, reproduksi, dan menghindari malapetaka saat bergerak. Memikirkan visi sebagai aktivitas yang diarahkan pada tujuan seringkali berguna, tetapi perlu dilakukan pada tingkat yang lebih rinci.

besar. Ini adalah hal yang baik untuk grafis komputer bahwa visi beroperasi dengan cara ini. Perangkat tampilan secara fisik terbatas dalam kemampuannya untuk memproyeksikan cahaya dengan kekuatan dan jangkauan dinamis yang khas dari scene alam. Tampilan grafis tidak akan efektif jika mereka perlu menghasilkan pola cahaya yang identik dengan dunia fisik yang sesuai. Untungnya, yang diperlukan hanyalah tampilan yang mampu menghasilkan pola perubahan spasial dan temporal yang serupa dengan dunia nyata.

20.3 KECERAHAN DAN KONTRAS

Dalam cahaya terang, sistem visual manusia mampu membedakan kisi-kisi yang terdiri dari garis terang dan gelap paralel kontras tinggi sehalus 50–60 siklus/derajat. (Dalam hal ini, "siklus" terdiri dari sepasang bilah terang dan gelap yang berdekatan.) Sebagai perbandingan, monitor komputer LCD terbaik yang tersedia saat ini, pada jarak pandang normal, menampilkan pola kandil dengan baik sekitar 20 siklus/derajat. Perbedaan kontras minimum pada tepi yang dapat dideteksi oleh sistem visual manusia dalam cahaya terang adalah sekitar 1% dari rata-rata pencahayaan di tepi. Di sebagian besar tampilan 8-bit, perbedaan tingkat abu-abu tunggal sering terlihat pada setidaknya sebagian dari rentang intensitas karena sifat mapping dari tingkat abu-abu ke pencahayaan tampilan aktual.



Gambar 20.2. Kontras antara garis-garis meningkat secara konstan dari atas ke bawah, namun ambang visibilitas bervariasi dengan frekuensi.

Mengkarakterisasi kemampuan sistem visual untuk mendeteksi pola skala halus (ketajaman visual) dan untuk mendeteksi perubahan kecerahan jauh lebih rumit daripada kamera dan perangkat akuisisi gambar serupa. Seperti yang ditunjukkan pada Gambar 20.2, ada interaksi antara kontras dan ketajaman dalam penglihatan manusia. Dalam gambar, skala pola menurun dari kiri ke kanan sementara kontras meningkat dari atas ke bawah. Jika Anda melihat gambar pada jarak pandang normal, akan jelas bahwa kontras terendah di mana sebuah pola terlihat adalah fungsi dari frekuensi spasial dari pola tersebut.

Ada hubungan linier antara intensitas cahaya L yang mencapai mata dari titik permukaan tertentu di dunia, intensitas cahaya I yang menyinari titik permukaan itu, dan ada ektivitas R dari permukaan pada titik yang diamati:

$$L = \alpha I \cdot R,$$

di mana bergantung pada hubungan antara geometri permukaan, pola iluminasi insiden, dan arah pandang. Sementara mata hanya dapat mengukur L secara langsung, penglihatan manusia jauh lebih baik dalam memperkirakan R daripada L. Untuk melihatnya, lihat Gambar 20.3 dalam cahaya langsung yang terang. Gunakan tangan Anda untuk membayangi salah satu pola, membiarkan yang lain langsung menyala. Sementara cahaya yang dipantulkan dari dua pola akan berbeda secara signifikan, kecerahan yang tampak dari dua kotak tengah akan tampak hampir sama. Istilah lightness sering digunakan untuk menggambarkan kecerahan yang tampak dari suatu permukaan, yang berbeda dari luminance yang sebenarnya. Dalam banyak situasi, ringan tidak berubah terhadap perubahan besar dalam iluminasi, sebuah fenomena yang disebut sebagai keteguhan cahaya.

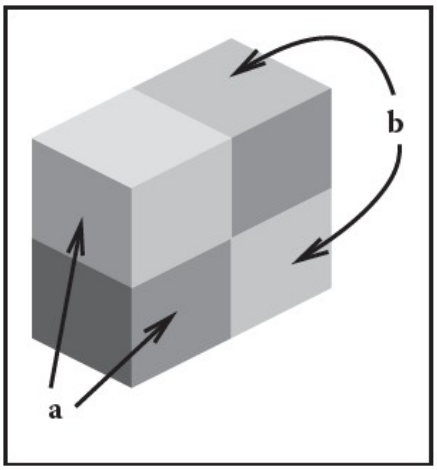
Gambar 20.3. Keteguhan ringan. Berikan bayangan di atas salah satu pola dengan tangan Anda dan perhatikan bahwa kecerahan yang tampak dari dua kotak tengah tetap hampir sama.

Mekanisme bagaimana sistem visual manusia mencapai keteguhan ringan tidak dipahami dengan baik. Seperti yang ditunjukkan pada Gambar 20.2, sistem penglihatan relatif tidak sensitif terhadap pola cahaya yang berubah-ubah secara perlahan, yang dapat berfungsi untuk mengabaikan efek iluminasi yang bervariasi secara perlahan. Kecerahan semu dipengaruhi oleh kecerahan daerah sekitarnya (Gambar 20.4). Ini dapat membantu keteguhan cahaya ketika area diterangi dengan cara yang berbeda. Sementara efek kontras simultan ini sering digambarkan sebagai modifikasi dari kecerahan yang dirasakan dari satu wilayah berdasarkan kontras kecerahan di wilayah sekitarnya, sebenarnya jauh lebih rumit dari itu (Gambar 20.5 dan 20.6). Untuk lebih lanjut tentang persepsi ringan, lihat (Gilchrist et al., 1999) dan (Adelson, 1999).

Gambar 20.4. (a) Kontras simultan: kecerahan semu dari batang tengah dipengaruhi oleh kecerahan area sekitarnya; (b) Batang yang sama tanpa variabel surround.

Gambar 20.5. Ilusi Munker-White menunjukkan kompleksitas kontras simultan.

Pada Gambar 20.4, wilayah tengah tampak lebih terang ketika area sekitarnya lebih gelap. Dalam (a), garis abu-abu di sebelah kiri terlihat lebih terang daripada garis abu-abu di sebelah kanan, meskipun mereka hampir dikelilingi oleh daerah putih; (b) menunjukkan strip abu-abu tanpa garis hitam.



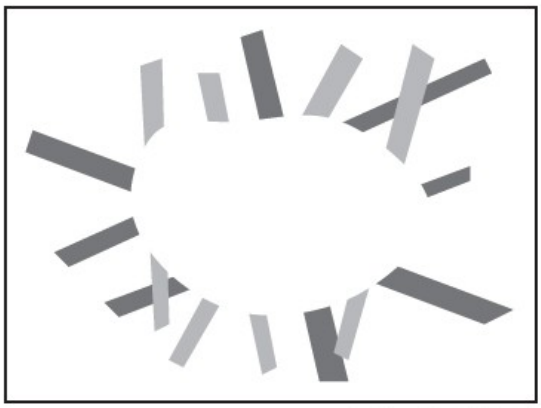
Gambar 20.6. Persepsi cahaya dipengaruhi oleh persepsi struktur 3D. Dua permukaan yang ditandai (a) memiliki kecerahan yang sama, seperti halnya dua permukaan yang ditandai (b) (setelah Adelson (1999)).

Sementara sistem visual sebagian besar mengabaikan pola intensitas yang bervariasi secara perlahan, sistem ini sangat sensitif terhadap tepi yang terdiri dari garis diskontinuitas dalam kecerahan. Tepi dalam intensitas cahaya yang dicitrakan sering sesuai dengan batas permukaan atau fitur penting lainnya di lingkungan (Gambar 20.7). Sistem penglihatan juga dapat mendeteksi perbedaan lokal dalam gerakan, disparitas stereo, tekstur, dan beberapa properti gambar lainnya. Namun, sistem penglihatan memiliki kemampuan yang sangat kecil untuk mendeteksi diskontinuitas spasial dalam warna bila tidak disertai dengan perbedaan dalam salah satu sifat lainnya.

Gambar 20.7. (a) Grayscale Image, (b) tepi image, yang merupakan garis dengan variabilitas spasial tinggi pada beberapa arah.

Gambar 20.8. Sistem visual terkadang melihat "tepian" bahkan ketika tidak ada diskontinuitas yang tajam dalam kecerahan, seperti yang terjadi di sisi kanan pola pusat pada gambar ini.

Persepsi tepi tampaknya berinteraksi dengan persepsi bentuk. Sementara tepi memberikan informasi yang dibutuhkan untuk menyimpan bentuk pengenalan pada sistem visual, kecerahan yang bervariasi secara perlahan dapat muncul sebagai tepi tajam jika tepi yang dihasilkan membuat bentuk yang lebih lengkap (Gambar 20.8). Gambar 20.9 menunjukkan kontur subjektif, bentuk ekstrim dari efek ini di mana kontur tertutup terlihat meskipun tidak ada kontur seperti itu pada gambar sebenarnya. Akhirnya, kepekaan sistem penglihatan terhadap tepi juga tampak sebagai bagian dari mekanisme yang terlibat dalam persepsi cahaya. Perhatikan bahwa daerah yang dilingkupi oleh kontur subyektif pada Gambar 20.9 tampak sedikit lebih terang daripada daerah sekitar halaman. Gambar 20.10 menunjukkan interaksi yang berbeda antara edge dan lightness. Dalam hal ini, profil kecerahan tertentu di tepi memiliki efek dramatis pada kecerahan yang tampak dari permukaan di kedua sisi tepi.



Gambar 20.9. Terkadang, sistem visual akan "melihat" kontur subyektif tanpa perubahan kecerahan yang terkait.

Gambar 20.10. Kecerahan yang dirasakan lebih bergantung pada kontras lokal daripada kecerahan di seluruh permukaan. Coba tutupi tepi vertikal di tengah gambar dengan pensil. Gambar ini adalah contoh dari ilusi Craik-O'Brien-Cornsweet.

Seperti yang ditunjukkan di atas, orang dapat mendeteksi perbedaan kecerahan antara dua wilayah yang berdekatan jika perbedaannya setidaknya 1% dari kecerahan rata-rata. Ini adalah contoh hukum Weber, yang menyatakan bahwa ada rasio konstan antara perbedaan yang nyata (jnd) dalam suatu stimulus dan besarnya stimulus:

Persamaan 20.2

$$\frac{\Delta I}{I} = k_1$$

di mana I adalah besarnya stimulus, I adalah besarnya perbedaan yang nyata, dan k_1 adalah konstanta khusus untuk stimulus. Hukum Weber dipostulasikan pada tahun 1846 dan masih tetap menjadi karakterisasi yang berguna dari banyak efek persepsi. Hukum Fechner, yang diusulkan pada tahun 1860, menggeneralisasikan hukum Weber dengan cara yang memungkinkan deskripsi kekuatan pengalaman indrawi apa pun, bukan hanya jnd:

Persamaan 20.3

$$S = k_2 \log(I),$$

di mana S adalah kekuatan persepsi dari pengalaman sensorik, I adalah besaran fisik dari stimulus yang sesuai, dan k_2 adalah konstanta scalling khusus untuk stimulus. Praktik saat ini

adalah memodelkan hubungan antara kekuatan stimulus yang dirasakan dan aktual menggunakan fungsi daya (hukum Stevens):

Persamaan 20.4

$$S = k_3 I^b,$$

di mana S dan I seperti sebelumnya, k3 adalah konstanta scalling lain, dan b adalah eksponen spesifik untuk stimulus. Untuk sejumlah besar kuantitas persepsi yang melibatkan penglihatan, b < 1. Ruang warna CIEL∗a∗b∗, dijelaskan di tempat lain, menggunakan representasi hukum Stevens yang dimodifikasi untuk mengkarakterisasi perbedaan persepsi antara nilai kecerahan. Perhatikan bahwa dalam dua karakterisasi pertama dari kekuatan persepsi stimulus dan dalam Hukum Stevens ketika b < 1, perubahan stimulus ketika besarnya rata-rata kecil menciptakan efek persepsi yang lebih besar daripada melakukan perubahan fisik yang sama pada stimulus ketika memiliki besaran yang lebih besar.

"Hukum" yang dijelaskan di atas bukanlah batasan fisik tentang bagaimana persepsi beroperasi. Sebaliknya, mereka adalah generalisasi tentang bagaimana sistem persepsi merespon rangsangan fisik tertentu. Di bidang psikologi persepsi, studi kuantitatif tentang hubungan antara rangsangan fisik dan efek persepsinya disebut psikofisika. Sementara hukum psikofisik adalah pengamatan yang diturunkan secara empiris daripada akun mekanistik, fakta bahwa begitu banyak efek persepsi yang dimodelkan dengan baik oleh fungsi kekuatan sederhana sangat mencolok dan dapat memberikan wawasan tentang mekanisme yang terlibat.

20.4 WARNA

Pada tahun 1666, Isaac Newton menggunakan prisma untuk menunjukkan bahwa sinar matahari yang tampaknya putih dapat diuraikan menjadi spektrum warna dan bahwa warna-warna ini dapat digabungkan kembali untuk menghasilkan cahaya yang tampak putih. Kita sekarang tahu bahwa energi cahaya terdiri dari kumpulan foton, masing-masing dengan panjang gelombang tertentu. Distribusi spektral cahaya adalah ukuran energi rata-rata cahaya pada setiap panjang gelombang. Untuk penerangan alami, distribusi spektral cahaya yang dipantulkan dari permukaan bervariasi secara signifikan tergantung pada bahan permukaan. Oleh karena itu, karakterisasi distribusi spektral ini dapat memberikan informasi visual untuk sifat permukaan di lingkungan.

Kebanyakan orang memiliki indera warna yang meresap ketika mereka melihat dunia. Persepsi warna tergantung pada distribusi frekuensi cahaya, dengan spektrum yang terlihat bagi manusia mulai dari panjang gelombang sekitar 370 nm hingga panjang gelombang sekitar 730 nm (lihat Gambar 20.11). Cara sistem visual memperoleh rasa warna dari distribusi spektral ini pertama kali diperiksa secara sistematis pada tahun 1801 dan tetap sangat kontroversial selama 150 tahun. Masalahnya adalah bahwa sistem visual merespons pola distribusi spektral sangat berbeda dari pola distribusi luminansi.

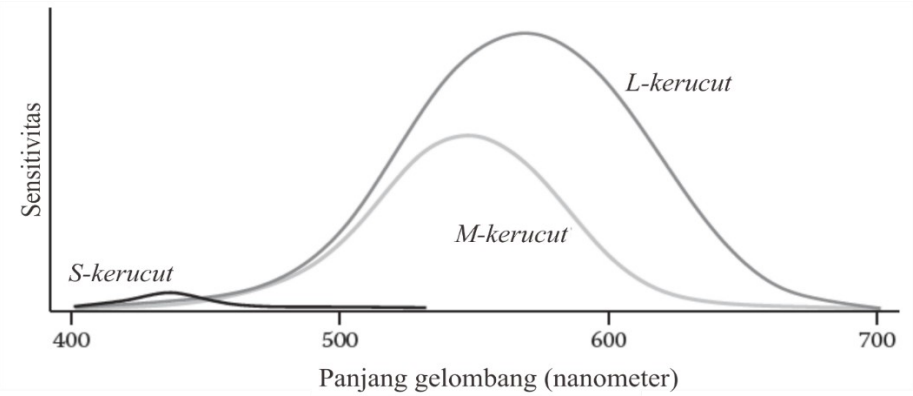
Bahkan akuntansi untuk fenomena seperti kekonstanan ringan, distribusi spasial yang sangat berbeda hampir selalu terlihat sangat berbeda. Lebih penting lagi mengingat bahwa tujuan dari sistem visual adalah untuk menghasilkan deskripsi stimulus distal yang diberikan stimulus proksimal, pola cahaya yang dirasakan paling tidak sesuai dengan pola kecerahan di atas permukaan di lingkungan.

Gambar 20.11. Spektrum yang terlihat. Panjang gelombang dalam nanometer.

Hal yang sama tidak berlaku untuk persepsi warna. Banyak distribusi spektral cahaya yang sangat berbeda dapat menghasilkan rasa warna tertentu. Sejalan dengan itu, perasaan bahwa permukaan adalah warna tertentu memberikan sedikit informasi langsung tentang distribusi spektral cahaya yang datang dari permukaan. Misalnya, distribusi spektral yang terdiri dari kombinasi cahaya pada panjang gelombang 700 nm dan 540 nm, dengan kekuatan relatif yang dipilih dengan tepat, akan terlihat tidak dapat dibedakan dari cahaya pada panjang gelombang tunggal 580 nm. (Warna yang secara persepsi tidak dapat dibedakan dengan komposisi spektral yang berbeda disebut sebagai metamer.) Jika kita melihat warna "kuning", kita tidak memiliki cara untuk mengetahui apakah itu dihasilkan oleh satu atau yang lain dari distribusi ini atau keluarga tak terbatas dari distribusi spektral lainnya. Untuk alasan ini, dalam konteks penglihatan, istilah warna mengacu pada kualitas persepsi murni, bukan properti fisik.

Ada dua kelas fotoreseptor di retina manusia. Kerucut terlibat dalam persepsi warna, sedangkan batang sensitif terhadap energi cahaya melintasi rentang yang terlihat dan tidak memberikan informasi tentang warna. Ada tiga jenis kerucut, masing-masing dengan sensitivitas spektral yang berbeda (Gambar 20.12). S-kerucut menanggapi panjang gelombang pendek dalam kisaran biru dari spektrum yang terlihat. Kerucut-M merespons panjang gelombang di wilayah tengah (kehijauan) dari spektrum yang terlihat. Kerucut-L merespons panjang gelombang yang agak lebih panjang yang menutupi bagian hijau dan merah dari spektrum yang terlihat.

Meskipun umum untuk menggambarkan tiga jenis kerucut sebagai merah, hijau, dan biru, ini bukanlah terminologi yang benar dan juga tidak secara akurat mencerminkan kepekaan kerucut yang ditunjukkan pada Gambar 20.12. Kerucut-L dan kerucut-M disetel secara luas, yang berarti bahwa mereka merespons berbagai frekuensi yang berbeda. Ada juga tumpang tindih substansial antara kurva sensitivitas dari tiga jenis kerucut. Secara bersama-sama, kedua sifat ini berarti bahwa tidak mungkin untuk merekonstruksi pendekatan ke distribusi spektral asli yang diberikan tanggapan dari tiga jenis kerucut. Ini berbeda dengan sampel spasial di sana (dan di kamera digital), di mana reseptor disetel secara sempit dalam sensitivitas spasialnya agar dapat mendeteksi detail halus dalam kontras lokal.



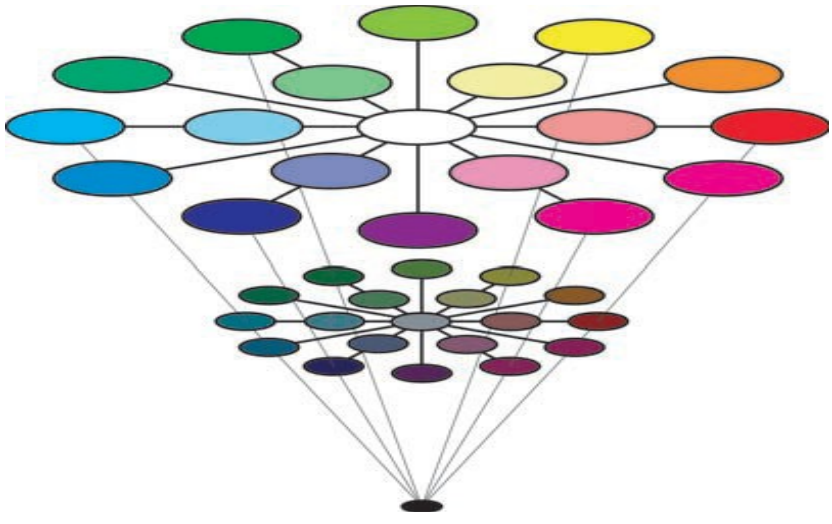
Gambar 20.12. Sensitivitas spektral kerucut pendek, sedang, dan panjang di retina manusia

Fakta bahwa hanya ada tiga jenis fotoreseptor peka warna di retina manusia sangat menyederhanakan tugas menampilkan warna pada monitor komputer dan tampilan grafis lainnya. Monitor komputer menampilkan warna sebagai kombinasi bobot dari tiga distribusi warna tetap. Paling sering, ketiga warna itu adalah merah yang berbeda, hijau yang berbeda, dan biru yang berbeda. Akibatnya, dalam grafis komputer, warna sering direpresentasikan dengan triple red-green-blue (RGB), mewakili intensitas warna primer merah, hijau, dan biru yang diperlukan untuk menampilkan warna tertentu. Tiga warna dasar sudah cukup untuk menampilkan warna yang paling terlihat, karena kombinasi bobot yang tepat dari tiga warna yang dipilih dengan tepat dapat menghasilkan metamer untuk warna yang terlihat ini.

Setidaknya ada dua masalah signifikan dengan representasi warna RGB. Yang pertama adalah bahwa monitor yang berbeda memiliki distribusi spektral yang berbeda untuk warna primer merah, hijau, dan biru. Akibatnya, rendisi warna yang benar secara persepsi melibatkan mapping ulang nilai RGB untuk setiap monitor. Ini, tentu saja, hanya mungkin jika nilai RGB asli memenuhi beberapa standar yang didefinisikan dengan baik, yang seringkali tidak demikian. (Lihat Bab 19 untuk informasi lebih lanjut tentang masalah ini.) Masalah kedua adalah bahwa nilai RGB tidak mendefinisikan warna tertentu dengan cara yang sesuai dengan persepsi subjektif. Ketika kita melihat warna “kuning”, kita tidak merasakan bahwa warna itu terdiri dari bagian yang sama dari cahaya merah dan hijau. Sebaliknya, itu terlihat seperti satu warna, dengan properti tambahan yang melibatkan kecerahan dan "jumlah" warna. Mewakili warna sebagai output kerucut S, kerucut M, dan kerucut L juga tidak membantu, karena kami memiliki beberapa rasa warna fenomenologis yang dicirikan oleh properti ini daripada yang kami lakukan seperti yang dicirikan oleh properti tampilan RGB.

Ada dua pendekatan berbeda untuk mengkarakterisasi warna dengan cara yang lebih mencerminkan persepsi manusia. Berbagai ruang warna CIE bertujuan untuk menjadi "perseptual seragam" sehingga besarnya perbedaan nilai yang diwakili dua warna sebanding dengan perbedaan warna yang dirasakan (Wyszecki & Stiles, 2000). Ini ternyata menjadi tujuan yang sulit untuk dicapai, dan ada beberapa modifikasi model CIE selama bertahun-tahun. Selain itu, sementara salah satu dimensi ruang warna CIE sesuai dengan kecerahan yang dirasakan, dua dimensi lainnya yang menentukan kromatisitas tidak memiliki makna intuitif.

Pendekatan kedua untuk mengkarakterisasi warna dengan cara yang lebih alami dimulai dengan pengamatan bahwa ada tiga sifat berbeda dan independen yang mendominasi rasa subjektif warna. Ringan, kecerahan permukaan yang tampak, telah dibahas. Saturasi mengacu pada kemurnian atau kejelasan warna. Warna dapat berkisar dari abu-abu tidak jenuh total hingga pastel jenuh sebagian hingga warna "murni" jenuh penuh. Sifat ketiga, rona, paling sesuai dengan arti informal kata "warna" dan dicirikan dengan cara yang mirip dengan warna dalam spektrum yang terlihat, mulai dari ungu tua hingga merah tua. Gambar 20.13 menunjukkan plot ruang warna hue-saturation-lightness (HSV). Karena hubungan antara kecerahan dan kecerahan keduanya kompleks dan tidak dipahami dengan baik, ruang warna HSV hampir selalu menggunakan kecerahan alih-alih mencoba memperkirakan kecerahan. Tidak seperti panjang gelombang dalam spektrum, bagaimanapun, rona biasanya diwakili dengan cara yang mencerminkan fakta bahwa ekstrem dari spektrum yang terlihat sebenarnya serupa dalam penampilan (Gambar 20.14). Transformasi sederhana ada antara representasi RGB dan HSV dari nilai warna tertentu. Akibatnya, sementara ruang warna HSV dimotivasi oleh pertimbangan perseptual, ia tidak mengandung lebih banyak informasi daripada representasi RGB.



Gambar 20.13. ruang warna HSV. Hue bervariasi di sekitar lingkaran, saturasi bervariasi dengan radius, dan nilainya bervariasi dengan ketinggian.

Gambar 20.14. Warna mana yang lebih dekat ke merah: hijau atau ungu?

Pendekatan hue-saturation-lightness untuk menggambarkan warna didasarkan pada distribusi spektral pada satu titik dan hanya mendekati respons persepsi terhadap distribusi spektral cahaya yang didistribusikan di ruang angkasa. Persepsi warna tunduk pada efek konstan dan kontras simultan yang serupa seperti kecerahan/kecerahan, keduanya tidak ditangkap dalam representasi RGB dan akibatnya tidak ditangkap dalam representasi HSV. Sebagai contoh keteguhan warna, lihatlah selembar kertas putih di dalam ruangan di bawah lampu pijar dan di luar ruangan di bawah sinar matahari langsung. Kertas akan terlihat "putih" dalam kedua kasus, meskipun cahaya pijar memiliki rona kuning yang jelas sehingga cahaya yang dipantulkan dari kertas juga akan memiliki rona kuning, sedangkan sinar matahari memiliki spektrum warna yang jauh lebih seragam.

Aspek lain dari persepsi warna yang tidak ditangkap oleh ruang warna CIE atau pengkodean HSV adalah kenyataan bahwa kita melihat sejumlah kecil warna yang berbeda ketika melihat spektrum kontinu cahaya tampak (Gambar 20.11) atau dalam pelangi yang terjadi secara alami. Bagi kebanyakan orang, spektrum tampak tampaknya dibagi menjadi empat menjadi enam warna yang berbeda: merah, kuning, hijau, dan biru, ditambah mungkin biru muda dan ungu. Mengingat warna non-spektral, hanya ada 11 istilah warna dasar yang umum digunakan dalam bahasa Inggris: red, green, blue, kuning, hitam, putih, abu-abu, oranye, ungu, coklat, dan merah muda. Pemisahan ruang distribusi spektral yang terus menerus secara intrinsik ke dalam seperangkat kategori persepsi yang relatif kecil yang terkait dengan istilah linguistik yang terdefinisi dengan baik tampaknya menjadi properti dasar persepsi, bukan hanya artefak budaya (Berlin & Kay, 1969). Sifat yang tepat dari proses ini, bagaimanapun, tidak dipahami dengan baik.

Dynmic Range/Rentang Dinamis

Penerangan alami bervariasi dalam intensitas lebih dari 6 orde besaran (Gambar 20.15). Sistem penglihatan manusia mampu beroperasi pada rentang tingkat kecerahan penuh ini. Namun, pada satu titik waktu, sistem visual hanya mampu mendeteksi variasi intensitas cahaya pada rentang yang jauh lebih kecil. Karena kecerahan rata-rata yang terkena sistem visual berubah dari waktu ke waktu, kisaran kecerahan yang dapat dibedakan berubah dengan cara yang sesuai. Efek ini paling jelas jika kita bergerak cepat dari area luar ruangan yang terang benderang ke ruangan yang sangat gelap. Pada awalnya, kita hanya bisa melihat sedikit. Namun, setelah beberapa saat, detail di ruangan itu mulai terlihat. Adaptasi gelap yang terjadi melibatkan sejumlah perubahan fisiologis pada mata. Dibutuhkan beberapa menit untuk adaptasi gelap yang signifikan terjadi dan 40 menit atau untuk menyelesaikan adaptasi gelap. Jika kita kemudian kembali ke cahaya terang, tidak hanya sulit untuk dilihat tetapi juga benar-benar menyakitkan. Adaptasi cahaya diperlukan sebelum dapat melihat dengan jelas lagi. Adaptasi cahaya terjadi jauh lebih cepat daripada adaptasi gelap, biasanya membutuhkan waktu kurang dari satu menit.

cahaya matahari langsung	10^5
pencahayaan indoor	10^2
cahaya bulan	10^{-1}
cahaya bintang	10^{-3}

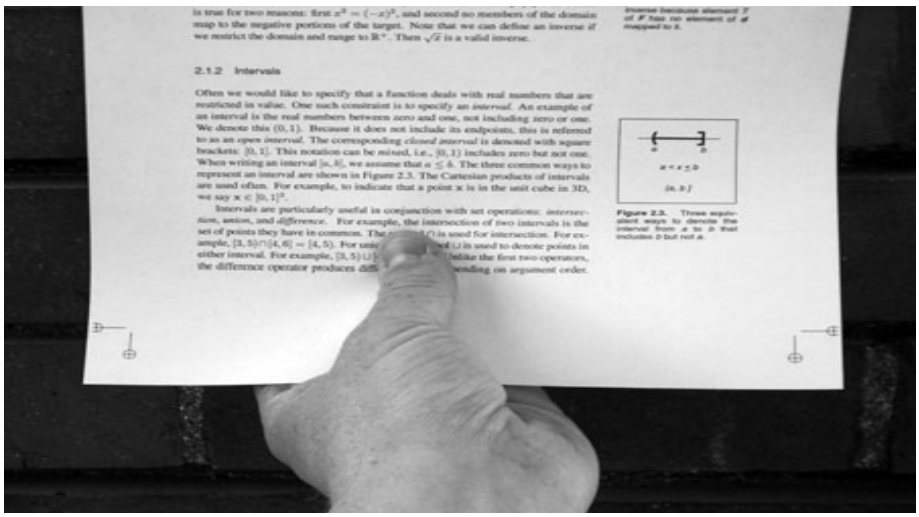
Gambar 20.15. Perkiraan tingkat luminansi permukaan putih di bawah berbagai jenis iluminasi dalam kandela per meter persegi (cd/m2). (Wandel, 1995).

Dua kelas fotoreseptor di retina manusia peka terhadap rentang kecerahan yang berbeda. Kerucut memberikan informasi visual atas sebagian besar dari apa yang kami anggap kondisi pencahayaan normal, mulai dari cahaya matahari yang terang hingga pencahayaan dalam ruangan yang redup. Batang hanya efektif pada tingkat cahaya yang sangat rendah. Penglihatan fotopik melibatkan cahaya terang di mana hanya sel kerucut yang efektif. Scotopicvision melibatkan cahaya gelap di mana hanya batang yang efektif. Ada kisaran intensitas di mana kerucut dan batang peka terhadap perubahan cahaya, yang disebut sebagai kondisi mesopik (lihat Bab 21).

20.5 BIDANG PANDANG DAN KETAJAMAN

Setiap mata dalam sistem visual manusia memiliki bidang pandang kira-kira 160◦ horizontal kali 135◦ vertikal. Dengan tampilan binokular, hanya ada sebagian tumpang tindih antara bidang pandang kedua mata. Ini menghasilkan lebih banyak bidang pandang (sekitar 200◦ horizontal dengan 135◦ vertikal), dengan wilayah tumpang tindih menjadi sekitar 120◦ horizontal dengan 135◦ vertikal.

Dengan penglihatan normal atau terkoreksi ke normal, kita biasanya memiliki pengalaman subjektif untuk dapat melihat detail yang relatif halus ke mana pun kita memandang. Namun, ini adalah ilusi. Hanya sebagian kecil dari bidang visual setiap mata yang benar-benar sensitif terhadap detail halus. Untuk melihat ini, pegang selembar kertas yang ditutupi dengan teks berukuran normal di sepanjang lengan, seperti yang ditunjukkan pada Gambar 20.16. Tutup satu mata dengan tangan yang tidak memegang kertas. Sambil menatap ibu jari Anda dan tidak menggerakkan mata Anda, perhatikan bahwa teks tepat di atas ibu jari Anda dapat dibaca sementara teks di kedua sisi tidak. Ketajaman penglihatan yang tinggi terbatas pada sudut visual yang sedikit lebih besar dari panjang ibu jari pada lengan Anda. Kami biasanya tidak memperhatikan hal ini karena mata biasanya sering bergerak, memungkinkan daerah yang berbeda dari bidang visual untuk dilihat pada resolusi tinggi. Sistem visual kemudian mengintegrasikan informasi ini dari waktu ke waktu untuk menghasilkan pengalaman subjektif dari seluruh bidang visual yang terlihat pada resolusi tinggi.



Gambar 20.16. Jika Anda memegang halaman teks sepanjang lengan dan menatap ibu jari Anda, hanya teks di dekat ibu jari Anda yang dapat dibaca.

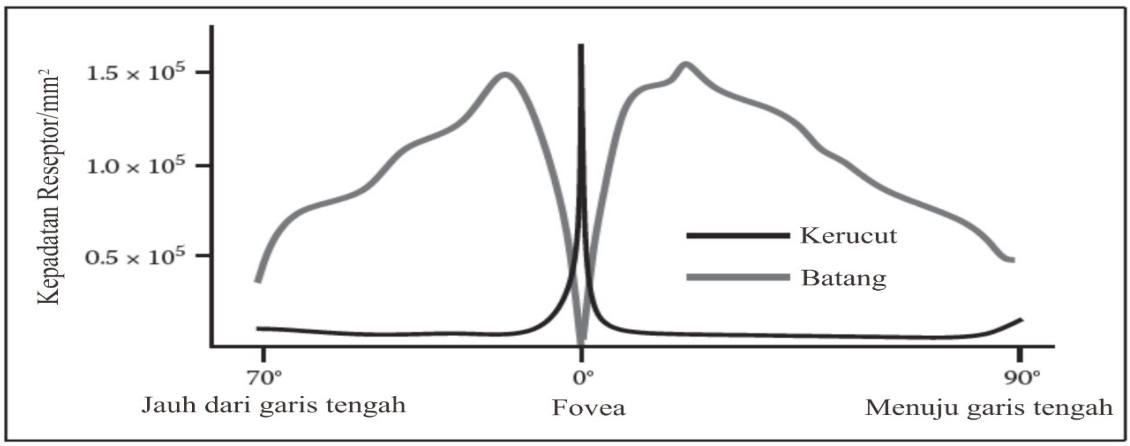
Tidak ada cukup lebar pita dalam korteks visual manusia untuk memproses informasi yang akan menghasilkan sampel intensitas gambar di seluruh retina. Kombinasi pengepakan fotoreseptor densitas variabel di retina dan mekanisme untuk gerakan mata yang cepat untuk menunjuk pada area yang diminati menyediakan cara untuk mengoptimalkan ketajaman dan bidang pandang secara bersamaan. Hewan lain telah mengembangkan cara yang berbeda untuk menyeimbangkan ketajaman dan bidang pandang yang tidak bergantung pada gerakan mata yang cepat. Beberapa hanya memiliki ketajaman penglihatan yang tinggi, tetapi terbatas pada bidang pandang yang sempit. Yang lain memiliki visi bidang pandang yang luas, tetapi kemampuan terbatas untuk melihat detail.

Gerak mata yang memfokuskan bidang minat di lingkungan pada fovea disebut saccades. Saccades terjadi dengan sangat cepat. Waktu dari stimulus pemicu hingga penyelesaian gerakan mata adalah 150–200 ms. Sebagian besar waktu ini dihabiskan dalam perencanaan sistem visi saccade. Gerak sebenarnya membutuhkan waktu rata-rata 20 ms atau lebih. Mata bergerak sangat cepat selama saccade, dengan kecepatan rotasi maksimum seringkali melebihi 500◦/detik. Di antara saccades, mata mengarah ke area yang diinginkan (fiksat), membutuhkan waktu sekitar 300 ms untuk memperoleh informasi visual detail yang halus. Mekanisme di mana beberapa fiksasi diintegrasikan untuk membentuk rasa subjektif keseluruhan dari detail halus pada bidang pandang yang luas tidak dipahami dengan baik.

Gambar 20.17 menunjukkan kepadatan pengepakan variabel kerucut dan batang di retina manusia. Kerucut, yang bertanggung jawab untuk penglihatan di bawah pencahayaan normal, terletak paling dekat di fovea retina (Gambar 20.17). Ketika mata terpaku pada suatu titik tertentu di lingkungan, bayangan titik tersebut jatuh pada fovea. Kepadatan pengepakan kerucut yang lebih tinggi di fovea menghasilkan frekuensi pengambilan sampel yang lebih tinggi dari cahaya yang dicitrakan (lihat Bab 9) dan karenanya detail yang lebih besar dalam pola sampel. Penglihatan foveal mencakup sekitar 1,7◦, yang merupakan sudut visual yang sama dengan lebar ibu jari Anda sepanjang lengan.

Sementara versi Gambar 20.17 muncul di sebagian besar teks pengantar tentang persepsi visual manusia, versi ini hanya memberikan sebagian penjelasan untuk keterbatasan neurofisiologis pada ketajaman visual. Output dari masing-masing batang dan kerucut dikumpulkan dalam berbagai cara oleh interkoneksi saraf di mata, sebelum informasi dikirimkan melalui saraf optik ke korteks visual.[6] Penyatuan ini menyaring sinyal yang diberikan oleh pola iluminasi insiden dengan cara yang berdampak penting pada pola cahaya yang dapat dideteksi. Khususnya, semakin jauh dari fovea, semakin besar area di mana kecerahan dirata-ratakan. Akibatnya, ketajaman spasial turun tajam dari fovea. Sebagian besar gambar yang menunjukkan kepadatan pengepakan batang dan kerucut menunjukkan lokasi titik buta retina, di mana berkas saraf yang membawa informasi optik dari mata ke otak melewati retina, dan tidak ada kepekaan terhadap cahaya. Pada umumnya, satu-satunya dampak praktis dari bintik buta pada persepsi dunia nyata adalah penggunaannya sebagai ilusi dalam teks persepsi pengantar, karena gerakan mata normal sebaliknya mengkompensasi hilangnya informasi sementara.

[6] Semua sel di saraf optik dan hampir semua sel di korteks visual memiliki bidang reseptif retina yang terkait. Pola cahaya yang mengenai retina di luar bidang reseptif sel tidak berpengaruh pada laju penembakan sel itu.



Gambar 20.17. Kepadatan batang dan kerucut di retina manusia (setelah Osterberg (1935)).

Seperti ditunjukkan pada Gambar 20.17, kerapatan pengepakan batang turun menjadi nol di tengah fovea. Jauh dari fovea, densitas batang pertama kali meningkat dan kemudian menurun. Salah satu akibatnya adalah tidak adanya penglihatan foveal saat penerangan sangat rendah. Kurangnya batang di fovea dapat ditunjukkan dengan mengamati langit malam di malam tanpa bulan, jauh dari lampu kota mana pun. Beberapa bintang akan sangat redup sehingga akan terlihat jika Anda melihat suatu titik di langit sedikit ke sisi bintang, tetapi mereka akan menghilang jika Anda melihatnya secara langsung. Hal ini terjadi karena ketika Anda melihat langsung ke fitur-fitur ini, gambar fitur hanya jatuh pada kerucut di retina, yang tidak cukup peka cahaya untuk mendeteksi fitur tersebut. Melihat sedikit ke samping menyebabkan gambar jatuh pada kerucut yang lebih peka cahaya. Penglihatan skotopik juga terbatas dalam ketajaman, sebagian karena kepadatan batang yang lebih rendah di sebagian besar retina dan sebagian karena pengumpulan sinyal yang lebih besar dari batang terjadi di retina untuk meningkatkan sensitivitas cahaya dari informasi visual yang diteruskan kembali ke otak.

Motion - Gerakan

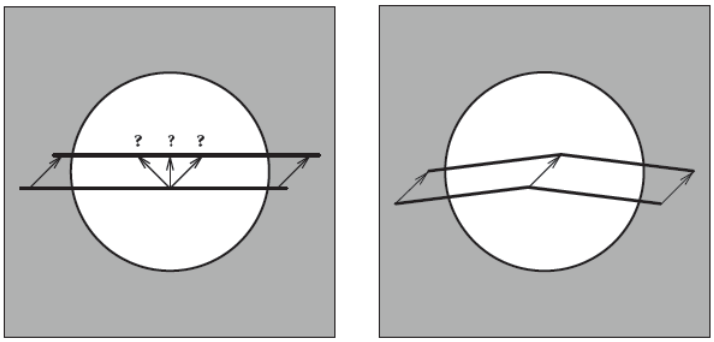
Saat membaca tentang persepsi visual dan melihat gambar statis pada halaman cetak, mudah untuk melupakan bahwa gerakan meresap dalam pengalaman visual kita. Pola cahaya yang jatuh di sana terus berubah karena gerakan mata dan tubuh serta gerakan benda-benda di dunia. Bagian ini mencakup kemampuan kita untuk mendeteksi gerakan visual. Bagian 20.3.4 menjelaskan bagaimana gerakan visual dapat digunakan untuk menentukan informasi geometris tentang lingkungan. Bagian 20.4.3 berhubungan dengan penggunaan gerakan untuk memandu gerakan kita melalui lingkungan.

Deteksi gerakan dalam pola tertentu dari cahaya yang jatuh pada retina adalah fungsi kompleks dari kecepatan, arah, ukuran pola, dan kontras. Masalah ini semakin rumit karena efek kontras simultan terjadi untuk persepsi gerakan dengan cara yang mirip dengan yang diamati pada persepsi kecerahan. Dalam kasus ekstrim dari satu pola kecil yang bergerak melawan latar belakang yang kontras dan homogen, gerakan yang dapat dipahami membutuhkan laju gerakan yang sesuai dengan sudut visual 0,2◦–0,3◦/detik. Gerakan dari pola yang sama bergerak melawan pola bertekstur dapat dideteksi pada kecepatan sekitar sepersepuluh ini.

Dengan kepekaan terhadap gerakan retina ini, dikombinasikan dengan frekuensi dan kecepatan gerakan mata saccadic, mengejutkan bahwa dunia biasanya tampak stabil dan tidak bergerak ketika kita melihatnya. Sistem visi menyelesaikan ini dalam tiga cara. Sensitivitas kontras berkurang selama saccades, mengurangi efek visual yang dihasilkan oleh perubahan posisi mata yang cepat ini. Di antara saccades, berbagai mekanisme canggih dan kompleks menyesuaikan posisi mata untuk mengimbangi gerakan kepala dan tubuh serta gerakan objek yang menarik di dunia. Akhirnya, sistem visual mengeksploitasi informasi tentang posisi mata

untuk merakit sebuah mosaik dari petak-petak kecil gambar resolusi tinggi dari beberapa fiksasi menjadi satu kesatuan yang stabil.

Pergerakan garis lurus dan tepi menjadi ambigu jika tidak ada titik akhir atau sudut yang terlihat, fenomena yang tidak disebut sebagai masalah bukaan (Gambar 20.18). Apertur bermasalah karena komponen gerak yang sejajar dengan garis atau tepi tidak menghasilkan perubahan visual apa pun. Geometri dunia nyata cukup kompleks sehingga jarang menyebabkan kesulitan dalam praktik, kecuali untuk ilusi yang disengaja seperti tiang tukang cukur. Geometri dan tekstur yang disederhanakan ditemukan di beberapa rendering grafis komputer, bagaimanapun, memiliki potensi untuk memperkenalkan ketidakakuratan dalam gerakan yang dirasakan.

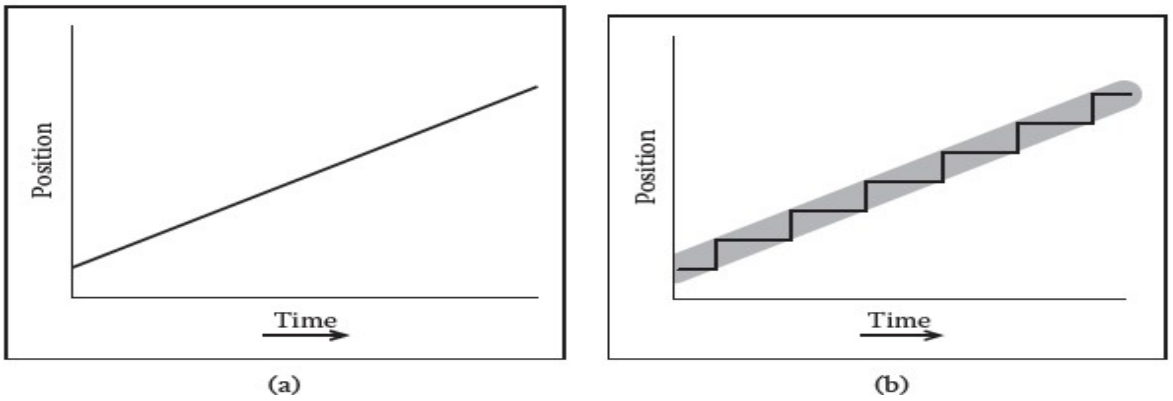


Gambar 20.18. Masalah bukaan: (a) Jika garis lurus atau tepi bergerak sedemikian rupa sehingga titik ujungnya tersembunyi, informasi visual tidak cukup untuk menentukan gerakan garis yang sebenarnya. (b) Gerak 2D dari sebuah garis tidak ambigu jika ada sudut atau tanda pembeda lainnya pada garis tersebut.

Grafis komputer real-time, film, dan video tidak akan mungkin terjadi tanpa fenomena persepsi penting: gerakan terputus-putus, di mana serangkaian gambar statis terlihat untuk interval waktu yang berbeda dan kemudian bergerak dengan interval diskrit dalam ruang, hampir tidak dapat dibedakan. dari gerakan terus menerus. Efeknya disebut gerakan semu untuk menyoroti bahwa munculnya gerakan terus menerus adalah ilusi.

Gambar 20.19 mengilustrasikan perbedaan antara gerakan kontinu, yang merupakan ciri khas dunia nyata, dan gerakan semu, yang dihasilkan oleh hampir semua perangkat tampilan gambar dinamis. Gerak yang diplot pada Gambar 20.19 (b) terdiri dari gerakan rata-rata yang sebanding dengan yang ditunjukkan pada Gambar 20.19 (a), dimodulasi oleh frekuensi ruang-waktu tinggi yang menyebabkan pergantian antara pola stasioner dan pola yang bergerak terputus-putus ke lokasi baru. Persepsi nyata dari gerakan terus menerus terjadi karena sistem visual tidak peka terhadap komponen frekuensi tinggi dari gerakan.

Rasa yang menarik dari gerakan nyata terjadi ketika kecepatan munculnya gambar individu di atas sekitar 10 Hz, selama perubahan posisi antara gambar yang berurutan tidak terlalu besar. Namun, kecepatan ini tidak cukup cepat untuk menghasilkan sensasi gerak berkelanjutan yang memuaskan bagi sebagian besar perangkat tampilan gambar. Hampir semua perangkat tersebut memperkenalkan variasi kecerahan saat satu gambar dialihkan ke gambar berikutnya. Dalam kondisi pencahayaan yang baik, sistem visual manusia peka terhadap kecerahan yang bervariasi ini untuk tingkat variasi hingga sekitar 80 Hz. Dalam cahaya yang lebih rendah, kemampuan deteksi hadir hingga sekitar 40 Hz. Ketika tingkat kecerahan bolak-balik cukup tinggi, fusi yang lebih cepat terjadi dan variasi tidak lagi terlihat.



Gambar 20.19. (a) Gerak terus menerus. (b) Gerak terputus-putus dengan kecepatan rata-rata yang sama. Dalam beberapa keadaan, persepsi kedua pola gerak ini mungkin serupa.

Untuk menghasilkan sensasi gerak visual yang menarik, tampilan gambar harus memenuhi dua batasan terpisah:

- gambar harus diperbarui pada kecepatan $\geq$10 Hz;
- setiap kedipan yang terjadi dalam proses memperbarui gambar harus terjadi pada kecepatan $\geq$60–80Hz.

Salah satu solusinya adalah mengharuskan kecepatan pembaruan gambar lebih besar dari atau sama dengan 60–80Hz. Namun, dalam banyak situasi, ini sama sekali tidak mungkin. Untuk tampilan grafis komputer, waktu komputasi bingkai seringkali jauh lebih besar dari 12–15 mdtk. Bandwidth transmisi dan batasan teknologi monitor lama membatasi siaran televisi normal hingga 25–30 gambar per detik. (Beberapa format HDTV beroperasi pada 60 gambar/dtk.) Film memperbarui gambar pada 24 bingkai/detik karena persyaratan waktu pencahayaan dan kesulitan mekanis dari film yang bergerak secara fisik lebih cepat dari itu.

Teknologi tampilan yang berbeda memecahkan masalah ini dengan cara yang berbeda. Tampilan komputer menyegarkan gambar yang ditampilkan pada 70–80 Hz, terlepas dari seberapa sering kontensoftheimage berubah. Istilah kecepatan bingkai tidak jelas untuk tampilan seperti itu, karena dua nilai diperlukan untuk mengkarakterisasi tampilan ini: kecepatan refresh, yang menunjukkan kecepatan gambar ditampilkan ulang dan kecepatan pembaruan bingkai, yang menunjukkan kecepatan gambar baru dihasilkan untuk tampilan. Televisi siaran nonHDTV standar menggunakan kecepatan refresh 60 Hz (NTSC, digunakan di Amerika Utara dan beberapa lokasi lain) atau 50 Hz (PAL, digunakan di sebagian besar belahan dunia lainnya). Kecepatan pembaruan bingkai adalah setengah dari kecepatan refresh. Alih-alih menampilkan setiap gambar baru dua kali, tampilan tersebut saling terkait dengan membagi garis horizontal gambar bolak-balik menjadi bidang genap dan ganjil dan tampilan bidang genap dan ganjil ini bergantian. Kedipan dihindari dalam film dengan menggunakan rana mekanis untuk mengedipkan setiap bingkai film tiga kali sebelum pindah ke bingkai berikutnya, menghasilkan kecepatan refresh 72 Hz sambil mempertahankan kecepatan pembaruan bingkai 24 Hz.

Penggunaan gerakan semu untuk mensimulasikan gerakan kontinu terkadang menghasilkan artefak yang tidak diinginkan. Yang paling terkenal dari ini adalah ilusi roda gerobak di mana jari-jari roda yang berputar tampak berputar ke arah yang berlawanan dari apa yang diharapkan mengingat gerakan translasi roda. Ilusi roda gerobak adalah contoh aliasing temporal. Jari-jari, atau pola periodik spasial lainnya pada piringan yang berputar, menghasilkan sinyal periodik temporal untuk melihat lokasi yang tetap terhadap pusat roda atau piringan. Tingkat pembaruan bingkai tetap memiliki efek pengambilan sampel sinyal periodik temporal ini dalam waktu. Jika frekuensi temporal dari pola sampel terlalu tinggi, hasil undersampling di alias, frekuensi temporal yang lebih rendah muncul saat gambar ditampilkan. Dalam beberapa keadaan, distorsi frekuensi temporal ini menyebabkan distorsi

spasial di mana roda tampak bergerak mundur. Ilusi roda gerobak lebih mungkin terjadi dengan film dibandingkan dengan video, karena tingkat pengambilan sampel temporal lebih rendah.

Masalah juga dapat terjadi ketika gambar gerak semu diubah dari satu media ke media lainnya. Ini menjadi perhatian khusus ketika film 24 Hz ditransfer ke video. Tidak hanya format non-interlaced yang perlu diterjemahkan ke format interlaced, tetapi tidak ada cara langsung untuk berpindah dari 24 frame per detik ke 50 atau 60 bidang per detik. Beberapa perangkat tampilan kelas atas memiliki kemampuan untuk mengkompensasi sebagian artefak yang diperkenalkan saat film dikonversi ke video.

20.6 VISI SPASIAL

Salah satu operasi kritis yang dilakukan oleh sistem visual adalah estimasi sifat geometris dari lingkungan yang terlihat, karena ini adalah pusat untuk menentukan informasi tentang objek, lokasi, dan peristiwa. Penglihatan kadang-kadang telah dijelaskan dalam optik terbalik, untuk menekankan bahwa salah satu fungsi sistem visual adalah membalikkan proses pembentukan gambar untuk menentukan geometri, bahan, dan pencahayaan di dunia yang menghasilkan pola tertentu pada cahaya di retina. Masalah utama untuk sistem penglihatan adalah bahwa sifat-sifat lingkungan yang terlihat dikacaukan dalam pola cahaya yang dicitrakan pada retina. Kecerahan adalah fungsi dari iluminasi dan pantulan, dan dapat bergantung pada sifat lingkungan di seluruh wilayah ruang yang luas karena kompleksitas transportasi cahaya. Lokasi gambar dari lokasi lingkungan yang diproyeksikan paling baik dapat digunakan untuk membatasi posisi lokasi tersebut menjadi setengah garis. Akibatnya, sangat jarang mungkin untuk secara unik menentukan sifat dunia yang menghasilkan pola bayangan tertentu dari cahaya.

Menentukan tata letak permukaan—lokasi dan orientasi permukaan yang terlihat di lingkungan—dianggap sebagai langkah kunci dalam penglihatan manusia. Sebagian besar diskusi tentang bagaimana sistem penglihatan mengekstrak informasi tentang tata letak permukaan dari pola cahaya yang diterimanya membagi masalah menjadi satu set isyarat visual, dengan setiap isyarat menggambarkan pola visual tertentu yang dapat digunakan untuk menyimpulkan sifat tata letak permukaan bersama dengan aturan inferensi yang diperlukan. Karena tata letak permukaan jarang dapat ditentukan secara akurat dan jelas dari penglihatan saja, proses menyimpulkan tata letak permukaan biasanya memerlukan informasi nonvisual tambahan. Ini bisa datang dari indra atau asumsi lain tentang apa yang mungkin terjadi di dunia nyata.

Isyarat visual biasanya dikategorikan ke dalam empat kategori. Isyarat motorik okuler melibatkan informasi tentang posisi dan fokus mata. Isyarat disparitas melibatkan informasi yang diambil dari melihat titik permukaan yang sama dengan dua mata, di luar itu tersedia hanya dari posisi mata. Isyarat gerak memberikan informasi tentang dunia yang muncul baik dari pergerakan pengamat atau pergerakan objek. Isyarat bergambar dihasilkan dari proses memproyeksikan bentuk permukaan 3D ke pola cahaya 2D yang jatuh pada retina. Bagian ini berhubungan dengan isyarat visual yang relevan dengan ekstraksi informasi geometris tentang titik individu pada permukaan. Ekstraksi lebih umum informasi lokasi dan bentuk tercakup dalam Bagian 20.4.

Kerangka Acuan dan Skala Pengukuran

Deskripsi lokasi dan orientasi titik pada permukaan yang terlihat harus dilakukan dalam konteks kerangka acuan tertentu yang menentukan asal, orientasi, dan skala sistem koordinat yang digunakan dalam merepresentasikan informasi geometrik. Sistem penglihatan manusia menggunakan banyak kerangka acuan, sebagian karena jenis informasi yang berbeda tersedia dari isyarat visual yang berbeda dan sebagian karena tujuan yang berbeda dari

informasi tersebut (Klatzky, 1998). Representasi egosentris didefinisikan sehubungan dengan tubuh pemirsa. Mereka dapat dibagi lagi menjadi sistem koordinat yang dipasang pada mata, kepala, atau tubuh. Representasi allocentric, juga disebut representasi eksosentris, didefinisikan sehubungan dengan sesuatu di luar pemirsa. Kerangka acuan alosentris dapat bersifat lokal untuk beberapa konfigurasi objek di lingkungan atau dapat didefinisikan secara global dalam hal lokasi, gravitasi, atau properti geografis yang berbeda.

Jarak dari penonton ke lokasi tertentu yang terlihat di lingkungan, dinyatakan dalam representasi egosentris, sering disebut sebagai kedalaman dalam literatur persepsi. Orientasi permukaan dapat direpresentasikan dalam koordinat egosentris atau alosentris. Dalam representasi orientasi egosentris, istilah miring digunakan untuk merujuk pada sudut antara garis pandang ke titik dan permukaan normal pada titik tersebut, sedangkan istilah kemiringan mengacu pada orientasi proyeksi permukaan normal pada bidang yang tegak lurus dengan bidang. pandangan.

Jarak dan orientasi dapat dinyatakan dalam berbagai skala pengukuran. Deskripsi mutlak ditentukan menggunakan standar yang bukan merupakan bagian dari informasi yang dirasakan itu sendiri. Ini dapat berupa standar yang ditentukan secara budaya (misalnya, meter), atau standar yang relatif terhadap tubuh penonton (misalnya, tinggi mata, lebar bahu). Deskripsi relatif menghubungkan satu properti geometris yang dirasakan dengan yang lain (misalnya, titik a dua kali lebih jauh dari titik b). Deskripsi ordinal adalah kasus khusus ukuran relatif di mana hanya tanda, tetapi bukan besaran, dari hubungan yang diwakili. Tabel 20.1 memberikan daftar isyarat visual yang paling umum dipertimbangkan, bersama dengan karakterisasi dari jenis informasi yang berpotensi dapat mereka berikan.

Isyarat motorik okuler

Informasi motorik okuler tentang kedalaman dihasilkan langsung dari kontrol otot mata. Ada dua jenis informasi ocularmotor yang berbeda. Akomodasi adalah proses dimana mata secara optik memfokuskan pada jarak tertentu. Konvergensi (sering disebut sebagai vergensi) adalah proses di mana kedua mata diarahkan ke titik yang sama dalam ruang tiga dimensi. Akomodasi dan konvergensi memiliki potensi untuk memberikan informasi absolut tentang kedalaman.

Secara fisiologis, pemfokusan pada mata manusia dilakukan dengan mendistorsi bentuk lensa di bagian depan mata. Sistem penglihatan dapat menyimpulkan kedalaman dari jumlah distorsi ini. Akomodasi adalah isyarat yang relatif lemah untuk jarak dan tidak efektif lebih dari sekitar 2 m. Kebanyakan orang mengalami kesulitan yang semakin meningkat dalam memfokuskan diri pada rentang jarak yang jauh saat mereka mencapai usia sekitar 45 tahun. Bagi mereka, akomodasi menjadi kurang efektif.

Mereka yang tidak akrab dengan spesifikasi persepsi visual terkadang mengacaukan estimasi kedalaman dari akomodasi dengan informasi kedalaman yang timbul dari kekaburan yang terkait dengan kedalaman bidang yang terbatas di mata. Isyarat kedalaman akomodasi memberikan informasi tentang jarak ke bagian bidang visual yang menjadi fokus. Itu tidak tergantung pada sejauh mana bagian lain dari bidang visual tidak fokus, selain itu blur digunakan oleh sistem visual untuk menyesuaikan fokus. Kedalaman lapangan tampaknya memberikan tingkat informasi kedalaman ordinal (Gambar 20.20), meskipun efek ini hanya menerima penyelidikan terbatas.

Jika dua mata terpaku pada titik yang sama dalam ruang, trigonometri dapat digunakan untuk menentukan jarak dari pengamat ke lokasi yang dilihat (Gambar 20.21). Untuk kasus yang paling sederhana, di mana tempat tujuan berada tepat di depan pemirsa,

Persamaan 20.5

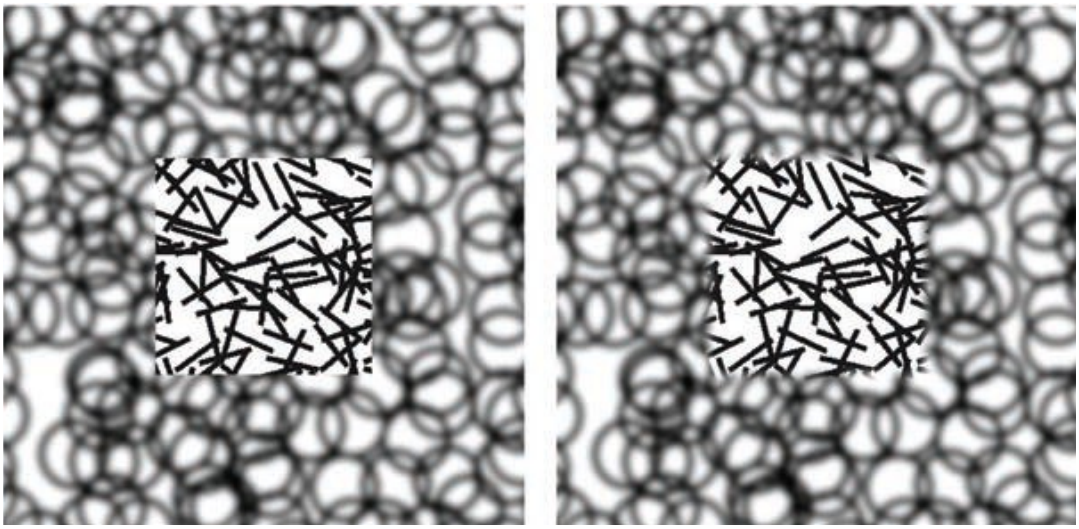
$$z = \frac{ipd/2}{\tan\theta}$$

di mana z adalah jarak ke suatu titik di dunia, ipd adalah jarak antar pupil yang menunjukkan jarak antara mata, dan θ adalah sudut vergensi yang menunjukkan orientasi mata relatif terhadap lurus ke depan. Untuk kecil, yang merupakan kasus untuk konfigurasi geometris mata manusia, tan θ ≈ θ ketika dinyatakan dalam radian. Dengan demikian, perbedaan sudut vergensi menentukan perbedaan kedalaman dengan hubungan berikut:

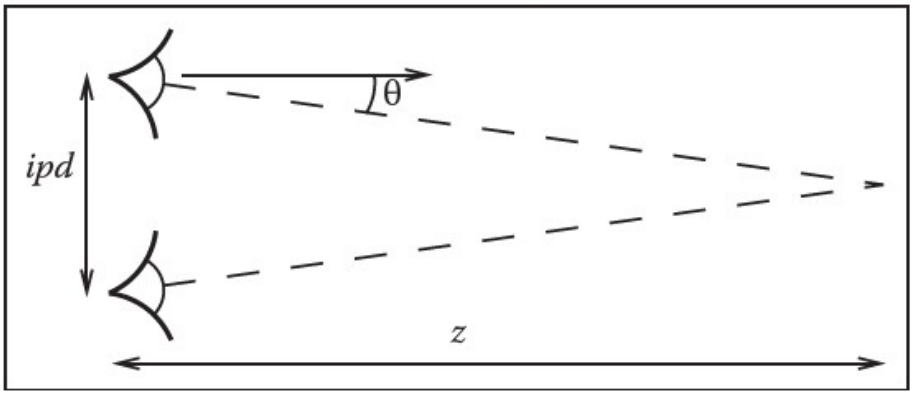
Persamaan 20.6

$$\Delta\theta \approx \frac{ipd}{2} \cdot \frac{1}{\Delta_z}$$

Saat θ → 0 dalam langkah seragam, Δz semakin besar. Ini berarti bahwa penglihatan stereo kurang sensitif terhadap perubahan kedalaman saat kedalaman keseluruhan meningkat. Konvergensi sebenarnya hanya memberikan informasi tentang kedalaman absolut untuk jarak beberapa meter. Di luar itu, perubahan jarak menghasilkan perubahan vergence yang terlalu kecil untuk berguna.



Gambar 20.20. Apakah bujur sangkar tengah muncul di depan pola lingkaran atau terlihat muncul melalui lubang bujur sangkar dalam pola lingkaran? Satu-satunya perbedaan dalam dua gambar adalah ketajaman tepi antara pola garis dan lingkaran (Marshall, Burbeck, Arely, Rolland, dan Martin (1999), digunakan dengan izin).



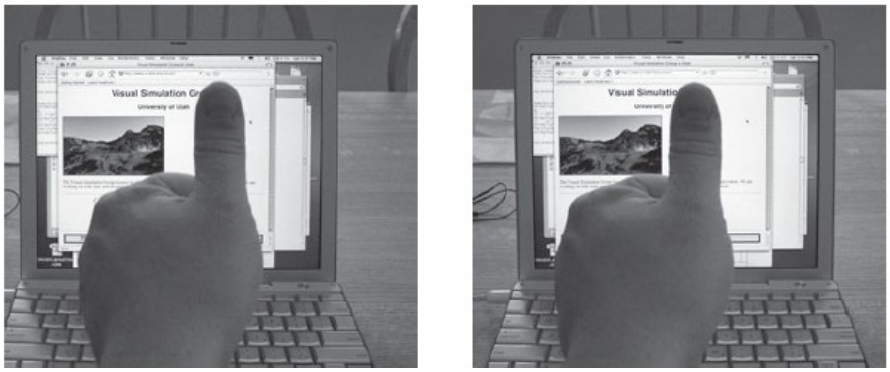
Gambar 20.21. Vergensi kedua mata memberikan informasi tentang jarak ke titik di mana mata terpaku.

Ada interaksi antara akomodasi dan konvergensi dalam sistem visual manusia: akomodasi digunakan untuk membantu menentukan sudut vergensi yang tepat, sedangkan

vergensi digunakan untuk membantu mengatur jarak fokus. Biasanya, ini membantu sistem visual ketika ada ketidakpastian dalam pengaturan akomodasi atau vergensi. Namun, tampilan komputer stereografis memutuskan hubungan antara fokus dan konvergensi yang terjadi di dunia nyata, yang menyebabkan sejumlah kesulitan persepsi (Wann, Rushton, & Mon-Williams, 1995).

20.7 DISPARITAS BINOKULAR

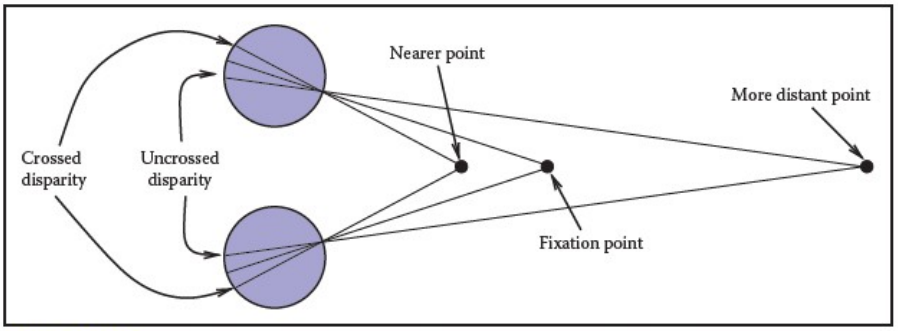
Sudut kemiringan mata, ketika difiksasi pada titik bersama di ruang angkasa, hanyalah salah satu cara agar sistem visual dapat menentukan kedalaman dari stereo binokular. Mekanisme kedua melibatkan perbandingan gambar retina di kedua mata dan tidak memerlukan informasi tentang ke mana mata diarahkan. Contoh sederhana menunjukkan efeknya. Pegang lengan Anda lurus di depan Anda, dengan ibu jari mengarah ke atas. Tatap ibu jari Anda lalu tutup satu mata. Sekarang, secara bersamaan buka mata tertutup dan tutup mata terbuka. Jempol Anda akan tampak kurang lebih tidak bergerak, sedangkan permukaan yang lebih jauh terlihat di belakang ibu jari Anda akan tampak bergerak dari sisi ke sisi (Gambar 20.22). Perubahan posisi retinal titik-titik dalam scene antara mata kiri dan kanan disebut disparitas.



Gambar 20.22. Disparitas binokular. Scene dari mata kiri dan kanan menunjukkan offset untuk titik permukaan pada kedalaman yang berbeda dari titik fiksasi.

Isyarat disparitas binokular mensyaratkan bahwa sistem penglihatan dapat mencocokkan gambar titik-titik di dunia di satu mata dengan lokasi yang dicitrakan dari titik-titik itu di mata yang lain, sebuah proses yang disebut sebagai masalah korespondensi. Ini adalah proses yang relatif rumit dan hanya dipahami sebagian. Setelah korespondensi telah ditetapkan, posisi relatif di mana titik-titik tertentu di dunia diproyeksikan ke retina kiri dan kanan menunjukkan apakah titik-titik tersebut lebih dekat atau lebih jauh daripada titik fiksasi. Disparitas silang terjadi ketika titik-titik yang bersesuaian dipindahkan ke luar relatif terhadap fovea dan menunjukkan bahwa titik permukaan lebih dekat daripada titik fiksasi. Disparitas tidak bersilangan terjadi ketika titik-titik yang bersesuaian dipindahkan ke dalam relatif terhadap fovea dan menunjukkan bahwa titik permukaan lebih jauh dari titik fiksasi (Gambar 20.23).[7] Disparitas binokular adalah petunjuk kedalaman relatif, tetapi dapat memberikan informasi tentang kedalaman absolut ketika diskalakan oleh konvergensi. Persamaan (20.5) berlaku untuk disparitas binokular serta konvergensi binokular. Seperti halnya konvergensi, sensitivitas disparitas binokular terhadap perubahan kedalaman menurun seiring dengan kedalaman.

[7] Secara teknis, disparitas silang dan tidak bersilangan menunjukkan bahwa titik permukaan yang menghasilkan disparitas lebih dekat atau lebih jauh dari horopter. Horopter bukanlah jarak tetap dari mata melainkan permukaan melengkung yang melewati titik fiksasi.



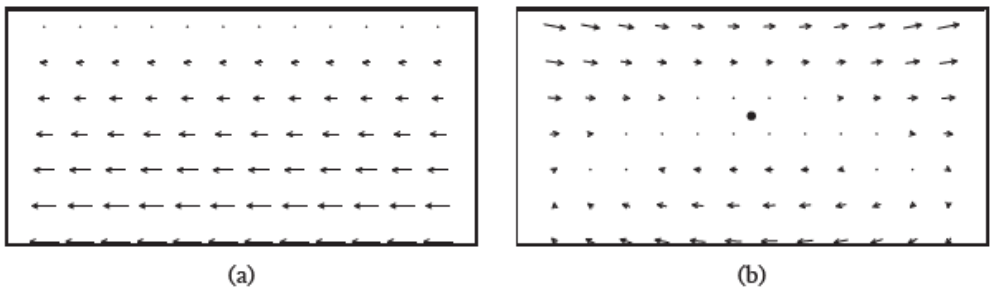
Gambar 20.23. Dekat garis pandang, titik permukaan lebih dekat dari titik fiksasi menghasilkan disparitas dalam arah yang berlawanan dari yang terkait dengan titik permukaan lebih jauh dari titik fiksasi.

20.8 ISYARAT GERAK

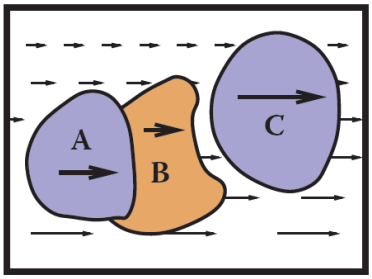
Gerak relatif antara mata dan permukaan yang terlihat akan menghasilkan perubahan pada gambar permukaan tersebut di sana. Gerak relatif tiga dimensi antara mata dan titik permukaan menghasilkan gerak proyeksi dua dimensi dari titik permukaan pada retina. Gerakan retina ini diberi nama aliran optik. Aliran optik berfungsi sebagai dasar untuk beberapa jenis isyarat kedalaman. Selain itu, aliran optik dapat digunakan untuk menentukan informasi tentang bagaimana seseorang bergerak di dunia dan apakah tabrakan akan terjadi atau tidak (Bagian 20.4.3).

Jika seseorang bergerak ke samping sambil terus memperbaiki beberapa titik permukaan, maka aliran optik memberikan informasi tentang kedalaman yang mirip dengan disparitas stereo. Ini disebut sebagai paralaks gerak. Untuk titik permukaan lain yang memproyeksikan ke lokasi retina dekat titik fiksasi, aliran optik nol menunjukkan kedalaman yang setara dengan titik fiksasi; aliran dalam arah yang berlawanan dengan translasi kepala menunjukkan titik-titik yang lebih dekat, setara dengan disparitas silang; dan aliran dalam arah yang sama dengan translasi kepala menunjukkan titik-titik yang lebih jauh, setara dengan disparitas yang tidak bersilangan (Gambar 20.24). Paralaks gerak adalah isyarat yang kuat untuk kedalaman relatif. Pada prinsipnya, paralaks gerak dapat memberikan informasi kedalaman mutlak jika sistem visual memiliki akses ke informasi tentang kecepatan gerak kepala. Dalam praktiknya, paralaks gerak tampak paling baik sebagai isyarat lemah untuk kedalaman absolut.

Selain informasi kedalaman egosentris akibat paralaks gerak, gerak visual juga dapat memberikan informasi tentang bentuk tiga dimensi benda yang bergerak relatif terhadap pemirsa. Dalam literatur persepsi, ini dikenal sebagai efek kedalaman kinetik. Dalam visi komputer, ini disebut sebagai struktur-dari gerak. Efek kedalaman kinetik mengandaikan bahwa salah satu komponen gerak benda adalah rotasi dalam, artinya terdapat komponen rotasi di sekitar sumbu yang tegak lurus terhadap garis pandang.



Gambar 20.24. (a) Gerak paralaks yang dihasilkan oleh gerakan menyamping ke kanan sambil melihat bidang tanah yang diperpanjang. (b) Gerakan yang sama, dengan pelacakan mata dari titik fiksasi.



Gambar 20.25. Diskontinuitas dalam batas permukaan sinyal aliran optik. Dalam banyak kasus, tanda perubahan kedalaman (yaitu, kedalaman ordinal) dapat ditentukan.

Aliran optik juga dapat memberikan informasi tentang bentuk dan lokasi batas permukaan, seperti yang ditunjukkan pada Gambar 20.25. Diskontinuitas spasial dalam aliran optik hampir selalu berhubungan dengan diskontinuitas kedalaman atau hasil dari objek yang bergerak secara independen. Perbandingan sederhana dari magnitudo aliran optik tidak cukup untuk menentukan tanda perubahan kedalaman, kecuali dalam kasus khusus penampil yang bergerak melalui dunia yang statis. Bahkan ketika ada objek yang bergerak secara independen, bagaimanapun, tanda perubahan kedalaman melintasi batas permukaan seringkali dapat ditentukan dengan cara lain. Gerak sering mengubah bagian permukaan yang lebih jauh yang terlihat pada batas permukaan. Munculnya (pertambahan) atau hilangnya (penghapusan) tekstur permukaan terjadi karena semakin dekat, permukaan oklusi semakin membuka atau menutupi bagian dari permukaan oklusi yang lebih jauh. Perbandingan gerakan tekstur permukaan ke kedua sisi batas juga dapat digunakan untuk menyimpulkan kedalaman ordinal, bahkan tanpa adanya pertambahan atau penghapusan tekstur. Diskontinuitas dalam aliran optik dan pertambahan/penghapusan tekstur permukaan disebut sebagai isyarat oklusi dinamis dan merupakan sumber informasi visual yang kuat tentang struktur spasial lingkungan.

Kecepatan yang ditempuh pemirsa relatif terhadap titik-titik di dunia tidak dapat ditentukan dari gerakan visual saja (lihat Bagian 20.4.3). Meskipun keterbatasan ini, adalah mungkin untuk menggunakan informasi visual untuk menentukan waktu yang diperlukan untuk mencapai titik yang terlihat di dunia, bahkan ketika kecepatan tidak dapat ditentukan. Ketika kecepatan konstan, waktu-untuk-kontak (sering disebut sebagai waktu-tabrakan) diberikan oleh ukuran retina dari suatu entitas ke arah mana pengamat bergerak, dibagi dengan tingkat di mana ukuran gambar meningkat.[8] Dalam literatur visi biologis, ini sering disebut fungsi (Lee & Reddish, 1981). Jika informasi jarak ke struktur di dunia yang berdasarkan perkiraan waktu tumbukan tersedia, maka ini dapat digunakan untuk menentukan kecepatan.

20.9 ISYARAT BERGAMBAR

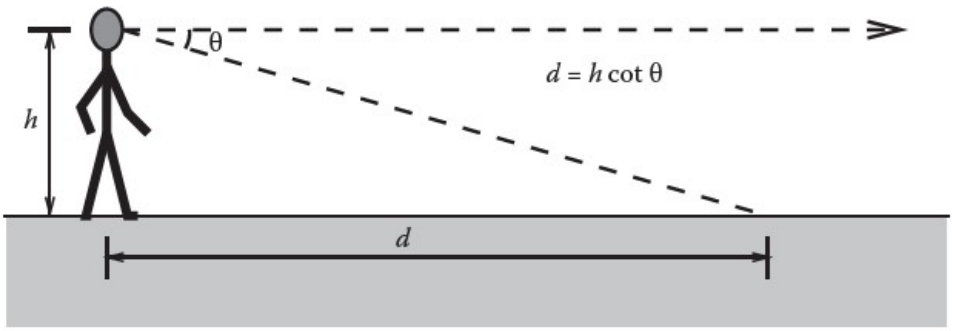
Sebuah gambar dapat berisi banyak informasi tentang struktur spasial dunia dari mana ia muncul, bahkan tanpa adanya stereo binokular atau gerakan. Sebagai bukti untuk ini, perhatikan bahwa dunia masih tampak tiga dimensi bahkan jika kita menutup satu mata, memegang kepala kita diam, dan tidak ada yang bergerak di lingkungan. (Seperti yang dibahas dalam Bagian 20.5, situasinya lebih rumit dalam kasus foto dan gambar lain yang ditampilkan.) Ada tiga kelas isyarat kedalaman bergambar tersebut. Yang paling terkenal dari ini melibatkan perspektif linier. Ada juga sejumlah isyarat oklusi yang memberikan informasi tentang kedalaman ordinal bahkan tanpa adanya perspektif. Akhirnya, isyarat iluminasi yang melibatkan bayangan, bayangan dan interrefleksi, dan perspektif udara juga memberikan informasi visual tentang tata ruang.

[8] Istilah time-to-collision dan time-to-contact menyesatkan, karena kontak hanya akan terjadi jika lintasan pemirsa benar-benar melewati atau mendekati entitas yang dilihat.

Gambar 20.26. Efek perspektif linier klasik mencakup ukuran objek yang diskalakan berdasarkan jarak, konvergensi garis paralel, bidang dasar yang meluas ke cakrawala yang terlihat, dan posisi pada bidang dasar relatif terhadap cakrawala. Gambar milik Sam Pullara.

Istilah perspektif linier sering digunakan untuk merujuk pada properti gambar yang melibatkan ukuran objek dalam gambar yang diskalakan berdasarkan jarak, konvergensi garis paralel, bidang dasar yang memanjang hingga cakrawala yang terlihat, dan hubungan antara jarak ke objek di bidang dasar dan lokasi bayangan objek tersebut relatif terhadap cakrawala (Gambar 20.26). Lebih formal, isyarat perspektif linier adalah isyarat visual yang mengeksploitasi fakta bahwa di bawah proyeksi perspektif, lokasi gambar ke mana titik-titik di dunia diproyeksikan diskalakan oleh 1/z, di mana z adalah jarak dari titik proyeksi ke titik di lingkungan. Konsekuensi langsung dari hubungan ini adalah bahwa titik-titik yang lebih jauh diproyeksikan ke titik-titik yang lebih dekat ke pusat gambar (konvergensi garis paralel) dan bahwa jarak antara bayangan titik-titik di dunia berkurang untuk titik dunia yang lebih jauh (ukuran objek dalam gambar diskalakan berdasarkan jarak)[9] bayangan permukaan datar tak berhingga di dunia berakhir pada horizon berhingga dijelaskan dengan memeriksa persamaan proyeksi perspektif sebagai $z \to \infty$.

Dengan pengecualian efek terkait ukuran yang dijelaskan dalam Bagian 20.4.2, sebagian besar petunjuk kedalaman gambar yang melibatkan perspektif linier bergantung pada objek yang diinginkan yang bersentuhan dengan bidang dasar. Akibatnya, isyarat ini tidak memperkirakan jarak ke objek tetapi, sebaliknya, jarak ke titik kontak di bidang tanah. Dengan asumsi pengamat dan objek keduanya berada di atas bidang tanah horizontal, maka lokasi di bidang tanah yang lebih rendah dalam pandangan akan dekat. Gambar 20.27 mengilustrasikan efek ini secara kuantitatif. Untuk titik pandang h di atas tanah dan sudut deklinasi antara cakrawala dan suatu tempat menarik di tanah, titik yang dimaksud adalah jarak $d = h\cot\theta$ dari titik di mana pengamat berdiri. Sudut deklinasi memberikan informasi kedalaman relatif untuk sudut pandang tetap yang sewenang-wenang dan dapat memberikan kedalaman absolut ketika scalling dengan ketinggian mata (h) dimungkinkan.



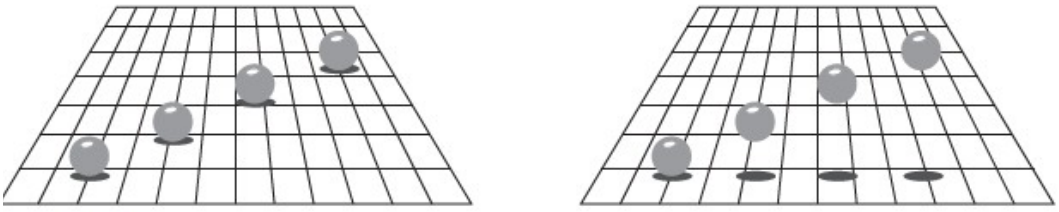
Gambar 20.27. Jarak mutlak ke lokasi pada bidang tanah dapat ditentukan berdasarkan sudut deklinasi dari cakrawala dan ketinggian mata.

[9] Matematika sebenarnya untuk menganalisis spesifikasi penglihatan biologis berbeda, karena mata tidak didekati dengan baik oleh formulasi proyeksi planar yang digunakan dalam grafis komputer dan sebagian besar aplikasi pencitraan lainnya.

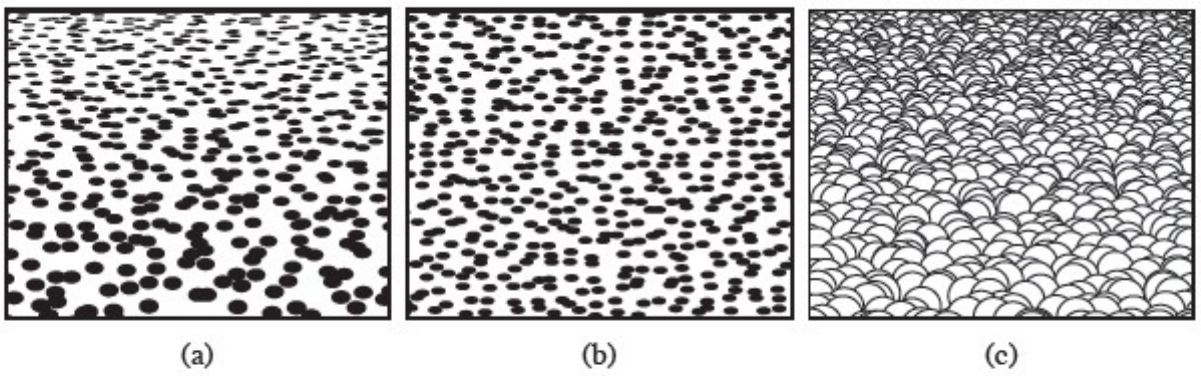
Sementara sistem visual manusia hampir pasti menggunakan sudut deklinasi sebagai petunjuk kedalaman, mekanisme pasti yang digunakan untuk memperoleh informasi yang dibutuhkan masih belum jelas. Sudut θ dapat diperoleh relatif terhadap gravitasi atau cakrawala yang terlihat. Ada beberapa bukti bahwa keduanya digunakan dalam penglihatan manusia. Tinggi mata h dapat didasarkan pada postur tubuh, ditentukan secara visual dengan melihat tanah di kaki seseorang, atau dipelajari dari pengalaman dan dianggap konstan. Sementara sejumlah peneliti telah menyelidiki masalah ini, apakah dan bagaimana nilai-nilai ini ditentukan belum diketahui secara pasti.

Bayangan menyediakan berbagai jenis informasi tentang tata ruang tiga dimensi. Bayangan terlampir menunjukkan bahwa objek berada dalam kontak dengan permukaan lain, sering kali terdiri dari bidang tanah. Bayangan terpisah menunjukkan bahwa suatu objek dekat dengan beberapa permukaan, tetapi tidak bersentuhan dengan permukaan itu. Bayangan dapat berfungsi sebagai isyarat kedalaman tidak langsung dengan menyebabkan suatu objek muncul pada kedalaman lokasi bayangan di groundplane (Yonas, Goldsmith, & Hallstrom, 1978). Saat menggunakan isyarat ini, sistem visual tampaknya membuat asumsi bahwa cahaya datang langsung dari atas (Gambar 20.28).

Visi memberikan informasi tentang orientasi permukaan serta jarak. Lebih mudah untuk mewakili orientasi permukaan yang ditentukan secara visual dalam hal kemiringan, yang didefinisikan sebagai orientasi dalam gambar proyeksi normal permukaan, dan kemiringan, yang didefinisikan sebagai sudut antara permukaan normal dan garis pandang.



Gambar 20.28. Bayangan secara tidak langsung dapat berfungsi sebagai isyarat kedalaman dengan mengasosiasikan kedalaman suatu objek dengan lokasi pada bidang dasar (setelah Kersten, Mamassian, dan Knill (1997)).

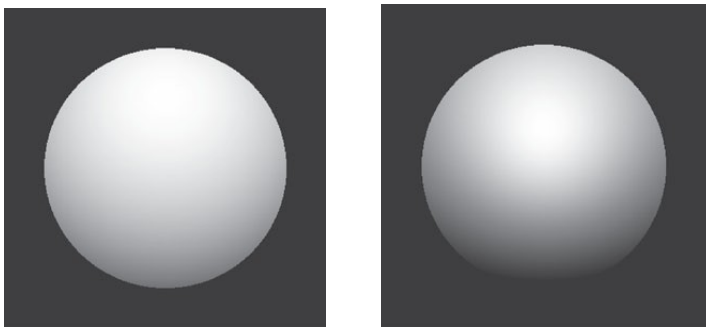


Gambar 20.29. Isyarat tekstur untuk kemiringan. (a) Dekat permukaan yang menunjukkan kompresi dan gradien tekstur; (b) permukaan jauh yang hanya memperlihatkan kompresi; (c) variabilitas dalam penampilan dekat permukaan dengan variabilitas geometris teratur.

Cakrawala permukaan yang terlihat dapat digunakan untuk menemukan orientasi permukaan (secara efektif tak terbatas) relatif terhadap pengamat. Menentukan kemiringan

sangatlah mudah, karena kemiringan permukaan adalah orientasi dari cakrawala yang terlihat. Kemiringan juga dapat dipulihkan, karena garis pandang dari titik mata ke cakrawala menentukan bidang yang sejajar dengan permukaan. Dalam banyak situasi, baik cakrawala permukaan tidak terlihat atau permukaannya cukup kecil sehingga tepi jauhnya tidak sesuai dengan cakrawala yang sebenarnya. Dalam kasus seperti itu, tekstur yang terlihat masih dapat digunakan untuk memperkirakan orientasi.

Dalam konteks persepsi, istilah tekstur mengacu pada pola visual yang terdiri dari sub-pola yang direplikasi di atas permukaan. Sub-pola dan distribusinya dapat tetap dan teratur, seperti untuk papan catur, atau konsisten dalam arti yang lebih statistik, seperti dalam tampilan lapangan berumput.[10] Ketika permukaan bertekstur dilihat dari sudut miring, tampilan yang diproyeksikan tekstur terdistorsi relatif terhadap tanda yang sebenarnya di permukaan. Dua jenis distorsi yang cukup berbeda terjadi (Knill, 1998), keduanya dipengaruhi oleh jumlah kemiringan. Posisi dan ukuran elemen tekstur tunduk pada efek perspektif linier yang dijelaskan di atas. Ini menghasilkan gradien tekstur (Gibson, 1950) karena ukuran elemen dan jarak berkurang dengan jarak (Gambar 20.29(a)). Baik gambar elemen tekstur individu maupun distribusi elemen dipersingkat di bawah tampilan miring (Gambar 20.29(b)). Ini menghasilkan kompresi ke arah kemiringan. Misalnya, lingkaran yang tampak miring muncul sebagai elips, dengan rasio sumbu minor ke mayor sama dengan cosinus kemiringan. Perhatikan bahwa pemendekan itu sendiri bukanlah hasil dari perspektif linier, meskipun dalam praktiknya, baik perspektif linier maupun pemendekan memberikan informasi tentang kemiringan.[11]



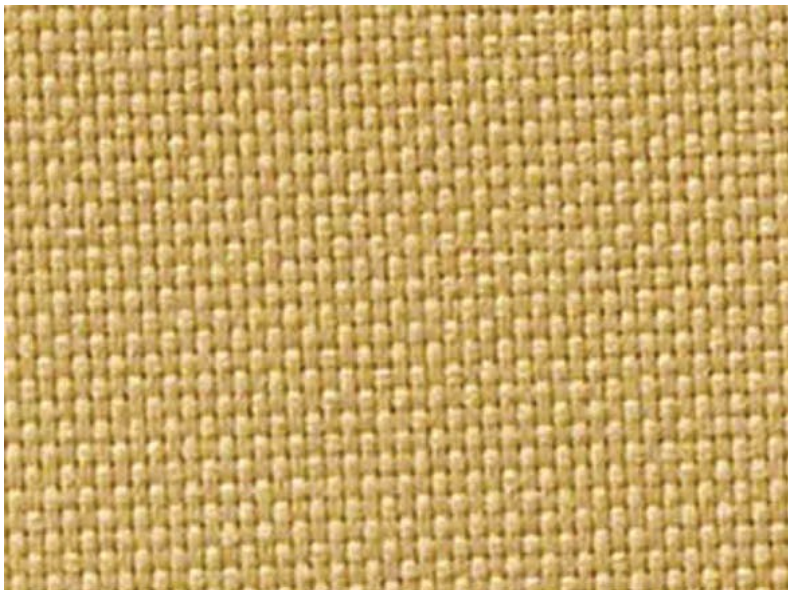
Gambar 20.30. Bentuk-dari-shading. Gambar pada (a) dan (b) tampak memiliki bentuk 3D yang berbeda karena perbedaan laju perubahan kecerahan pada permukaannya.

Agar gradien tekstur berfungsi sebagai isyarat untuk kemiringan permukaan, ukuran rata-rata dan jarak elemen tekstur harus konstan di atas permukaan bertekstur. Jika variabilitas spasial dalam ukuran dan jarak pada gambar tidak sepenuhnya disebabkan oleh proses proyeksi, maka upaya untuk membalikkan efek proyeksi akan menghasilkan kesimpulan yang salah tentang orientasi permukaan. Demikian juga, isyarat pemendekan gagal jika bentuk elemen tekstur tidak isotropik, sejak itu bentuk gambar elemen tekstur asimetris akan terjadi dalam situasi yang tidak terkait dengan tampilan miring. Ini adalah contoh asumsi yang sering diperlukan agar isyarat visual spasial menjadi efektif. Asumsi seperti itu masuk akal sampai tingkat yang mencerminkan sifat-sifat dunia yang umum terjadi.

Shading juga memberikan informasi tentang bentuk permukaan (Gambar 20.30). Kecerahan titik yang dilihat pada permukaan tergantung pada pantulan permukaan dan orientasi permukaan terhadap sumber cahaya terarah dan titik pengamatan. Ketika posisi

[10] Dalam grafis komputer, istilah tekstur memiliki arti yang berbeda, mengacu pada gambar apa pun yang diterapkan ke permukaan sebagai bagian dari proses rendering.
[11] Bentuk ketiga dari distorsi visual terjadi ketika permukaan dengan relief permukaan 3D yang berbeda dilihat secara miring (Leung & Malik, 1997), seperti yang ditunjukkan pada Gambar 20.29(c). Saat ini tidak ada yang diketahui tentang apakah atau bagaimana efek ini dapat digunakan oleh sistem penglihatan manusia untuk menentukan kemiringan.

relatif suatu objek, arah pandang, dan arah iluminasi tetap, perubahan kecerahan di atas permukaan pemantulan yang konstan merupakan indikasi perubahan orientasi permukaan objek. Shape-from-shading adalah proses pemulihan bentuk permukaan dari variasi kecerahan yang diamati. Hampir tidak pernah mungkin untuk memulihkan orientasi sebenarnya dari permukaan dari shading saja, meskipun shading sering dapat dikombinasikan dengan isyarat lain untuk memberikan indikasi bentuk permukaan yang efektif. Untuk permukaan dengan variabilitas geometris skala halus, bayangan dapat memberikan tampilan tiga dimensi yang menarik, bahkan untuk gambar yang dirender pada permukaan dua dimensi (Gambar 20.31).

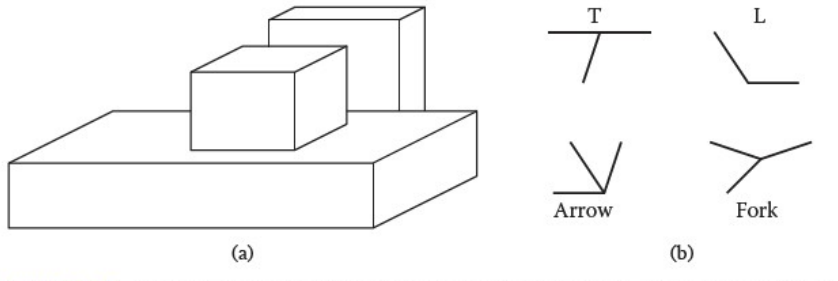


Gambar 20.31. Shading dapat menghasilkan persepsi yang kuat dari bentuk tiga dimensi.

Dalam gambar 20.31, efeknya lebih kuat jika Anda melihat gambar dari jarak beberapa meter menggunakan satu mata. Akan menjadi lebih kuat lagi jika Anda meletakkan selembar karton di depan gambar dengan lubang yang dipotong sedikit lebih kecil dari gambar (lihat Bagian 20.5). Gambar milik Albert Yonas.

Ada sejumlah isyarat bergambar yang menghasilkan informasi ordinal tentang kedalaman, tanpa secara langsung menunjukkan jarak sebenarnya. Gambar sebaris, jenis sambungan yang berbeda memberikan batasan pada geometri 3D yang dapat menghasilkan gambar (Gambar 20.32). Banyak dari efek ini juga terjadi pada gambar yang lebih alami. Persepsi yang paling efektif dari isyarat persimpangan adalah persimpangan-T, yang merupakan indikator kuat bahwa permukaan yang berlawanan dengan batang T menutupi setidaknya satu permukaan yang lebih jauh. Persimpangan-T sering menghasilkan rasa penyelesaian amodal, di mana satu permukaan terlihat berlanjut di belakang permukaan yang lebih dekat dan tertutup (Gambar 20.33).

Efek atmosfer menyebabkan perubahan visual yang dapat memberikan informasi tentang kedalaman, terutama di luar ruangan dalam jarak jauh. Leonardo da Vinci adalah orang pertama yang menggambarkan perspektif udara (juga disebut perspektif atmosfer), di mana hamburan mengurangi kontras bagian yang jauh dari scene dan menyebabkannya tampak lebih kebiruan daripada jika mereka lebih dekat (da Vinci, 1970) (lihat Gambar 20.34). Perspektif udara sebagian besar merupakan isyarat kedalaman relatif, meskipun ada beberapa spekulasi yang mungkin mempengaruhi persepsi jarak mutlak juga. Sementara banyak orang percaya bahwa objek yang lebih jauh terlihat lebih kabur karena efek atmosfer, hamburan atmosfer sebenarnya menyebabkan sedikit buram.



Gambar 20.32. (a) Persimpangan memberikan informasi tentang oklusi dan kecembungan atau kecekungan sudut. (b) Jenis sambungan umum untuk objek permukaan planar.

Gambar 20.33. T-junction menyebabkan piringan kiri tampak melanjutkan di belakang persegi panjang, sedangkan piringan kanan muncul di depan persegi panjang, yang terlihat berlanjut di belakang piringan.

20.10 OBJEK, LOKASI, DAN ACARA

Meskipun ada kesepakatan yang cukup luas di antara para ilmuwan visi saat ini bahwa tujuan penglihatan adalah untuk mengekstrak informasi tentang objek, lokasi, dan peristiwa, ada sedikit konsensus tentang fitur kunci dari informasi apa yang diekstraksi, bagaimana diekstraksi, atau bagaimana informasi digunakan untuk melakukan tugas. Kontroversi signifikan ada tentang sifat pengenalan objek dan interaksi potensial antara pengenalan objek dan aspek persepsi lainnya. Sebagian besar dari apa yang kita ketahui tentang lokasi melibatkan penglihatan spasial tingkat rendah, bukan masalah yang terkait dengan hubungan spasial antara objek kompleks atau proses visual yang diperlukan untuk bernavigasi di lingkungan yang kompleks. Kita tahu cukup banyak tentang bagaimana orang memandang kecepatan dan arah mereka ketika mereka bergerak di seluruh dunia, tetapi hanya memiliki pemahaman yang terbatas tentang persepsi peristiwa faktual. Perhatian visual melibatkan aspek persepsi objek, lokasi, dan kejadian. Sementara ada banyak data tentang fenomenologi perhatian visual untuk rangsangan yang relatif sederhana dan terkontrol dengan baik, kita tahu lebih sedikit tentang bagaimana perhatian visual melayani tujuan persepsi tingkat tinggi.

Gambar 20.34. Perspektif udara, di mana efek atmosfer mengurangi kontras dan menggeser warna ke arah biru, memberikan isyarat kedalaman jarak jauh.

Pengenalan objek melibatkan pemisahan gambar menjadi bagian-bagian konstituen yang sesuai dengan entitas fisik yang berbeda dan menentukan identitas entitas tersebut. Gambar 20.35 menggambarkan keamanan dari kompleksitas yang terkait dengan proses ini. Kami memiliki sedikit kesulitan untuk mengenali bahwa gambar di sebelah kiri adalah semacam kendaraan, meskipun kami belum pernah melihat pandangan khusus tentang kendaraan ini dan kebanyakan dari kita biasanya tidak mengaitkan kendaraan dengan konteks ini. Gambar di sebelah kanan kurang mudah dikenali sampai halaman dibalik, menunjukkan preferensi orientasi dalam pengenalan objek manusia.

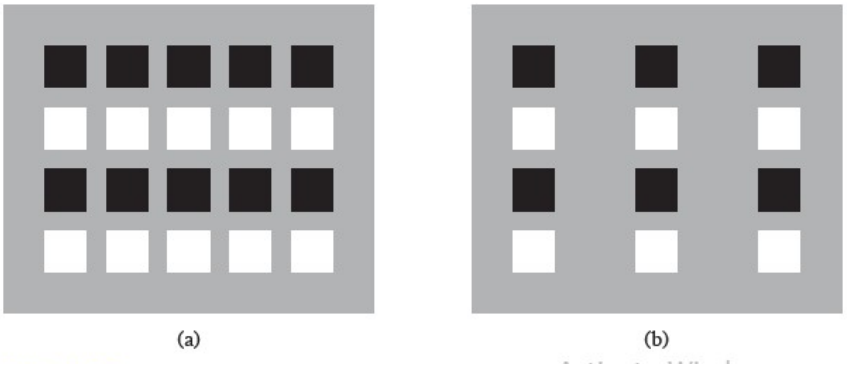
Pengenalan objek dianggap melibatkan dua langkah yang cukup berbeda. Langkah pertama mengatur bidang visual ke dalam kelompok-kelompok yang mungkin sesuai dengan objek dan permukaan. Proses pengelompokan ini sangat kuat (lihat Gambar 20.36), meskipun ada sedikit atau tanpa kesadaran dari fitur gambar tingkat rendah yang menghasilkan efek pengelompokan.[12] Pengelompokan didasarkan pada interaksi kompleks dari kedekatan, kesamaan dalam kecerahan, warna, bentuk, dan orientasi struktur primitif dalam gambar, gerakan umum, dan berbagai hubungan yang lebih kompleks.

(a) (b)

Gambar 20.35. Kompleksitas pengenalan objek. (a) Kami mengenali objek seperti kendaraan meskipun kami mungkin belum pernah melihat scene kendaraan ini sebelumnya. (b) Gambar

[12] Bentuk kamuflase visual yang paling umum melibatkan penambahan tekstur visual yang menipu proses pengelompokan persepsi sehingga pandangan dunia tidak dapat diatur dengan cara yang memisahkan objek yang disamarkan.

sulit dikenali berdasarkan tampilan cepat. Menjadi lebih mudah untuk mengenali jika buku itu terbalik.



Gambar 20.36. Gambar secara persepsi diatur ke dalam pengelompokan berdasarkan seperangkat kesamaan dan kriteria organisasi yang kompleks. (a) Kesamaan dalam kecerahan menghasilkan empat pengelompokan horizontal. (b) Kedekatan menghasilkan tiga pengelompokan vertikal.

Langkah kedua dalam pengenalan objek adalah menafsirkan pengelompokan sebagai objek yang diidentifikasi. Sebuah analisis komputasi menunjukkan bahwa ada sejumlah cara yang jelas berbeda di mana suatu objek dapat diidentifikasi. Data persepsi tidak jelas tentang laut mana yang benar-benar digunakan dalam penglihatan manusia. Pengenalan objek membutuhkan sistem visi yang memiliki deskripsi masing-masing kelas objek yang cukup untuk membedakan setiap kelas dari yang lain. Teori pengenalan objek berbeda dalam sifat informasi yang menggambarkan setiap kelas dan mekanisme yang digunakan untuk mencocokkan deskripsi ini dengan pandangan dunia yang sebenarnya.

Gambar 20.37. Pencocokan template. Titik terang di gambar kanan menunjukkan lokasi yang paling cocok dengan template di gambar kiri.

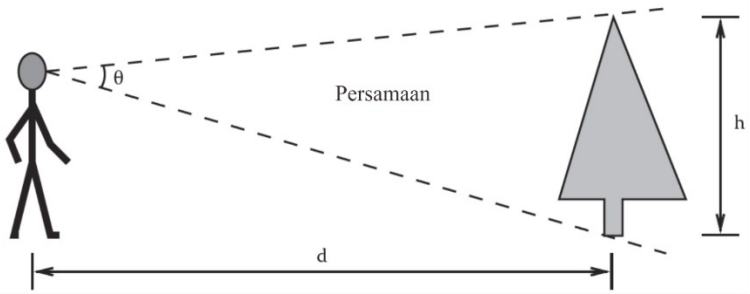
Tiga jenis deskripsi umum dimungkinkan. Template merepresentasikan kelas-kelas objek dalam istilah-istilah dari tampilan prototipikal dari objek di setiap kelas. Gambar 20.37 menunjukkan contoh sederhana. Deskripsi Struktural mewakili kelas-kelas objek dalam istilah fitur-fitur khas dari setiap kelas yang cenderung mudah dideteksi dalam tampilan objek, bersama dengan informasi tentang hubungan geometris antara fitur-fitur tersebut. Deskripsi struktural dapat direpresentasikan dalam 2D atau 3D. Untuk model 2D dari tipe objek, harus ada deskripsi terpisah untuk setiap potensi tampilan objek yang sangat berbeda. Untuk model 3D, dua bentuk strategi pencocokan yang berbeda dimungkinkan. Dalam satu, struktur tiga dimensi dari objek yang dilihat ditentukan sebelum klasifikasi menggunakan isyarat spasial apa pun yang tersedia, dan kemudian deskripsi tampilan 3D ini dicocokkan dengan prototipe 3D dari objek yang diketahui. Kemungkinan lainnya adalah bahwa beberapa mekanisme memungkinkan penentuan orientasi objek yang belum diidentifikasi yang sedang dilihat. Informasi orientasi ini digunakan untuk memutar dan memproyeksikan potensi deskripsi 3D dengan cara yang memungkinkan pencocokan 2D dari deskripsi dan objek yang dilihat.

Terakhir, opsi terakhir untuk mendeskripsikan properti kelas objek melibatkan fitur invarian yang mendeskripsikan kelas objek dalam kaitannya dengan properti geometris yang lebih umum, terutama yang mungkin tidak sensitif terhadap pandangan objek yang berbeda.

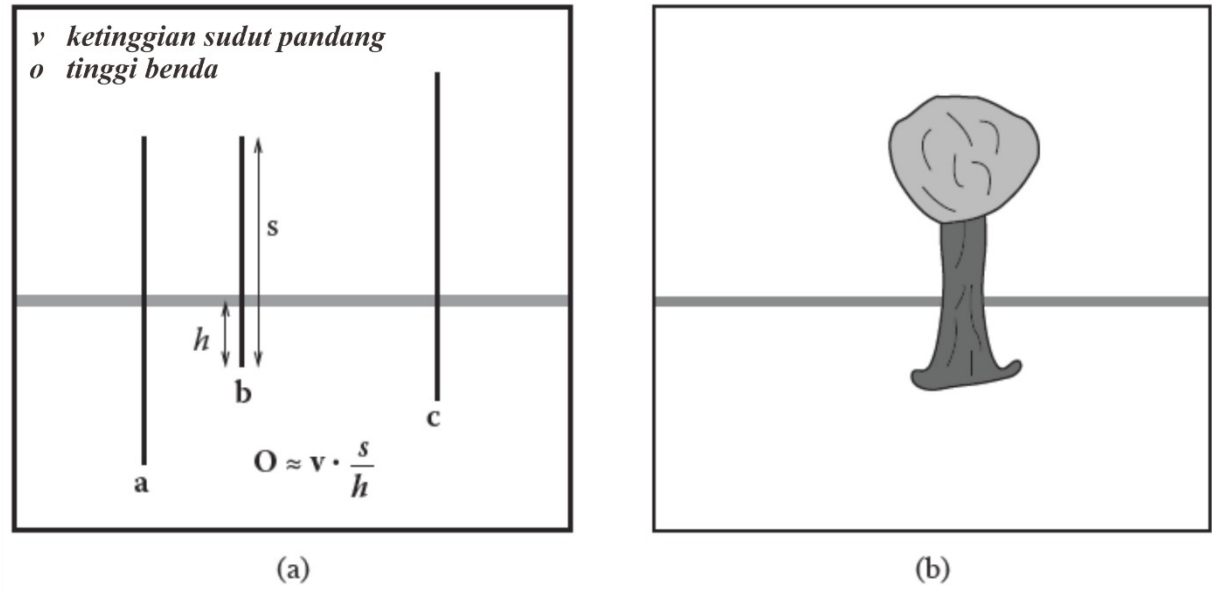
20.11 UKURAN DAN JARAK

Dengan tidak adanya informasi yang lebih pasti tentang kedalaman, objek yang diproyeksikan ke area retina yang lebih besar terlihat lebih dekat dibandingkan dengan objek yang diproyeksikan ke area retina yang lebih kecil, efek yang disebut ukuran relatif. Isyarat yang lebih kuat melibatkan ukuran yang dikenal, yang dapat memberikan informasi tentang jarak absolut ke objek yang dapat dikenali dengan ukuran yang diketahui. Kekuatan ukuran yang sudah dikenal sebagai isyarat kedalaman dapat dilihat dalam ilusi, di mana sebuah benda ditempatkan di kedalaman bidang dasar, berbasis perspektif. Ukuran yang familier adalah salah satu bagian dari hubungan ukuran-jarak, yang menghubungkan ukuran fisik suatu objek, ukuran optik dari objek yang sama yang diproyeksikan ke retina, dan jarak objek dari mata (Gambar 20.39).

Ketika objek berada di atas bidang datar, sumber tambahan untuk informasi kedalaman tersedia, terutama ketika cakrawala terlihat atau dapat diturunkan dari informasi perspektif lainnya. Sudut deklinasi ke titik kontak di tanah adalah isyarat kedalaman relatif dan memberikan jarak egosentris absolut ketika diukur dengan ketinggian mata, seperti yang ditunjukkan sebelumnya pada Gambar 20.27. Rasio cakrawala, di mana ketinggian total yang terlihat dari suatu objek dibandingkan dengan luas yang terlihat dari bagian objek yang muncul di bawah cakrawala, dapat digunakan untuk menentukan ukuran sebenarnya dari objek, bahkan ketika jarak ke objek tidak. diketahui (Gambar 20.40). Mendasari rasio cakrawala adalah fakta bahwa untuk bidang datar, garis pandang ke cakrawala memotong objek pada posisi yang tepat setinggi mata di atas tanah.



Gambar 20.39. Hubungan ukuran-jarak memungkinkan jarak ke objek dengan ukuran yang diketahui ditentukan berdasarkan sudut visual yang dibentuk oleh objek. Demikian juga ukuran suatu benda pada jarak yang diketahui dapat ditentukan berdasarkan sudut pandang yang dibentuk oleh benda tersebut.



Gambar 20.40. (a) Rasio horizon dapat digunakan untuk menentukan kedalaman dengan membandingkan bagian yang terlihat dari suatu objek di bawah cakrawala dengan luas total vertikal yang terlihat dari objek tersebut. (b) Contoh dunia nyata.

Sistem visual manusia cukup mampu menentukan ukuran absolut dari objek yang paling banyak dilihat; persepsi kita tentang ukuran didominasi oleh ukuran fisik yang sebenarnya, dan kita hampir tidak memiliki kesadaran akan ukuran retina yang sesuai dari objek. Ini mirip dengan kekonstanan ringan, yang dibahas sebelumnya, bahwa persepsi kita didominasi oleh sifat-sifat dunia yang disimpulkan, bukan fitur tingkat rendah yang sebenarnya dirasakan oleh fotoreseptor di retina. Gregory (1997) menjelaskan sebuah contoh sederhana dari keteguhan ukuran. Pegang kedua tangan Anda di depan Anda, satu sepanjang lengan dan yang lainnya pada jarak setengah dari Anda (Gambar 20.41(a)). Kedua tangan Anda akan terlihat hampir sama, meskipun ukuran retinanya berbeda dua kali lipat. Efeknya kurang kuat jika tangan yang lebih dekat menutup sebagian tangan yang lebih jauh, terutama jika Anda menutup satu mata (Gambar 20.41(b)). Sistem visual juga menunjukkan kekonstanan bentuk, di mana persepsi struktur geometris mendekati geometri objek sebenarnya daripada yang mungkin diharapkan mengingat distorsi gambar retina karena perspektif (Gambar 20.42).

Gambar 20.41. (a) Keteguhan ukuran membuat tangan yang diposisikan pada jarak yang berbeda dari mata tampak memiliki ukuran yang hampir sama untuk dilihat di dunia nyata, meskipun ukuran retina sangat berbeda. (b) Efeknya kurang kuat ketika satu tangan tertutup

sebagian oleh tangan yang lain, terutama ketika satu mata tertutup. Gambar milik Peter Shirley dan Pat Moulis.

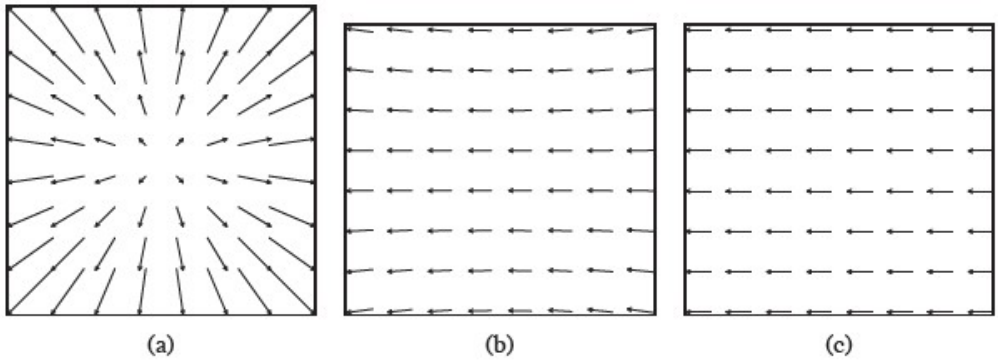
Gambar 20.42. Keteguhan bentuk—tabel terlihat persegi panjang meskipun bentuknya pada gambar adalah poligon empat sisi yang tidak beraturan.

20.12 ACARA

Sebagian besar aspek persepsi peristiwa berada di luar cakupan bab ini, karena melibatkan proses kognitif nonvisual yang kompleks. Namun, tiga jenis persepsi peristiwa terutama visual, dan juga memiliki relevansi yang jelas dengan grafis komputer. Penglihatan mampu memberikan informasi tentang bagaimana seseorang bergerak di dunia, keberadaan benda-benda yang bergerak bebas di dunia, dan potensi tumbukan baik karena gerak pengamat maupun karena benda bergerak ke arah pengamat.

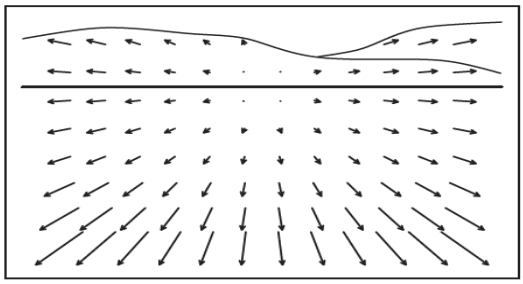
Visi dapat digunakan untuk menentukan rotasi dan arah translasi relatif terhadap lingkungan. Kasus paling sederhana melibatkan gerakan menuju permukaan datar yang berorientasi tegak lurus terhadap garis pandang. Dengan asumsi bahwa ada tekstur permukaan yang cukup untuk memungkinkan pemulihan aliran optik, bidang aliran akan membentuk pola simetris seperti yang ditunjukkan pada Gambar 20.43(a). Lokasi di bidang pandang fokus perluasan bidang aliran akan memiliki garis pandang terkait yang sesuai dengan arah translasi. Sementara aliran optik dapat digunakan untuk menentukan arah gerakan secara visual, aliran optik tidak mengandung informasi yang cukup untuk menentukan kecepatan. Untuk melihat ini, pertimbangkan situasi di mana dunia dibuat dua kali lebih besar dan penonton bergerak dua kali lebih cepat. Penurunan besaran nilai aliran akibat penggandaan jarak secara tepat dikompensasikan dengan kenaikan besaran nilai aliran akibat penggandaan kecepatan, yang menghasilkan medan aliran yang identik.

Gambar 20.43(b) menunjukkan medan aliran optik yang dihasilkan dari penampil (atau lebih tepatnya, mata pengamat) berputar di sekitar sumbu vertikal. Berbeda dengan situasi sehubungan dengan gerak translasi, aliran optik memberikan informasi yang cukup untuk menentukan baik sumbu rotasi maupun kecepatan (sudut) rotasi. Masalah praktis dalam memanfaatkan ini adalah bahwa aliran yang dihasilkan dari gerakan rotasi murni di sekitar sumbu yang tegak lurus terhadap garis pandang cukup mirip dengan aliran yang dihasilkan dari translasi murni dalam arah yang tegak lurus terhadap garis pandang dan sumbu rotasi ini, sehingga sulit untuk membedakan secara visual antara keduanya. jenis gerak (Gambar 20.43(c)). Gambar 20.44 menunjukkan pola aliran optik yang dihasilkan oleh gerakan melalui lingkungan yang lebih realistis.



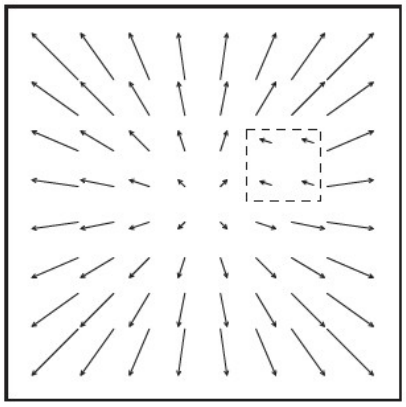
Gambar 20.43. (a) Gerakan menuju permukaan yang rata dan bertekstur menghasilkan medan aliran yang meluas, dengan fokus pemuaian yang menunjukkan garis pandang yang sesuai dengan arah gerakan. (b) Medan aliran yang dihasilkan dari rotasi di sekitar sumbu vertikal saat melihat permukaan datar yang berorientasi tegak lurus terhadap garis pandang. (c) Medan aliran yang dihasilkan dari translasi sejajar dengan permukaan yang rata dan bertekstur.

Jika tampilan benar-benar stasioner, deteksi visual dari objek bergerak menjadi mudah, karena objek tersebut akan diasosiasikan dengan satu-satunya aliran optik tidak nol di bidang pandang. Situasinya jauh lebih rumit ketika pengamat bergerak, karena bidang visual akan didominasi oleh aliran bukan nol, yang sebagian besar atau semuanya disebabkan oleh gerakan relatif antara pengamat dan lingkungan statis (Thompson & Pong, 1990). Dalam kasus tersebut, sistem visual harus peka terhadap pola dalam bidang aliran optik yang tidak konsisten dengan bidang aliran yang terkait dengan gerakan pengamat relatif terhadap lingkungan statis (Gambar 20.45).



Gambar 20.44. Aliran optik yang dihasilkan dengan bergerak melalui lingkungan statis memberikan informasi tentang gerakan relatif terhadap lingkungan dan jarak ke titik di lingkungan. Dalam hal ini, arah pandang ditekan dari cakrawala, tetapi seperti yang ditunjukkan oleh fokus pemuaian, gerakannya sejajar dengan bidang tanah.

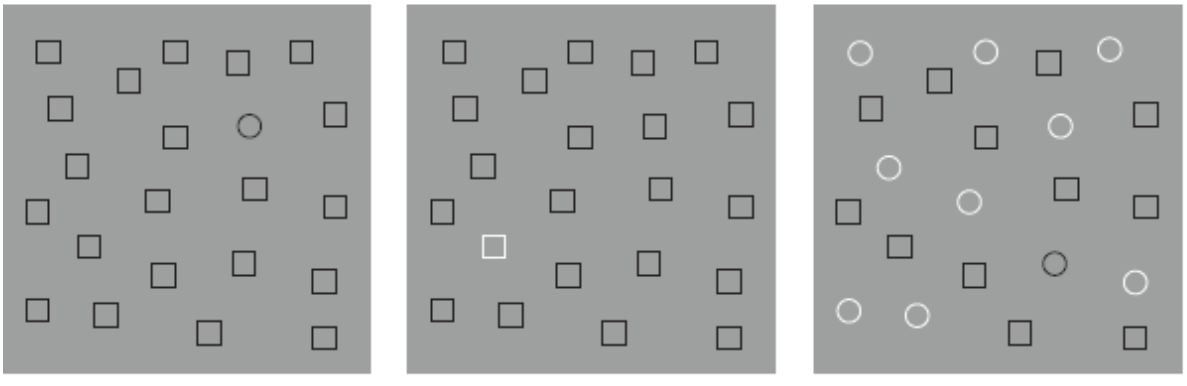
Bagian 20.3.4 menjelaskan bagaimana penglihatan dapat digunakan untuk menentukan waktu untuk kontak dengan suatu titik di lingkungan bahkan ketika kecepatan gerak tidak diketahui. Dengan asumsi seorang pengamat bergerak dengan lintasan lurus, kecepatan konstan dan tidak ada objek yang bergerak secara independen di dunia, kontak akan dibuat dengan permukaan apa pun yang searah dengan garis pandang yang sesuai dengan fokus pemuaian pada waktu yang ditunjukkan oleh hubungan . Sebuah objek yang bergerak secara independen memperumit masalah menentukan apakah tabrakan benar-benar akan terjadi. Pelaut menggunakan metode untuk mendeteksi potensi tabrakan yang juga dapat digunakan dalam sistem visual manusia: untuk gerakan garis lurus yang tidak dipercepat, tabrakan akan terjadi dengan objek yang secara visual meluas tetapi sebaliknya tetap diam secara visual dalam kerangka acuan egosentris.



Gambar 20.45. Deteksi visual objek bergerak dari titik pengamatan yang bergerak memerlukan pengenalan pola dalam aliran optik yang tidak dapat dikaitkan dengan gerakan melalui lingkungan statis.

Salah satu bentuk persepsi peristiwa yang lebih kompleks patut dibahas disini, karena sangat penting dalam grafisa komputer interaktif. Orang-orang sangat sensitif terhadap gerakan yang berhubungan dengan gerakan manusia. Penggerak dapat dikenali ketika satu-satunya fitur yang terlihat adalah lampu pada sendi alat bantu jalan (Johansson, 1973). Tampilan lampu bergerak seperti itu seringkali bahkan cukup untuk mengenali sifat-sifat seperti jenis kelamin pejalan kaki dan berat beban yang mungkin dibawa pejalan kaki. Dalam rendering grafis komputer, pemirsa akan melihat ketidakakuratan bahkan kecil dalam karakter animasi, terutama jika mereka dimaksudkan untuk meniru gerakan manusia.

Istilah perhatian visual mencakup berbagai fenomena dari mana kita mengarahkan mata kita ke efek kognitif yang melibatkan apa yang kita lihat dalam scene yang kompleks dan bagaimana kita menafsirkan apa yang kita lihat (Pashler, 1998). Gambar 20.46 memberikan contoh bagaimana proses perhatian mempengaruhi penglihatan, bahkan untuk gambar yang sangat sederhana. Di dua panel kiri, satu pola yang berbeda dalam bentuk atau warna dari yang lain segera "muncul" dan mudah terlihat. Pada panel di sebelah kanan, satu pola yang berbeda baik dalam bentuk maupun warna lebih sulit ditemukan. Alasan untuk ini adalah bahwa sistem visual dapat melakukan pencarian paralel untuk item yang dibedakan berdasarkan properti individu, tetapi membutuhkan lebih banyak pencarian kognitif dan berurutan ketika mencari item yang ditunjukkan oleh kehadiran simultan dari dua fitur pembeda. Antarmuka manusia-komputer berbasis grafis harus (tetapi seringkali tidak!) dirancang dengan pemahaman tentang bagaimana memanfaatkan proses perhatian visual pada orang-orang untuk mengomunikasikan informasi penting dengan cepat dan efektif.



Gambar 20.46. Dalam (a) dan (b), perhatian visual dengan cepat tertuju pada benda yang bentuk atau warnanya berbeda. Dalam (c), pencarian berurutan tampaknya diperlukan untuk menemukan satu item yang berbeda baik dalam bentuk maupun warna.

20.13 PERSEPSI GAMBAR

Sejauh ini, bab ini telah membahas persepsi visual yang terjadi ketika dunia dicitrakan secara langsung oleh mata manusia. Ketika kita melihat hasil grafis komputer, tentu kita melihat gambar yang dirender dan bukan dunia nyata. Ini memiliki implikasi persepsi yang penting. Pada prinsipnya, seharusnya dimungkinkan untuk menghasilkan grafis komputer yang tampak tidak dapat dibedakan dari dunia nyata, setidaknya untuk tampilan bermata tanpa objek atau gerakan pengamat. Bayangkan melihat dunia melalui windows kaca. Sekarang, pertimbangkan untuk mewarnai setiap titik di windows agar sama persis dengan warna dunia yang semula terlihat pada titik itu.[13] Cahaya yang mencapai mata tidak berubah dengan operasi ini, artinya persepsi harus sama apakah kaca yang dicat dilihat atau dunia nyata dilihat melalui windows. Tujuan grafis komputer dapat dianggap sebagai menghasilkan windows berwarna tanpa benar-benar memiliki tampilan dunia nyata yang setara.

Masalah untuk grafis komputer dan seni visual lainnya adalah bahwa dalam praktiknya kita tidak dapat mencocokkan pandangan dunia nyata dengan mewarnai permukaan yang datar. Kecerahan dan rentang dinamis cahaya di dunia nyata tidak mungkin dibuat ulang menggunakan teknologi tampilan apa pun saat ini. Resolusi gambar yang dirender juga seringkali kurang dari detail terbaik yang dapat dilihat oleh penglihatan manusia. Cahaya dan keteguhan warna kurang terlihat dalam gambar daripada di dunia nyata, kemungkinan karena sistem visual mencoba untuk mengkompensasi variabilitas dalam kecerahan dan warna iluminasi berdasarkan iluminasi sekitar di lingkungan tampilan, daripada iluminasi yang terkait dengan gambar yang dirender. Inilah sebabnya mengapa tampilan warna yang realistis dalam foto bergantung pada keseimbangan warna film untuk sifat sumber cahaya yang ada saat foto diambil dan mengapa warna realistis dalam video memerlukan langkah keseimbangan putih. Sementara banyak yang diketahui tentang bagaimana keterbatasan dalam resolusi, kecerahan, dan jangkauan dinamis mempengaruhi pendeteksian pola sederhana, hampir tidak ada yang diketahui tentang bagaimana properti tampilan ini mempengaruhi penglihatan spasial atau identifikasi objek.

Kami memiliki pemahaman yang lebih baik tentang aspek lain dari masalah ini, yang disebut psikolog sebagai persepsi ruang bergambar (S. Rogers, 1995). Satu perbedaan antara melihat gambar dan melihat dunia nyata adalah bahwa akomodasi, stereo binokular, paralaks gerak, dan mungkin isyarat kedalaman lainnya dapat menunjukkan bahwa permukaan yang dilihat jauh berbeda dari jarak di dunia yang dimaksudkan untuk diwakili. Kedalaman yang terlihat dalam situasi seperti itu cenderung berada di suatu tempat antara kedalaman yang ditunjukkan oleh isyarat gambar dalam gambar dan jarak ke gambar itu sendiri. Saat melihat foto atau tampilan komputer, ini sering menghasilkan skala rasa yang lebih kecil dari yang dimaksudkan. Di sisi lain, menonton film di bioskop layar lebar menghasilkan rasa kelapangan yang lebih menarik daripada menonton film yang sama di televisi, meskipun jarak ke TV sedemikian rupa sehingga sudut visualnya sama, karena film layar lebih jauh.

Grafis komputer yang dirender menggunakan proyeksi perspektif memiliki sudut pandang, yang ditentukan sebagai posisi dan arah dalam ruang model, dan frustum tampilan, yang menentukan bidang pandang horizontal dan vertikal dan beberapa aspek lain dari transformasi tampilan. Jika gambar yang dirender tidak dilihat dari lokasi yang benar, sudut visual ke tepi gambar tidak akan cocok dengan frustum yang digunakan dalam membuat gambar. Semua sudut visual dalam gambar juga akan terdistorsi, menyebabkan distorsi di

[13] Ide ini pertama kali dijelaskan oleh pelukis Leon Battista Alberti pada tahun 1435 dan sekarang dikenal sebagai Windows Alberti. Ini terkait erat dengan kamera obscura.

semua kedalaman gambar dan isyarat orientasi berdasarkan perspektif linier. Efek ini sering terjadi dalam praktiknya, ketika pemirsa diposisikan terlalu dekat atau terlalu jauh dari foto atau permukaan tampilan. Jika penampil terlalu dekat, petunjuk perspektif untuk kedalaman akan dikompresi, dan isyarat untuk kemiringan permukaan akan menunjukkan bahwa permukaan lebih dekat dengan garis pandang daripada yang sebenarnya terjadi. Situasinya terbalik jika pemirsa terlalu jauh dari foto atau layar. Situasinya bahkan lebih rumit jika garis pandang tidak melewati pusat area tampilan, seperti yang biasa terjadi pada berbagai situasi tampilan.

Sistem visual manusia mampu mengkompensasi sebagian distorsi perspektif yang timbul dari melihat gambar di lokasi yang salah, itulah sebabnya kita dapat duduk di kursi yang berbeda di bioskop dan mengalami rasa yang sama dari ruang yang digambarkan. Saat mengontrol posisi tampilan sangat penting, tabung tampilan dapat digunakan. Ini adalah tabung berukuran tepat, dipasang dalam posisi tetap relatif terhadap layar, dan melalui mana pemirsa melihat layar. Tabung penglihatan membatasi titik pengamatan ke (semoga) posisi yang benar. Melihat tabung juga cukup efektif dalam mengurangi konflik informasi mendalam antara isyarat gambar dalam gambar dan permukaan tampilan sebenarnya. Mereka menghilangkan baik stereo dan paralaks gerak, yang, jika ada, akan sesuai dengan permukaan tampilan, bukan tampilan yang diberikan. Jika diameternya cukup kecil, mereka juga mengurangi petunjuk lain ke lokasi permukaan layar dengan menyembunyikan bingkai gambar atau tepi perangkat layar. Perangkat tampilan imersif visual yang eksotis seperti *head-mounted display* (HMDs) melangkah lebih jauh dalam upaya menyembunyikan isyarat visual ke posisi permukaan tampilan sambil menambahkan stereo binokular dan paralaks gerak yang konsisten dengan geometri dunia yang ditampilkan.

BAB 21
REPRODUKSI NADA

Seperti yang dibahas dalam Bab 20, sistem visual manusia beradaptasi dengan berbagai kondisi tampilan. Di bawah tampilan normal, kita dapat melihat kisaran sekitar 4 hingga 5 log unit iluminasi, yaitu, rasio antara area paling terang dan paling gelap di mana kita dapat melihat detail mungkin sebesar 100.000 : 1. Melalui proses adaptasi, kita dapat beradaptasi dengan jangkauan iluminasi yang lebih besar. Kami menyebut gambar yang sesuai dengan kemampuan sistem visual manusia rentang dinamis tinggi.

Simulasi visual secara rutin menghasilkan gambar dengan rentang dinamis yang tinggi (Ward Larson & Shakespeare, 1998). Perkembangan terbaru dalam teknik pengambilan gambar memungkinkan beberapa eksposur untuk disejajarkan dan digabungkan kembali menjadi satu gambar rentang dinamis tinggi (Debevec & Malik, 1997). Beberapa teknik eksposur juga tersedia untuk video. Selain itu, kami berharap perangkat keras masa depan dapat memotret atau memfilmkan scene rentang dinamis tinggi secara langsung. Secara umum, kita mungkin menganggap setiap piksel sebagai triplet dari tiga angka titik mengambang.

Karena semakin mudah untuk membuat gambar rentang dinamis tinggi, kebutuhan untuk menampilkan data tersebut meningkat pesat. Sayangnya, sebagian besar perangkat tampilan, monitor, dan printer saat ini hanya mampu menampilkan sekitar 2 unit log rentang dinamis. Kami menganggap perangkat tersebut memiliki rentang dinamis rendah. Sebagian besar gambar yang ada saat ini diwakili dengan saluran byte per piksel per warna, yang dicocokkan dengan perangkat tampilan saat ini, bukan dengan scene yang diwakilinya.

Biasanya, gambar rentang dinamis rendah tidak dapat mewakili scene tanpa kehilangan informasi. Contoh umum adalah ruangan dalam ruangan dengan area luar ruangan yang terlihat melalui windows. Manusia dengan mudah dapat melihat detail baik bagian dalam maupun bagian luar. Sebuah foto konvensional biasanya tidak menangkap informasi lengkap ini—fotografer harus memilih apakah bagian dalam atau luar dari scene tersebut diekspos dengan benar (lihat Gambar 21.1). Keputusan ini dapat dihindari dengan menggunakan pencitraan rentang dinamis tinggi dan menyiapkan gambar ini untuk ditampilkan menggunakan teknik yang dijelaskan dalam bab ini (lihat Gambar 21.2).

Gambar 21.1. Dengan fotografi konvensional, beberapa bagian scene mungkin kurang atau terlalu terang. Untuk memvisualisasikan meja snooker, tampilan melalui windows dibakar di gambar kiri. Di sisi lain, meja snooker akan terlalu gelap jika bagian luar dari scene ini diekspos dengan benar. Bandingkan dengan Gambar 21.2, yang menunjukkan gambar rentang dinamis tinggi yang disiapkan untuk ditampilkan menggunakan algoritme reproduksi nada.

Ada dua strategi yang tersedia untuk menampilkan gambar rentang dinamis tinggi. Pertama, kami dapat mengembangkan perangkat tampilan yang dapat secara langsung mengakomodasi rentang dinamis tinggi (Seetzen, Whitehead, & Ward, 2003; Seetzen et al., 2004). Kedua, kami dapat menyiapkan gambar rentang dinamis tinggi untuk ditampilkan pada perangkat tampilan rentang dinamis rendah (Upstill, 1985). Saat ini pendekatan yang lebih umum dan topik bab ini. Meskipun kami memperkirakan bahwa perangkat tampilan rentang dinamis tinggi akan digunakan secara luas (dalam waktu dekat), kebutuhan untuk mengompresi rentang dinamis gambar mungkin berkurang, tetapi tidak akan hilang. Secara khusus, media cetak seperti buku ini, pada dasarnya, memiliki jangkauan dinamis yang rendah.

Gambar 21.2. Gambar nada rentang dinamis tinggi yang dipetakan untuk ditampilkan menggunakan operator reproduksi nada terkini (Reinhard & Devlin, 2005). Dalam gambar ini, baik bagian dalam ruangan maupun scene melalui windows diekspos dengan benar.

Mengompresi rentang nilai gambar untuk tujuan tampilan pada perangkat tampilan rentang dinamis rendah disebut mapping nada atau reproduksi nada.

Gambar 21.3. Scalling linier gambar rentang dinamis tinggi agar sesuai dengan perangkat tampilan tertentu dapat menyebabkan detail yang signifikan hilang (kiri dan tengah). Gambar kiri diskalakan secara linier. Di tengah gambar nilai tinggi dijepit. Sebagai perbandingan, gambar kanan dipetakan nada, memungkinkan detail di daerah terang dan gelap terlihat.

Fungsi kompresi sederhana adalah menormalkan gambar (lihat Gambar 21.3 (kiri)). Ini merupakan scalling linier yang cenderung cukup hanya jika rentang dinamis gambar hanya sedikit lebih tinggi daripada rentang dinamis perangkat tampilan. Untuk gambar dengan

rentang dinamis yang lebih tinggi, perbedaan intensitas kecil akan dikuantisasi ke nilai tampilan media yang sama sehingga detail yang terlihat hilang. Pada Gambar 21.3 (tengah) semua nilai piksel yang lebih besar dari maksimum yang ditentukan pengguna diatur ke maksimum ini (yaitu, mereka dijepit). Ini membuat normalisasi kurang bergantung pada outlier yang bising, tetapi di sini kita kehilangan informasi di area terang gambar. Sebagai perbandingan, Gambar 21.3 (kanan) adalah versi yang dipetakan dengan nada yang menunjukkan detail di wilayah gelap dan terang.

Gambar 21.4. Gambar yang digunakan untuk mendemonstrasikan tujuan reproduksi nada pada Gambar 21.5.

Secara umum, scalling linier tidak sesuai untuk reproduksi nada. Isu utama dalam reproduksi nada kemudian adalah mengompresi gambar sementara pada saat yang sama mempertahankan satu atau lebih atribut gambar. Algoritme reproduksi nada yang berbeda berfokus pada atribut yang berbeda seperti kontras, detail yang terlihat, kecerahan, atau tampilan.

Idealnya, menampilkan gambar yang dipetakan nada pada perangkat tampilan rentang dinamis rendah akan menciptakan respons visual yang sama di pengamat seperti scene aslinya. Mengingat keterbatasan perangkat tampilan, hal ini tidak akan dapat dicapai, meskipun kami dapat menargetkan untuk mendekati tujuan ini sedekat mungkin.

Sebagai contoh, kami membuat gambar rentang dinamis tinggi yang ditunjukkan pada Gambar 21.4. Gambar ini kemudian dipetakan nada dan ditampilkan pada perangkat tampilan. Perangkat layar itu sendiri kemudian ditempatkan dalam scene sedemikian rupa sehingga menampilkan latar belakang sendiri (Gambar 21.5). Dalam kasus yang ideal, tampilan akan tampak transparan. Tergantung pada kualitas operator reproduksi nada, serta sifat scene yang digambarkan, tujuan ini mungkin sedikit banyak dapat dicapai.

Gambar 21.5. Setelah tonemapping gambar pada Gambar 21.4 dan menampilkannya pada monitor, monitor ditempatkan pada scene kira-kira pada lokasi pengambilan gambar. Tergantung pada kualitas operator reproduksi nada, hasilnya akan tampak seperti monitor transparan.

21.1 KLASIFIKASI

Meskipun dimungkinkan untuk mengklasifikasikan operator reproduksi nada dengan atribut mana yang ingin mereka pertahankan, atau untuk tugas mana mereka dikembangkan, kami mengklasifikasikan algoritme menurut teknik umumnya. Ini akan memungkinkan kita untuk menunjukkan perbedaan dan persamaan antara sejumlah besar operator yang berbeda, dan dengan demikian, diharapkan, berkontribusi pada pemilihan operator tertentu yang berarti untuk tugas reproduksi nada yang diberikan.

Skema klasifikasi utama yang kami ikuti berdasarkan realisasi bahwa operator reproduksi nada didasarkan pada wawasan yang diperoleh dari berbagai disiplin ilmu. Secara khusus, beberapa operator didasarkan pada pengetahuan tentang persepsi visual manusia. Sistem visual manusia mendeteksi cahaya menggunakan fotoreseptor yang terletak di retina. Cahaya diubah menjadi sinyal listrik yang sebagian diproses di retina dan kemudian ditransmisikan ke otak. Kecuali untuk beberapa lapisan sel pertama di retina, sinyal yang berasal dari cahaya yang terdeteksi ditransmisikan menggunakan rangkaian impuls. Kuantitas pembawa informasi adalah frekuensi terjadinya pulsa elektrik ini.

Rentang cahaya yang dapat dideteksi oleh sistem visual manusia jauh lebih besar daripada rentang frekuensi yang digunakan oleh otak manusia untuk mengirimkan informasi. Dengan demikian, sistem visual manusia dengan mudah memecahkan masalah reproduksi nada—sejumlah besar luminans diubah menjadi rentang kecil frekuensi rangkaian impuls. Meniru aspek yang relevan dari sistem visual manusia karena itu merupakan pendekatan yang berharga untuk reproduksi nada; pendekatan ini dijelaskan secara lebih rinci dalam Bagian 21.7.

Operator kelas kedua didasarkan pada fisika. Cahaya berinteraksi dengan permukaan dan volume sebelum diserap oleh fotoreseptor. Dalam grafis komputer, interaksi cahaya umumnya dimodelkan dengan persamaan rendering. Untuk permukaan difus murni, persamaan ini dapat disederhanakan menjadi produk antara insiden cahaya pada permukaan (penerangan), dan kemampuan permukaan ini untuk memantulkan cahaya (pemantulan) (Oppenheim, Schafer, & Stockham, 1968).

Karena pantulan adalah sifat pasif permukaan, untuk permukaan difus, menurut definisi, rentang dinamis rendah—biasanya antara 0,005 dan 1 (Stockham, 1972). Pantulan permukaan tidak boleh lebih besar dari 1, karena itu akan memantulkan lebih banyak cahaya daripada yang terjadi di permukaan. Penerangan, di sisi lain, dapat menghasilkan nilai besar yang tidak wajar dan hanya dibatasi oleh intensitas dan kedekatan sumber cahaya.

Rentang dinamis gambar dengan demikian sebagian besar diatur oleh komponen iluminasi. Dalam menghadapi scene yang menyebar, pendekatan yang layak untuk reproduksi nada mungkin karena itu untuk memisahkan pantulan dari iluminasi, mengompresi komponen iluminasi, dan kemudian menggabungkan kembali gambar.

Namun, asumsi bahwa semua permukaan dalam suatu scene tersebar umumnya tidak benar. Banyak gambar rentang dinamis tinggi menggambarkan sorotan dan/atau sumber cahaya yang terlihat langsung (Gambar 21.3). Pencahayaan yang dipantulkan oleh permukaan specular mungkin hampir setinggi sumber cahaya yang dipantulkannya.

Operator reproduksi berbagai nada yang saat ini digunakan membagi gambar menjadi lapisan dasar rentang dinamis tinggi dan lapisan detail rentang dinamis rendah. Lapisan-lapisan ini akan mewakili penerangan dan pantulan jika scene yang digambarkan seluruhnya tersebar. Untuk scene yang mengandung sumber cahaya yang terlihat langsung atau sorotan khusus, pemisahan menjadi lapisan dasar dan detail masih memungkinkan desain operator reproduksi nada yang efektif, meskipun tidak ada makna langsung yang dapat dilampirkan pada lapisan yang terpisah. Operator tersebut dibahas dalam Bagian 21.5.

21.2 DYNAMIC RANGE (RENTANG DINAMIS)

Gambar konvensional disimpan dengan satu byte per piksel untuk masing-masing komponen merah, hijau dan biru. Rentang dinamis yang diberikan oleh pengkodean seperti itu tergantung pada rasio antara nilai terkecil dan terbesar yang dapat diwakili, serta ukuran langkah antara nilai yang berurutan. Jadi, untuk gambar rentang dinamis rendah, hanya ada 256 nilai berbeda per saluran warna.

Gambar rentang dinamis tinggi mengkodekan serangkaian nilai yang mungkin lebih besar secara signifikan; nilai representasi maksimum mungkin jauh lebih besar dan ukuran langkah antara nilai-nilai yang berurutan mungkin jauh lebih kecil. Oleh karena itu, ukuran file gambar rentang dinamis tinggi umumnya juga lebih besar, meskipun setidaknya satu standar (format file rentang dinamis tinggi OpenEXR (Kainz, Bogart, & Hess, 2003)) menyertakan skema kompresi yang sangat mumpuni.

Sebuah pendekatan yang berbeda untuk membatasi ukuran adalah untuk menerapkan operator produksi nada ke data dinamis tinggi. Hasilnya kemudian dapat dikodekan dalam format JPEG. Selain itu, gambar input dapat dibagi berdasarkan piksel dengan gambar yang dipetakan nada. Hasil dari pembagian ini kemudian dapat dijadikan subsampel dan menyimpan sejumlah kecil data dalam header dari gambar JPEG yang sama (G. Ward & Simmons, 2004). Ukuran file dari gambar yang dikodekan dengan sub-pita memiliki urutan yang sama dengan gambar yang dikodekan JPEG konvensional. Program tampilan dapat menampilkan gambar JPEG secara langsung atau dapat merekonstruksi gambar rentang dinamis tinggi dengan mengalikan gambar yang dipetakan nada dengan data yang disimpan di header.

Secara umum, kombinasi ukuran langkah terkecil dan rasio nilai terkecil dan terbesar yang dapat diwakili menentukan rentang dinamis yang diberikan skema pengkodean gambar. Untuk gambar yang dihasilkan komputer, sebuah gambar biasanya disimpan sebagai triplet dari nilai floating point sebelum ditulis ke file atau ditampilkan di layar, meskipun skema pengkodean yang lebih efisien dimungkinkan (Reinhard, Ward, Debevec, & Pattanaik, 2005). Karena sebagian besar perangkat layar masih dilengkapi dengan konverter D/A delapan bit, kita mungkin menganggap reproduksi nada sebagai mapping angka titik mengambang ke byte sehingga hasilnya dapat ditampilkan pada perangkat layar rentang dinamis rendah.

Rentang dinamis gambar individu umumnya lebih kecil, dan ditentukan oleh pencahayaan terkecil dan terbesar yang ditemukan di tempat kejadian. Oleh karena itu, pendekatan sederhana untuk mengukur rentang dinamis suatu gambar dapat menghitung rasio antara nilai piksel terbesar dan terkecil dari suatu gambar. Sensitivitas terhadap outlier dapat dikurangi dengan mengabaikan sebagian kecil piksel paling gelap dan paling terang.

Atau, rasio yang sama dapat dinyatakan sebagai perbedaan dalam domain logaritmik. Ukuran ini kurang sensitif terhadap outlier. Gambar yang ditampilkan di margin halaman ini adalah contoh gambar dengan rentang dinamis yang berbeda. Perhatikan bahwa scene malam dalam hal ini tidak memiliki rentang dinamis yang lebih kecil daripada scene siang hari. Sementara semua nilai dalam scene malam lebih kecil, rasio antara nilai terbesar dan terkecil tidak.

Namun, perangkat perekam atau algoritme rendering dapat menimbulkan noise yang akan menurunkan rentang dinamis yang berguna. Dengan demikian, pengukuran rentang dinamis suatu gambar harus memperhitungkan noise. Oleh karena itu, ukuran rentang dinamis yang lebih baik akan menjadi rasio sinyal-ke-noise, yang dinyatakan dalam desibel, seperti yang digunakan dalam pemrosesan sinyal.

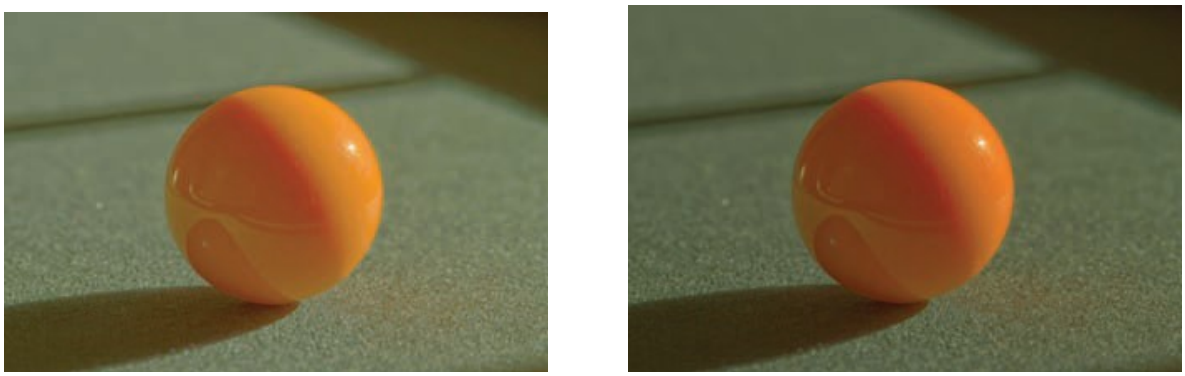
Gambar 21.6. Rentang dinamis 2,65 log2 unit.

Gambar 21.7. Rentang dinamis 3,96 unit log2.

Gambar 21.8. Rentang dinamis 4,22 unit log2.

Gambar 21.9. Rentang dinamis 5,01 log2 unit.

Gambar 21.10. Rentang dinamis 6,56 log2 unit.



Gambar 21.11. Koreksi gamma per saluran dapat menghilangkan saturasi gambar. Gambar kiri desaturasi dengan nilai s = 0,5. Gambar kanan tidak desaturasi (s = 1).

21.3 WARNA

Operator reproduksi nada biasanya memampatkan nilai luminansi, daripada bekerja langsung pada komponen merah, hijau, dan biru dari gambar berwarna. Setelah nilai luminansi ini dikompresi menjadi nilai tampilan Ld(x,y), gambar berwarna dapat direkonstruksi dengan menjaga rasio antara saluran warna sama seperti sebelum kompresi (menggunakan s = 1) (Schlick, 1994b):

$$
\begin{aligned}
I_{r,d}(x,y) &= \left(\frac{I_r(x,y)}{L_v(x,y)}\right)^s L_d(x,y), \\
I_{g,d}(x,y) &= \left(\frac{I_g(x,y)}{L_v(x,y)}\right)^s L_d(x,y), \\
I_{b,d}(x,y) &= \left(\frac{I_b(x,y)}{L_v(x,y)}\right)^s L_d(x,y).
\end{aligned}
$$

Hasilnya sering kali tampak terlalu jenuh, karena persepsi warna manusia tidak linier sehubungan dengan tingkat pencahayaan keseluruhan. Ini berarti bahwa jika kita melihat gambar scene luar ruangan yang terang pada monitor di lingkungan yang redup, mata kita akan beradaptasi dengan lingkungan dimensi daripada pencahayaan di luar ruangan. Dengan menjaga rasio warna tetap konstan, kami tidak memperhitungkan efek ini.

Atau, konstanta saturasi s dapat dipilih lebih kecil dari satu. Koreksi per-channelgamma seperti itu dapat menurunkan saturasi hasil ke tingkat yang sesuai, seperti yang ditunjukkan pada Gambar 21.11 (Fattal, Lischinski, & Werman, 2002). Solusi yang lebih komprehensif adalah dengan memasukkan ide-ide dari bidang pemodelan penampilan

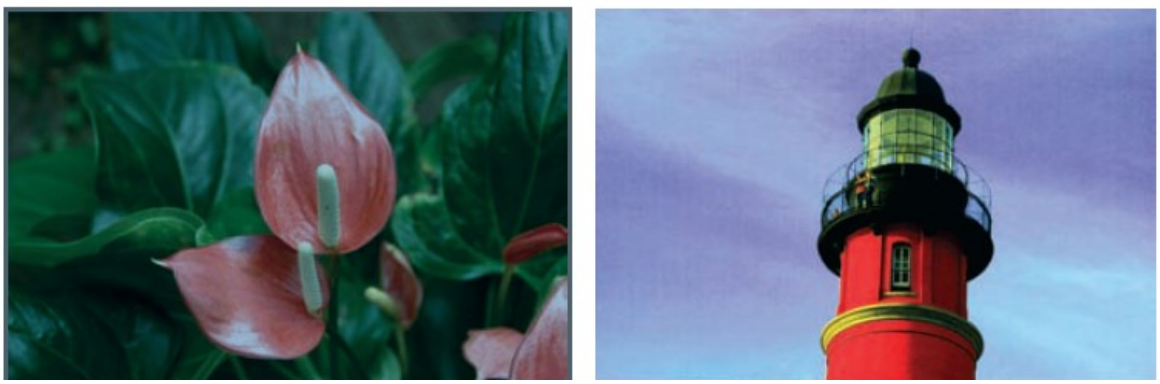
warna ke dalam operator reproduksi nada (Pattanaik, Ferwerda, Fairchild, & Greenberg, 1998; Fairchild & Johnson, 2004; Reinhard & Devlin, 2005).

Terakhir, jika contoh gambar dengan skema warna yang representatif sudah tersedia, skema warna ini dapat diterapkan ke gambar baru. Mapping warna antar gambar seperti itu dapat digunakan untuk koreksi warna yang halus, seperti penyesuaian saturasi atau untuk mapping warna yang lebih kreatif. Mapping dilanjutkan dengan mengonversi gambar sumber dan target ke ruang warna yang berhubungan dengan dekorasi. Dalam ruang warna seperti itu, nilai piksel di setiap saluran warna dapat diperlakukan secara independen tanpa memasukkan terlalu banyak artefak (Reinhard, Ashikhmin, Gooch, & Shirley, 2001).

Memetakan warna dari satu gambar ke gambar lain dalam ruang warna yang berhubungan dengan dekorasi kemudian menjadi mudah: hitung rata-rata dan simpangan baku semua piksel dalam gambar sumber dan target untuk tiga saluran warna secara terpisah. Kemudian, geser dan skalakan gambar target sehingga pada setiap saluran warna mean dan standar deviasi dari gambar target sama dengan gambar sumber. Gambar yang dihasilkan kemudian diperoleh dengan mengubah dari ruang warna terkait ke RGB dan menjepit piksel negatif menjadi nol. Rentang dinamis gambar mungkin telah berubah sebagai akibat dari penerapan algoritma ini. Oleh karena itu disarankan untuk menerapkan algoritme ini pada gambar rentang dinamis tinggi dan menerapkan algoritme reproduksi nada konvensional sesudahnya. Ruang warna terkait dekorasi yang sesuai adalah ruang lawan dari Bagian 19.2.4.

Gambar 21.12. Gambar yang digunakan untuk mendemonstrasikan teknik transfer warna. Hasil ditunjukkan pada Gambar 21.13 dan 21.31.

Hasil penerapan transformasi warna tersebut pada gambar pada Gambar 21.12 ditunjukkan pada Gambar 21.13.



Gambar 21.13. Gambar di sebelah kiri digunakan untuk mengatur warna gambar yang ditunjukkan pada Gambar 21.12. Hasilnya ditunjukkan di sebelah kanan

21.4 FORMASI GAMBAR

Untuk saat ini, kita asumsikan bahwa gambar terbentuk sebagai hasil dari cahaya yang dipantulkan secara difus dari permukaan. Kemudian dalam bab ini, kami mengendurkan batasan ini pada scene yang secara langsung menggambarkan sumber cahaya dan sorotan. LuminanceLv dari setiap piksel kemudian didekati dengan produk berikut:

$$Lv(x, y) = r(x, y)\, Ev(x, y).$$

Di sini, r menunjukkan reektansi permukaan, dan Ev menunjukkan iluminasi. Subskrip v menunjukkan bahwa kita menggunakan besaran berbobot fotometrik. Atau, kita dapat menulis ekspresi ini dalam domain logaritmik (Oppenheim et al., 1968):

$$\begin{aligned} D(x, y) &= \log(Lv(x, y)) \\ &= \log(r(x, y)\, Ev(x, y)) \\ &= \log(r(x, y)) + \log(Ev(x, y)). \end{aligned}$$

Transparansi fotografi merekam gambar dengan memvariasikan kepadatan bahan. Dalam fotografi tradisional, variasi ini memiliki hubungan logaritmik dengan luminance. Jadi, dalam analogi dengan praktik umum dalam fotografi, kita akan menggunakan istilah representasi densitas (D) untuk log luminance. Ketika direpresentasikan dalam domain log, pantulan dan iluminasi menjadi aditif. Ini memfasilitasi pemisahan dua komponen ini, meskipun fakta bahwa mengisolasi baik pantulan atau penerangan adalah masalah yang kurang dibatasi. Dalam praktiknya, pemisahan hanya dimungkinkan pada tingkat tertentu dan tergantung pada komposisi gambar. Meskipun demikian, reproduksi nada dapat didasarkan pada pemisahan kedua komponen pembentukan gambar ini, seperti yang ditunjukkan pada dua bagian berikut.

21.5 OPERATOR BERBASIS FREKUENSI

Untuk scene difus yang khas, komponen pantulan cenderung menunjukkan frekuensi spasial yang tinggi karena permukaan bertekstur serta adanya tepi permukaan. Di sisi lain, iluminasi cenderung menjadi fungsi yang bervariasi perlahan di atas ruang.

Ketulusan adalah rentang dinamis rendah dan pencahayaan adalah rentang dinamis tinggi, kita dapat mencoba untuk memisahkan kedua komponen tersebut. Ketergantungan frekuensi dari pantulan dan penerangan memberikan solusi. Kita dapat, misalnya, menghitung transformasi Fourier dari suatu gambar dan hanya mengurangi frekuensi rendah. Ini memampatkan komponen penerangan sementara membiarkan komponen pemantulan sebagian besar tidak terpengaruh—operator reproduksi nada digital pertama yang kita kenal mengambil pendekatan ini (Oppenheimet al., 1968).

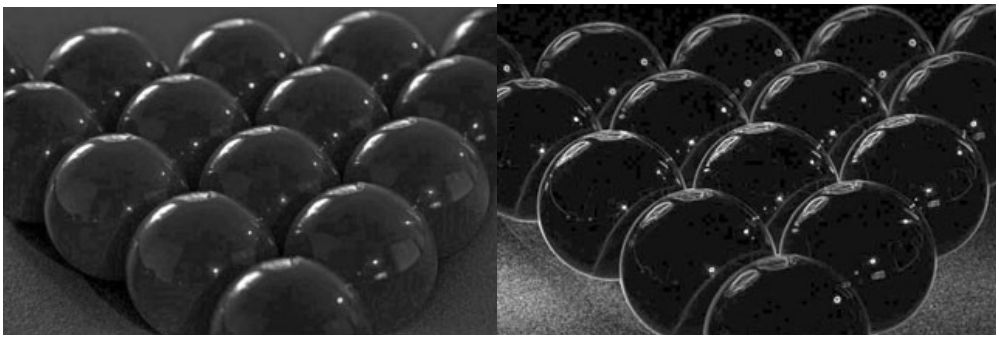
Gambar 21.14. Pemfilteran bilateral menghilangkan detail kecil tetapi mempertahankan gradien yang tajam (kiri). Lapisan detail terkait ditampilkan di sebelah kanan.

Baru-baru ini, operator lain juga mengikuti alur pemikiran ini. Secara khusus, filter bilateral dan trilateral digunakan untuk memisahkan gambar menjadi lapisan dasar dan lapisan detail (Durand & Dorsey, 2002; Choudhury & Tumblin, 2003). Kedua filter tersebut merupakan operator penghalus tepi yang dapat digunakan dalam berbagai cara yang berbeda. Menerapkan operator perataan tepi-pemelihara ke gambar kepadatan menghasilkan gambar kabur di mana tepi tajam tetap ada (Gambar 21.14 (kiri)). Kita mungkin melihat gambar seperti itu sebagai lapisan dasar. Jika kita kemudian membagi piksel gambar rentang dinamis tinggi dengan lapisan dasar, kita memperoleh lapisan detail yang berisi semua detail frekuensi tinggi (Gambar 21.14(kanan)).

Untuk scene difus, lapisan dasar dan detail mirip dengan representasi iluminansi dan pantulan. Untuk gambar yang menggambarkan sorotan dan sumber cahaya, paralel ini tidak berlaku. Namun, pemisahan gambar menjadi lapisan dasar dan detail dimungkinkan terlepas dari konten gambar. Dengan mengompresi lapisan dasar sebelum digabungkan kembali menjadi gambar dengan kerapatan terkompresi, gambar kerapatan rentang dinamis rendah dapat dibuat (Gambar 21.15). Setelah eksponensial, gambar yang dapat ditampilkan diperoleh.

Gambar 21.15. Nada gambar dipetakan menggunakan penyaringan bilateral. Lapisan dasar dan detail yang ditunjukkan pada Gambar 21.14 digabungkan kembali setelah mengompresi lapisan dasar.

Operator penghalusan pengawet tepi juga dapat digunakan untuk menghitung tingkat adaptasi lokal untuk setiap piksel, yang dapat digunakan dalam operator reproduksi nada yang bervariasi secara spasial atau lokal. Kami menjelaskan penggunaan filter bilateral dan trilateral ini di Bagian 21.7.



Gambar 21.16. Gambar di sebelah kiri (dipetakan dengan menggunakan kompresi domain gradien) menunjukkan scene dengan sorotan. Sorotan ini muncul sebagai gradien besar di

sebelah kanan, di mana besarnya gradien dipetakan ke skala abu-abu (hitam adalah gradien 0, putih adalah gradien maksimum dalam gambar).

21.6 OPERATOR DOMAIN-GRADIEN

Argumen yang dibuat untuk operator berbasis frekuensi di bagian sebelumnya juga berlaku untuk bidang gradien. Dengan asumsi bahwa tidak ada sumber cahaya yang tidak terlihat secara langsung, komponen pantulan akan menjadi fungsi konstan dengan lonjakan tajam di bidang gradien. Demikian pula, komponen iluminasi akan menyebabkan gradien kecil di mana-mana.

Manusia umumnya dapat memisahkan iluminasi dari pantulan dalam scene biasa. Persepsi pantulan permukaan setelah mengabaikan iluminan disebut lightness. Untuk menilai kecerahan gambar yang hanya menggambarkan permukaan difus, B. K. P. Horn adalah orang pertama yang memisahkan pantulan dan pencahayaan menggunakan bidang gradien (Horn, 1974). Dia menggunakan ambang batas sederhana untuk menghapus semua gradien kecil dan kemudian mengintegrasikan gambar, yang melibatkan penyelesaian persamaan Poisson menggunakan Metode Multigrid Penuh (Teukolsky, Vetterling, & Flannery, 1992).

Hasilnya mirip dengan filter pemulusan tepi-pemelihara. Ini sesuai dengan harapan karena operator berbasis frekuensi Oppenheim bekerja di bawah asumsi refleksi scene dan pembentukan gambar yang sama. Secara khusus, karya Horn secara langsung ditujukan pada “mini-worlds of Mondrians”, yang merupakan versi sederhana dari scene-scene difus yang menyerupai lukisan abstrak karya pelukis terkenal Belanda Piet Mondrian.

Pekerjaan Horn tidak dapat digunakan secara langsung sebagai operator reproduksi nada, karena sebagian besar gambar rentang dinamis tinggi menggambarkan sumber cahaya. Namun, variasi yang relatif kecil akan mengubah pekerjaan ini menjadi operator reproduksi nada yang sesuai. Jika sumber cahaya atau permukaan spekular digambarkan dalam gambar, maka gradien besar akan dikaitkan dengan tepi sumber cahaya dan sorotan. Ini menyebabkan gambar memiliki rentang dinamis yang tinggi. Sebuah contoh ditunjukkan pada Gambar 21.16, di mana sorotan pada bola snooker menyebabkan gradien yang tajam.

Oleh karena itu, kami dapat mengompresi gambar rentang dinamis tinggi dengan melemahkan gradien besar, daripada membatasi bidang gradien. Pendekatan ini diambil oleh Fattal et al. yang menunjukkan bahwa gambar rentang dinamis tinggi dapat berhasil dikompresi dengan mengintegrasikan bidang gradien terkompresi (Gambar 21.17) (Fattal et al., 2002). Kompresi domain gradien Fattal tidak terbatas pada scene yang menyebar.

Gambar 21.17. Gambar yang dipetakan nadanya menggunakan kompresi domain gradien.

21.7 OPERATOR SPASIAL

Pada bagian berikut, kita membahas operator reproduksi nada yang menerapkan kompresi langsung pada piksel tanpa transformasi ke domain lain. Seringkali operator global dan lokal dibedakan. Operator reproduksi nada di kelas sebelumnya mengubah nilai luminansi

setiap piksel sesuai dengan fungsi tekan yang sama untuk setiap piksel. Istilah global berasal dari fakta bahwa banyak fungsi seperti itu perlu dikaitkan dengan beberapa nilai yang ditentukan dengan menganalisis gambar penuh. Dalam praktiknya, sebagian besar operator menggunakan rata-rata geometrik Lv untuk mengarahkan kompresi:

$$L_v = \exp ¿¿$$

Dalam Persamaan (21.1), konstanta kecil diperkenalkan untuk mencegah rata-rata menjadi nol dengan adanya piksel hitam. Rata-rata geometrik biasanya dipetakan ke nilai tampilan yang telah ditentukan sebelumnya. Pengaruh mapping rata-rata geometrik terhadap tampilan nilai yang berbeda ditunjukkan pada Gambar 21.18. Atau, terkadang pencahayaan gambar minimum atau maksimum digunakan. Tantangan utama yang dihadapi dalam desain sebuah operator global terletak pada pilihan fungsi tekan.

Di sisi lain, operator lokal mengompresi setiap piksel sesuai dengan fungsi kompresi tertentu yang dimodulasi oleh informasi yang diperoleh dari pemilihan piksel tetangga, bukan gambar penuh. Alasannya adalah bahwa piksel terang di lingkungan yang terang dapat dipersepsikan secara berbeda dari piksel terang di lingkungan yang redup. Tantangan desain dalam pengembangan operator lokal melibatkan pemilihan fungsi tekan, ukuran lingkungan lokal untuk setiap piksel, dan cara nilai piksel lokal digunakan. Secara umum, operator lokal mencapai kompresi yang lebih baik daripada operator global (Gambar 21.19), meskipun dengan biaya komputasi yang lebih tinggi.

Gambar 21.18. Operator mapping nada spasial diterapkan setelah memetakan rata-rata geometrik ke nilai tampilan yang berbeda (kiri: 0,12, kanan: 0,38).

Gambar 21.19. Operator reproduksi nada global (kiri) dan operator reproduksi nada lokal (kanan) (Reinhard, Stark, Shirley, & Ferwerda, 2002) dari setiap gambar. Operator lokal menunjukkan lebih detail; misalnya, lencana logam di sebelah kanan menunjukkan kontras yang lebih baik dan sorotan lebih tajam

Baik operator global maupun lokal sering kali terinspirasi oleh sistem visual manusia. Sebagian besar operator menggunakan salah satu dari dua fungsi kompresi yang berbeda, yang ortogonal dengan perbedaan antara operator lokal dan global. Nilai tampilan $L_d(x, y)$ paling sering diturunkan dari pencahayaan gambarLv(x, y) dengan dua bentuk fungsi berikut:

$$L_d(x,y) = \frac{L_v(x,y)}{f(x,y)}$$

$$L_d(x,y) = \frac{L_v(x,y)}{L_v(x,y) f^n(x,y)}$$

Dalam persamaan-persamaan tersebut, f(x, y) dapat menjadi suatu fungsi konstanta yang bervariasi per piksel. Dalam kasus sebelumnya, kami memiliki operator global, sedangkan fungsi yang bervariasi secara spasial f(x, y) menghasilkan operator lokal. Eksponen n biasanya adalah konstanta yang ditetapkan untuk operator tertentu.

Persamaan (21.2) membagi luminansi setiap piksel dengan nilai yang diturunkan dari gambar penuh atau lingkungan lokal. Persamaan (21.3)hasan kurva berbentuk S pada plot log-linear dan disebut sigmoid karena alasan itu. Bentuk fungsional ini sesuai dengan data yang diperoleh dari pengukuran respons listrik fotoreseptor terhadap kilatan cahaya di berbagai spesies. Pada bagian berikut, kita membahas kedua bentuk fungsional.

21.8 DIVISI

Setiap piksel dapat dibagi dengan konstanta untuk menghadirkan gambar rentang dinamis tinggi dalam rentang yang dapat ditampilkan. Pembagian seperti itu pada dasarnya merupakan scalling linier, seperti yang ditunjukkan pada Gambar 21.3. Sementara Gambar 21.3 menunjukkan scalling linier ad-hoc, pendekatan ini dapat disempurnakan dengan menggunakan data psikofisik untuk menurunkan konstanta scalling f(x,y)=k dalam Persamaan (21.2) (GJ Ward, 1994; Ferwerda, Pattanaik, Shirley, & Greenberg ,1996).

Atau, ada beberapa pendekatan yang menghitung pembagi yang bervariasi secara spasial. Dalam setiap kasus ini, f(x, y) adalah versi buram dari gambar, yaitu f(x, y) = L^{blur}_v (x,y). Kekaburan dicapai dengan menggulung gambar dengan filter Gaussian (Chiu et al., 1993; Rahman, Jobson, & Woodell, 1996). Selain itu, perhitungan f(x, y) dengan mengaburkan gambar dapat dikombinasikan dengan shifting titik putih untuk tujuan pemodelan tampilan warna (Fairchild & Johnson, 2002; GM Johnson & Fairchild, 2003; Fairchild & Johnson, 2004).

Ukuran dan berat filter Gaussian memiliki dampak besar pada gambar yang dapat ditampilkan yang dihasilkan. Filter Gaussian memiliki efek memilih rata-rata lokal tertimbang. Reproduksi nada kemudian menjadi masalah membagi setiap piksel dengan rata-rata lokal tertimbang yang terkait. Jika ukuran kernel filter dipilih terlalu kecil, maka artefak haloing akan terjadi (Gambar 21.20(kiri)). Haloing adalah masalah umum dengan operator lokal dan terutama terlihat ketika mapping nada bergantung pada pembagian.

Gambar 21.20. Gambar dipetakan nada dengan membaginya dengan versi kabur Gaussian.

Ukuran kernel filter adalah 64 piksel untuk gambar kiri dan 512 piksel untuk gambar kanan. Algoritma berbasis divisi, artefak halo diminimalkan dengan memilih kernel filter besar.

Secara umum, artefak haloing dapat diminimalkan dalam pendekatan ini dengan membuat kernel filter menjadi besar (Gambar 21.20 (kanan)). Hasil yang masuk akal dapat diperoleh dengan memilih ukuran filter setidaknya seperempat dari gambar. Kadang-kadang bahkan kernel filter yang lebih besar diinginkan untuk meminimalkan artefak. Perhatikan, bahwa dalam batasnya, ukuran filter menjadi sebesar gambar itu sendiri. Dalam hal ini, operator lokal menjadi global, dan kompresi ekstra yang biasanya diberikan oleh pendekatan lokal hilang.

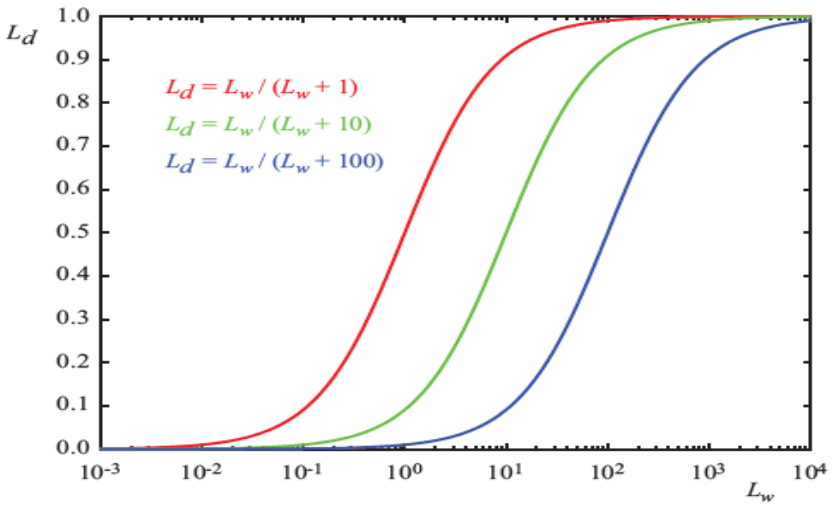
Bentuk fungsional di mana setiap piksel dibagi dengan piksel kabur Gaussian pada posisi spasial yang sama sehingga memerlukan pertukaran yang tidak diinginkan antara jumlah kompresi dan tingkat keparahan artefak.

21.9 SIGMOID

Persamaan (21.3) mengikuti bentuk fungsional yang berbeda dari pembagian sederhana, dan, oleh karena itu, memberikan tradeoff yang berbeda antara jumlah kompresi, keberadaan artefak, dan kecepatan komputasi.

Sigmoid memiliki beberapa sifat yang diinginkan. Untuk nilai luminansi yang sangat kecil, mappingnya mendekati linier, sehingga kontras dipertahankan di area gelap gambar. Fungsi tersebut memiliki asimtot pada satu, yang berarti bahwa output mapping selalu dibatasi antara 0 dan 1.

Dalam Persamaan (21.3), fungsi f(x,y) dapat dihitung sebagai konstanta global atau sebagai fungsi yang bervariasi secara spasial. Mengikuti praktik umum dalam elektrofisiologi, kita menyebut f(x,y) konstanta semi-jenuh. Nilainya menentukan nilai mana dalam gambar input yang terlihat optimal setelah mapping nada. Secara khusus, jika kita mengasumsikan bahwa eksponen n sama dengan 1, maka nilai luminansi yang sama dengan konstanta semi-saturasi akan dipetakan ke 0,5. Pengaruh pemilihan konstanta setengah jenuh yang berbeda ditunjukkan pada Gambar 21.21.



Gambar 21.21. Pilihan konstanta semi-saturasi menentukan bagaimana nilai input dipetakan untuk menampilkan nilai.

Fungsi f(x,y) dapat dihitung dalam beberapa cara yang berbeda (Reinhard et al., 2005). Dalam bentuknya yang paling sederhana, f(x,y) diatur ke Lv/k, sehingga rata-rata

geometrik dipetakan ke parameter pengguna k (Gambar 21.22) (Reinhard et al., 2002). Dalam hal ini, nilai awal yang baik untuk k adalah 0,18, meskipun untuk scene yang sangat terang atau gelap, nilai ini dapat dinaikkan atau diturunkan. Nilainya dapat diperkirakan dari gambar itu sendiri (Reinhard, 2003). Eksponen dalam Persamaan (21.3) mungkin menjadi 1.

Dalam pendekatan ini, konstanta semi-jenuh adalah fungsi dari rata-rata geometrik, dan karena itu operatornya adalah global. Variasi dari operator global ini menghitung konstanta semi-saturasi dengan interpolasi linier antara rata-rata geometrik dan pencahayaan setiap piksel:

$$f(x, y) = aL_v(x, y) + (1 - a)\, L_v.$$

Interpolasi diatur oleh parameter pengguna a yang memiliki efek memvariasikan jumlah kontras pada gambar yang dapat ditampilkan (Gambar 21.23) (Reinhard & Devlin, 2005). Lebih kontras berarti detail yang kurang terlihat di area terang dan gelap dan sebaliknya. Interpolasi ini dapat dilihat sebagai rumah tengah antara operator yang sepenuhnya global dan sepenuhnya lokal dengan melakukan interpolasi antara dua ekstrem tanpa menggunakan operasi blurring yang mahal.

Gambar 21.22. Gambar dengan skala linier (kiri) dan nada gambar dipetakan menggunakan kompresi sigmoid (kanan).

Gambar 21.23. Interpolasi linier memvariasikan kontras dalam gambar yang dipetakan nada. Parameter a diatur ke 0.0 di gambar kiri, dan ke 1.0 di gambar kanan.

Meskipun operator biasanya memampatkan nilai luminansi, operator khusus ini dapat diperluas untuk menyertakan bentuk sederhana dari adaptasi kromatik. Dengan demikian memberikan kesempatan untuk menyesuaikan tingkat kejenuhan yang biasanya terkait dengan mapping nada, seperti yang dibahas di awal bab ini.

Daripada hanya mengompresi saluran luminance, kompresi sigmoidal diterapkan pada masing-masing dari tiga saluran warna:

$$f(x, y) = aL_v(x, y) + (1 - a)\, L_v.$$

Perhitungan f(x, y) juga dimodifikasi untuk melakukan interpolasi bilinear antara luminansi rata-rata geometrik dan luminans piksel dan antara setiap saluran warna independen dan nilai luminansi piksel. Oleh karena itu, kami menghitung nilai luminansi rata-rata geometrik Lv, serta rata-rata geometrik saluran merah, hijau, dan biru ($\bar{}$ Ir, Ig, dan Ib). Dari nilai-nilai ini, kami menghitung (x, y) untuk setiap piksel dan untuk setiap saluran warna secara independen. Parameter interpolasia mengarahkan jumlah kontras sebelumnya, dan parameter interpolasi baru memungkinkan bentuk koreksi warna yang sederhana (Gambar 21.24).

Gambar 21.24. Interpolasi linier untuk koreksi warna. Parameter c diatur ke 0.0 di gambar kiri, dan ke 1.0 di gambar kanan.

Sejauh ini kita belum membahas nilai eksponen n dalam Persamaan (21.3). Studi dalam nilai laporan elektrofisiologi antara n = 0.2 dan n = 0 .9 (Hood, Finkelstein, & Buckingham, 1979). Sementara eksponen mungkin ditentukan pengguna, untuk berbagai macam gambar kami dapat memperkirakan nilai yang wajar dari rata-rata geometrik luminansi Lv dan luminansi minimum dan maksimum pada gambar (L_{min} dan L_{max}) dengan persamaan empiris berikut:

$$n = 0.3 + 0.7 \left(\frac{L_{max} - L_v}{L_{max} - L_{min}} \right)^{1.4}$$

Beberapa varian kompresi sigmoidal yang ditunjukkan sejauh ini semuanya bersifat global. Ini memiliki keuntungan bahwa mereka cepat untuk menghitung, dan mereka sangat cocok untuk gambar rentang dinamis menengah hingga tinggi. Untuk gambar rentang dinamis yang sangat tinggi, mungkin perlu menggunakan operator lokal, karena ini dapat memberikan beberapa kompresi ekstra. Metode langsung untuk memperluas kompresi sigmoidal menggantikan konstanta setengah jenuh global dengan fungsi yang bervariasi secara spasial, yang dapat dihitung dengan beberapa cara berbeda.

Dengan kata lain, fungsi f(x,y) sejauh ini diasumsikan konstan, tetapi juga dapat dihitung sebagai rata-rata terlokalisasi spasial. Mungkin cara paling sederhana untuk mencapai ini adalah dengan sekali lagi menggunakan gambar buram Gaussian. Setiap piksel dalam gambar kabur mewakili nilai rata-rata lokal yang dapat dilihat sebagai pilihan yang sesuai untuk konstanta semi-saturasi[14].

Seperti halnya operator berbasis divisi yang dibahas di bagian sebelumnya, kita harus mempertimbangkan artefak halo. Namun, ketika sebuah gambar dibagi dengan versi Gaussianblurred itu sendiri, ukuran kernel filter Gaussian harus besar untuk meminimalkan lingkaran cahaya. Jika sigmoid digunakan dengan konstanta semisaturasi varian spasial, kernel filter Gaussian perlu dibuat kecil untuk meminimalkan artefak. Ini adalah peningkatan yang signifikan, karena sejumlah kecil Gaussianblur mungkin secara efisien dihitung secara langsung dalam domain spasial. Dengan kata lain, tidak perlu melakukan transformasi Fourier

[14] Meskipun f(x,y) sekarang bukan lagi konstanta, kita tetap menyebutnya sebagai konstanta semi-jenuh

yang mahal. Dalam praktiknya, kernel filter dengan lebar hanya beberapa piksel sudah cukup untuk menekan artefak yang signifikan sementara pada saat yang sama menghasilkan lebih banyak kontras lokal pada gambar yang dipetakan nada.

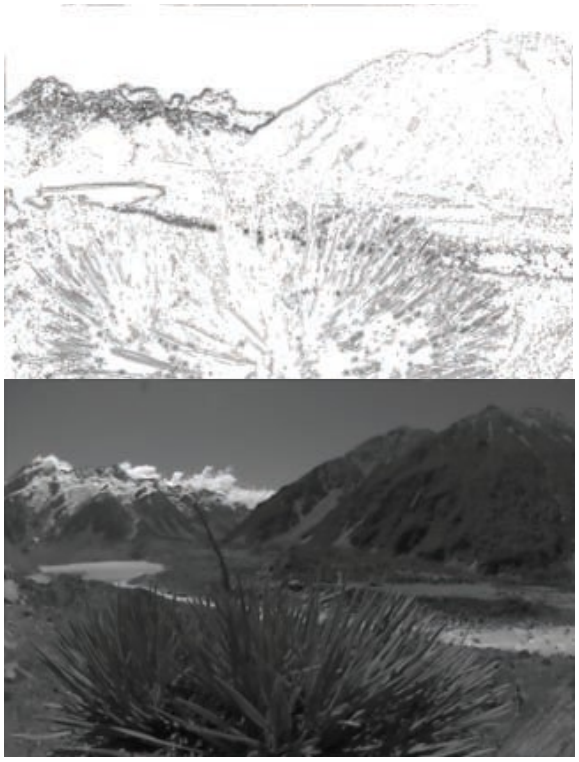
Salah satu masalah potensial dengan Gaussian blur adalah bahwa filter mengaburkan tepi kontras yang tajam dengan cara yang sama seperti mengaburkan detail kecil. Dalam praktiknya, jika ada gradien kontras yang besar di sekitar piksel yang dipertimbangkan, hal ini menyebabkan piksel yang diburamkan Gaussian menjadi berbeda secara signifikan dari piksel itu sendiri. Ini adalah penyebab langsung lingkaran cahaya. Dengan menggunakan kernel filter yang sangat besar dalam pendekatan berbasis divisi, kontras besar seperti itu dirata-ratakan.

Gambar 21.25. Contoh gambar yang digunakan untuk mendemonstrasikan mekanisme pemilihan skala ditunjukkan pada Gambar 21.26.

Dalam skema kompresi sigmoidal, filter Gaussian kecil meminimalkan kemungkinan tumpang tindih dengan gradien kontras yang tajam. Dalam hal ini, lingkaran cahaya masih terjadi, tetapi ukurannya sedemikian rupa sehingga biasanya tidak diperhatikan dan dianggap sebagai peningkat kontras.

Cara lain untuk mengaburkan gambar, sambil meminimalkan efek negatif dari langkah-langkah kontras besar yang hampir mendekati, adalah dengan menghindari blurring pada tepi tersebut. Cara yang sederhana, tetapi secara komputasi mahal, adalah dengan menghitung setumpuk gambar kabur Gaussian dengan ukuran kernel yang berbeda. Untuk setiap piksel, kita dapat memilih Gaussian terbesar yang tidak tumpang tindih dengan gradien yang signifikan.

Dalam lingkungan yang relatif seragam, nilai piksel kabur Gaussian harus sama terlepas dari ukuran kernel filter. Jadi, perbedaan antara piksel yang difilter dengan dua Gauss yang berbeda harus mendekati nol. Perbedaan ini hanya akan berubah secara signifikan jika kernel filter yang lebih luas tumpang tindih dengan lingkungan yang mengandung langkah kontras yang tajam, sedangkan kernel filter yang lebih kecil tidak.



Gambar 21.26. Mekanisme pemilihan skala: gambar kiri menunjukkan skala yang dipilih untuk setiap piksel gambar yang ditunjukkan pada Gambar 21.25; semakin gelap pikselnya, semakin kecil skalanya. Sebanyak delapan skala yang berbeda digunakan untuk menghitung gambar ini. Gambar kanan menunjukkan rata-rata lokal yang dihitung untuk setiap piksel berdasarkan mekanisme pemilihan lingkungan.

Oleh karena itu, dimungkinkan untuk menemukan lingkungan terbesar di sekitar piksel yang tidak mengandung tepi tajam dengan memeriksa perbedaan Gauss pada ukuran kernel yang berbeda. Untuk gambar yang ditunjukkan pada Gambar 21.25, skala yang dipilih untuk setiap piksel ditunjukkan pada Gambar 21.26 (kiri). Mekanisme pemilihan skala seperti itu digunakan oleh operator reproduksi nada fotografis (Reinhard et al., 2002) serta operator Ashikhmin (Ashikhmin, 2002).

Setelah lingkungan yang sesuai untuk setiap piksel diketahui, Lblur rata-rata Gaussianblurred untuk lingkungan ini (ditunjukkan di sebelah kanan Gambar 21.26) dapat digunakan untuk mengarahkan konstanta semi-saturasi, seperti misalnya yang digunakan oleh operator reproduksi nada fotografis.

Pendekatan alternatif, dan bisa dibilang lebih baik, adalah menggunakan operator penghalusan yang mempertahankan tepi, yang dirancang secara khusus untuk menghilangkan detail-detail kecil sambil menjaga kontras yang tajam. Beberapa filter seperti itu, seperti filter bilateral (Gambar 21.27), filter trilateral, filter Susan, algoritma LCIS dan algoritma shifting rata-rata cocok, meskipun beberapa di antaranya mahal untuk dihitung (Durand & Dorsey, 2002; Choudhury & Tumblin, 2003; Pattanaik & Yee, 2002; Tumblin & Turk, 1999; Comaniciu & Meer, 2002).

21.10 PENDEKATAN LAINNYA

Meskipun bagian sebelumnya membahas sebagian besar operator produksi batu bara, ada satu atau dua operator yang tidak langsung masuk ke dalam kategori di atas. Yang paling sederhana adalah variasi kompresi logaritmik, dan yang lainnya adalah pendekatan berbasis histogram.

Gambar 21.27. Kompresi sigmoidal (kiri) dan kompresi sigmoidal menggunakan penyaringan bilateral untuk menghitung konstanta semi-saturasi (kanan). Perhatikan peningkatan kontras di langit pada gambar yang tepat

Reduksi rentang dinamis dapat dicapai dengan mengambil logaritma, asalkan angka ini lebih besar dari 1. Setiap angka positif kemudian dapat diskalakan secara nonlinier antara 0 dan 1 menggunakan persamaan berikut:

$$L_d(x,y) = \frac{\log_b(1+L_v(x,y))}{\log_b(1+¿L_{max}❑)}$$

Sementara basis b dari logaritma di atas tidak ditentukan, pilihan basis apa pun dapat digunakan. Kebebasan untuk memilih basis logaritma ini dapat digunakan untuk memvariasikan basis dengan pencahayaan input, dan dengan demikian mencapai operator yang lebih cocok dengan gambar yang dikompresi (Drago, Myszkowski, Annen, & Chiba, 2003). Metode ini menggunakan fungsi bias Perlin dan Hoffert yang mengambil parameter pengguna p (Perlin & Hoffert, 1989):

$$\text{bias}_p(x) = x_{\log 10}(p)/\log_{10}(1/2).$$

Untuk parameter p pengguna, nilai awal sekitar 0,85 cenderung menghasilkan hasil yang masuk akal (Gambar 21,28 (kanan)).

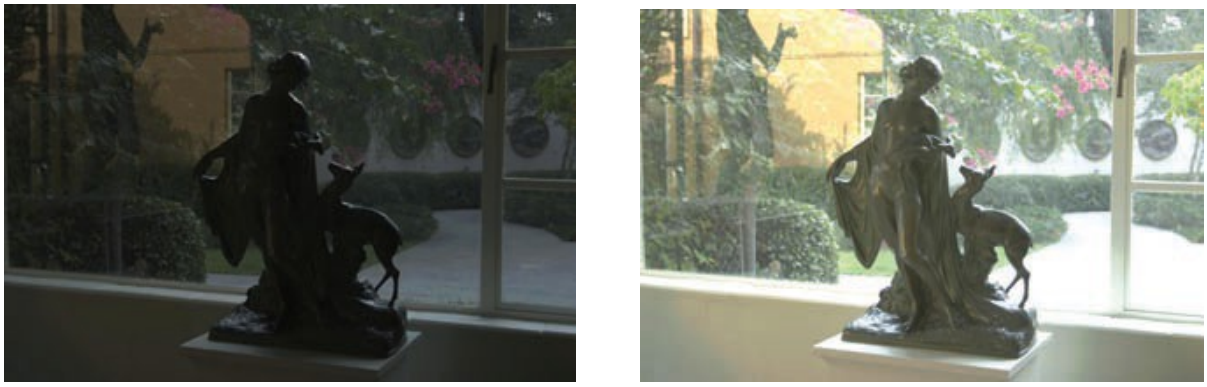
Gambar 21.28. Kompresi logaritma menggunakan logaritma basis 10 (kiri) dan kompresi logaritmik dengan basis bervariasi (kanan).

Atau, produksi nada dapat bebas dari pemerataan histogram. Penyetaraan histogram tradisional bertujuan untuk memberikan setiap nilai luminansi yang sama dengan peluang kemunculan dalam gambar output. GregWard menyempurnakan metode ini dengan cara yang mempertahankan kontras (Ward Larson, Rushmeier, & Piatko, 1997).

Pertama, ahistogram dihitung dari luminansi dalam gambar rentang dinamis tinggi. Dari histogram ini, histogram kumulatif dihitung sedemikian rupa sehingga setiap bin berisi jumlah piksel yang memiliki nilai luminance kurang dari atau sama dengan nilai luminance yang diwakili bin. Histogram kumulatif adalah fungsi yang meningkat secara monoton.

Memplot nilai di setiap nampan terhadap nilai luminansi yang diwakili oleh setiap nampan menghasilkan fungsi yang dapat dilihat sebagai fungsi mapping luminansi. Scalling fungsi ini, sehingga sumbu vertikal mencakup rentang perangkat tampilan, menghasilkan operator reproduksi nada. Teknik ini disebut pemerataan histogram.

Ward selanjutnya menyempurnakan metode ini dengan memastikan bahwa gradien fungsi ini tidak pernah melebihi 1. Ini berarti, bahwa jika perbedaan antara nilai tetangga dalam histogram kumulatif terlalu besar, perbedaan ini dijepit ke 1. Ini menghindari masalah bahwa perubahan kecil dalam luminansi pada input dapat menghasilkan perbedaan besar dalam gambar output. Dengan kata lain, dengan membatasi gradien histogram kumulatif menjadi 1, kontras tidak pernah dilebih-lebihkan. Algoritma yang dihasilkan disebut penyesuaian histogram (lihat Gambar 21.29).



Gambar 21.29. Gambar dengan skala linier (kiri) dan gambar yang disesuaikan dengan histogram (kanan). Gambar dibuat dengan izin dari museum Albin Polasek, Winter Park, Florida.

21.11 TONE MAPPING MALAM HARI

Operator reproduksi nada yang dibahas sejauh ini hampir semuanya berasumsi bahwa gambar mewakili scene di bawah kondisi tampilan fotopik, yaitu, seperti yang terlihat pada tingkat cahaya normal. Untuk scene skotopik, yaitu scene yang sangat gelap, sistem visual manusia menunjukkan perilaku yang sangat berbeda. Secara khusus, kontras yang dirasakan lebih rendah, ketajaman visual (yaitu, detail terkecil yang dapat kita bedakan) lebih rendah, dan semuanya memiliki tampilan yang agak biru.

Untuk memungkinkan gambar tersebut dilihat dengan benar pada monitor yang ditempatkan dalam kondisi pencahayaan fotopik, kami dapat memproses gambar terlebih dahulu sedemikian rupa sehingga tampak seolah-olah kami diadaptasi ke lingkungan tampilan yang sangat gelap. Preprocessing tersebut sering mengambil bentuk pengurangan kecerahan dan kontras, desaturasi gambar, shifting biru, dan pengurangan ketajaman visual (Thompson, Shirley, & Ferwerda, 2002).

Pendekatan tipikal dimulai dengan mengonversi gambar dari RGB ke XYZ. Kemudian, pencahayaan skotopik V dapat dihitung untuk setiap piksel:

$$V = Y\left[1.33\left(1+\frac{Y+Z}{X}\right)-1.68\right]$$

Gambar saluran tunggal ini kemudian dapat diskalakan dan dikalikan dengan abu-abu kebiruan yang dipilih secara empiris. Contohnya ditunjukkan pada Gambar 21.30. Jika beberapa piksel berada dalam rentang fotopik, maka gambar malam dapat dibuat dengan memadukan gambar abu-abu kebiruan secara linier dengan gambar input. Pecahan yang akan digunakan untuk setiap piksel bergantung pada V .

Gambar 21.30. Simulasi scene malam menggunakan gambar yang ditunjukkan pada Gambar 21.12.

Hilangnya ketajaman visual dapat dimodelkan dengan penyaringan low-pass pada gambar malam, meskipun hal ini akan memberikan kesan kabur yang salah. Pendekatan yang lebih baik adalah menerapkan filter bilateral untuk mempertahankan tepi tajam sambil mengaburkan detail yang lebih kecil (Tomasi & Manduchi, 1998).

Akhirnya, teknik transfer warna yang diuraikan dalam Bagian 21.3 juga dapat digunakan untuk mengubah gambar siang hari menjadi scene malam. Keefektifan pendekatan ini tergantung pada ketersediaan gambar malam yang cocok untuk mentransfer warna. Sebagai contoh, gambar pada Gambar 21.12 ditransformasikan menjadi gambar malam pada Gambar 21.31.

Gambar 21.31. Gambar di sebelah kiri digunakan untuk mengubah gambar Gambar 21.12 menjadi scene malam, yang ditunjukkan di sini di sebelah kanan.

21.12 DISKUSI

Karena algoritme iluminasi global secara alami menghasilkan gambar rentang dinamis tinggi, tampilan langsung dari gambar yang dihasilkan tidak dimungkinkan. Daripada menggunakan scalling linier atau penjepitan, operator produksi penebusan harus digunakan. Operator reproduksi nada apa pun lebih baik daripada menggunakan tidak satu pun reproduksi. Tergantung pada persyaratan aplikasi, salah satu dari beberapa operator mungkin cocok.

Misalnya, aplikasi rendering real-time mungkin harus menggunakan kompresi sigmoidal sederhana, karena ini cukup cepat untuk juga berjalan secara real time. Selain itu, kualitas visual mereka seringkali cukup baik. Teknik penyesuaian histogram (Ward Larson et al., 1997) mungkin juga cukup cepat untuk operasi waktu nyata.

Untuk scene yang mengandung rentang dinamis yang sangat tinggi, kompresi yang lebih baik dapat dicapai dengan operator lokal. Namun, biaya komputasi seringkali jauh lebih tinggi, sehingga operator ini hanya cocok untuk aplikasi noninteraktif. Di antara operator lokal tercepat adalah filter bilateral karena optimasi yang diberikan oleh teknik ini (Durand & Dorsey, 2002).

Filter ini menarik sebagai operator reproduksi nada dengan sendirinya, atau dapat digunakan untuk menghitung tingkat adaptasi lokal untuk digunakan dalam fungsi kompresi sigmoidal. Dalam kedua kasus, filter menghormati perubahan kontras yang tajam dan menghaluskan kontras yang lebih kecil. Ini adalah fitur penting yang membantu meminimalkan artefak halo, yang merupakan masalah umum dengan operator lokal.

Pendekatan alternatif untuk meminimalkan artefak halo adalah mekanisme pemilihan skala yang digunakan dalam operator reproduksi nada fotografi (Reinhardetal., 2002), meskipun teknik ini lebih lambat untuk dihitung.

Singkatnya, sementara sejumlah besar operator reproduksi nada saat ini tersedia, hanya ada sejumlah kecil pendekatan yang berbeda secara mendasar. Operator domain-fourier dan domain-gradien keduanya berakar pada pengetahuan tentang pembentukan citra. Operator domain spasial adalah varian spasial (lokal) atau sifat global. Operator ini biasanya didasarkan pada wawasan yang diperoleh dari mempelajari sistem visual manusia (dan sistem visual dari banyak spesies lainnya).

BAB 22

MODELLING IMPLISIT

Pemodelan implisit (juga dikenal sebagai permukaan implisit) dalam grafis komputer mencakup banyak metode yang berbeda untuk mendefinisikan model. Ini termasuk pemodelan resmi kerangka, permukaan offset, set level, permukaan variasi, dan permukaan aljabar. Dalam bab ini, kami membahas metode ini dan menjelaskan cara membangun model implisit kerangka secara lebih rinci. Kurva dapat ditentukan oleh persamaan implisit dari bentuk:

$$f(x, y) = 0.$$

Jika kita mempertimbangkan kurva tertutup, seperti lingkaran, dengan jari-jari r, maka persamaan implisit dapat ditulis sebagai:

Persamaan 22.1

$$f(x, y) = x^2 + y^2 - r^2 = 0.$$

Nilai f(x,y) dapat bernilai positif (di luar lingkaran), negatif (di dalam lingkaran), atau nol untuk titik-titik tepat pada lingkaran. Setara dalam tiga dimensi adalah permukaan tertutup di sekitar sekumpulan titik yang menempati volume atau wilayah ruang tertentu. Volume membentuk medan skalar, yaitu, kita dapat menghitung nilai untuk setiap titik dan seperti yang dapat dilihat untuk lingkaran, nilai negatif dibatasi oleh kurva atau permukaan implisit. Permukaan dapat divisualisasikan sebagai kontur di lapangan, menghubungkan titik-titik dengan nilai tertentu seperti nol (lihat Persamaan (22.1)). Menghitung permukaan seperti itu menyiratkan pencarian melalui ruang untuk menemukan titik-titik yang memenuhi persamaan implisit; metode ini tidak mungkin menghasilkan algoritma yang efisien untuk menggambar lingkaran (dan bahkan lebih kecil kemungkinannya dalam tiga dimensi). Ini mungkin alasan bahwa metode algoritma untuk pemodelan dengan kurva parametrik dan permukaan diselidiki sebelum metode implisit; namun, ada beberapa alasan bagus untuk mengembangkan algoritme untuk memvisualisasikan permukaan implisit. Dalam bab ini kami mengeksplorasi implikasi dari memperoleh data dari proses pemodelan daripada dari pemindai.

Meskipun overhead komputasi menemukan permukaan implisit, merancang dengan teknik pemodelan implisit menawarkan beberapa keuntungan dibandingkan metode pemodelan lainnya. Banyak operasi geometris disederhanakan menggunakan metode implisit termasuk:

- definisi campuran;
- operasi himpunan standar (penyatuan, persimpangan, perbedaan, dll.) dari geometri padat konstruktif (CSG);
- komposisi fungsional dengan fungsi implisit lainnya (misalnya, fungsi-R, campuran Barthe, campuran Ricci, dan warping);
- pengujian di dalam/luar, (misalnya, untuk deteksi tabrakan).

Memvisualisasikan permukaan dapat dilakukan baik dengan penelusuran sinar langsung menggunakan algoritma seperti yang dijelaskan dalam (Kalra & Barr, 1989; Mitchell, 1990; Hart & Baker, 1996; deGroot & Wyvill, 2005) atau dengan terlebih dahulu mengonversi ke poligon (Wyvill, McPheeters, & Wyvill, 1986).

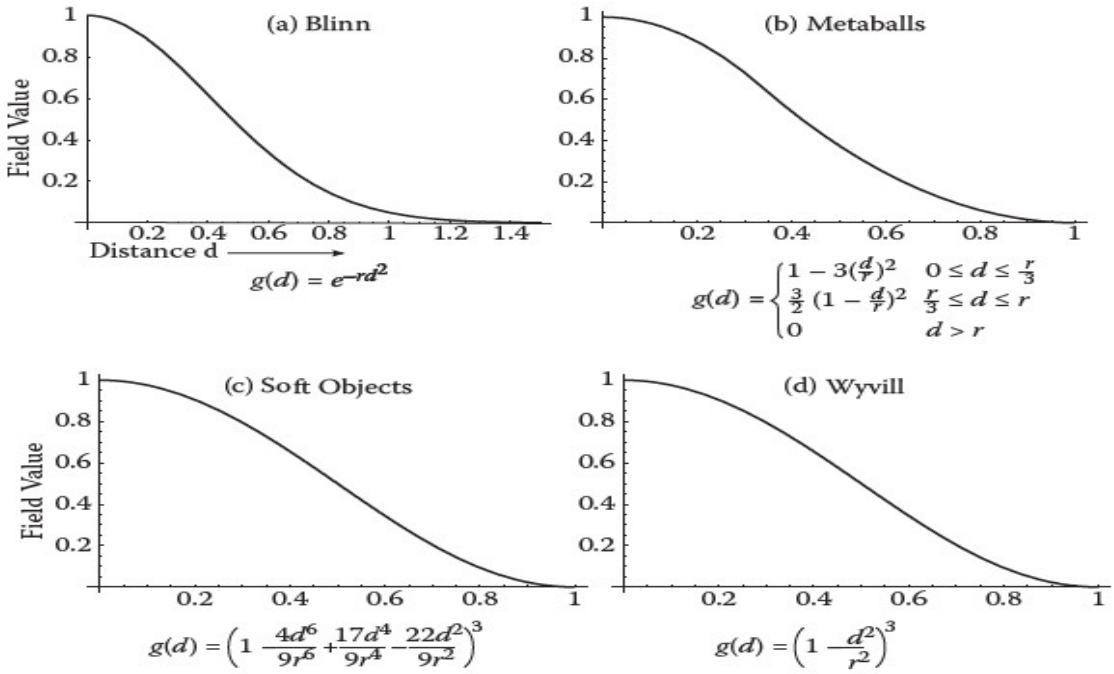
Gambar 22.1. Blinn's Blobby Man 1980. Gambar milik Jim Blinn.

Salah satu metode pertama diusulkan oleh Ricci sejak tahun 1973 (Ricci, 1973), yang juga memperkenalkan CSG dalam makalah yang sama. Algoritma Jim Blinn untuk menemukan kontur dalam medan kerapatan elektron, yang dikenal sebagai molekul Blobby (J. Blinn, 1982), Metaballs Nishimura (Nishimura et al., 1985) dan Soft Objects Wyvills (Wyvill et al., 1986) adalah contoh awal. metode pemodelan implisit. Blobby Man Jim Blinn (lihat Gambar 22.1) adalah rendering pertama dari model implisit nonaljabar.

22.1 FUNGSI IMPLISIT, KERANGKA PRIMITIF, DAN PENJUMLAHAN CAMPURAN

Dalam konteks pemodelan, fungsi implisit didefinisikan sebagai fungsi f yang diterapkan pada suatu titik$p \in E^3$ yang menghasilkan nilai skalar$\in \mathbb{R}$. Fungsi implisit $f_i(x, y, z)$ dapat dipecah menjadi fungsi jarak di(x, y, z) dan fungsi filter turunan[15] $g_i(r)$, di mana adalah jarak dari kerangka dan subskrip mengacu pada elemen kerangka ke-i.

Gambar 22.2. Fungsi filter jatuh (0 r 1). (a) fungsi Gaussian atau “Blobby” Blinn; (b) fungsi “Metaball” Nishimura; (c) fungsi "soft object" Wyvill dkk; (d) fungsi Wyvill.



Kami akan menggunakan notasi berikut:

[15] Fungsi-fungsi ini telah diberi banyak nama oleh para peneliti di masa lalu, misalnya, filter, potensial, basis radial, kernel, tetapi kami menggunakan filter fall-off sebagai istilah sederhana untuk menggambarkan penampilan mereka.

$$f_i(x, y, z) = g_i \circ d_i(x, y, z)$$

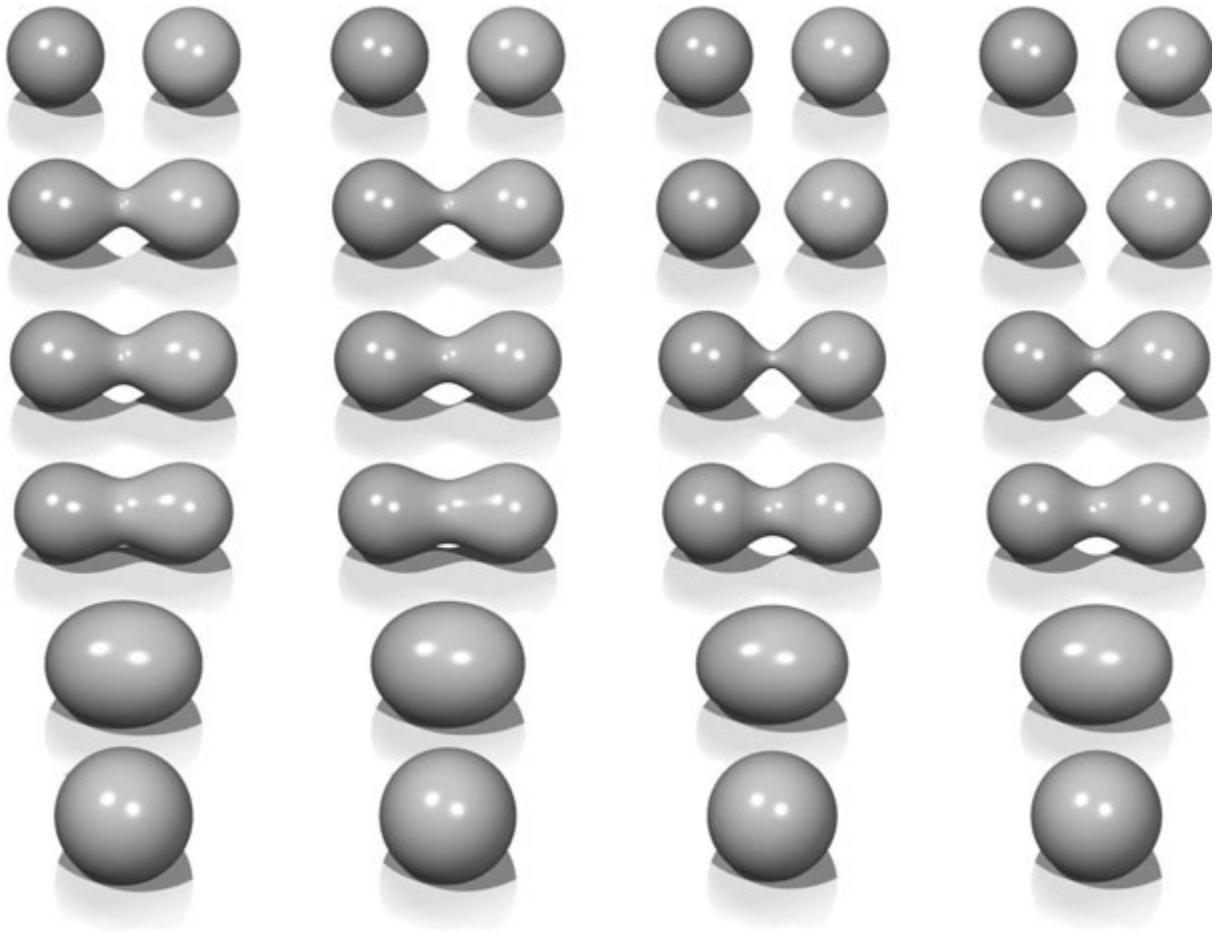
Contoh sederhananya adalah titik primitif, dan kita mengambil analogi sebuah bintang yang memancarkan panas ke luar angkasa. Nilai medan (suhu dalam contoh ini) dapat diukur pada sembarang titik p dan dapat ditemukan dengan mengambil jarak dari p ke pusat bintang dan memberikan nilai tersebut ke fungsi filter jatuh yang serupa dengan salah satu yang diberikan dalam Gambar 22.2. Dalam fungsi sampel ini, medan diberi nilai 1 di pusat bintang; nilai jatuh dengan jarak. Permukaan model dapat diturunkan dari fungsi implisit f(x, y, z) sebagai titik-titik ruang yang nilainya sama dengan beberapa nilai iso(iso) yang diinginkan;dalam contoh bintang, kulit bola untuk nilai iso $\in (0,1)$.

Secara umum, fungsi filter (g_i) dipilih sehingga nilai medan dimaksimalkan pada kerangka dan turun ke nol pada beberapa jarak yang dipilih dari kerangka. Dalam kasus sederhana di mana permukaan yang dihasilkan dicampur bersama, medan global f(x, y, z) dari suatu objek, fungsi implisit, dapat didefinisikan sebagai:

Persamaan 22.3

$$F(x,y,z)=\sum_{i=1}^{i=n} f_i(x,y,z)$$

di mana n elemen kerangka berkontribusi pada nilai medan yang dihasilkan. Sebuah contoh ditunjukkan pada Gambar 22.3 di mana medan pada setiap titik (x, y, z) dihitung seperti pada Persamaan (22.3).



Gambar 22.3. Setiap kolom menunjukkan dua titik primitif yang saling mendekat. Dari kiri ke kanan: fungsi fall-off filter yang digunakan adalah Blobby, Metaball, soft objects, dan Wyvill. Gambar milik Erwin DeGroot.

Dalam hal ini, dua titik primitif ditempatkan berdekatan. Saat dua titik disatukan, permukaannya menonjol dan kemudian menyatu. Istilah fungsi filter digunakan karena fungsi

menyebabkan primitif menjadi kabur bersama-sama agak mirip dengan fungsi filter untuk gambar. Penjumlahan campuran adalah operasi pencampuran yang paling kompak dan efisien yang dapat diterapkan pada permukaan implisit (lihat Persamaan (22.3)).

Salah satu keuntungan menggunakan fungsi filter dengan dukungan terbatas adalah bahwa primitif yang jauh dari p akan memiliki kontribusi nol dan dengan demikian tidak perlu dipertimbangkan (Wyvill et al., 1986).

22.2 KONTINUITAS C_1 DAN GRADIEN

Bentuk paling dasar dari kontinuitas adalah kontinuitas C^0, yang memastikan bahwa tidak ada “loncatan” dalam suatu fungsi. Kontinuitas orde tinggi didefinisikan dalam turunan fungsi (lihat Bab 15). Dalam kasus bidang skalar 3D, turunan pertama adalah fungsi vektor yang dikenal sebagai gradien, ditulis ∇f dan didefinisikan sebagai:

$$\nabla f(p) = ¿$$

Jika ∇f didefinisikan sebagai semua titik, dan turunan parsial tiga dimensi masing-masing adalah C^0, maka f adalah C^1. Secara informal, kontinuitas permukaan C^1 berarti bahwa normal permukaan bervariasi mulus di atas permukaan. Normal permukaan adalah vektor satuan yang tegak lurus permukaan. Jika tidak ada permukaan normal yang unik yang dapat didefinisikan pada tepi sebuah kubus, misalnya, maka permukaan tersebut bukan C^1. Untuk titik pada permukaan implisit, normal permukaan dapat dihitung dengan menormalkan vektor gradien ∇f. Dalam contoh lingkaran, titik-titik di dalam bernilai negatif dan yang di luar bernilai positif. Untuk banyak jenis permukaan implisit, pengertian di dalam dan di luar dibalik, dan karena vektor normal harus selalu menunjuk ke luar, itu bisa berlawanan dengan arah gradien.

Kerangka implisit primitif dibuat dengan menerapkan fungsi fall-off filter ke medan jarak tak bertanda seperti pada Persamaan (22.2). Meskipun medan jarak tidak pernah C1 pada kerangka, diskontinuitas ini dapat dihilangkan dengan menggunakan fungsi penurunan yang sesuai (Akleman & Chen, 1999). Jika sebuah operator, g, menggabungkan fungsi implisit, f_1 dan f_2, di mana semua titiknya adalah C^1, maka(f_1, f_2) belum tentu C^1. Misalnya, dimungkinkan untuk membuat persimpangan CSG yang tajam menggunakan operator min dan maks. Kombinasi ini tidak kontinu C^1 karena operator min dan maks tidak memiliki properti itu (lihat Bagian 22.5).

Analisis operator rumit oleh fakta bahwa kadang-kadang diinginkan untuk membuat diskontinuitas aC^1. Kasus ini terjadi setiap kali lipatan di permukaan diinginkan. Misalnya, kubus bukan C^1 karena diskontinuitas tangen terjadi di setiap sisi. Untuk membuat lipatan menggunakan primitif C_1, operator harus memasukkan diskontinuitas C^1, dan karenanya tidak bisa menjadi C^1 itu sendiri.

22.3 BIDANG JARAK, FUNGSI-R, DAN F-REPS

Medan jarak didefinisikan terhadap beberapa objek geometris T:

$$F(T, p) = \lim_{q \in T} ¿ q - p \vee ¿¿$$

Secara visual, $F(T, _p)$ adalah jarak terpendek dari p ke T. Oleh karena itu, ketika p terletak pada T, $F(T, _p) = 0$ dan permukaan yang dibuat oleh fungsi implisit adalah objek T. Di luar T,jarak noldikembalikan. FungsiT dapat berupa entitas geometri yang tertanam dalam 3D —titik, kurva, permukaan, atau benda padat. Pemodelan prosedural dengan bidang jarak dimulai dengan Ricci (Ricci, 1973); R-fungsi (Rvachev, 1963) pertama kali diterapkan untuk

membentuk pemodelan lebih dari 20 tahun kemudian (lihat (Saphiro, 1994) dan (A. Pasko, Adzhiev, Sourin, & Savchenko, 1995)).

Fungsi R atau fungsi Rvachev adalah fungsi yang tandanya dapat berubah jika dan hanya jika tanda dari salah satu argumennya berubah; yaitu, tandanya ditentukan semata-mata oleh argumennya. R-fungsi menyediakan kerangka teoritis yang kuat untuk komposisi boolean fungsi nyata, memungkinkan konstruksi operator Cn CSG (Saphiro, 1988). Operator CSG ini dapat digunakan untuk membuat operator pencampuran hanya dengan menambahkan offset tetap ke hasil (A. Pasko et al., 1995). Meskipun fungsi pencampuran ini tidak lagi secara teknis fungsi-R, mereka memiliki sebagian besar sifat yang diinginkan dan dapat dicampur secara bebas dengan fungsi-R untuk membuat model hierarki yang kompleks (Saphiro, 1988). Operator pencampuran dan CSG berbasis fungsi-R ini disebut sebagai operator-R (lihat Bagian 22.4). Sistem Hyperfun (Adzhiev et al., 1999) didasarkan pada F-reps (representasi fungsi), nama lain untuk permukaan implisit. Sistem ini menggunakan bahasa seperti C prosedural untuk menggambarkan banyak jenis permukaan implisit.

Set Tingkat

Ini berguna untuk mewakili medan implisit secara terpisah melalui grid biasa (Barthe, Mora, Dodgson, & Sabin, 2002) atau grid adaptif (Frisken, Perry, Rockwood, & Jones, 2000). Inilah yang dilakukan algoritma poligonisasi dalam kasus set level; selain itu, grid dapat digunakan untuk berbagai keperluan lain selain membangun poligon. Representasi diskrit dari f biasanya diperoleh dengan mengambil sampel fungsi kontinu pada interval reguler. Misalnya, fungsi sampel dapat didefinisikan oleh representasi model volume lainnya (V. V. Savchenko, Pasko, Sourin, & Kunii, 1998). Data juga dapat berupa objek fisik yang diambil sampelnya menggunakan teknik pencitraan tiga dimensi. Data volume diskrit paling sering digunakan bersama dengan metode set level (Osher & Sethian, 1988), yang menentukan cara untuk memodifikasi struktur data secara dinamis menggunakan fungsi kecepatan yang bergantung pada kurvatur. Lingkungan pemodelan interaktif berdasarkan set level telah didefinisikan (Museth, Breen, Whitaker, & Barr, 2002), meskipun set level hanya satu metode yang menggunakan representasi diskrit dari bidang implisit. Metode untuk mendefinisikan representasi diskrit secara interaktif menggunakan teknik permukaan implisit standar juga telah dieksplorasi (Baerentzen & Christensen, 2002).

Sebuah keuntungan utama untuk menggunakan struktur data diskrit adalah kemampuannya untuk bertindak sebagai pendekatan pemersatu untuk semua berbagai model volume yang didefinisikan oleh bidang potensial (diskrit atau tidak) (V. V. Savchenko et al., 1998). Konversi dari setiap fungsi kontinu ke representasi diskrit memperkenalkan masalah bagaimana merekonstruksi fungsi kontinu, yang diperlukan untuk tujuan gabungan operasi pemodelan tambahan dan visualisasi bidang potensial yang dihasilkan. Solusi terkenal untuk masalah ini adalah menerapkan filter g menggunakan operator konvolusi (lihat Bab 9). Pemilihan filter dipandu oleh properti yang diinginkan dari rekonstruksi, dan banyak filter telah dieksplorasi (Marschner & Lobb, 1994). Poin penting adalah bahwa biasanya ada tradeoff antara efisiensi filter yang dipilih dan kelancaran rekonstruksi yang dihasilkan; lihat juga Bagian 22.9.

Agar interaktif, sistem diskrit harus membatasi ukuran grid relatif terhadap daya komputasi yang tersedia. Ini, pada gilirannya, membatasi kemampuan pemodel untuk memasukkan detail frekuensi tinggi. Selain itu, filter triquadratic smoothing tidak memungkinkan untuk memasukkan tepi yang tajam, jika diinginkan. Sebuah solusi parsial untuk masalah ini adalah penggunaan grid adaptif, meskipun dengan representasi diskrit akan ada keterbatasan. Sebuah grid diskrit digunakan dalam (Schmidt, Wyvill, & Galin, 2005) untukactasacacherepresentingaBlobTreenode. Grid dalam pekerjaan ini digunakan untuk pembuatan prototipe cepat dan menggunakan interpolasi trilinear untuk posisi dan interpolasi trikuadrat yang lebih lambat dan lebih akurat untuk menghitung nilai gradien, karena mata lebih jeli dalam mengamati kesalahan gradien daripada kesalahan posisi.

22.4 PERMUKAAN IMPLISIT VARIASI

Seringkali diperlukan untuk mengubah data sampel menjadi representasi implisit. Permukaan implisit variasional interpolasi atau perkiraan himpunan titik menggunakan jumlah tertimbang dari fungsi basis yang didukung secara global (V. Savchenko, Pasko, Okunev, & Kunii, 1995;Turk&O'Brien,1999;Carretal.,2001;Turk&O'Brien,2002). Fungsi basis simetri radial ini diterapkan pada setiap titik sampel. Kontinuitas permukaan seperti itu tergantung pada pilihan fungsi basis. Spline pelat tipis C2 paling sering digunakan (Turk & O'Brien, 2002; Carr et al., 2001). Seperti fungsi eksponensial Blinn (lihat Gambar 22.2), fungsi ini tidak terbatas seperti permukaan implisit variasi yang dihasilkan.

Jika bidangnya adalah C2 secara global, lipatan tidak dapat ditentukan;2 namun, fungsi dasar anisotropik dapat digunakan untuk menghasilkan bidang yang berubah lebih cepat dan mungkin tampak memiliki lipatan (Dinh, Slabaugh, & Turk, 2001). Pada skala yang sesuai, permukaannya masih halus. Bidang halus menyiratkan bahwa perpotongan-diri tidak terjadi, dan karenanya volume selalu terdefinisi dengan baik. Spline pelat tipis menjamin bahwa kelengkungan global diminimalkan (Duchon, 1977). Interpolasi variasi memiliki banyak properti yang diinginkan untuk pemodelan 3D; namun, mengontrol permukaan yang dihasilkan bisa jadi sulit.

Permukaan implisit variasi juga dapat didasarkan pada fungsi basis radial yang didukung secara kompak (CS-RBFs) untuk mengurangi biaya komputasi teknik interpolasi variasi (Morse,Yoo,Rheingans,Chen,&Subramanian,2001). Setiap CS-RBF hanya mempengaruhi wilayah lokal, sehingga komputasi f(p) hanya memerlukan evaluasi fungsi dasar dalam beberapa lingkungan kecil dari p. Seperti rekanan yang didukung secara global, bidang yang dihasilkan adalah Ck, lipatan tidak didukung, dan persimpangan-diri tidak dapat terjadi.[16] Dukungan lokal dari setiap fungsi basis menghasilkan bidang global yang berlimpah. Ini juga menjamin bahwa kontur tambahan akan hadir, seperti yang dicatat oleh berbagai peneliti (Ohtake, Belyaev, & Pasko, 2003; Reuter, 2003).

Permukaan Konvolusi

Permukaan konvolusi, yang diperkenalkan oleh Bloomenthal dan Shoemake (Bloomenthal & Shoemake, 1991) dihasilkan dengan menggulung kerangka geometris S dengan fungsi kernel h. Oleh karena itu, nilai pada setiap posisi dalam ruang didefinisikan oleh integral di atas kerangka:

$$f(p) = \int_s g(r)\,h(p-r)\,dr$$

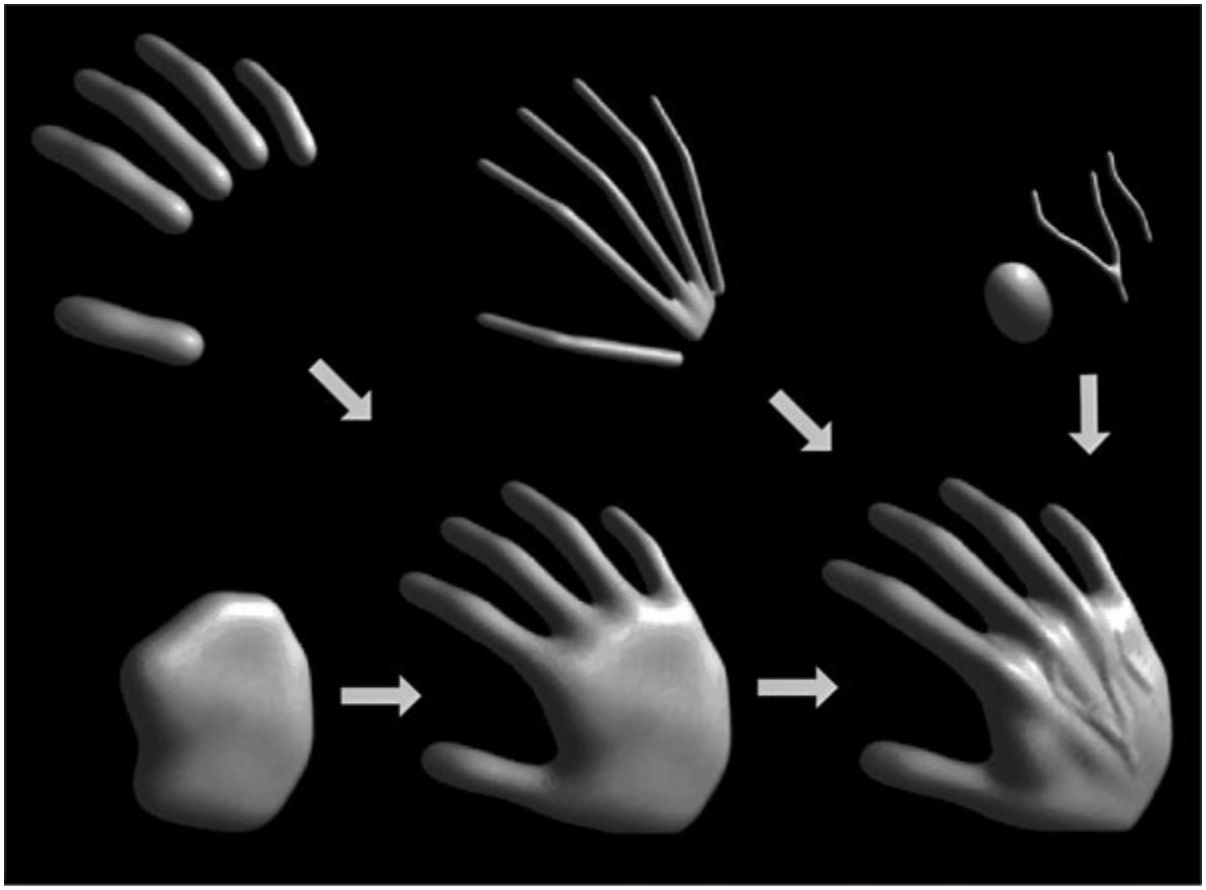
Setiap fungsi yang didukung secara terbatas dapat digunakan sebagai h; lihat (Sherstyuk, 1999) untuk analisis rinci kernel yang berbeda.

Seperti primitif kerangka, permukaan konvolusi memiliki bidang yang dibatasi. Blinn "Blobbymolecules" adalah bentuk paling sederhana dari permukaan konvolusi (J.Blinn, 1982); dalam hal ini, kerangka hanya terdiri dari titik-titik. Ide ini diperluas oleh Bloomenthal untuk memasukkan kerangka garis, busur, segitiga, dan poligon (Bloomenthal & Shoemake, 1991). Ini mewakili primitif 1D dan 2D; Primitif 3D kemudian dijelaskan oleh Bloomenthal (Bloomenthal, 1995).

[16] Perhatikan, k>0 tergantung pada RBF.

Gambar 22.4. Dua silinder campuran. Kiri: campuran penjumlahan; kanan: permukaan konvolusi dengan tonjolan yang hampir tidak terlihat (Bloomenthal, 1997). Gambar milik Erwin DeGroot.

Kombinasi permukaan konvolusi ditentukan oleh komposisi kerangka geometris yang mendasarinya dan memiliki keuntungan menghilangkan tonjolan yang cenderung terjadi saat menyusun beberapa kerangka dasar dengan pencampuran aditif. Permukaan yang dihasilkan dari konvolusi kerangka gabungan tidak memiliki tonjolan, seperti pada Gambar 22.4, dan bidangnya kontinu bahkan jika kerangka gabungan tidak cembung. Permukaan konvolusi diimbangi jarak tetap dari bagian cembung kerangka, tetapi menghasilkan fillet sepanjang bagian cekung kerangka. Contoh elemen rangka yang dirangkai untuk membangun model kompleks ditunjukkan pada Gambar 22.5. Model tangan berisi empat belas primitif.

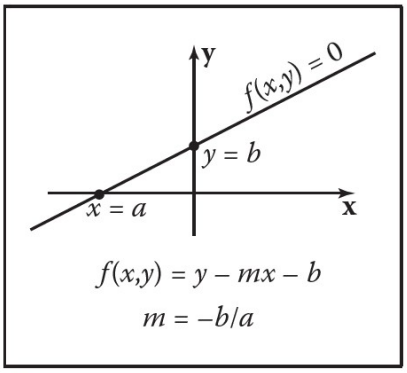


Gambar 22.5. Elemen rangka dililitkan untuk membangun dan model. ImagecourtesyJulesBloomenthal

22.5 MENDEFINISIKAN KERANGKA PRIMITIF

Seperti yang akan kita lihat di bagian berikut, rendering model implisit membutuhkan pencarian nilai medan dan gradien untuk sejumlah besar titik. Kita membutuhkan jarak untuk memasok ke Persamaan (22.2) dan gradien berguna untuk pencarian akar serta perhitungan pencahayaan. Menyediakan jarak ke fungsi filter jatuh pada Gambar 22.2 adalah masalah menghitung jarak terdekat ke primitif kerangka, sederhana untuk primitif titik tetapi sedikit lebih rumit untuk bentuk geometris yang lebih kompleks. Segmen garisprimitif (AB) dapat didefinisikan sebagai silinder di sekitar garis dengan ujung hemisferis (lihat Gambar 22.6). Titik P0 terletak pada permukaan di mana f(P0) = iso dan f(P1)=0karena terletak di luar pengaruh garis primitif. Jarak dari beberapa P_i ke garis ditemukan dengan hanya

memproyeksikan ke garis AB dan menghitung jarak tegak lurus, mis.,$|CP_0|$; ini dapat ditemukan dari AC, karena A, P_0, dan B, semuanya diketahui:



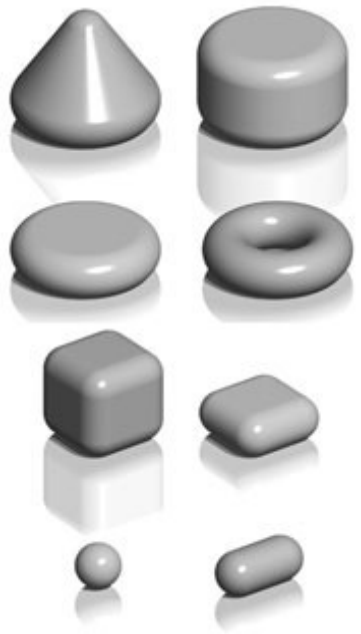
Gambar 22.6. Garis primitif ab dan titik contoh p_0, p_1, p_2 menunjukkan perhitungan jarak.

$$AC = AB\frac{AP_0 . AB}{¿|AB|\vee¿^2¿}$$

Pada Gambar 22.6, nilai medan dari $P_2 > 0$, karena P_2 berada di ujung hemisferis, yang dapat diperiksa secara terpisah. Variasi dari sisi ini dapat menentukan primitif dengan ujung jari-jari yang berbeda menghasilkan bentuk kerucut yang menarik. Contohnya ditunjukkan pada Gambar 22.7.

Gambar22.7. Silinder primitif dicampur dengan bola. Gambar milik Erwin DeGroot.

Berbagai macam kerangka geometris telah dideskripsikan, dan, pada prinsipnya, ini hanya untuk menentukan jarak kerangka dari suatu titik p dan juga gradien di p. Sebagai contoh, sebuah permukaan offset dari sebuah segitiga dapat didefinisikan dari simpul-simpul segitiga dan jari-jari r. Cara sederhana untuk menerapkan ini adalah dengan menggunakan primitif segmen garis untuk menggambarkan silinder pembatas yang menghubungkan simpul (jari-jari r). Jarak dari titik q dalam segitiga yang tidak jatuh di dalam bidang batas salah satu ruas garis primitif dikembalikan sebagai jarak tegak lurus terhadap bidang segitiga. Contoh lain termasuk cakram implisit, yang ditentukan oleh parameter lingkaran dan ketebalan, atorus juga ditentukan oleh lingkaran dan jari-jari penampang (atau jari-jari lingkaran dalam dan luar), kerucut melingkar dari piringan dan tinggi, kubus dengan sudut membulat, dll. (lihat Gambar 22.8) .



Gambar 22.8. Model implisit dari berbagai kerangka primitif. Gambar milik Erwin DeGroot.

Gambar 22.9. Model dinosaurus yang dilacak dengan sinar menunjukkan kerangka primitif yang mendasarinya. Gambar milik Erwin DeGroot.

22.6 RENDERING

Metode pemodelan, seperti permukaan parametrik, cocok untuk visualisasi, karena mudah untuk mengulang titik di permukaan yang dapat ditemukan langsung dari persamaan yang mendefinisikan; misalnya $(x, y) = (\cos\theta, \sin\theta)$, $\theta \in [0, 2\pi)$ menghasilkan lingkaran.

Ada dua teknik yang biasa digunakan untuk membuat permukaan implisit: raytracing dan surfacetiling. Dalam praktiknya, seorang desainer ingin memvisualisasikan model permukaan implisit dengan cepat, mengorbankan kualitas demi kecepatan untuk tujuan interaksi. Algoritma prototyping berkaitan dengan menghasilkan poligon mesh yang dapat dirender secara real time pada workstation modern. Menemukan jaring poligonal yang paling mendekati permukaan yang diinginkan disebut sebagai poligonisasi atau tiling permukaan. Untuk animasi atau untuk visualisasi akhir, di mana kualitas lebih disukai daripada kecepatan, ray tracing secara implisit permukaan langsung tanpa poligonisasi pertama menghasilkan hasil yang sangat baik

Seperti disebutkan sebelumnya, menemukan permukaan implisit membutuhkan pencarian melalui ruang untuk menemukan titik-titik yang memenuhi, $f(p)=0$. Ada dua pendekatan utama untuk melakukan pencarian seperti itu: partisi ruang—mempartisi ruang menjadi unit yang dapat dikelola seperti kubus, dan partisi non-ruang, misalnya, marchingtriangle (Hartmann, 1998; Akkouche & Galin, 2001) dan algoritma shrinkwrap (van Overveld & Wyvill, 2004).).

Dalam bab ini, kami menjelaskan algoritma partisi ruang asli dan menyerahkannya kepada pembaca untuk mengeksplorasi metode yang lebih maju. Algoritma ini bersama dengan postprocessing untuk perbaikan mesh (lihat Bab 12) dan caching menyediakan metode untuk tampilan interaktif model implisit pada workstation modern.

22.7 PEMBAGIAN TEMPAT

Pencarian Lengkap

Algoritma dasar partisi ruang kubik untuk tiling permukaan implisit pertama kali diterbitkan di (Wyvillet al., 1986) dan algoritma serupa yang berorientasi pada visualisasi volume, yang disebut marching cubes di (Lorensen & Cline, 1987). Sejak itu telah ada banyak penyempurnaan dan perluasan.

Pendekatan pertama untuk menemukan permukaan implisit mungkin dengan membagi ruang secara seragam menjadi kisi reguler sel kubik dan menghitung nilai untuk setiap titik. Setiap sel diganti dengan satu set poligon yang paling mendekati bagian permukaan yang terkandung di dalam sel itu. Masalah dengan metode ini adalah bahwa banyak sel akan sepenuhnya berada di luar atau sepenuhnya di dalam volume; dengan demikian, banyak sel yang tidak mengandung bagian permukaan yang diproses. Untuk grid data yang besar, ini bisa sangat memakan waktu dan memori.

Untuk menghindari penyimpanan seluruh grid, tabel hash digunakan untuk menyimpan hanya kubus yang berisi bagian dari permukaan, berdasarkan struktur data yang digunakan (Wyvill et al., 1986). Perangkat lunak yang berfungsi diterbitkan dalam Graphics Gems IV (Bloomenthal, 1990). Algoritma ini didasarkan pada kelanjutan numerik; dimulai dengan kubus benih yang memotong bagian permukaan dan membangun kubus tetangga seperlunya untuk mengikuti permukaan.

Algoritma memiliki dua bagian. Pada bagian pertama, ditemukan sel kubik yang berisi permukaan dan pada bagian kedua, setiap kubus diganti dengan segitiga. Bagian pertama dari algoritma didorong oleh antrian kubus, yang masing-masing berisi bagian dari permukaan; bagian kedua dari algoritma adalah table-driven.

Deskripsi Algoritma

Gambaran singkat dari algoritma ini adalah sebagai berikut:

- membagi ruang menjadi voxel kubik;
- mencari permukaan, mulai dari elemen kerangka;
- tambahkan voxel ke antrian, tandai dikunjungi;
- cari tetangga;
- setelah selesai, ganti voxel dengan poligon.

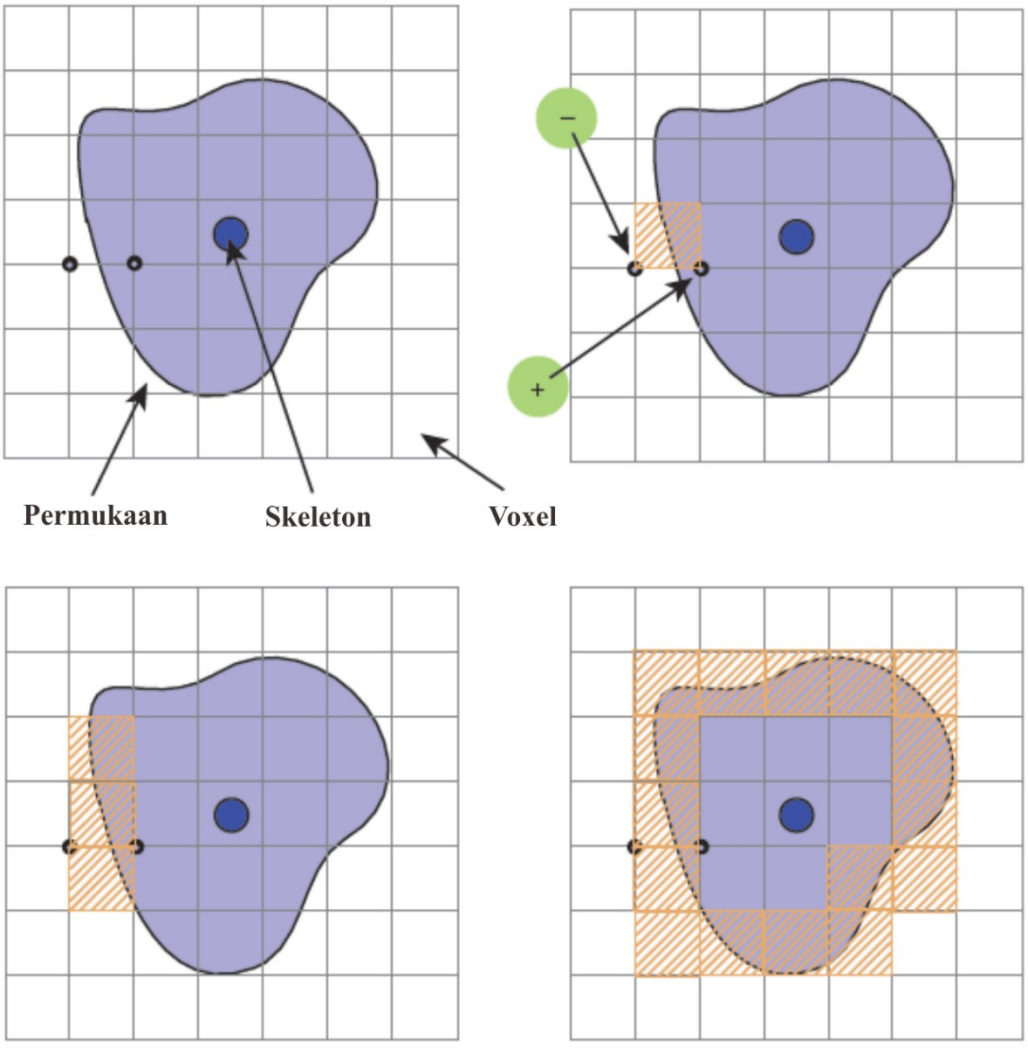
Pertama, ruang dibagi menjadi kisi kubik, dan selanjutnya adalah kubus biji yang berisi bagian permukaan. Sebuah titik kubus vi di dalam permukaan akan memiliki nilai medan $v_i >= $ iso dan sebuah titik di luar permukaan akan memiliki nilai medan vi < iso; dengan demikian, tepi dengan salah satu dari setiap jenis simpul akan berpotongan dengan permukaan. Kami menyebutnya tepi berpotongan. Nilai medan pada simpul kubus terdekat dengan primitif pertama dapat dievaluasi dengan menjumlahkan kontribusi primitif sesuai Persamaan (22.3), meskipun operator lain juga dapat digunakan seperti yang akan dilihat nanti. Kita akan mengasumsikan bahwa $f(v_0) >$ iso, yang menunjukkan bahwa v0 terletak di dalam zat padat. Nilai iso dipilih oleh pengguna; contohnya adalah iso = 0,5 saat menggunakan fungsi soft fall-off, yang memiliki beberapa sifat simetri yang menghasilkan pencampuran yang bagus (lihat Gambar 22.3). Simpul sepanjang satu sumbu dievaluasi secara

bergantian sampai nilai v_i < iso ditemukan. Kubus yang memuat sisi berpotongan adalah kubus biji.

Tetangga dari kubus benih diperiksa, dan yang mengandung setidaknya satu tepi berpotongan ditambahkan ke antrian yang siap untuk diproses. Untuk memproses kubus, kami memeriksa setiap wajah. Jika salah satu tepi pembatas memiliki simpul bertanda berlawanan, permukaan akan melewati wajah itu dan tetangga wajah harus diproses. Ketika proses ini telah selesai untuk semua wajah, fase kedua dari algoritma diterapkan ke kubus. Jika permukaan ditutup, akhirnya sebuah kubus akan ditinjau kembali dan tidak ada lagi tetangga yang tidak bertanda yang ditemukan, dan algoritma pencarian akan dihentikan. Memproses sebuah kubus melibatkan menandainya sebagai diproses dan memproses tetangganya yang tidak bertanda. Yang mengandung tepi berpotongan diproses sampai seluruh permukaan tertutup (lihat Gambar 22.10).

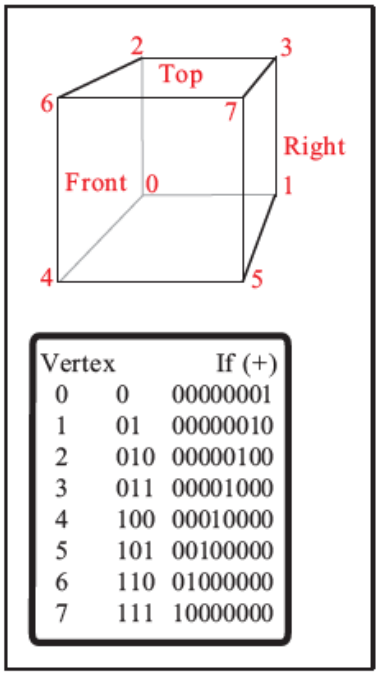
Setiap kubus diindeks oleh pengidentifikasi simpul yang kita tentukan menjadi sudut jauh kiri bawah (yaitu, simpul dengan nilai koordinat (x, y, z) terendah (lihat Gambar 22.11)). Untuk setiap simpul yang berada di dalam permukaan, bit yang sesuai akan diatur untuk membentuk alamat dalam tabel 8-bit (lihat Gambar 22.11 dan Bagian 22.3.3).

Titik pengidentifikasi dialamatkan oleh bilangan bulat i, j, k, dihitung dari (x, y, z)-koordinat lokasi kubus sedemikian rupa sehingga x = sisi∗i, dst., di mana adalah ukuran kubus. Titik pengidentifikasi setiap kubus dapat muncul sebanyak delapan kubus lainnya, dan akan tidak efisien untuk menyimpan titik-titik ini lebih dari satu kali. Dengan demikian, simpul disimpan secara unik dalam tabel hash berantai. Karena sebagian besar ruang tidak mengandung bagian permukaan, hanya kubus yang dikunjungi yang akan disimpan. Nilai fungsi implisit ditemukan untuk setiap simpul seperti yang disimpan dalam tabel hash.



Gambar 22.10. Sebuah bagian melalui kisi kubik. Tanda + menunjukkan titik di dalam permukaan (f ($v_i \geq$ iso) dan - di luar f ($v_i <$ iso).

Tidak ada yang diketahui tentang topologi permukaan sehingga pencarian harus dimulai dari setiap primitif untuk menghindari bagian permukaan yang tidak terhubung terlewatkan. Skalar dapat digunakan untuk mengukur pengaruh primitif. Jika skalar bisa kurang dari nol, maka dimungkinkan untuk mencari sepanjang sumbu tanpa menemukan sisi yang berpotongan. Dalam hal ini, pencarian yang lebih canggih harus dilakukan untuk menemukan kubus benih (Galin & Akkouche, 1999).



Vertex		If (+)
0	0	00000001
1	01	00000010
2	010	00000100
3	011	00001000
4	100	00010000
5	101	00100000
6	110	01000000
7	111	10000000

Gambar 22.11. Penomoran simpul.

Struktur data

Entri tabel hash memiliki lima nilai:

- indeks kisi i,j,k dari titik pengidentifikasi (lihat Gambar 22.11);
- f, nilai fungsi implisit dari titik pengidentifikasi;
- Boolean untuk menunjukkan apakah kubus ini telah dikunjungi.

Fungsi hash menghitung alamat dalam tabel hash dengan memilih beberapa bit dari masing-masing i,j,k dan menggabungkannya secara aritmatika. Misalnya, lima bit paling tidak signifikan menghasilkan alamat 15-bit untuk sebuah tabel, yang harus memiliki panjang 215. Fungsi hash seperti itu dapat diimplementasikan dengan rapi di C-preprocessor sebagai berikut:

```
#define NBITS 5
#define BMASK 037
#define HASH(a,b,c) (((a&BMASK)<<NBITS|b&BMASK)
<<NBITS|c&BMASK)
#define HSIZE 1<<NBITS*3
```

Antrian (daftar FIFO) digunakan sebagai penyimpanan sementara untuk mengidentifikasi tetangga untuk diproses. Algoritma dimulai dengan kubus benih yang ditandai sebagai dikunjungi dan ditempatkan pada antrian. Kubus pertama pada antrian di-dequeued dan semua tetangganya yang belum dikunjungi ditambahkan ke antrian. Setiap kubus diproses dan

diteruskan ke fase kedua dari algoritma jika berisi bagian dari permukaan. Antrian tersebut kemudian diproses hingga kosong.

22.8 ALGORITMA POLIGONISASI

Tahap kedua dari algoritma memperlakukan setiap kubus secara independen. Sel diganti dengan sekumpulan segitiga yang paling cocok dengan bentuk bagian permukaan yang melewati sel. Algoritma harus memutuskan bagaimana melakukan poligonisasi sel yang diberikan nilai fungsi implisit pada setiap simpul. Nilai-nilai ini akan positif atau negatif (yaitu, kurang dari atau lebih besar dari nilai iso), memberikan 256 kombinasi simpul positif atau negatif untuk delapan simpul kubus. Tabel 256 entri menyediakan simpul yang tepat untuk digunakan di setiap segitiga (Gambar 22.12). Misalnya, entri 4(000000100) menunjuk ke tabel kedua yang mencatat simpul yang mengikat tepi yang berpotongan. Dalam contoh ini, simpul nomor 2 berada di dalam permukaan (f(V 2) >= iso) dan, oleh karena itu, kami ingin menggambar segitiga yang menghubungkan titik-titik pada permukaan yang berpotongan dengan tepi yang dibatasi oleh (V 2,V0), (V 2,V3), dan (V 2,V6) seperti yang ditunjukkan pada Gambar 22.13.

Menemukan Persimpangan Kubus-Permukaan

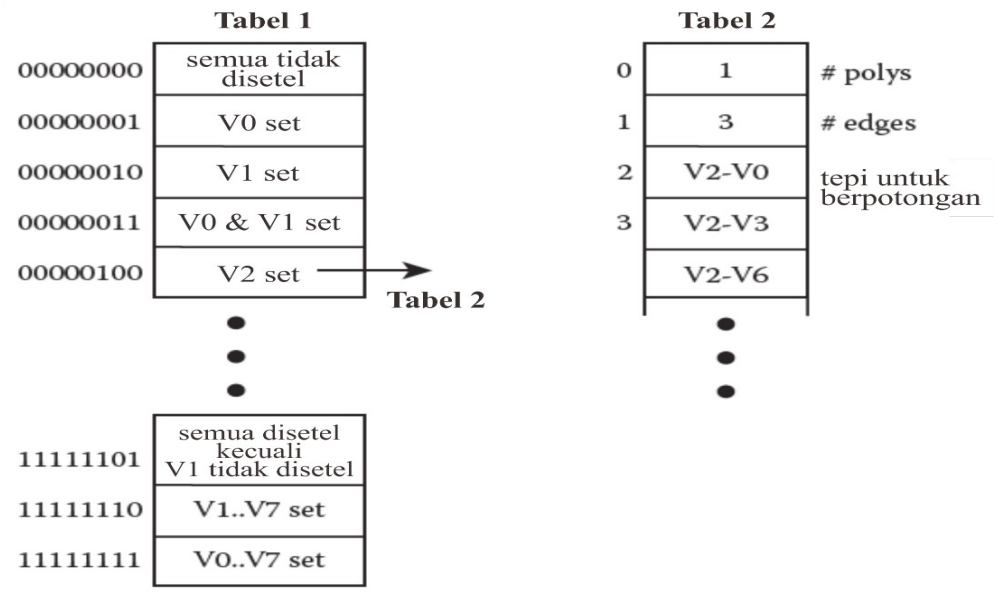
Gambar 22.13 menunjukkan sebuah kubus di mana simpul V2 berada di dalam permukaan dan semua simpul lainnya berada di luar. Persimpangan dengan permukaan terjadi pada tiga sisi seperti yang ditunjukkan. Permukaan memotong tepi V2 V6 di titik A. Cara tercepat, tetapi tidak akurat, untuk menghitung A adalah dengan menggunakan interpolasi linier:

$$\frac{f(A)-f(V¿¿2)}{f(V¿¿6)-f(V¿¿2)=¿A-V_2 \vee \frac{¿}{sisi}¿¿¿}¿$$

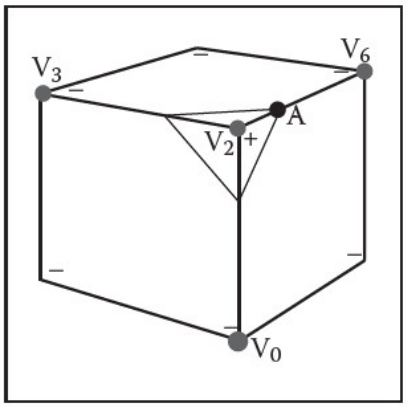
Jika sisi kubus adalah 1 dan nilai iso yang dicari untuk f(A) adalah 0,5, maka

$$A=V_3+\frac{0.5-f(V¿¿2)}{f(V¿¿6)-f(V¿¿2)¿¿}¿$$

Ini bekerja dengan baik untuk gambar statis, tetapi dalam kesalahan animasi, perbedaan antara bingkai akan sangat terlihat. Metode pencarian akar seperti regulafalsi harus diterapkan. Hal ini menjadi lebih komputasional karena gradien diperlukan untuk mengevaluasi titik potong. Gradien juga diperlukan pada titik permukaan untuk rendering. Untuk banyak jenis primitif, lebih mudah untuk menemukan pendekatan numerik menggunakan titik sampel di sekitar p, Nilai wajar untuk telah ditemukan secara empiris menjadi $0{,}01 * \text{sisi}$ di mana adalah panjang rusuk kubus.

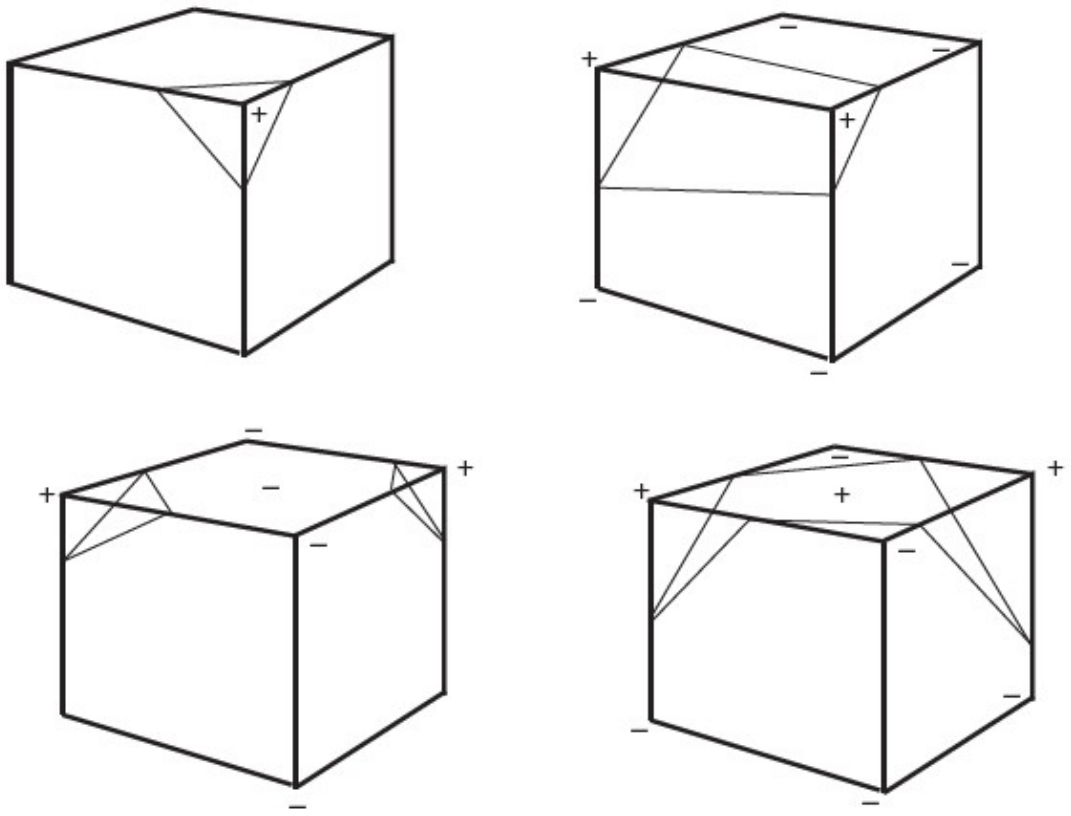


Gambar 22.12. Tabel 2 berisi tepi yang berpotongan dengan permukaan. Tabel 1 menunjukkan entri yang sesuai pada Tabel 2.



Gambar 22.13. Menemukan perpotongan permukaan dengan rusuk kubus.

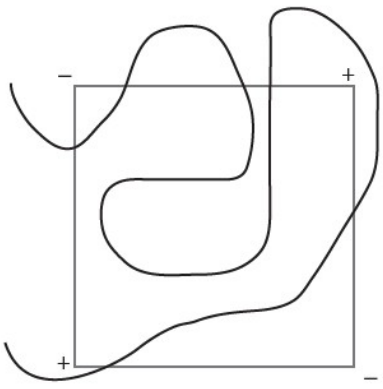
Untuk pembuatan mesh, sebagai lawan dari satu set segitiga independen, tabel hash kedua dapat mempertahankan penurunan tepi yang berpotongan. Karena setiap tepi kubus dibagi hingga empat tetangga, tabel hash tepi mencegah pengulangan perhitungan persimpangan tepi kubus permukaan. Alamat hash dapat diturunkan dari fungsi hash yang sama seperti untuk simpul (diterapkan ke titik akhir tepi).



Gambar 22.14. Contoh simpul di dalam (+) dan di luar (-) permukaan. Perhatikan sampel tambahan memberikan petunjuk untuk menghindari kasus ambigu

22.9 MASALAH PENGAMBILAN SAMPEL

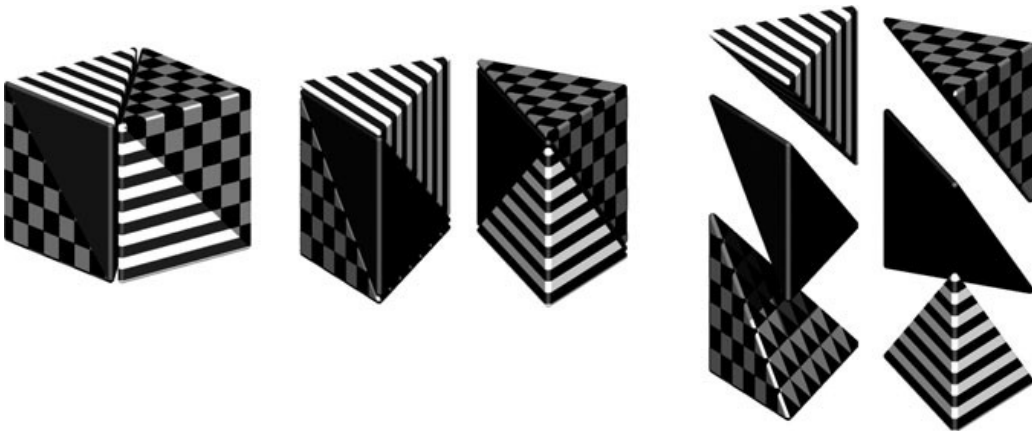
Ambiguitas terjadi ketika sudut yang berlawanan dari sebuah wajah (atau kubus) memiliki tanda yang sama dan pasangan simpul lainnya pada wajah memiliki tanda yang berlawanan (lihat Gambar 22.14). Sampel yang diambil di tengah wajah akan memberikan petunjuk apakah kubus mewakili pertemuan dua permukaan atau pelana. Harus diperjelas bahwa grid spasial menyimpan sampel fungsi implisit di setiap simpul. Jika fungsinya sangat bervariasi di dalam sel, representasi poligonal tidak akan menunjukkan variasi tersebut (lihat Gambar 22.15). Permukaan tidak dapat diselesaikan dengan pengambilan sampel saja kecuali ada sesuatu yang diketahui tentang kelengkungan permukaan. Diskusi yang baik tentang topik ini muncul di (Kalra & Barr, 1989).



Gambar 22.15. Kubus terlalu besar untuk menangkap variasi kecil dalam fungsi implisit.

Masalah ambiguitas ini (bukan masalah undersampling) dihindari dengan membagi sel kubik menjadi tetrahedra. Tetrahedra kemudian dapat dipoligonisasi dengan jelas. Karena ada empat simpul di setiap tetrahedron, tabel 16 entri akan memberikan informasi segitiga yang benar. Kerugiannya adalah sekitar dua kali jumlah poligon yang akan dihasilkan.

Membagi Kubus Tanpa memerlukan simpul sel tambahan, kubus dapat didekomposisi menjadi lima atau enam tetrahedra seperti yang ditunjukkan pada Gambar 22.16. Dekomposisi ini memperkenalkan diagonal pada permukaan kubus, dan untuk mempertahankan arah diagonal yang konsisten antara tetangga, enam dekomposisi lebih disukai. Pengenalan tepi diagonal menghasilkan resolusi permukaan yang lebih tinggi daripada mengganti setiap kubus langsung dengan segitiga. Dekomposisi menjadi tetrahedra dan penggantian tetrahedra dengan segitiga cepat, algoritma berbasis tabel, yang menghasilkan mesh yang konsisten secara topologi.



Gambar 22.16. Menguraikan kubus menjadi enam tetrahedra. Gambar milik Erwin DeGroot.

Poligonisasi Sel

Dua masalah nyata muncul dari penggunaan pembagian ruang yang seragam. Ukuran segitiga yang dihasilkan oleh algoritma ini tidak beradaptasi dengan kelengkungan permukaan dan sampel lebih lanjut diperlukan untuk memecahkan ambiguitas, di mana sel kubik diganti dengan poligon. Sebuah algoritma pembagian ruang berdasarkan oktre dikembangkan oleh Bloomenthal (Bloomenthal, 1988), yang beradaptasi dengan kelengkungan permukaan. Sel dibagi menjadi delapan oktan dan retak dihindari dengan menggunakan skema oktre terbatas, yaitu, sel tetangga tidak dapat berbeda lebih dari satu tingkat subdivisi. Ini memang mengurangi jumlah poligon yang dihasilkan, tetapi keuntungan penuh dari sel-sel besar hanya dapat diambil jika daerah-daerah datar pada permukaan jatuh seluruhnya dalam oktan yang sesuai. Algoritma terbukti dalam praktiknya menjadi jauh lebih lambat daripada algoritma seragam voxel dan lebih rumit untuk diimplementasikan.

Blending

Bagian 22.1 menunjukkan bahwa pencampuran dapat terjadi ketika nilai-nilai bidang dijumlahkan. Ricci, dalam makalahnya yang terkenal (Ricci, 1973), menjelaskan pencampuran super-elips. Mengingat dua fungsi F_A dan F_B, sebelumnya kami hanya menemukan nilai implisit sebagai $F_{total} = F_A + F_B$. Kita dapat menyatakan operator pencampuran yang lebih umum ini sebagai A B. Campuran Ricci didefinisikan sebagai:

Persamaan 22.4

$$f_{A\_B} = (f_A^n + f_B^n)^{1/2}$$

Selain itu, pencampuran umum ini bersifat asosiatif, yaitu f(A B) C = fA (B C). Blendingoperator+ standar terbukti menjadi kasus khusus dari campuran super-elips dengan n =1. Ketika n bervariasi dari 1 hingga tak terhingga, itu menciptakan satu set campuran

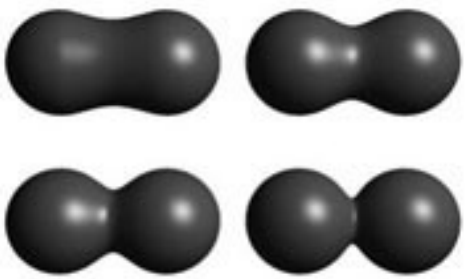
interpolasi antara pencampuran A + B dan gabungan A ∪ B (lihat Gambar 22.17). Gambar 22.27 menunjukkan node menjadi biner atau unary; sebenarnya node biner dapat dengan mudah diperluas menggunakan rumus di atas menjadi n-ary node.

Kekuatan operator Ricci adalah bahwa mereka tertutup di bawah operasi pada ruang dari semua volume implisit yang mungkin, yang berarti bahwa aplikasi operator hanya menghasilkan medan skalar lain yang mendefinisikan volume implisit lain. Bidang baru ini dapat dikomposisikan dengan bidang lain, sekali lagi menggunakan operator Ricci. Persamaan (22.4) akan selalu menghasilkan penyatuan yang tepat dari dua volume implisit, terlepas dari seberapa kompleksnya. Dibandingkan dengan kesulitan yang terlibat dalam menerapkan operasi boolean CSG ke permukaan B-rep, pemodelan padat dengan volume implisit sangat sederhana.

Mengikuti representasi fungsional Pasko (A. Pasko et al., 1995), fungsi pencampuran umum lainnya dapat didefinisikan:

$$f_{AoB} = ¿$$

Ketika $\alpha \in [-1,1]$ bervariasi dari −1 ke 1, itu menciptakan satu set campuran interpolasi serikat pekerja dan operator persimpangan. Namun, operator ini tidak lagi bersifat asosiatif yang tidak sesuai dengan definisi operator n-ary.



Gambar 22.17. Dengan memvariasikan n, campuran Ricci dapat dibuat untuk berubah dengan lancar dari campuran ke gabungan. Gambar milik Erwin DeGroot.

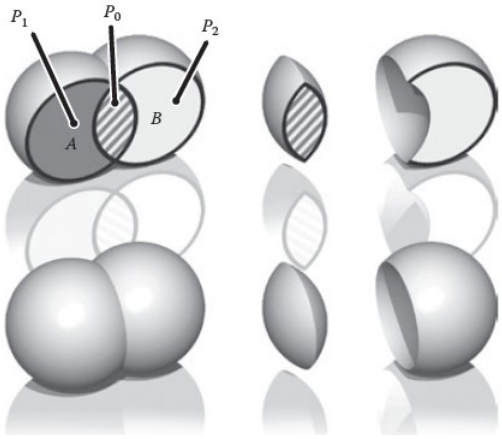
22.10 GEOMETRI SOLID KONSTRUKTIF

Model implisit sering disebut permukaan implisit; Namun, mereka secara inheren model volume dan berguna untuk operasi pemodelan yang solid. Ricci memperkenalkan geometri konstruktif untuk menentukan bentuk kompleks dari operasi seperti penyatuan, persimpangan, perbedaan, dan campuran primitif (Ricci, 1973). Permukaan dianggap sebagai batas antara setengah ruang f(p) < 1, yang mendefinisikan bagian dalam, dan f(p) > 1 yang mendefinisikan bagian luar. Pendekatan awal untuk pemodelan padat ini berkembang menjadi geometri padat konstruktif atau CSG (Ricci, 1973; Requicha, 1980). CSG biasanya dievaluasi dari bawah ke atas menurut pohon biner, dengan primitif polinomial derajat rendah sebagai simpul daun dan simpul internal yang mewakili operasi himpunan Boolean. Metode-metode ini mudah diadaptasi untuk digunakan dalam pemodelan implisit, dan dalam kasus permukaan implisit kerangka, operasi himpunan Boolean serikat max, persimpangan min dan perbedaan $\setminus_{\text{minmax}}$ didefinisikan sebagai berikut (Wyvill, Galin, & Guy, 1999):

Persamaan 22.5

$$\begin{aligned} \cup_{\max} \quad f &= \max_{i=0}^{k-1}(f_i), \\ \cap_{\min} \quad f &= \min_{i=0}^{k-1}(f_i), \\ \setminus_{\text{minmax}} \quad f &= \min\left(f_0, 2*\text{iso} - \max_{j=1}^{k-1}(f_j)\right). \end{aligned}$$

Operator Ricci diilustrasikan pada Gambar 22.18 untuk titik primitif A dan B. Untuk union (kiri bawah) medan di semua titik di dalam union akan lebih besar dari $f_A()$ dan $f_B()$. Untuk perpotongan (pusat), titik-titik pada daerah bertanda P1 akan memiliki nilai $\min(f_A(P_1),f_B(P_1)) = 0$, karena kontribusi B akan menjadi nol di luar jangkauan pengaruhnya. Demikian pula, untuk daerah yang ditandai sebagai P_2, (pengaruh A adalah nol, yaitu minimum) hanya menyisakan daerah persimpangan dengan nilai positif. Perbedaan bekerja dengan cara yang sama menggunakan nilai iso di tiga wilayah yang ditandai (P_i) sebagai berikut:

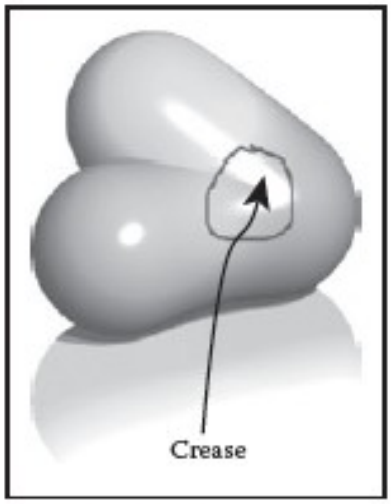


Gambar 22.18. Operator Ricci untuk CSG. Gambar milik Erwin DeGroot.

Persamaan 22.6

$$
\begin{aligned}
f(P_0) &= \min(f_B(P_0),\ 2 * \text{iso} - f_A(P_0)) \\
&= \min([\text{iso},\ 1],\ [2 * \text{iso} - 1,\ \text{iso}]) \\
&= [2*\text{iso} - 1,\ \text{iso}] < \text{iso} \\
f(P_1) &= \min(f_B(P_1),\ 2 * \text{iso} - f_A(P_1)) \\
&= \min([0,\ \text{iso}],\ [2 * \text{iso} - 1,\ \text{iso}]) < \text{iso} \\
f(P_2) &= \min(f_B(P_2),\ 2* \text{iso} - f_A(P_2)) \\
&= \min([\text{iso},\ 1],\ [\text{iso},\ 2 * \text{iso}]) >= \text{iso}
\end{aligned}
$$

CSGoperator membuat lipatan, yaitu diskontinuitas C1. Misalnya, operator themin() (Persamaan (22.5)) membuat diskontinuitas C1 di semua titik di mana f1(p)= f2(p). Ketika diterapkan pada dua bola, diskontinuitas yang dihasilkan oleh operator gabungan ini menghasilkan lipatan pada permukaan, seperti yang ditunjukkan pada Gambar 22.18, yang merupakan hasil yang diinginkan. Sayangnya, diskontinuitas meluas ke bidang di luar permukaan, yang tidak terlihat dalam gambar ini. Jika campuran kemudian diterapkan pada hasil penyatuan, bidang diskontinu C1 di lapangan menghasilkan diskontinuitas naungan (Gambar 22.19).



Gambar 22.19. Dua titik primitif di sebelah kiri dihubungkan oleh serikat Ricci. Primitif ketiga dicampur dengan hasilnya, menciptakan lipatan yang tidak diinginkan di bidang. Gambar milik Erwin DeGroot.

Masalah dapat dihindari sampai batas tertentu (G. Pasko, Pasko, Ikeda, & Kunii, 2002), dan operator CSG telah dikembangkan yang C1 di semua titik kecuali di mana f1(p)=f2(p) = iso (Barthe, Dodgson, Sabin, Wyvill, & Gaildrat, 2003).

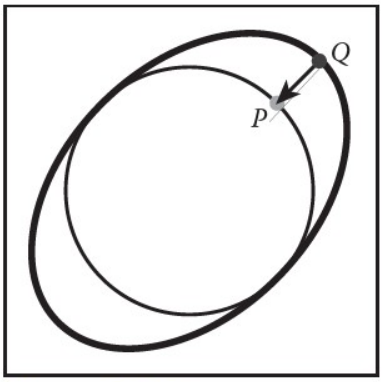
Warping

Kemampuan untuk mendistorsi bentuk permukaan dengan membelokkan ruang di sekitarnya adalah alat pemodelan yang berguna. Warp adalah fungsi kontinu w(x, y, z) yang memetakan $\mathbb{R}^3$ ke $\mathbb{R}^3$. Sederberg memberikan analogi yang baik untuk warping ketika menggambarkan deformasi bentuk bebas (Sederberg & Parry, 1986). Dia menyarankan bahwa ruang melengkung dapat disamakan dengan paralelepiped plastik yang jelas, fleksibel, di mana objek yang akan dilengkungkan tertanam. Sebuah elemen melengkung dapat didefinisikan dengan hanya menerapkan beberapa fungsi melengkung w(p) ke persamaan implisit:

Persamaan 22.6

$$f_i(x, y, z) = g_i \circ d_i \circ w_i(x, y, z).$$

Sebuah elemen melengkung mungkin sepenuhnya dicirikan oleh jarak ke kerangkanya di(x, y, z), fungsi filter jatuhnya gi(r), dan akhirnya fungsi lengkungannya wi(x, y, z). Untuk merender atau melakukan operasi pada permukaan implisit, nilai implisit dari banyak titik f(P) harus ditemukan. Pertama, P ditransformasikan oleh fungsi warp ke beberapa titik baru Q, dan f(Q) dikembalikan menggantikan f(P). Pada Gambar 22.20, alih-alih mengembalikan nilai implisit dari beberapa titik f(Q), nilai untuk f(P)dikembalikan. Dalam hal ini, nilai iso dikembalikan dan permukaan implisit (kurva dalam 2D) melewati Q dan bukan P. Jadi, lingkaran dilengkungkan menjadi elips.



Gambar 22.20. Titik Q mengembalikan nilai bidang untuk titik P.

Barr memperkenalkan gagasan deformasi global dan lokal menggunakan operasi puntiran, lancip, dan tekukan yang diterapkan pada permukaan parametrik (Barr, 1984). Deformasi dapat disarangkan untuk menghasilkan model seperti yang ditunjukkan pada Gambar 22.27. Secara konseptual, ini mudah diterapkan pada permukaan implisit, seperti yang ditunjukkan dalam Persamaan (22.6).

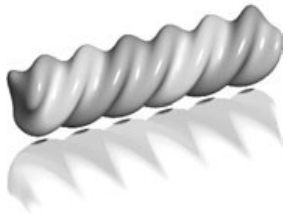
Perhatikan bahwa normal tidak dapat dihitung dengan cara yang mirip dengan melengkungkan suatu titik. Masalah ini mirip dengan masalah yang diuraikan dalam Bagian 13.2 tentang pembuatan contoh. Dalam hal ini, normal paling mudah dapat didekati dengan

menggunakan Persamaan (22.3.3) meskipun penggunaan Jacobian, seperti yang disarankan (Barr, 1984), memberikan hasil yang tepat. Warps Barr dijelaskan di bagian berikut.

Memutar

Dalam contoh ini, puntiran di sekitar sumbu-z sebesar (lihat Gambar 22.21) untuk tiga silinder implisit campuran dengan lilitan puntir diterapkan padanya. Putaran di sekitar z dinyatakan sebagai

$$w(x,y,z)=\begin{cases} x*\sin(\theta(z))-y*\sin(\theta(z)) \\ x*\sin(\theta(z))+y*\cos(\theta(z)) \\ z \end{cases}$$



Gambar 22.21. Tiga silinder implisit campuran dipilin menjadi satu. Gambar milik Erwin DeGroot.

Gambar 22.22. Tiga silinder implisit campuran, dipelintir lalu diruncingkan. Gambar milik Erwin DeGroot.

Lancip

Taper diterapkan di sepanjang satu sumbu utama. Sebuah lancip linier telah terbukti menjadi yang paling berguna meskipun lancip kuadrat dan kubik mudah diterapkan. Misalnya, lancip linier sepanjang sumbu y melibatkan perubahan koordinat x dan z. (Lihat Gambar 22.22.) Skala linier diterapkan pada y antara ymax dan ymin:

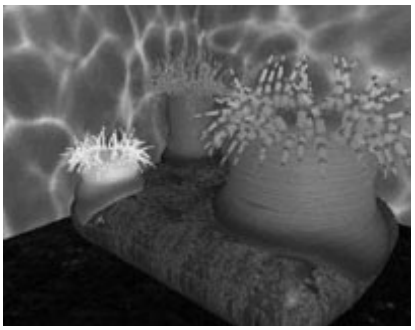
$$s(y)=¿\frac{y_{max}-y}{y_{max}-y_{min}}w(x,y,z)\begin{Bmatrix} s(y)x \\ y \\ s(y)z \end{Bmatrix}$$

Bengkok

Tekuk juga diterapkan di sepanjang satu sumbu utama. (Lihat Gambar 22.23.) Untuk contoh tikungan di bawah ini, laju pembengkokan adalah k diukur dalam radian per satuan panjang, sumbu tikungan adalah (x0,1/k), dan sudut didefinisikan sebagai (x x0) k. Belokan di sekitar z adalah

$$w(x,y,z)=\begin{cases} -\sin(\theta)\wedge(y-\frac{1}{k})+x0 \\ \cos(\theta)*\left(y-\frac{1}{k}\right)+\frac{1}{k} \end{cases}$$

Gambar 22.23. Tiga silinder implisit campuran, dipelintir bersama, diruncingkan dan ditekuk. Gambar milik Erwin DeGroot.



Gambar 22.24. Anemon laut berubah bentuk menjadi batuan implisit. Gambar milik Mai Nur dan X. Liang.

22.11 MODELLING KONTAK YANG TEPAT

Pemodelan kontak yang tepat (PCM) adalah metode deforming primitif permukaan implisit dalam situasi kontak sambil mempertahankan permukaan kontak yang tepat dengan kontinuitas C1 (Gascuel, 1993). PCM penting karena merupakan cara sederhana dan otomatis untuk menunjukkan bagaimana model dapat bereaksi terhadap lingkungannya. Ini tidak dapat dengan mudah dilakukan dengan metode non-implisit (lihat Gambar 22.24).

PCM diimplementasikan dengan memasukkan fungsi deformasi (p) yang memodifikasi nilai bidang yang dikembalikan untuk setiap titik. Untuk setiap pasangan objek, tumbukan pertama kali dideteksi menggunakan uji kotak pembatas. Setelah dipastikan bahwa tabrakan mungkin terjadi, PCM diterapkan. Suku deformasi geometrik lokal si dihitung dan ditambahkan ke fungsi implisit f_i. Volume benda yang bertabrakan dibagi menjadi daerah interpenetrasi dan daerah deformasi. Hasil penerapan si adalah bahwa daerah interpenetrasi dikompresi sehingga kontak dipertahankan tanpa terjadi interpenetrasi (lihat Gambar 22.25). Efek s_i dilemahkan menjadi nol di dalam daerah propagasi sehingga volume di luar kedua daerah tersebut tidak terdeformasi.

Diberikan dua elemen kerangka yang menghasilkan medan $f_1(p)$ dan $f_2(p)$, permukaan di sekitar masing-masing dihitung sebagai

$$f1(p) + s1(p) = 0,$$
$$f2(p) + s2(p) = 0.$$

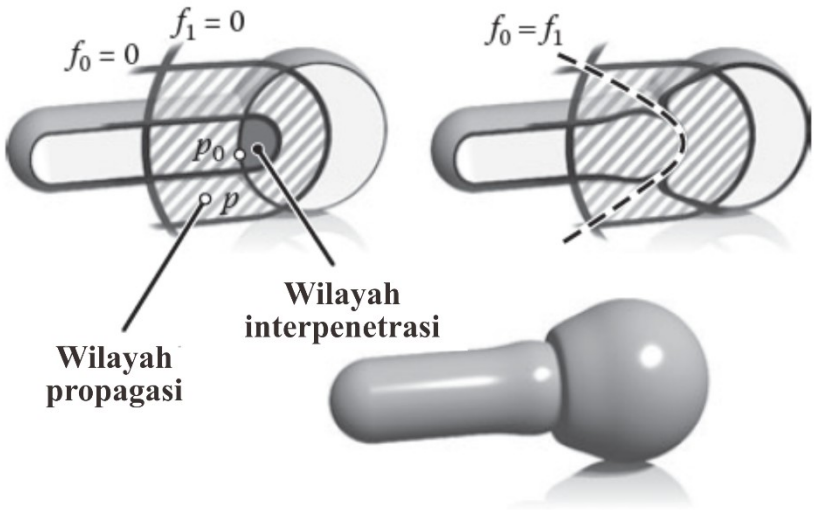
Kita perlu menghasilkan permukaan yang sama untuk kedua elemen (garis putus-putus pada Gambar 22.25), yaitu, di mana mereka berbagi solusi dalam wilayah interpenetrasi untuk beberapa p di wilayah itu:

Persamaan 22.7

$$s1(p) - f1(p) = \text{iso},$$

$$s2(p) - f2(p) = \text{iso}.$$

Secara intuitif, semakin dalam objek 1 yang ditembus objek 2, semakin tinggi nilai implisit objek 1 dan dengan demikian semakin banyak objek 2 yang akan dikompresi.



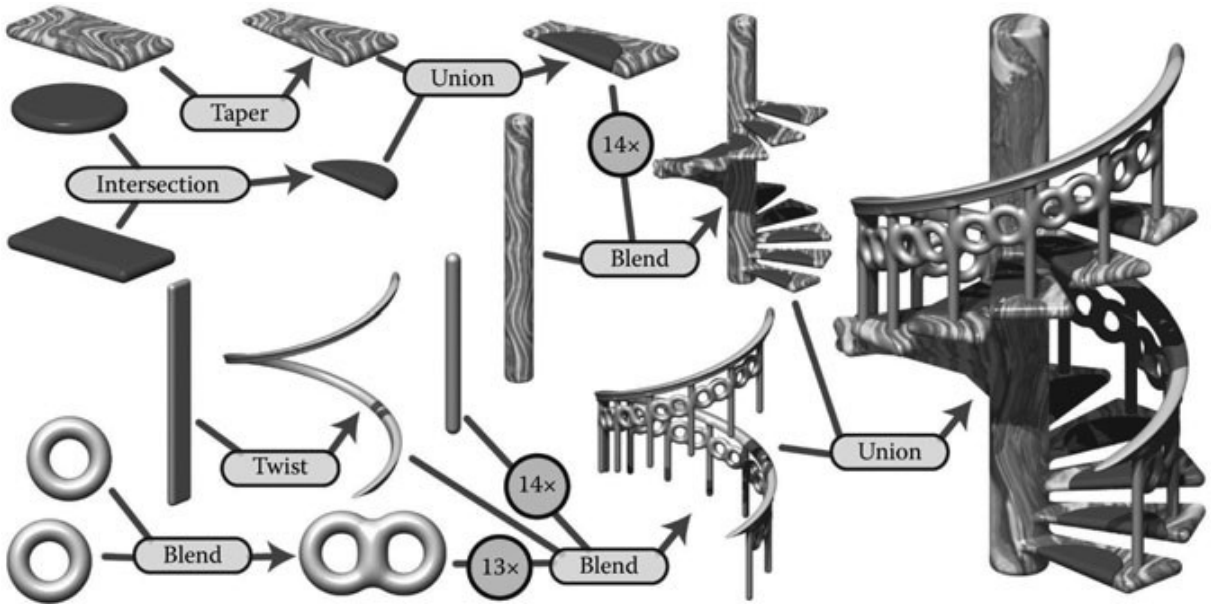
Gambar 22.25. Irisan 2D melalui objek dalam tabrakan yang menunjukkan berbagai wilayah dan deformasi PCM. Gambar milik Erwin DeGroot.

Fungsi, si didefinisikan untuk menghasilkan persimpangan mulus pada batas daerah interpenetrasi, dengan kata lain di mana $s_i = 0$tetapi turunannya lebih besar dari nol. Dari sini ke batas daerah propagasi, si digunakan untuk melemahkan propagasi menjadi nol. Titik terdekat pada batas wilayah interpenetrasi p_0 ditemukan dengan mengikuti gradien.

Dalam daerah propagasi $s_i(p) = h_i(r)$, di mana p = (x, y, z) adalah titik yang nilai implisitnya sedang dihitung dan $r = ||p - p_0||$(lihat Gambar 22.26). Nilai r_i, yang ditetapkan oleh pengguna, menentukan ukuran wilayah propagasi; tidak ada deformasi yang terjadi di luar wilayah ini. Untuk mengontrol seberapa banyak objek mengembang di wilayah propagasi, pengguna memberikan nilai untuk parameter α. Nilai maksimum hi adalah Mi. Minimum saat ini si negatif di wilayah interpenetrasi dan diberikan sebagai simin, di mana $M_i = \alpha_i s_{i\ min}$. Dengan demikian suatu benda akan terkompresi pada daerah interpenetrasi dan akan mengembang pada daerah propagasi. Persamaan untuk h_i dibentuk dalam dua bagian oleh dua polinomial kubikyang dirancang untuk bergabung pada $r = r_i/2$, di mana kemiringannya nol.Kita memiliki kontinuitas C1 saat kita berpindah dari interpenetrasi ke wilayah propagasi. Jadi, $h'_i(0) = k$ pada Gambar 22.26, adalah turunan arah dari s_i di persimpangan (ditandai sebagai p_0 pada Gambar 22.25). Seperti ditunjukkan dalam Persamaan (22.7), $s_i = f_i$ di daerah interpenetrasi, sehingga:

$$K = ||\nabla(f_i,\ p_0)||$$

PCM hanya merupakan pendekatan untuk permukaan yang terdeformasi dengan benar, tetapi merupakan algoritma yang menarik karena kesederhanaannya.

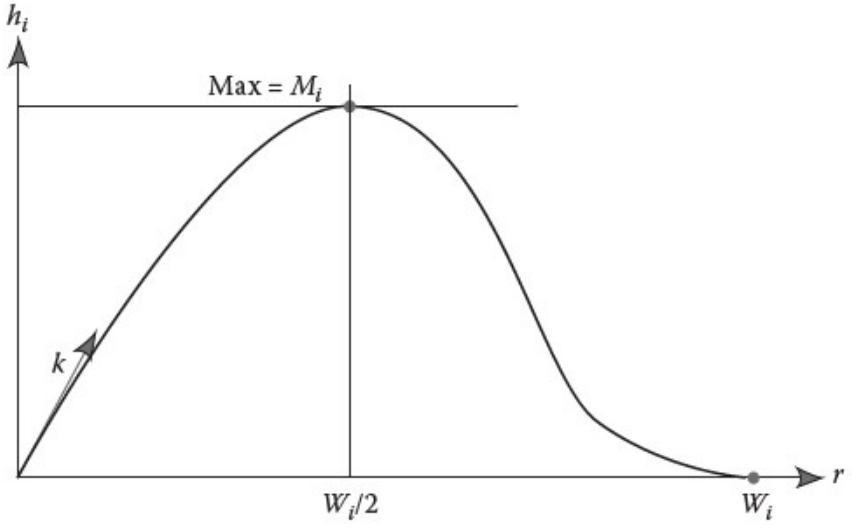


Gambar 22.26. Fungsi, $h_i = (r)$ adalah nilai fungsi deformasi w_i di daerah propagasi

22.12 BLOBTREE

BlobTree adalah metode yang menggunakan struktur pohon yang memperluas pohon CSG untuk memasukkan berbagai operasi pencampuran menggunakan kerangka primitif (Wyvill et al., 1999). Sebuah sistem dengan kemampuan serupa, proyek Hyperfun, menggunakan bahasa khusus untuk menggambarkan objek F-rep (Adzhiev et al., 1999).

Dalam sistem BlobTree, model didefinisikan oleh ekspresi yang menggabungkan primitif implisit dan operator (union), (intersection), (difference), + (blend), (super-elliptic blend), dan w (warp) . BlobTree tidak hanya struktur data yang dibangun dari ekspresi ini tetapi juga cara memvisualisasikan struktur model. Operator yang tercantum di atas adalah biner dengan pengecualian warp, yang merupakan operator unary. Secara umum lebih efisien menggunakan n-ary daripada operator biner. BlobTree menggabungkan transformasi halus sebagai node sehingga juga grafis scene dan primitif (misalnya, kerangka) membentuk node daun.



Gambar 22.27. BlobTree. Tangga spiral dibangun dari silinder bertekstur pusat tempat tangga dan pagar berpadu.

Pagar terdiri dari serangkaian silinder yang dicampur dengan dua lingkaran (torus) primitif, dicampur bersama dan selanjutnya dicampur dengan silinder vertikal. TheBlobTree juga merupakan grafis scene dan simpul instans mengulangi berbagai bagian yang ditransformasikan oleh matriks yang sesuai. Setiap tangga terbuat dari poligon primitif atap (yang menjadi permukaan offset); persimpangan dan simpul serikat menggabungkan disk yang digelembungkan dengan tangga.

Lintasan Bob Tree

Contoh operasi BlobTree termasuk Barrwarps dan CSG ditunjukkan pada Gambar 22.27. Node lain dapat mencakup tekstur 2D (Schmidt, Grimm, & Wyvill, 2006), pemodelan kontak yang tepat, serta animasi dan atribut lainnya. PenjelajahanBlobTreeisiness sangat sederhana. Semua yang diperlukan untuk membuat objek baik dengan poligonisasi atau penelusuran sinar adalah menemukan nilai implisit dari sembarang titik (dan gradien yang sesuai). Ini dapat dilakukan dengan melintasi pohon. Algoritma poligonisasi dan ray-tracing perlu mengevaluasi fungsi medan implisit pada sejumlah besar titik dalam ruang. Fungsi f(N,M) mengembalikan nilai field untuk nodeN pada titik M, yang bergantung pada jenis node. Nilai L dan R menunjukkan bahwa cabang kiri atau kanan pohon dieksplorasi. Algoritma di bawah ini ditulis (untuk kesederhanaan) seolah-olah pohon itu biner:

fungsi f(N,M):

- primitive: f(M);
- warp: f(L(N), w(M));
- campuran: f(L(N),M) +f(R(N),M));
- blend: maks(f(L(N),M), f(R(N),M));
- union: $\max(f(L(N),M),\ f(R(N),M))$;
- Intersection: min(f(L(N),M), f(R(N),M));
- different: min(f(L(N),M), −f(R(N),M)).

Model BlobTree yang kompleks menunjukkan banyak fitur yang telah terintegrasi ditunjukkan pada Gambar 22.28.

Gambar 22.28. “Tangga Spiral.” Model implisit BlobTree kompleks yang dibuat dalam sistem BlobTree.net Erwin DeGroot.

Gambar 22.29. Garis besar meningkat. Gambar milik Erwin DeGroot.

22.13 SISTEM MODELLING IMPLISIT INTERAKTIF

Sistem pemodelan berbasis sketsa awal, seperti Teddy (Igarashi, Matsuoka, & Tanaka, 1999), menggunakan beberapa goresan yang ditarik dari pengguna untuk model inferapoligonal dalam 3-ruang. Dengan perangkat keras yang lebih baik dan algoritma yang ditingkatkan, sistem pemodelan implisit berbasis sketsa sekarang dimungkinkan. Shapeshop menggunakan permukaan sapuan implisit untuk membuat goresan 3D dari sapuan pengguna 2D dan juga mempertahankan hierarki BlobTree tidak seperti sistem awal yang menghasilkan jerat homogen (Schmidt, Wyvill, Sousa, & Jorge, 2005). Hal ini memungkinkan pengguna untuk menghasilkan model kompleks topologi arbitrer dari beberapa goresan sederhana. Angka margin menunjukkan sapuan tertutup (Gambar 22.29) yang dikembangkan menjadi sapuan implisit dan sapuan kedua (Gambar 22.30) yang memiliki objek sapuan yang lebih kecil dikurangi menggunakan CSG.

Salah satu perbaikan yang memungkinkan ini adalah sistem caching yang menggunakan grid 3D tetap dari nilai implisit pada setiap node dari BlobTree yang mewakili nilai yang ditemukan dengan melintasi pohon di bawah node (Schmidt, Wyvill, & Galin, 2005). Jika nilai suatu titik p diperlukan pada simpul N, nilai dapat dikembalikan tanpa melintasi pohon di bawah N, asalkan bagian pohon tidak diubah. Sebaliknya, skema interpolasi (lihat Bab 9) digunakan untuk menemukan nilai p. Skema ini mempercepat traversal untuk BlobTrees yang kompleks dan merupakan salah satu faktor yang memungkinkan sistem berjalan dengan kecepatan interaktif.

Gambar 22.30. Operasi BlobTree dapat diterapkan, misalnya, perbedaan CSG. Gambar milik Erwin DeGroot.

Generasi berikutnya dari sistem pemodelan implisit akan memanfaatkan kemajuan perangkat keras dan perangkat lunak untuk dapat menangani model hierarkis yang semakin kompleks secara interaktif. Contoh Shapeshop yang lebih kompleks ditunjukkan pada Gambar 22.31.

Gambar 22.31. "Langkah selanjutnya." Model implisit BlobTree yang kompleks dibuat secara interaktif di Shapeshop Ryan Schmidt oleh seniman Corien Clapwijk (Andusan).

22.14 LATIHAN

1. Dalam sistem pemodelan permukaan implisit, fungsi fall-off filter didefinisikan sebagai

$$f(r)=\begin{cases} 0 & r>R \\ 1-\dfrac{r}{R} & jika tidak \end{cases}$$

   di mana R adalah konstanta. Titik primitif yang ditempatkan pada (−1,0) dan titik lainnya pada (1,0) dirender untuk menunjukkan permukaan iso f = 0, 5. Nilai R, jarak di mana potensial karena titik jatuh ke nol dalam kedua kasus, adalah 1,5. Hitung potensial pada titik (0,0) dan pada interval +0,5 sampai titik (2.5, 0). Buat sketsa kontur 0,5 dan kontur di mana bidang jatuh ke nol.

2. Mengapa kasus ambigu dalam algoritma poligonisasi dianggap sebagai masalah sampling?
3. Hitung kesalahan yang terlibat dalam penggunaan interpolasi linier untuk memperkirakan perpotongan permukaan implisit dan voxel kubik.
4. Rancang fungsi primitif implisit menggunakan kerangka pilihan Anda. Fungsi tersebut harus mengambil titik input dan mengembalikan nilai implisit dan juga gradien pada titik tersebut.

BAB 23

ILUMINASI GLOBAL

Banyak permukaan di dunia nyata menerima sebagian besar atau semua cahaya datang dari permukaan reflektif lainnya. Ini sering disebut pencahayaan tidak langsung atau pencahayaan timbal balik. Misalnya, langit-langit di sebagian besar ruangan menerima sedikit atau tidak sama sekali penerangan langsung dari luminer (benda yang memancarkan cahaya). Komponen penerangan langsung dan tidak langsung ditunjukkan pada Gambar 23.1.

Meskipun menghitung interrefleksi cahaya antara permukaan secara langsung, ini berpotensi mahal karena semua permukaan dapat memantulkan permukaan tertentu, menghasilkan interaksi $O(N^2)$ sebanyak N permukaan. Karena seluruh database objek global dapat menerangi objek apa pun, akuntansi untuk iluminasi tidak langsung sering disebut masalah iluminasi global.

Ada beberapa literatur yang kompleks dan kompleks untuk memecahkan masalah iluminasi global (misalnya, Appel, 1968;Goral, Torrance, Greenberg, & Battaile, 1984; Cooketal., 1984; Immeletal., 1986; Kajiya, 1986; Malley, 1988). Dalam bab ini, kita akan membahas dua algoritma sebagai contoh: penelusuran partikel dan penelusuran jalur. Yang pertama berguna untuk aplikasi panduan seperti permainan labirin, dan sebagai komponen rendering batch. Yang kedua berguna untuk rendering batch yang realistis. Kemudian kita membahas pemisahan pencahayaan "langsung" di mana cahaya mengambil tepat satu kali pantulan antara luminer dan kamera.

Gambar 23.1. Di gambar kiri dan tengah, pencahayaan tidak langsung dan langsung, masing-masing, dipisahkan. Di sebelah kanan, jumlah kedua komponen ditampilkan. Algoritma iluminasi global memperhitungkan pencahayaan langsung dan tidak langsung.

23.1 TRACING PARTIKEL UNTUK LAMBERTIAN SCENE

Ingat persamaan transport dari Bagian 18.2:

$$L_s(\mathbf{k}_o) = \int_{\text{all } \mathbf{k}_i} \rho(\mathbf{k}_i, \mathbf{k}_o) L_f(\mathbf{k}_i) \cos\theta_i d\sigma_i.$$

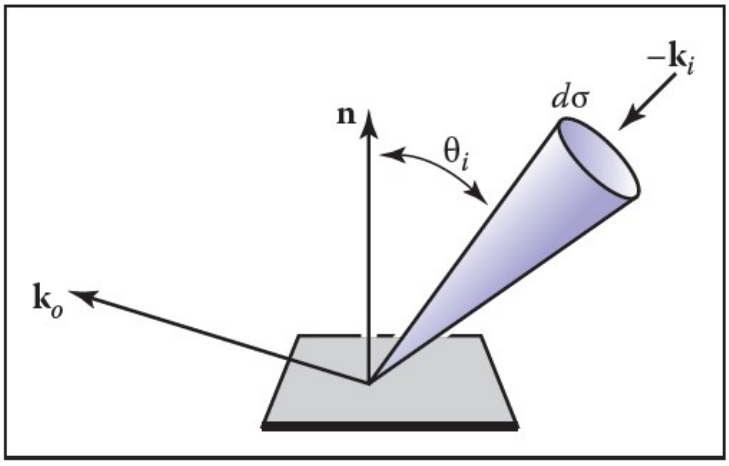
Geometri untuk persamaan ini ditunjukkan pada Gambar 23.2. Ketika titik yang diterangi adalah Lambertian, persamaan ini direduksi menjadi:

$$L_s = \frac{R}{\pi} \int_{\text{all } \mathbf{k}_i} L_f(\mathbf{k}_i) \cos\theta_i d\sigma_i,$$

di mana R adalah reflektansi difus. Salah satu cara untuk mendekati solusi persamaan ini adalah dengan menggunakan metode elemen hingga. Pertama, kami memecah scene menjadi N permukaan masing-masing dengan pancaran permukaan yang tidak diketahui L_i, memantulkan R_i, dan memancarkan pancaran E_i. Ini menghasilkan himpunan N persamaan linier simultan

$$L_i = E_i + \frac{R_i}{\pi} \sum_{j=1}^{N} k_{ij} L_j,$$

di mana kij adalah konstanta yang terkait dengan representasi integral asli. Kami kemudian memecahkan himpunan persamaan garis ini, dan kami dapat merender N poligon berwarna konstan. Pendekatan elemen hingga ini sering disebut radiositas.



Gambar 23.2. Geometri untuk persamaan transportasi dalam bentuk arahnya.

Metode alternatif untuk radioositas adalah pendekatan simulasi statistik timur secara acak mengikuti "partikel" cahaya dari luminer melalui lingkungan. Ini adalah jenis pelacakan partikel. Ada banyak algoritma yang menggunakan beberapa bentuk penelusuran partikel; kita akan membahas bentuk penelusuran partikel yang menyimpan cahaya dalam tekstur pada segitiga. Pertama, kami meninjau beberapa hubungan radiometrik dasar. Pancaran L dari permukaan Lambert dengan luasA berbanding lurus dengan daya datang per satuan luas:

Persamaan 23.1

$$L = \frac{\Phi}{\pi A}$$

di mana Φ_e adalah daya keluar dari permukaan. Perhatikan bahwa dalam diskusi ini, semua besaran radiometrik adalah spektral atau RGB, tergantung pada implementasinya. Jika permukaan telah memancarkan daya Φ_e, daya datang Φ_i, dan memantulkan R, maka persamaan ini menjadi:

$$L = \frac{\Phi e + R\,\Phi i}{\pi A}$$

Jika memakai model tertentu dengan Φ_e dan R yang ditentukan untuk setiap segitiga, kita dapat melanjutkan luminer dengan luminer, daya tembak dalam bentuk partikel dari setiap luminer. Kami mengasosiasikan peta tekstur dengan setiap segitiga untuk menyimpan akumulasi cahaya, dengan semua teks diinisialisasi ke:

$$L = \frac{\Phi e}{\pi A}$$

Jika sebuah segitiga memiliki luas A dan n_t texel, dan terkena partikel yang membawa daya Φ, maka pancaran texel tersebut bertambah sebesar

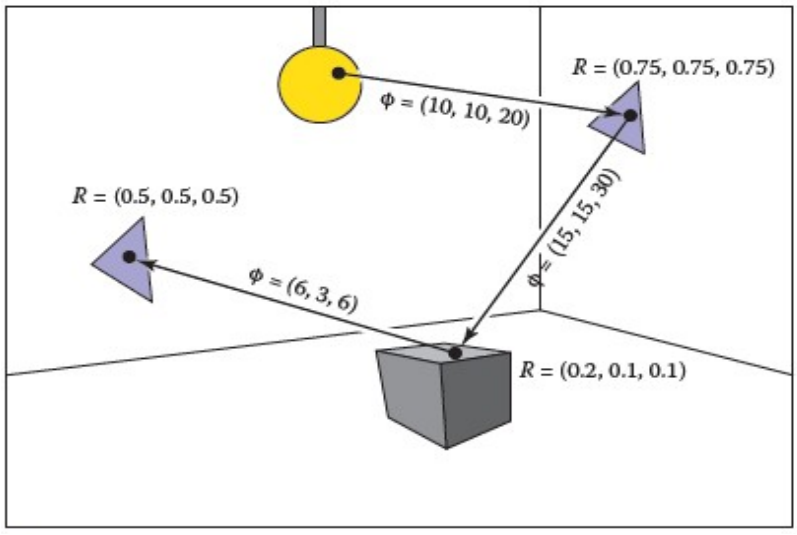
$$\Delta L = \frac{n t \varnothing}{\pi A}$$

Begitu sebuah partikel mengenai permukaan, kita meningkatkan pancaran texel yang ditabraknya, secara probabilistik memutuskan apakah akan memantulkan partikel, dan jika kita memantulkannya, kita memilih arah dan menyesuaikan kekuatannya.

Perhatikan bahwa kita ingin partikel berhenti di suatu titik. Untuk setiap permukaan kita dapat menetapkan probabilitas refleksi p untuk setiap interaksi permukaan. Pilihan yang wajar adalah membiarkan p = R sebagaimana adanya dengan cahaya di alam. Partikel kemudian akan menyebar di sekitar lingkungan, tidak kehilangan atau mendapatkan energi apapun sampai diserap. Pendekatan ini bekerja dengan baik ketika partikel membawa panjang gelombang tunggal (Walter, Hubbard, Shirley, & Greenberg, 1997). Namun, ketika spektrum atau rangkap tiga RGB dibawa oleh sinar seperti yang sering diterapkan (Jensen, 2001), tidak ada R tunggal dan beberapa kompromi untuk nilai p harus dipilih. Daya Φ' untuk partikel yang dipantulkan harus disesuaikan untuk memperhitungkan kemungkinan kepunahan partikel:

$$\Phi' = \frac{R\Phi}{p}$$

Perhatikan bahwa p dapat disetel ke konstanta positif apa pun, dan konstanta ini dapat berbeda untuk setiap interaksi. Ketika p > R untuk panjang gelombang tertentu, partikel akan memperoleh daya pada panjang gelombang tersebut, dan ketika p < R_{it} akan kehilangan daya pada panjang gelombang tersebut. Kasus di mana ia memperoleh daya tidak akan mengganggu konvergensi karena partikel akan berhenti menyebar dan dihentikan pada beberapa titik selama p<1. Untuk sisa diskusi ini, kami menetapkan p = 0,5. Lintasan partikel tunggal dalam sistem seperti itu ditunjukkan pada Gambar 23.3.



Gambar 23.3. Lintasan partikel yang bertahan dengan probabilitas 0,5 dan diserap pada perpotongan terakhir. Daya RGB ditampilkan untuk setiap segmen jalur.

Bagian penting dari algoritme ini adalah bahwa kami menyebarkan cahaya dengan distribusi yang sesuai untuk permukaan Lambertian. Sebagaimana dibahas dalam Bagian 14.4.1, kita dapat menemukan vektor dengan distribusi kosinus (Lambertian) dengan mentransformasikan dua bilangan acak kanonik (ξ_1, ξ_2) sebagai berikut:

$$a = ¿¿$$

Perhatikan bahwa ini mengasumsikan vektor normal sejajar dengan sumbu z. Untuk sebuah segitiga, kita harus membangun sebuah basis ortonormal dengan w sejajar dengan vektor normal. Kita dapat mencapai ini sebagai berikut:

$$w = \frac{n}{\|n\|}$$

$$u = \frac{p1 - p0}{\|p1 - p0\|}$$

$$v = w \times u$$

di mana p_i adalah simpul segitiga. Kemudian, menurut definisi, vektor kita dalam koordinat yang sesuai:

Persamaan 23.3

$$a = ¿¿$$

Dalam algoritma pseudocodeour untuk p =0,5 dan satu luminer adalah:

for (Each of *n* particles) **do**
RGB *phi* = Φ/*n*
compute uniform random point **a** on luminaire
compute random direction **b** with cosine density
done = false
while not done **do**
if (ray **a** + *t***b** hits at some point **c**) **then**
add *ntRφ*/(*πA*) to appropriate texel
if (*ξ*1 > 0.5) **then**
φ = 2*Rφ*
a = **c**
b = random direction with cosine density
else
done = true

Di sini i adalah bilangan acak kanonik. Setelah kode ini dijalankan, peta tekstur menyimpan pancaran setiap segitiga dan dapat dirender secara langsung untuk sudut pandang apa pun tanpa perhitungan tambahan.

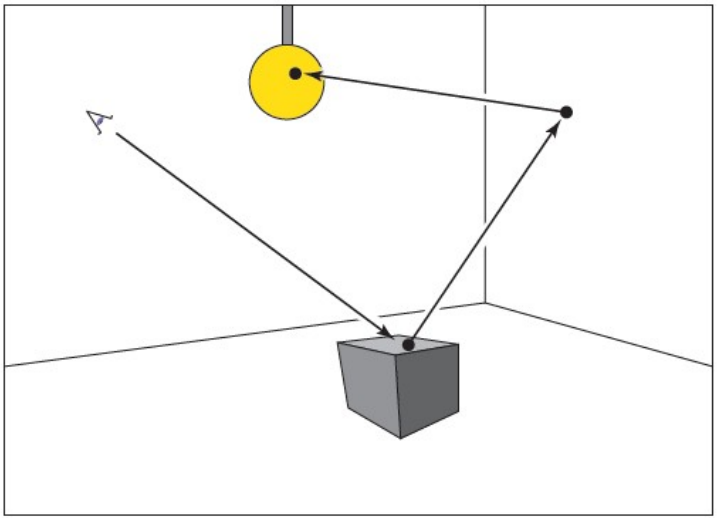
23.2 *TRACING PATH*

Sementara penelusuran partikel sangat cocok untuk pra-perhitungan pancaran scene yang menyebar, ini bermasalah untuk membuat gambar scene dengan BRDF umum atau scene yang berisi banyak objek. Cara paling mudah untuk membuat gambar scene seperti itu adalah menggunakan penelusuran jalan (Kajiya, 1986). Ini adalah metode probabilitas yang mengirimkan sinar dari mata dan menelusurinya kembali ke cahaya. Seringkali penelusuran jalur hanya digunakan untuk menghitung pencahayaan tidak langsung. Di sini kami akan menyajikannya dengan cara yang menangkap semua pencahayaan, yang bisa jadi tidak efisien. Ini kadang-kadang disebut penelusuran jalur brute force. Dalam Bagian 23.3, teknik yang lebih efisien untuk pencahayaan langsung dapat ditambahkan. Dalam penelusuran jalur, kita mulai dengan persamaan transport penuh:

$$L_a(k_0) = L_e(k_0) + \int_{semua\ k} p(¿k_{i,}k_0) L_f(k¿¿i) cos\theta_i d\sigma i¿¿$$

Kami menggunakan integrasi Monte Carlo untuk memperkirakan solusi persamaan ini untuk setiap sinar pandang. Ingat dari Bagian 14.3, bahwa kita dapat menggunakan sampel acak untuk mendekati integral:

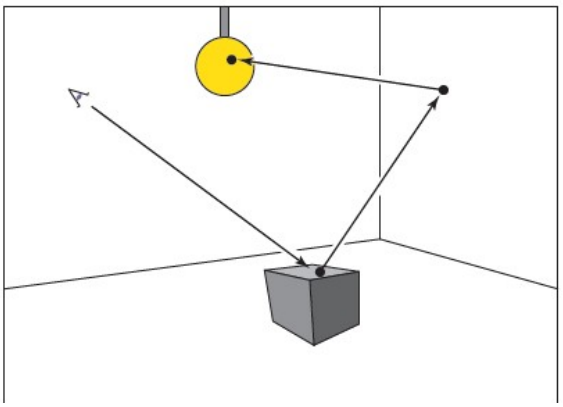
$$\int_{x \epsilon S} g(x)\, d\mu \approx \frac{1}{N} \sum_{i=1}^{N} \frac{g(xi)}{p(xi)}$$



di mana xi adalah titik acak dengan fungsi kepadatan probabilitas p. Jika kita terapkan ini langsung ke persamaan transport dengan N = 1 kita dapatkan

$$Lo(ko) \approx \leq (ko) + \frac{p(ki, ko)\, Lf(ki) \cos\theta i d\sigma i}{p(ki)}$$

Jadi, jika kita memiliki cara untuk memilih arah acak ki dengan kerapatan p yang diketahui, kita bisa mendapatkan perkiraan. Tangkapannya adalah bahwa $L_f(k_i)$ itu sendiri tidak diketahui. Untungnya, kita dapat menerapkan rekursi dan menggunakan perkiraan statistik untuk $L_f(k_i)$ dengan mengirimkan sinar ke arah itu untuk menemukan permukaan yang terlihat di arah itu. Kita berakhir ketika kita menabrak luminer dan L_e bukan nol (Gambar 23.4). Metode ini mengasumsikan lampu memiliki pantulan nol, atau kita akan terus berulang.



Gambar 23.4. Dalam penelusuran jalur, sinar diikuti melalui piksel dari mata dan dihamburkan melalui scene hingga mengenai luminer.

Dalam kasus BRDF Lambertian (ρ = R/π), kita dapat menggunakan fungsi kerapatan kosinus:

$$p(k_i) = \frac{\cos\theta_i}{\pi}$$

Arah dengan kerapatan ini dapat dipilih menurut Persamaan (23.3). Ini memungkinkan beberapa pembatalan istilah kosinus dalam perkiraan kami:

$$L_s(\mathbf{k}_o) \approx L_e(\mathbf{k}_o) + RL_f(\mathbf{k}i).$$

Dalam pseudocode, seperti pelacak jalur untuk permukaan Lambertian akan beroperasi seperti pelacak sinar yang dijelaskan dalam Bab 4, tetapi fungsi raycolor akan dimodifikasi:

```
RGB raycolor(ray a + tb, int depth)
if (ray hits at some point c ) then
RGB c = Le(−b)
if (depth < maxdepth) then
compute random direction d
return c + R raycolor(c + sd, depth+1)
else
return background color
```

Ini akan menghasilkan gambar yang sangat bising kecuali jika luminer besar atau sampel dalam jumlah sangat besar digunakan. Perhatikan warna luminer harus di atas satu (kadang ribuan atau puluhan ribu) untuk membuat permukaan memiliki warna akhir yang mendekati satu, karena hanya sinar yang mengenai luminer secara kebetulan yang akan memberikan kontribusi, dan sebagian besar sinar hanya akan menyumbang warna mendekati nol. Untuk menghasilkan arah acak d, kami menggunakan teknik yang sama seperti yang kami lakukan dalam penelusuran partikel (lihat Persamaan (23.2)).

Dalam kasus umum, kami mungkin ingin menggunakan warna spektrum atau menggunakan BRDF yang lebih umum. Dalam praktiknya, kita harus memiliki kelas materi yang berisi fungsi anggota untuk menghitung arah acak serta menghitung p yang terkait dengan arah itu. Dengan cara ini materi dapat ditambahkan secara transparan ke sebuah implementasi.

23.3 AKURASI PENCAHAYAAN LANGSUNG

Bagian ini menyajikan metode pencahayaan langsung yang lebih berbasis fisik daripada Bab 10. Metode ini akan berguna dalam membuat algoritme pencahayaan global lebih efisien. Ide utamanya adalah mengirim sinar bayangan ke luminer seperti yang dijelaskan dalam Bab 4, tetapi melakukannya dengan pembukuan yang cermat berdasarkan persamaan transportasi dari bab sebelumnya. Algoritme iluminasi global dapat disesuaikan untuk memastikan mereka menghitung komponen langsung tepat satu kali. Misalnya, dalam penelusuran partikel, partikel yang datang langsung dari luminer tidak akan dicatat, sehingga partikel hanya akan mengkodekan pencahayaan tidak langsung. Ini membuat bayangan yang terlihat bagus jauh lebih efisien daripada menghitung pencahayaan langsung dalam konteks iluminasi global.

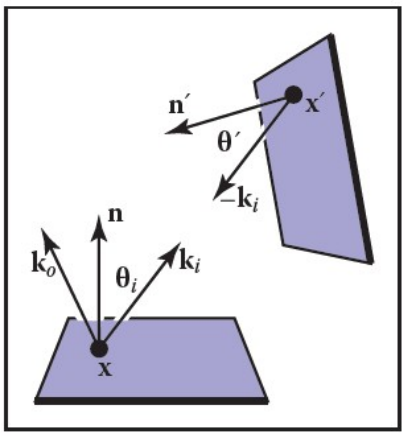
Kerangka Matematika

Untuk menghitung cahaya langsung dari satu luminer (objek pemancar cahaya) ke satu permukaan pemancar, kami memecahkan bentuk persamaan transpor dari Bagian 18.2:

Persamaan 23.4

$$L_s(\mathbf{x}, \mathbf{k}_o) = \int_{\text{all } \mathbf{x}'} \frac{\rho(\mathbf{k}_i, \mathbf{k}_o) L_e(\mathbf{x}', -\mathbf{k}_i) v(\mathbf{x}, \mathbf{x}') \cos\theta_i \cos\theta'}{\|\mathbf{x} - \mathbf{x}'\|^2} dA'.$$

Ingat bahwa Le adalah pancaran pancaran sumber, v adalah fungsi visibilitas yang sama dengan 1 jika x "melihat" x′ dan nol sebaliknya, dan variabel lainnya seperti yang diilustrasikan pada Gambar 23.5.



Gambar 23.5. Istilah pencahayaan langsung untuk Persamaan (23.4).

Jika kita ingin mengambil sampel Persamaan (23.4) menggunakan integrasi Monte Carlo, kita perlu untuk memilih titik x acak pada permukaan luminer dengan fungsi kerapatan p (jadi x p). Cukup masukkan ke Persamaan (14.5) dengan satu hasil sampel

Persamaan 23.5

$$L_s(\mathbf{x}, \mathbf{k}_o) \approx \frac{\rho(\mathbf{k}_i, \mathbf{k}_o) L_e(\mathbf{x}', -\mathbf{k}_i) v(\mathbf{x}, \mathbf{x}') \cos\theta_i \cos\theta'}{p(\mathbf{x}') \|\mathbf{x} - \mathbf{x}'\|^2}.$$

Jika kita memilih titik acak seragam pada luminer, maka p = 1/A, di mana A adalah luas luminer. Ini memberi

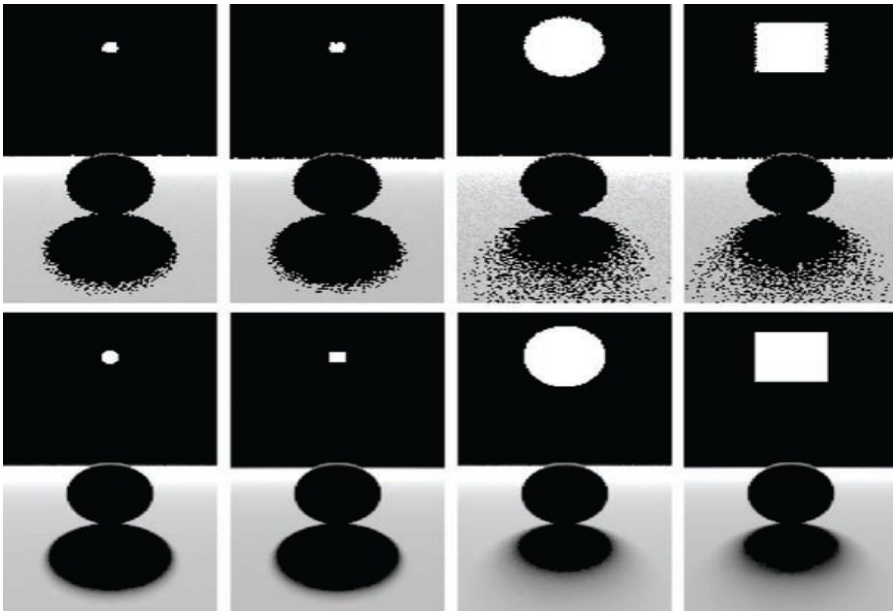
Persamaan 23.6

$$L_s(\mathbf{x}, \mathbf{k}_o) \approx \frac{\rho(\mathbf{k}_i, \mathbf{k}_o) L_e(\mathbf{x}', -\mathbf{k}_i) v(\mathbf{x}, \mathbf{x}') A \cos\theta_i \cos\theta'}{\|\mathbf{x} - \mathbf{x}'\|^2}.$$

Kita dapat menggunakan Persamaan (23.6) untuk mengambil sampel luminer planar (misalnya, persegi panjang) secara langsung. Kami hanya memilih titik acak pada setiap luminer. Kode untuk satu luminer adalah:

```
color directLight( x, ko, n)
pick random point x_ with normal vector n_ on light
d = x_ − x
ki = d/_d_
if (ray x + td has no hits for t < 1 −        ) then
return ρ(ki, ko)Le(x_,−ki)(n · d)(−n_ · d)/_d_4
else
return 0
```

Kode di atas memerlukan beberapa tes tambahan seperti menjepit kosinus ke nol jika negatif. Perhatikan bahwa suku d 4 berasal dari suku kuadrat jarak dan dua cosinus, misalnya n·d = || d|| cosθ karena d belum tentu merupakan vektor satuan. Beberapa contoh bayangan lembut ditunjukkan pada Gambar 23.6.

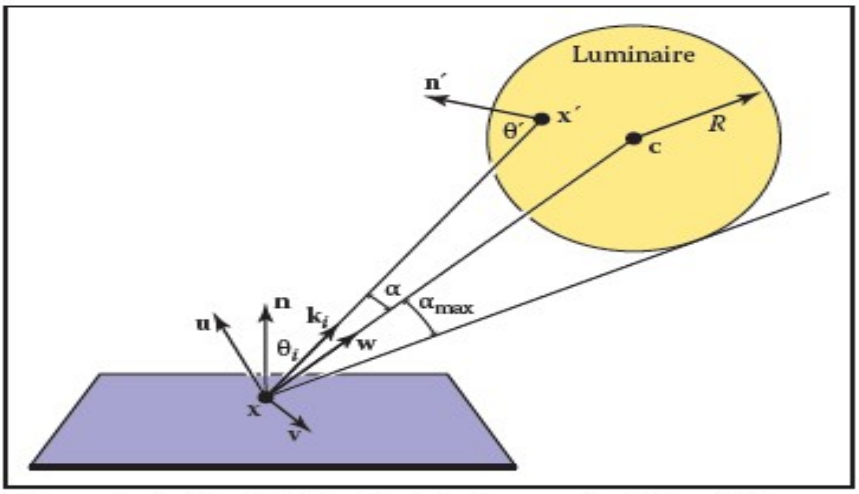


Gambar 23.6. Berbagai bayangan lembut pada bola lampu latar dengan persegi dan sumber cahaya area. Atas: 1 sampel. Bawah: 100 sampel. Perhatikan bahwa bentuk sumber cahaya kurang penting daripada ukurannya dalam menentukan tampilan bayangan

23.4 MENCICIPI LUMINER BULAT

Meskipun sebagai bidang dengan pusat candradiusR dapat diambil sampelnya menggunakan Persamaan (23.6), pengambilan sampel ini akan menghasilkan gambar yang sangat bising karena banyak sampel akan berada di bagian belakang bola, dan istilah kosnya sangat bervariasi. Sebagai gantinya, kita dapat menggunakan p(x) yang lebih kompleks untuk mengurangi noise.

Ketidakseragaman pertama yang mungkin kita coba adalah $p(x') \propto \cos\theta'$. Ini ternyata sama rumitnya dengan pengambilan sampel dengan $p(x') \propto \cos\theta'/||x' - x^2||$, jadi kita bahas di sini. Kami mengamati bahwa pengambilan sampel pada luminer dengan cara ini sama dengan menggunakan fungsi kerapatan konstan q(ki)=konstan yang didefinisikan dalam ruang arah yang diwakili oleh luminer seperti yang terlihat dari x. Kami sekarang menggunakan sistem koordinat yang didefinisikan dengan x di titik asal, dan basis ortonormal tangan kanan dengan $w = (c-x)/||c - x||$, dan $v = (w \times n)/||(w \times n)||$ (lihat Gambar 23.7). Kami juga mendefinisikan (α, ϕ) sebagai sudut azimut dan kutub terhadap sistem koordinat uvw.



Gambar 23.7. Geometri untuk penerangan langsung di titik x dari luminer bola.

α maksimum yang mencakup luminer bola diberikan oleh

$$\alpha_{\max} = \arcsin\left(\frac{R}{\|\mathbf{x}-\mathbf{c}\|}\right) = \arccos\sqrt{1-\left(\frac{R}{\|\mathbf{x}-\mathbf{c}\|}\right)^2}.$$

Jadi, kerapatan seragam (terhadap sudut padat) dalam kerucut arah yang dibatasi oleh bola hanyalah kebalikan dari sudut padat $2\pi(1-\cos\alpha_{max})$ yang diwakili oleh bola:

$$q(\mathbf{k}_i) = \frac{1}{2\pi\left(1-\sqrt{1-\left(\frac{R}{\|\mathbf{x}-\mathbf{c}\|}\right)^2}\right)}.$$

Dan kita mendapatkan

$$\begin{bmatrix} \cos\alpha \\ \phi \end{bmatrix} = \begin{bmatrix} 1-\xi_1+\xi_1\sqrt{1-\left(\frac{R}{\|\mathbf{x}-\mathbf{c}\|}\right)^2} \\ 2\pi\xi_2 \end{bmatrix}.$$

Ini memberi kita arah ki. Untuk menemukan titik sebenarnya, kita perlu mencari titik pertama pada bola ke arah itu. Sinar dalam arah itu adil $(x + tk_i)$, di mana k_i diberikan oleh:

$$\mathbf{k}_i = \begin{bmatrix} u_x & v_x & w_x \\ u_y & v_y & w_y \\ u_z & v_z & w_z \end{bmatrix} \begin{bmatrix} \cos\phi\sin\alpha \\ \sin\phi\sin\alpha \\ \cos\alpha \end{bmatrix}.$$

Kita juga harus menghitung p(x), fungsi densitas probabilitas sehubungan dengan ukuran luas (ingat bahwa fungsi densitas q didefinisikan dalam ruang sudut padat). Karena kita tahu bahwa q adalah fungsi densitas probabilitas yang valid dengan menggunakan ukuran , dan kita tahu bahwa $d\Omega=dA(x')\cos\theta / x'- x^2$, kita dapat menghubungkan setiap fungsi densitas probabilitas $q(k_i)$ dengan fungsi densitas probabilitas yang terkait fungsi densitas probabilitas p(x'):

$$q(k¿¿1)=\frac{p(x')\cos\theta'}{¿|x'-x|\vee¿^2¿}¿$$

Jadi kita dapat memecahkan p(x):

$$p(\mathbf{x}') = \frac{\cos\theta'}{2\pi\|\mathbf{x}'-\mathbf{x}\|^2\left(1-\sqrt{1-\left(\frac{R}{\|\mathbf{x}-\mathbf{c}\|}\right)^2}\right)}.$$

Kasus yang baik untuk ini ditunjukkan pada Gambar 23.8.

Gambar 23.8. Sebuah bola dengan Le = 1 menyentuh bola pantul 1. Jika kedua bola bersentuhan, bola pantul harus memiliki L(x) = 1 . Kiri: 1 sampel. Tengah: 100 sampel. Kanan: 100 sampel, close-up.

23.5 TOKOH-TOKOH NONDIFUS

Tidak ada alasan mengapa luminer luminer tidak dapat bervariasi dengan arah dan posisi. Misalnya, dapat bervariasi dengan posisi jika luminer adalah televisi. Ini dapat bervariasi dengan arah untuk lampu mobil dan sumber arah lainnya. Sedikit dalam analisis kami perlu perubahan dari bagian sebelumnya, kecuali bahwa $L_e(x')$ harus berubah menjadi $L_e(x', -k_i)$. Cara paling sederhana untuk memvariasikan intensitas dengan arah adalah menggunakan pola seperti Phong terhadap vektor normal n'. Untuk menghindari penggunaan eksponen dalam istilah untuk output cahaya total, kita dapat menggunakan bentuk:

$$L_e ¿¿$$

di mana E(x') adalah radianexitance(powerper satuan luas) di titik x', dan n adalah eksponen Phong. Anda mendapatkan cahaya difus untuk n = 1. Jika cahaya tidak seragam di seluruh areanya, misalnya, seperti pesawat televisi, maka E tidak akan menjadi konstanta.

Pertanyaan yang Sering Diajukan

- ***Nilai piksel saya tidak lagi dalam kisaran nol-ke-satu yang masuk akal. Apa yang harus saya tampilkan?***

Anda harus menggunakan salah satu teknik reproduksi nada yang dijelaskan dalam Bab 21.

- ***Teknik iluminasi global apa yang digunakan dalam praktik?***

Untuk rendering batch scene kompleks, penelusuran jalur dengan satu tingkat refleksi sering digunakan. Penelusuran jalur sering ditambah dengan praproses penelusuran partikel seperti yang dijelaskan dalam buku Jensen di catatan bab. Untuk permainan walkthrough, beberapa bentuk praproses ruang-dunia sering digunakan, seperti pelacakan partikel yang dijelaskan dalam bab ini. Untuk scene dengan transportasi spekuler yang sangat rumit, metode yang elegan namun melibatkan, Metropolis LightTransport (Veach&Guibas, 1997) mungkin merupakan pilihan terbaik.

- ***Bagaimana komponen ambien berhubungan dengan iluminasi global?***

Untuk scene difus, pancaran suatu permukaan sebanding dengan produk pancaran sinar pada permukaan dan pantulan permukaan. Komponen ambien hanyalah perkiraan terhadap radiasi yang diskalakan dengan kebalikan dari . Jadi meskipun ini adalah perkiraan kasar, mungkin ada beberapa metodologi untuk menebaknya (M. F. Cohen, Chen, Wallace, & Greenberg, 1988), dan mungkin lebih akurat daripada tidak melakukan apa-apa, yaitu, menggunakan nol untuk istilah ambient. Karena penyinaran tidak langsung dapat sangat bervariasi dalam sebuah scene, menggunakan konstanta yang berbeda untuk setiap permukaan dapat digunakan untuk hasil yang lebih baik daripada menggunakan istilah lingkungan global.

- ***Mengapa sebagian besar algoritme menghitung pencahayaan langsung menggunakan penelusuran sinar tradisional?***

Meskipun algoritme iluminasi global secara otomatis menghitung pencahayaan langsung, dan pada kenyataannya, sedikit lebih rumit untuk membuatnya hanya menghitung pencahayaan tidak langsung, biasanya lebih cepat untuk menghitung pencahayaan langsung secara terpisah. Ada tiga alasan untuk ini. Pertama, pencahayaan tidak langsung cenderung halus dibandingkan dengan pencahayaan langsung (lihat Gambar 23.1) sehingga representasi yang lebih kasar dapat digunakan, misalnya, peta tekstur resolusi rendah untuk penelusuran partikel. Alasan kedua bahwa sumber cahaya cenderung kecil, dan jarang mengenainya secara kebetulan dalam metode "dari mata" seperti penelusuran jalur, sementara sinar bayangan

langsung efisien. Alasan ketiga adalah bahwa pencahayaan langsung memungkinkan pengambilan sampel bertingkat, sehingga konvergensinya cepat dibandingkan dengan pengambilan sampel tidak bertingkat. Masalah stratifikasi adalah alasan bahwa sinar bayangan digunakan di Transportasi Cahaya Metropolis meskipun teknik standarnya memiliki kemampuan untuk menangani pencahayaan langsung hanya sebagai satu jenis jalur yang harus ditangani.

Gambar 23.9. Perbandingan antara rendering dan foto. Gambar milik Sumant Pattanaik dan Cornell Program of Computer Graphics.

- ***Seberapa artifisial untuk mengasumsikan perilaku difus dan specular yang ideal?***

Untuk lingkungan yang hanya memiliki permukaan matte dan cermin, asumsi Lambertian/spekular bekerja dengan baik. Perbandingan antara rendering menggunakan asumsi itu dan fotografi ditunjukkan pada Gambar 23.9.

- ***Berapa banyak sinar bayangan yang dibutuhkan per piksel?***

Biasanya antara 16 dan 400. Menggunakan penumbra sempit, istilah ambient yang besar (atau komponen tidak langsung yang besar), dan tekstur penutup (Ferwerda, Shirley, Pattanaik, & Greenberg, 1997) dapat mengurangi jumlah yang dibutuhkan.

- ***Bagaimana saya mengambil sampel sesuatu seperti filamen dengan reflektor logam di mana sebagian besar cahaya dipantulkan dari filamen?***

Biasanya, seluruh cahaya digantikan oleh sumber sederhana yang mendekati perilaku agregatnya. Untuk melihat sinar, sumber yang rumit digunakan. Jadi lampu depan mobil akan terlihat rumit bagi yang melihatnya, tetapi kode pencahayaannya mungkin melihat lampu berbentuk cakram sederhana.

- ***Bukankah sesuatu seperti langit adalah luminer?***

Ya, dan Anda dapat memperlakukannya sebagai satu. Namun, sumber cahaya besar seperti itu mungkin tidak terbantu dengan pencahayaan langsung; teknik brute force cenderung bekerja lebih baik.

23.6 CATATAN

Globaliluminasi berakar pada bidang perpindahan panas dan rekayasa iluminasi seperti yang didokumentasikan dalam Radiosity: A Programmer's Perspective (Ashdown, 1994). Buku bagus lainnya yang berkaitan dengan iluminasi global termasuk Radiosity and

Global Illumination (MF Cohen & Wallace, 1993), Radiosity and Realistic Image Synthesis (Sillion & Puech, 1994), Principles of Digital Image Synthesis (Glassner, 1995), Realistic Image Synthesis Using Photon Mapping (Jensen, 2001), Penerangan Global Lanjutan (Dutre´, Bala, & Bekaert, 2002), dan Rendering Berbasis Fisik (Pharr & Humphreys, 2004). Metode probabilistik yang dibahas dalam bab ini berasal dari Teknik Monte Carlo untuk Perhitungan Pencahayaan Langsung (Shirley, Wang, & Zimmerman, 1996).

23.7 LATIHAN

1. Untuk lingkungan tertutup, di mana setiap permukaan adalah reflektor dan emitor difus dengan pantul R dan pancaran pancaran E, berapa pancaran total pada setiap titik? Petunjuk: untuk R =0 .5 dan E =0 .25 jawabannya adalah 0,5. Ini adalah kasus debug yang sangat baik.
2. Menggunakan definisi dari Bab 18, verifikasi Persamaan (23.1).
3. Jika kita ingin membuat ruangan berukuran biasa dengan tekstur pada resolusi sentimeter persegi, kira-kira berapa banyak partikel yang harus kita kirim untuk mendapatkan rata-rata sekitar 1000 hit per texel?
4. Kembangkan metode untuk mengambil sampel acak dengan kerapatan seragam dari piringan.
5. Kembangkan metode untuk mengambil sampel acak dengan kerapatan seragam dari sebuah segitiga.
6. Kembangkan metode untuk mengambil sampel acak yang seragam pada "kubah langit" (bagian dalam belahan bumi).

BAB 24

MODEL REFLEKSI

Seperti yang telah kita bahas di Bab 18, sifat reflektif permukaan dapat diringkas menggunakan BRDF (Nicodemus, Richmond, Hsia, Ginsberg, & Limperis, 1977; Cook & Torrance, 1982). Dalam bab ini, kita membahas beberapa aspek yang paling penting secara visual dari sifat material dan beberapa model yang cukup sederhana yang berguna dalam menangkap sifat-sifat ini. Ada banyak model BRDF yang digunakan dalam grafis, dan model yang disajikan di sini dimaksudkan untuk memberikan gambaran tentang BRDF yang tidak menyebar.

24.1 MATERI DUNIA NYATA

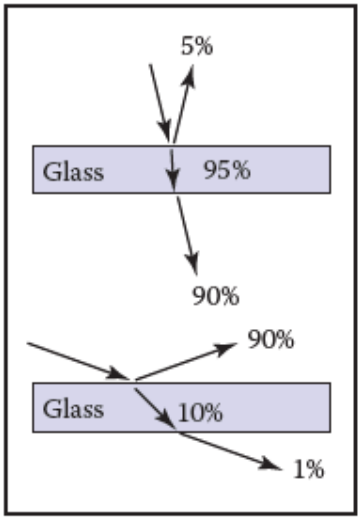
Banyak bahan nyata memiliki struktur yang terlihat pada jarak pandang normal. Misalnya, sebagian besar karpet memiliki tumpukan yang mudah terlihat yang berkontribusi pada penampilan. Untuk tujuan kita, struktur seperti itu bukan bagian dari properti material, melainkan bagian dari model geometris. Struktur yang detailnya tidak terlihat pada jarak pandang normal, tetapi yang menentukan kenampakan material secara makroskopis, adalah bagian dari properti material. Misalnya, serat di kertas memiliki tampilan kompleks di bawah pembesaran, tetapi mereka dikaburkan bersama-sama menjadi tampilan yang homogen jika dilihat dari jarak jauh. Perbedaan antara mikrostruktur yang terlipat menjadi BRDF ini agak sewenang-wenang dan tergantung pada apa yang didefinisikan sebagai jarak pandang "normal" dan ketajaman visual, tetapi perbedaan tersebut telah terbukti cukup berguna dalam praktik.

Pada bagian ini, kami mendefinisikan beberapa kategori bahan. Kemudian di bab ini, kami menyajikan model refleksi yang menargetkan setiap jenis materi. Dalam catatan di akhir bab ini, beberapa model yang menjelaskan materi yang lebih eksotis juga dibahas.

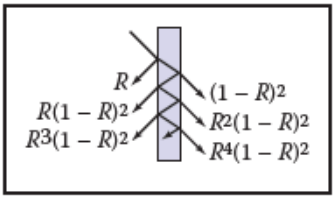
24.2 DIELEKTRIK DAN LOGAM HALUS

Dielektrik adalah bahan bening yang membiaskan cahaya; sifat dasarnya diringkas dalam Bab 4. Logam memantulkan dan membiaskan cahaya seperti dielektrik, tetapi mereka menyerap cahaya dengan sangat, sangat cepat. Jadi, hanya lembaran logam yang sangat tipis yang transparan sama sekali, misalnya pelapisan emas tipis pada beberapa benda kaca. Untuk bahan yang halus, hanya ada dua sifat penting:

1. Berapa banyak cahaya yang dipantulkan pada setiap sudut datang dan panjang gelombang.
2. Berapa fraksi cahaya yang diserap saat merambat melalui bahan untuk jarak dan panjang gelombang tertentu.



Gambar 24.1. Jumlah cahaya yang dipantulkan dan ditransmisikan oleh kaca bervariasi menurut sudutnya.



Gambar 24.2. Cahaya berulang kali dipantulkan dan dibiaskan oleh kaca, dengan fraksi energi yang ditunjukkan

Jumlah cahaya yang ditransmisikan adalah berapa pun yang tidak dipantulkan (hasil dari konservasi energi). Forametal, dalam prakteknya, kita dapat mengasumsikan semua cahaya segera diserap. Untuk dielektrik, fraksi ditentukan oleh konstanta yang digunakan dalam Hukum Beer seperti yang dibahas dalam Bab 13.

Jumlah cahaya yang dipantulkan ditentukan oleh persamaan Fresnel seperti yang dibahas dalam Bab 4. Persamaan ini sederhana, tetapi rumit. Efek utama dari persamaan Fresnel adalah untuk meningkatkan reflektansi ketika sudut datang meningkat, khususnya sudut penggembalaan dekat. Efek ini bekerja untuk cahaya yang ditransmisikan juga. Ide-ide ini ditunjukkan secara diagram pada Gambar 24.1. Perhatikan bahwa cahaya berulang kali dipantulkan dan dibiaskan seperti yang ditunjukkan pada Gambar 24.2. Biasanya hanya satu atau dua dari gambar yang dipantulkan yang mudah terlihat.

24.3 PERMUKAAN KASAR

Jika logam atau dielektrik dikasar hingga derajat kecil, tetapi tidak terlalu kecil sehingga terjadi difraksi, maka kita dapat menganggapnya sebagai permukaan dengan aspek mikro (Cook & Torrance, 1982). Permukaan seperti itu berperilaku spekular pada jarak yang lebih dekat, tetapi dilihat pada jarak yang lebih jauh tampaknya menyebarkan cahaya dalam suatu distribusi. Untuk logam, contoh permukaan kasar ini mungkin baja yang disikat, atau sisi "berawan" dari sebagian besar aluminium foil.

Untuk dielektrik, seperti lembaran kaca, goresan atau fitur permukaan tidak beraturan lainnya membuat kaca mengaburkan gambar yang dipantulkan dan ditransmisikan yang biasanya dapat kita lihat dengan jelas. Jika permukaannya sangat tergores, kami menyebutnya

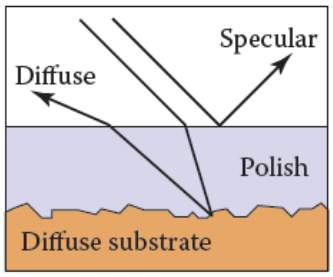
tembus daripada transparan. Ini adalah perbedaan yang agak sewenang-wenang, tetapi biasanya jelas apakah kita akan menganggap kaca tembus cahaya transparan.

Bahan Difusi

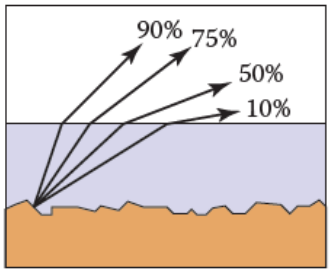
Amaterialisdiffuseifitismatte,yaitu,tidak mengkilap. Banyak permukaan yang kita lihat berserakan, seperti kebanyakan batu, kertas, dan kayu yang belum jadi. Untuk pendekatan pertama, permukaan difus dapat didekati dengan BRDF Lambertian (konstan). Bahan difus nyata biasanya menjadi agak specular untuk sudut penggembalaan. Ini adalah efek yang halus, tetapi dapat menjadi penting untuk realisme.

Bahan tembus pandang

Banyak benda tipis, seperti daun dan kertas, memancarkan dan memantulkan cahaya secara difus. Untuk semua tujuan praktis tidak ada gambar yang jelas ditransmisikan oleh benda-benda ini. Permukaan ini dapat menambah shifting ke cahaya yang ditransmisikan. Misalnya, kertas merah berwarna merah karena menyaring cahaya non-merah untuk cahaya yang menembus jarak pendek ke dalam kertas, dan kemudian menyebar kembali. Kertas juga mentransmisikan cahaya dengan warna merah karena mekanisme yang sama berlaku, tetapi cahaya yang ditransmisikan membuat semua jalan melalui kertas. Salah satu implikasi dari sifat ini adalah bahwa koefisien yang ditransmisikan harus sama di kedua arah.



Gambar 24.3. Cahaya yang mengenai permukaan berlapis dapat dipantulkan secara spekular, atau dapat ditransmisikan dan kemudian dihamburkan secara difus dari substrat.



Gambar 24.4. Cahaya yang dihamburkan oleh substrat semakin kecil kemungkinannya untuk keluar dari permukaan karena sudut relatif terhadap permukaan normal meningkat.

Bahan Berlapis

Banyak permukaan terdiri dari "lapisan" atau dielektrik dengan partikel tertanam yang memberikan permukaan properti difus (Phong, 1975). Permukaan bahan tersebut mencerminkan specularly seperti yang ditunjukkan pada Gambar 24.3, dan dengan demikian mematuhi persamaan Fresnel. Cahaya yang ditransmisikan diserap atau dihamburkan kembali ke permukaan dielektrik di mana ia dapat atau tidak dapat ditransmisikan. Cahaya yang ditransmisikan, dihamburkan, dan kemudian ditransmisikan kembali dalam arah yang berlawanan membentuk komponen “refleksi” difus.

Perhatikan bahwa komponen sekering juga dilemahkan dengan derajat sudut, karena persamaan Fresnel menyebabkan pemantulan kembali ke permukaan saat sudut bertambah seperti yang ditunjukkan pada Gambar 24.4. Jadi, daripada BRDF difus konstan, BRDF yang menghilang di dekat sudut penggembalaan lebih tepat.

24.4 IMPLEMENTASI MODEL REFLEKSI

Model BRDF, seperti yang dijelaskan dalam Bagian 18.1.6, akan menghasilkan rendering yang lebih berbasis fisik daripada rendering yang kita dapatkan dari sumber cahaya titik dan model mirip Phong. Sayangnya, BRDF nyata biasanya cukup rumit dan tidak dapat diturunkan dari prinsip pertama. Sebaliknya, mereka harus diukur dan didekati secara langsung dari data mentah, atau mereka harus didekati secara kasar secara empiris. Strategi empiris terakhir adalah apa yang biasanya dilakukan, dan pengembangan model perkiraan seperti itu masih merupakan bidang penelitian. Bagian ini membahas beberapa sifat yang diinginkan dari model empiris tersebut.

Pertama, kendala fisik menyiratkan dua sifat dari model BRDF. Kekekalan energi kendala pertama:

$$R(ki) = \int_{semua\,ko} p(ki, kpo) \cos\theta o \, d\sigma o \leq 1$$

Jika Anda mengirimkan seberkas cahaya pada permukaan dari segala arah ki, maka jumlah total cahaya yang dipantulkan ke segala arah akan paling banyak jumlah datang. Properti fisik kedua yang kami harapkan dimiliki oleh semua BRDF adalah timbal balik:

$$Untuk\ semua\ \mathbf{k}_i,\ \mathbf{k}_o,\ \rho(\mathbf{k}_i,\ \mathbf{k}_o) = \rho(\mathbf{k}_o,\ \mathbf{k}_i).$$

Kedua, kami ingin pemisahan yang jelas antara komponen difusi dan specular. Alasan untuk ini adalah bahwa, meskipun ada rumusan fungsi delta yang bersih secara matematis untuk komponen spekular ideal, fungsi delta harus diimplementasikan sebagai kasus khusus dalam praktiknya. Kasus khusus seperti itu hanya praktis jika model BRDF secara jelas menunjukkan apa yang spekular dan apa yang difus.

$$\int f(x) d\mu \approx \frac{1}{N} \sum_{j=1}^{N} \frac{f(x_j)}{p(x_j)}.$$

Ketiga, kami ingin parameter intuitif. Misalnya, salah satu alasan model Phong telah menikmati umur panjang seperti itu adalah karena konstanta dan eksponen difusnya jelas terkait dengan sifat intuitif permukaan, yaitu warna permukaan dan ukuran sorotan.

Terakhir, kami ingin agar fungsi BRDF dapat digunakan untuk pengambilan sampel Monte Carlo. Ingat dari Bab 14 bahwa suatu integral dapat diambil sampelnya dengan N titik acak xi p di mana p didefinisikan dengan ukuran yang sama dengan integral:

$$Ls(ko) = \int_{semua\,ki} p(ki, ko) Lf(ki) \cos\theta i \, d\sigma i$$

Jika kita mengambil sampel arah dengan pdf p(ki) seperti yang dibahas dalam Bab 23, maka kita dapat memperkirakan pancaran permukaan dengan sampel:

$$L_s(\mathbf{k}_o) \approx \frac{1}{N} \sum_{j=1}^{N} \frac{\rho(\mathbf{k}_j, \mathbf{k}_o) L_f(\mathbf{k}_j) \cos\theta_j}{p(\mathbf{k}_j)}.$$

Pendekatan ini akan konvergen untuk setiap p yang bukan nol di mana integrannya bukan nol. Namun, itu hanya akan konvergen dengan baik jika integran tidak terlalu besar relatif terhadap p. Idealnya, p(k) harus berbentuk seperti integral $(k_j, k_o)\ L_f(k_j)\cos\theta_j$. Dalam praktiknya, Lf rumit, dan hal terbaik yang dapat kita lakukan adalah membuat p(k) berbentuk seperti $(k, k_o)L_f(k)\cos\theta$.

Misalnya, jika BRDF adalah Lambertian, maka BRDF adalah konstan dan p(k) "ideal" sebanding dengan cosθ. Karena integral p harus satu, kita dapat menyimpulkan konstanta terdepan:

$$\int_{semua\ k\ dengan\ \theta<\pi/2} C\cos\theta d\sigma = 1$$

Ini menyiratkan bahwa C = 1/π, jadi kami memiliki

$$p(k) = \frac{1}{\pi}\cos\theta$$

Implementasi efisien yang dapat diterima akan dihasilkan selama p tidak menjadi terlalu kecil ketika integran bukan nol. Dengan demikian, pdf konstan juga akan cukup:

$$p(k) = \frac{1}{2\pi}$$

Ini menekankan bahwa banyak pdf mungkin dapat diterima untuk model BRDF tertentu.

24.5 MODEL REFLEKSI SPEKULAR

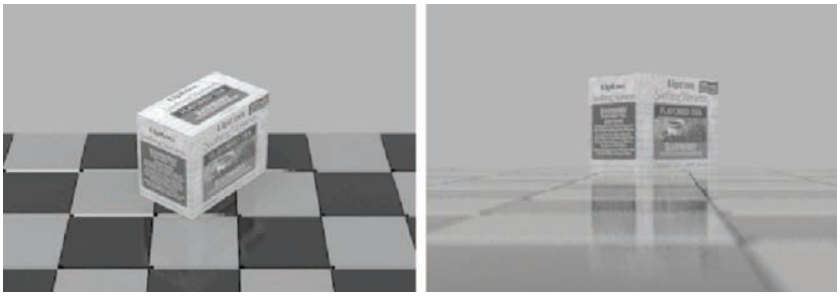
Untuk logam, kami biasanya menentukan pemantulan pada kejadian normal $R_0(\lambda)$. Refleksi harus bervariasi sesuai dengan persamaan Fresnel, dan pendekatan yang baik diberikan oleh (Schlick, 1994a)

$$R(\theta, \lambda) = R_0(\lambda) + (1 - R_0(\lambda))(1 - \cos\theta)^5 .$$

Pendekatan ini memungkinkan kita untuk menyetel pantulan normal logam baik dari data maupun mata. Untuk dielektrik, rumus yang sama bekerja untuk pemantulan. Namun, kita dapat menetapkan $R_0(\lambda)$ dalam bentuk indeks bias n(λ):

$$R0(\lambda) = \left(\frac{n(\lambda)-1}{n(\lambda)+1}\right)^2$$

Biasanya, n tidak bervariasi dengan panjang gelombang, tetapi untuk aplikasi di mana dispersi penting, n dapat bervariasi. Indeks bias yang sering berguna antara lain air (n =1 .33), kaca (n = 1 . 4 sampai n = 1 . 7), dan intan (n = 2 . 4).



Gambar24.5. Rendering tiling yang dipoles menggunakan model berpasangan. Gambar-gambar ini diproduksi menggunakan pelacak jalur Monte Carlo. Distribusi sampling untuk suku difus adalah cosθ/π.

24.6 MODEL LAYER-SMOOTH

Refleksi dalam bahan matte/spekular, seperti plastik atau kayu yang dipoles, diatur oleh persamaan Fresnel di permukaan dan dengan hamburan di dalam bawah permukaan. Contoh refleksi ini dapat dilihat pada tiling pada rendering pada Gambar 24.5. Perhatikan bahwa blurring dalam pantulan spekular sebagian besar vertikal karena kompresi jarak tonjolan yang tampak dalam arah tampilan. Efek ini menyebabkan pantulan garis vertikal yang terlihat di luar danau pada hari berangin; efek ini dapat dimodelkan dengan menggunakan mikrogeometri eksplisit dan model pantulan permukaan halus sederhana atau dengan model yang lebih umum yang menjelaskan asimetri ini.

Kita dapat menggunakan model spekular Lambertian tradisional untuk tiling, yang secara linier mencampurkan suku spekular dan Lambertian. Dalam suku radiometrik standar, ini dapat dinyatakan sebagai

$$p(\theta, \varnothing, \theta', \theta' \lambda) = \frac{Rd(\lambda)}{\pi} + Rsps(\theta, \varnothing, \theta', \theta')$$

di mana $R_d(\lambda)$ adalah pantulan hemisferis dari suku matte, Rs adalah pantulan spekular, dan s adalah BRDF spekular ternormalisasi (fungsi delta Dirac berbobot pada bola). Persamaan ini adalah versi sederhana dari BRDF di mana Rs tidak bergantung pada panjang gelombang. Ketergantungan panjang gelombang menyebabkan sorot yang merupakan warna luminer, dipoles sabun daripada penampilan logam akan dicapai. Ward (G. J. Ward, 1992) menyarankan untuk mengatur $R_d(\lambda) + R_s \leq 1$ untuk menghemat energi. Namun, model tersebut dengan R_s konstan gagal menunjukkan peningkatan spekularitas untuk sudut pandang yang curam. Ini adalah poin kuncinya: di dunia nyata proporsi relatif dari matte dan tampilan specular berubah dengan sudut pandang.

Salah satu cara untuk mensimulasikan perubahan penampilan teh adalah dengan secara eksplisit meredam $R_d(\lambda)$ seiring dengan meningkatnya R_s (Shirley, 1991):

$$p(\theta, \varnothing, \theta', \theta' \lambda) = Rf ¿$$

di mana $R_f(\theta)$ adalah pantulan Fresnel untuk antarmuka udara-poles. Masalah dengan persamaan ini adalah bahwa itu tidak timbal balik, seperti yang dapat dilihat dengan menukar dan ; ini mengubah nilai faktor redaman matte karena perkalian dengan $(1-R_f(\theta))$. Istilah spekuler, fungsi delta Dirac berskala, adalah timbal balik, tetapi ini tidak menggantikan non-timbal balik dari istilah materi. Meskipun BRDF ini bekerja dengan baik, kurangnya timbal baliknya dapat menyebabkan beberapa metode rendering memiliki solusi yang tidak jelas.

Kami sekarang menghadirkan model yang menghasilkan tradeoff matte/spekular dengan tetap mempertahankan timbal balik dan penghematan energi. Karena fitur kunci dari model baru adalah bahwa ia menggabungkan koefisien skala matte dan specular, itu disebut model digabungkan (Shirley, Smits, Hu, & Lafortune, 1997).

Permukaan yang tampak mengkilap seringkali merupakan dielektrik yang bening, seperti poliuretan atau minyak, dengan beberapa struktur bawah permukaan. Komponen spekular (mirip cermin) dari pantulan disebabkan oleh permukaan dielektrik yang halus dan tidak bergantung pada struktur di bawah permukaan ini. Besarnya suku spekular ini diatur oleh persamaan Fresnel.

Cahaya yang tidak dipantulkan secara spekular di permukaan ditransmisikan melalui permukaan. Di sana, baik itu diserap oleh bawah permukaan, atau dipantulkan dari pigmen atau bawah permukaan dan ditransmisikan kembali melalui permukaan cat. Cahaya yang ditransmisikan ini membentuk komponen refleksi matte. Karena komponen matte hanya dapat

terdiri dari cahaya yang ditransmisikan, maka secara alami akan berkurang total besarnya untuk meningkatkan sudut.

Untuk menghindari pemilihan antara model yang masuk akal secara fisik dan model dengan perilaku kualitatif yang baik pada rentang sudut datang, perhatikan bahwa persamaan Fresnel yang menjelaskan suku spekular, $R_f(\theta)$, diturunkan langsung dari fisika antarmuka dielektrik-udara. Oleh karena itu, masalahnya harus terletak pada istilah matte. Kita bisa menggunakan simulasi penuh hamburan bawah permukaan seperti yang diterapkan, tetapi teknik ini mahal dan membutuhkan pengetahuan rinci tentang struktur bawah permukaan, yang biasanya tidak diketahui atau mudah diukur. Sebagai gantinya, kita dapat memodifikasi istilah matte menjadi pendekatan sederhana yang menangkap perilaku sudut kualitatif penting yang ditunjukkan pada Gambar 24.4.

Mari kita asumsikan bahwa istilah matte bukan Lambertian, tetapi merupakan fungsi lain yang hanya bergantung pada , dan λ : $p_m(\theta, \theta', \lambda)$. Kami membuang perilaku yang bergantung pada ϕ atau ϕ' demi kesederhanaan. Kami mencoba untuk menjaga agar rumus tetap sederhana karena fisika istilah matte rumit dan terkadang memerlukan parameter yang tidak diketahui. Kami mengharapkan istilah matte mendekati konstan, dan kira-kira simetris secara rotasi (He et al., 1992).

Kandidat yang jelas untuk komponen matte $p_m(\theta, \theta', \lambda)$ yang akan resiprokal adalah bentuk yang dapat dipisahkan $kR_m(\lambda)\ f(\theta)\ f(\theta')$ untuk beberapa konstanta k dan parameter refleksi matte $R_m(\lambda)$. Kita dapat menggabungkan k dan $R_m(\lambda)$ menjadi satu suku, tetapi kita memilih untuk memisahkannya karena ini membuatnya lebih intuitif untuk menetapkan $R_m(\lambda)$ —yang harus antara 0 dan 1 untuk semua panjang gelombang. BRDF yang dapat dipisahkan telah terbukti memiliki beberapa keuntungan komputasi, oleh karena itu kami menggunakan model yang dapat dipisahkan:

$$Rf(\theta, \varnothing, \theta', \theta' \lambda) = Rf ¿$$

Kita tahu bahwa komponen matte hanya dapat mengandung energi yang tidak terpantul di permukaan (spekular) komponen. Ini berarti bahwa untuk $R_m(\lambda) = 1$, energi datang dan energi pantul adalah sama, yang menunjukkan batasan BRDF berikut untuk setiap kejadian θ dan λ:

$$Rf(\theta) + 2\pi kf(\theta) = \int_0^{\frac{\pi}{2}} f \vee (\theta) \cos\theta' SN\,\theta' D\,\theta' = 1$$

Kita dapat melihat bahwa f(θ) harus sebanding dengan $(1 - R_f(\theta))$. Jika kita berasumsi bahwa komponen matte yang menyerap beberapa energi memiliki pola arah yang sama dengan ideal ini, kita mendapatkan bentuk BRDF

$$P(\theta, \varnothing, \theta', \theta' \lambda) = rF ¿$$

Sekarang kita dapat menyisipkan bentuk lengkap persamaan Fresnel untuk mendapatkan $R_f(\theta)$, dan kemudian menggunakan kekekalan energi untuk menyelesaikan kendala pada k. Sebagai gantinya, kami akan menggunakan pendekatan yang dibahas dalam Bagian 24.1.1 Kami menemukan bahwa:

$$f(\theta) \propto (1 - (1 - \cos\theta)^5).$$

Menerapkan Persamaan (24.1) hasil

$$k = \frac{21}{2\pi(1 = Ro)}$$

Hasil running model digabungkan ditunjukkan pada Gambar 24.5. Perhatikan bahwa untuk sudut pandang tinggi, pantulan spekular hampir tidak terlihat, tetapi terlihat jelas pada gambar foto sudut rendah, di mana perilaku matte kurang jelas.

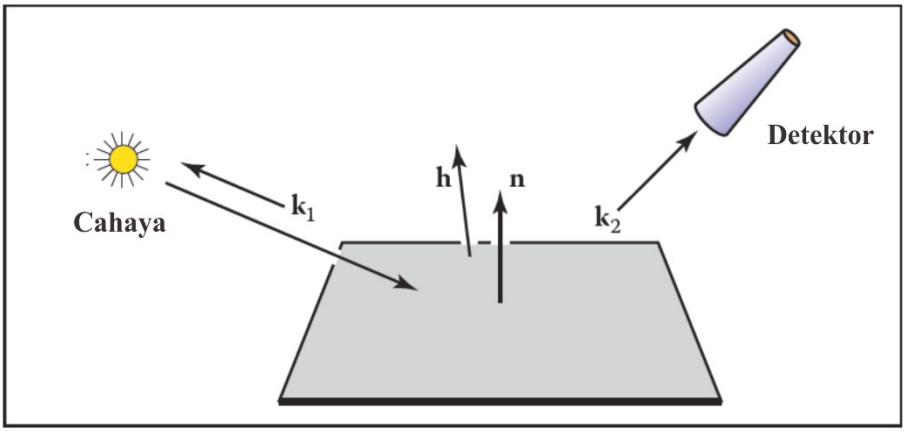
Untuk nilai indeks bias yang masuk akal, R0 dibatasi kira-kira pada kisaran 0,03 hingga 0,06 (nilai R_0 = 0,05 digunakan untuk Gambar 24.5). Nilai R_s dalam model Phong tradisional lebih sulit untuk dipilih, karena biasanya harus disetel untuk sudut pandang dalam gambar statis dan disetel untuk urutan kamera tertentu untuk animasi. Dengan demikian, model yang digabungkan lebih mudah digunakan dalam mode "lepas tangan".

24.7 MODEL LAYER-ROUGH

Model sebelumnya baik-baik saja jika permukaannya halus. Namun, jika permukaannya tidak ideal, diperlukan beberapa spread di komponen specular. Perpanjangan model digabungkan untuk kasus ini disajikan di sini (Ashikhmin & Shirley, 2000). Pada titik tertentu di permukaan, BRDF adalah fungsi dari dua arah, satu ke arah cahaya dan satu ke arah penonton. Kami ingin memiliki model BRDF yang berfungsi untuk permukaan "umum", seperti logam dan plastik, dan memiliki karakteristik sebagai berikut:

1. **Masuk akal**. Seperti yang didefinisikan oleh Lewis (R.R.Lewis, 1994), ini mengacu pada BRDF yang mematuhi konservasi energi dan timbal balik.
2. **Anisotropi**. Bahan harus memodelkan anisotropi sederhana, seperti yang terlihat pada logam yang disikat.
3. **Parameter intuitif.** Untuk bahan, seperti plastik, harus ada parameter Rd untuk substrat dan Rs untuk pantulan spekular normal serta dua parameter kekasaran nu dan nv.
4. **Perilaku Fresnel.** Spekularitas harus meningkat saat sudut datang berkurang.
5. **Istilah non-Lambertian difus**. Bahan harus memungkinkan untuk istilah difus, tetapi komponen harus non-Lambertian untuk memastikan konservasi energi dengan adanya perilaku Fresnel.
6. **MonteCarlo friendliness**. Harus ada beberapa fungsi kepadatan probabilitas yang masuk akal yang memungkinkan pembuatan sampel MonteCarlos untuk BRDF.

BRDF dengan sifat-sifat ini adalah model lobus kosinus gaya Phong berbobot Fresnel yang bersifat anisotropik.



Gambar 24.6. Geometri refleksi. Perhatikan bahwa k1, k2, andh berbagi bidang, yang biasanya tidak mencakup n.

Kami menguraikan lagi BRDF menjadi komponen specular komponen standar difus (Gambar 24.6). Dengan demikian, kami menulis BRDF kami sebagai jumlah klasik dari dua bagian:

Persamaan 24.4

$$\rho(\mathbf{k}_1, \mathbf{k}_2) = \rho_s(\mathbf{k}_1, \mathbf{k}_2) + \rho_d(\mathbf{k}_1, \mathbf{k}_2),$$

di mana istilah pertama menjelaskan refleksi spekular (ini akan disajikan pada bagian berikutnya). Meskipun dimungkinkan untuk menggunakan LambertianBRDF untuk suku difus d(k1,k2) dalam model kita, kita akan membahas solusi yang lebih baik di Bagian 24.5.2 dan bagaimana mengimplementasikan model di Bagian 24.5.3. Pembaca yang hanya ingin mengimplementasikan model harus melompat ke bagian itu.

24.8 BRDF SPEKULER ANISOTROPIK

Untuk memodelkan perilaku spekular, kami menggunakan lobus spekular gaya Phong tetapi membuat lobus ini anisotropik dan menggabungkan perilaku Fresnel sambil mencoba mempertahankan kesederhanaan mode awal. BRDF ini adalah

Persamaan 24.5

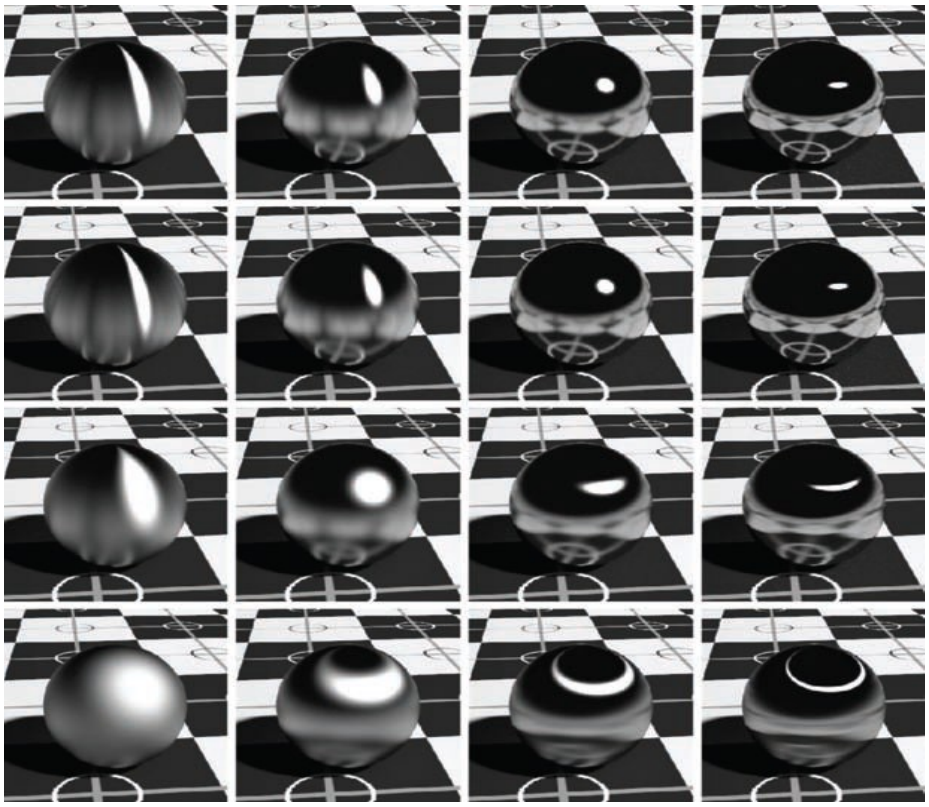
$$\rho(\mathbf{k}_1, \mathbf{k}_2) = \frac{\sqrt{(n_u+1)(n_v+1)}}{8\pi} \frac{(\mathbf{n}\cdot\mathbf{h})^{n_u \cos^2\phi + n_v \sin^2\phi}}{(\mathbf{h}\cdot\mathbf{k}_i)\max(\cos\theta_i, \cos\theta_o))} F(\mathbf{k}_i\cdot\mathbf{h})\,.$$

Sekali lagi kami menggunakan pendekatan Schlick untuk persamaan Fresnel:

Persamaan 24.6

$$F(\mathbf{k}_i \cdot \mathbf{h}) = R_s + (1 - R_s)(1 - (\mathbf{k}_i \cdot \mathbf{h}))^5,$$

di mana Rs adalah refleksi material untuk kejadian normal. Karena $k_i \cdot h = k_o \cdot h$, bentuk ini bersifat timbal balik. Kami memiliki model empiris yang istilahnya dipilih untuk menegakkan konservasi energi dan timbal balik. Rasionalisasi penuh untuk istilah-istilah tersebut diberikan dalam makalah oleh Ashikhmin, yang tercantum dalam catatan bab.



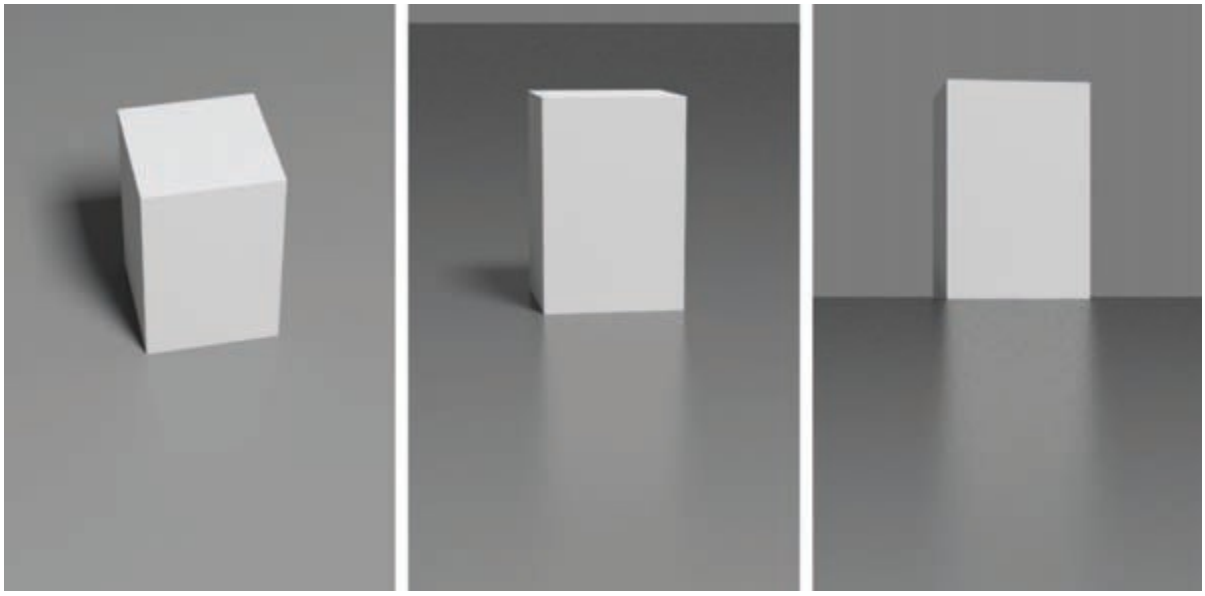
Gambar 24.7. Bola logam untuk eksponen 10, 100, 1000, dan 10.000 meningkat dari kiri ke kanan dan atas ke bawah.

BRDF khusus dari Persamaan (24.5) berguna untuk menggambarkan permukaan logam di mana komponen refleksi difus sangat kecil. Gambar 24.7 menunjukkan satu set bola

logam pada bidang Lambertian yang dipetakan tekstur. Ketika nilai parameter n_u dan n_v berubah, penampakan bola bergeser dari logam kasar ke cermin yang hampir sempurna, dan dari perilaku yang sangat anisotropik ke perilaku seperti Phong yang lebih dikenal.

Istilah Difusi untuk Model Phong Anisotropik

Dimungkinkan untuk menggunakan BRDF Lambertian bersama dengan istilah spekular anisotropik; ini dilakukan untuk kebanyakan model, tetapi tidak selalu menghemat energi. Pendekatan yang lebih baik bentuk bergantung-sudut dari komponen sekering yang menjelaskan fakta bahwa jumlah energi yang tersedia untuk hamburan difus bervariasi karena ketergantungan dari pantulan total suku pada sudut datang. Khususnya, warna difus dari suatu permukaan menghilang di dekat sudut penggembalaan, karena pantulan spekular total mendekati satu. Efek terkenal ini tidak dapat direproduksi dengan istilah Lamberti dan difus dan oleh karena itu diabaikan oleh sebagian besar model refleksi.



Gambar 24.8. Tiga tampilan untuk $n_u = n_v = 400$ dan substrat difus. Perhatikan perubahan intensitas pantulan spekular.

24.9 MENERAPKAN MODEL

Ingatlah bahwa BRDF adalah kombinasi dari komponen difus dan specular:

Persamaan 24.8

$$\rho(\mathbf{k}_1, \mathbf{k}_2) = \rho_s(\mathbf{k}_1, \mathbf{k}_2) + \rho_d(\mathbf{k}_1, \mathbf{k}_2).$$

Komponen difus diberikan dalam Persamaan (24.7); komponen specular diberikan dalam Persamaan (24.5). Tidak perlu memanggil fungsi trigonometri untuk menghitung eksponen, sehingga BRDF specular dapat ditulis:

Persamaan 24.9

$$\rho(\mathbf{k}_1, \mathbf{k}_2) = \frac{\sqrt{(n_u+1)(n_v+1)}}{8\pi}(\mathbf{n}\cdot\mathbf{h})^{\frac{(n_u(\mathbf{h}\cdot\mathbf{u})^2+n_v(\mathbf{h}\cdot\mathbf{v})^2)/(1-(\mathbf{hn})^2)}{(\mathbf{h}\cdot\mathbf{k}_i)\max(\cos\theta_i,\cos\theta_o)}} F(\mathbf{k}_i\cdot\mathbf{h}).$$

Dalam pengaturan Monte Carlo, kami tertarik pada masalah berikut: diberikan k1, menghasilkan sampel k2 dengan distribusi yang bentuknya mirip dengan BRDF berbobot kosinus. Perhatikan bahwa sangat di bawah sampling, nilai integral yang besar dan kesalahan serius, sementara sangat besar di bawah sampel, nilai kecil dapat diterima dalam praktik.

Pembaca dapat memverifikasi bahwa kepadatan yang disarankan di bawah ini memiliki sifat ini.

Cara yang cocok untuk membuat pdf untuk pengambilan sampel adalah dengan mempertimbangkan distribusi setengah vektor yang akan memunculkan BRDF kami. Fungsi seperti itu adalah:

Persamaan 24.10

$$p_h(\mathbf{h}) = \frac{\sqrt{(n_u+1)(n_v+1)}}{2\pi}(\mathbf{nh})^{n_u \cos^2\phi + n_v \sin^2\phi},$$

di mana konstanta dipilih untuk memastikan itu adalah pdf yang valid. Kita bisa menggunakan fungsi densitas probabilitas $p_h(h)$ dari Persamaan (24.10) untuk menghasilkan h acak. Namun, untuk mengevaluasi persamaan rendering, kita membutuhkan vektor yang direfleksikan k_o dan fungsi kepadatan probabilitas $p(k_o)$. Penting untuk dicatat bahwa jika Anda menghasilkan menurut $p_h(h)$ dan kemudian mengubahnya ke k_o yang dihasilkan:

Persamaan 24.11

$$\mathbf{k}_o = -\mathbf{k}_i + 2(\mathbf{k}_i \cdot \mathbf{h})\mathbf{h}$$

kerapatan k_o yang dihasilkan bukan $p_h(k_o)$. Ini karena perbedaan ukuran dalam h dan k_o. Jadi densitas sebenarnya $p(k_o)$ adalah

Persamaan 24.1

$$p(k¿¿0)=\frac{p_h(h)}{4(k¿¿ih)¿}¿$$

Perhatikan bahwa dalam implementasi di mana BRDF dikenal sebagai model ini, perkiraan persamaan rendering cukup sederhana karena banyak istilah yang dibatalkan.

Adalah mungkin untuk menghasilkan sebuah vektor yang vektor korespondennya akan menunjuk ke dalam permukaan, yaitu $\cos\theta_o < 0$. Berat sampel seperti itu harus disetel ke nol. Situasi ini sesuai dengan lobus specular yang berada di bawah cakrawala dan merupakan sumber utama kehilangan energi dalam model. Jelas, masalah ini menjadi semakin tidak parah seperti n_u, n_v menjadi lebih besar.

Satu-satunya yang tersisa sekarang adalah menjelaskan bagaimana menghasilkan h vektor dengan pdf dari Persamaan (24.10). Kita akan mulai dengan membangkitkan h dengan sudut bolanya dalam kisaran (θ, ϕ) $[0, \pi/2] \times [0, \beta/2]$. Perhatikan bahwa ini hanya kuadran pertama belahan bumi. Diberikan dua bilangan acak (ξ_1, ξ_2) yang terdistribusi merata di [0,1], kita dapat memilih dan kemudian gunakan nilai ini untuk mendapatkan sesuai dengan:

Persamaan 24.13

$$\phi = \arctan\left(\sqrt{\frac{n_u+1}{n_v+1}}\tan\left(\frac{\pi\xi_1}{2}\right)\right)|,$$

Persamaan 24.14

$$\cos\theta = (1-\xi_2)1/(n_u\cos^2\varphi + n_v\sin^2\varphi + 1).$$

Untuk sampel seluruh belahan bumi, kami menggunakan manipulasi standar di mana 1 dipetakan ke salah satu dari empat fungsi yang mungkin tergantung pada apakah itu di [0,0.25), [0.25, 0.5), [0.5, 0.75), atau [0.75, 1.0). Misalnya, untuk $\xi_1 \in [0.25, 0.5)$, cari $(\emptyset 1 - 4(0.5 - \xi_1))$ melalui Persamaan (24.13), dan kemudian "balikkan" tentang sumbu $\emptyset = \pi/2$. Ini memastikan cakupan dan stratifikasi penuh.

Untuk suku difus, gunakan pendekatan yang lebih sederhana dan hasilkan sampel menurut distribusi kosinus. Ini cukup dekat dengan BRDF difus lengkap untuk secara substansial mengurangi varians dari estimasi Monte Carlo.

Pertanyaan yang Sering Diajukan

- ***Gambar saya terlihat terlalu halus, bahkan dengan BRDF yang rumit. Apa yang saya lakukan salah?***

BRDF hanya menangkap detail subpiksel yang terlalu kecil untuk dipecahkan oleh mata. Sebagian besar permukaan nyata juga memiliki beberapa variasi kecil, seperti kerutan di kulit, yang dapat terlihat. Jika Anda menginginkan realisme sejati, diperlukan semacam tekstur atau peta perpindahan.

- ***Bagaimana cara mengintegrasikan BRDF dengan mapping tekstur?***

Mapping tekstur dapat digunakan untuk mengontrol parameter apa pun pada permukaan. Jadi segala jenis warna atau parameter kontrol yang digunakan oleh BRDF harus dapat diprogram.

- ***Saya memiliki kode yang sangat cantik kecuali untuk kelas materi saya. Apa yang saya lakukan salah?***

Anda mungkin tidak melakukan sesuatu yang salah. Kelas material cenderung menjadi hal yang jelek di program semua orang. Jika Anda menemukan cara yang bagus untuk menghadapinya, beri tahu saya! Kode saya sendiri menggunakan arsitektur shader (Hanrahan & Lawson, 1990) yang membuat materi menyertakan banyak algoritma rendering.

24.10 CATATAN

Ada banyak model BRDF yang dijelaskan dalam literatur, dan hanya sedikit yang telah dijelaskan di sini. Lainnya termasuk (Cook & Torrance, 1982; He et al., 1992; GJ Ward, 1992; Oren & Nayar, 1994; Schlick, 1994a; Lafortune, Foo, Torrance, & Greenberg, 1997; Stam, 1999; Ashikhmin, Premoˇze, & Shirley, 2000; Ershov, Kolchin, & Myszkowski, 2001; Matusik, Pfister, Brand, & McMillan, 2003; Lawrence, Rusinkiewicz, & Ramamoorthi, 2004; Stark, Arvo, & Smits, 2005). Karakteristik yang diinginkan dari model BRDF dibahas dalam Membuat Shader Lebih Plausible Secara Fisik (R. R. Lewis, 1994).

24.11 LATIHAN

1. Anggaplah sebagai ganti BRDF Lambertian kita menggunakan BRDF dari bentuk C cosa i. Apa yang harus C untuk menghemat energi?
2. BRDF dalam Latihan 1 tidak bersifat timbal balik. Bisakah Anda memodifikasinya menjadi timbal balik?
3. Sesuatu seperti rambu jalan raya adalah retroreector. Ini berarti BRDF besar ketika ki dan ko saling berdekatan. Buatlah model yang terinspirasi oleh model Phong yang menangkap perilaku retroreeksi sekaligus melakukan konservasi energi secara timbal balik.

BAB 25
GRAFIS KOMPUTER DALAM GAME

Dari semua aplikasi komputer grafis, komputer dan video game mungkin paling menarik perhatian. Metode grafis yang dipilih untuk game tertentu memiliki efek mendalam, tidak hanya pada kode mesin game, tetapi juga pada penciptaan aset seni, dan bahkan terkadang pada gameplay, atau mekanik game inti.

Meskipun grafis game bergantung pada materi di semua bab sebelumnya, dua bab sangat erat. Game perlu menggunakan perangkat keras grafis dengan sangat efisien, jadi pemahaman tentang materi di Bab 17 adalah penting.

Dalam bab ini, saya akan merinci pertimbangan khusus yang berlaku untuk grafis dalam pengembangan game, mulai dari platform tempat game dijalankan hingga proses produksi game.

25.1 *PLATFORM*

Di sini, saya menggunakan istilah platform untuk merujuk pada kombinasi spesifik dari perangkat keras, sistem operasi, dan API (antarmuka pemrograman aplikasi) yang dirancang untuk sebuah game. Game berjalan di berbagai platform, mulai dari mesin virtual yang digunakan untuk game berbasis browser hingga konsol game khusus yang menggunakan perangkat keras dan API khusus.

Di masa lalu, game dirancang untuk satu platform adalah hal yang umum. Meningkatnya biaya pengembangan game telah membuat ini jarang terjadi; pengembangan game multiplatform sekarang menjadi norma. Peningkatan bertahap dalam biaya pengembangan untuk mendukung beberapa platform lebih dari sekadar dilunasi oleh potensi penggandaan atau tiga kali lipat basis pelanggan.

Beberapa platform didefinisikan dengan cukup longgar. Misalnya, ketika mengembangkan game untuk platform PC Windows, pengembang harus memperhitungkan variasi yang sangat besar dari kemungkinan konfigurasi perangkat keras. Game bahkan diharapkan dapat berjalan (dan berjalan dengan baik) pada konfigurasi PC yang tidak ada saat game dikembangkan! Ini hanya mungkin karena abstraksi yang diberikan oleh API yang mendefinisikan platform Windows.

Salah satu cara di mana pengembang memperhitungkan variasi yang luas dalam kinerja grafis adalah dengan scalling—menyesuaikan kualitas grafis sebagai respons terhadap kemampuan sistem. Ini dapat memastikan kinerja yang wajar pada sistem kelas bawah, sambil tetap mencapai visual yang kompetitif pada sistem kinerja tinggi. Penyesuaian ini terkadang dilakukan secara otomatis dengan memprofilkan kinerja sistem, tetapi lebih sering kontrol ini diserahkan kepada pengguna, yang dapat menilai preferensi pribadinya untuk kualitas versus kecepatan. Resolusi tampilan paling mudah disesuaikan, diikuti dengan kualitas antialiasing. Hal ini juga cukup umum untuk menawarkan beberapa tingkat kualitas untuk efek visual seperti bayangan dan blur, termasuk pilihan untuk mematikan efek sepenuhnya.

Perbedaan dalam kinerja grafis bisa sangat besar sehingga beberapa mesin mungkin tidak menjalankan game pada kecepatan bingkai yang dapat dimainkan, bahkan dengan pengaturan kualitas terendah; untuk alasan ini, pengembang game PC menerbitkan spesifikasi mesin minimum dan yang direkomendasikan untuk setiap game.

Sebagai platform, konsol game didefinisikan secara ketat. Saat mengembangkan game untuk, misalnya, konsol Nintendo Wii, pengembang tahu persis perangkat keras apa yang akan digunakan game tersebut. Jika implementasi perangkat keras platform diubah (sering dilakukan untuk mengurangi biaya produksi), produsen konsol harus memastikan bahwa implementasi baru berperilaku persis seperti yang sebelumnya, termasuk waktu dan kinerja. Ini bukan untuk mengatakan bahwa tugas pengembang konsol itu mudah; API konsol cenderung jauh lebih abstrak dan lebih dekat dengan perangkat keras yang mendasarinya. Hal ini memberikan pengembangan konsol serangkaian kesulitannya sendiri. Dalam beberapa hal, pengembangan multiplatform (yang biasanya mencakup setidaknya dua platform konsol yang berbeda dan sering juga Windows) adalah yang paling sulit untuk gagal, karena pengembang game multiplatform tidak memiliki jaminan platform tetap atau kenyamanan API tingkat tinggi tunggal.

Mesin virtual berbasis browser seperti Adobe Flash adalah kelas platform game yang menarik. Meskipun mesin virtual tersebut berjalan pada kelas perangkat keras yang luas dari komputer pribadi hingga ponsel, tingkat abstraksi yang tinggi yang disediakan oleh mesin virtual menghasilkan platform pengembangan yang stabil dan terpadu. Kemudahan relatif pengembangan untuk platform ini dan kumpulan besar pelanggan potensial membuat mereka semakin menarik bagi pengembang game. Namun, platform ini ditentukan oleh denominator umum terendah dari perangkat keras yang didukung, dan mesin virtual memiliki kinerja yang lebih rendah daripada kode asli pada platform tertentu. Untuk alasan ini, platform tersebut paling cocok untuk game dengan persyaratan grafis sederhana.

Platform juga dapat dicirikan oleh keterbukaannya terhadap pengembangan, yang merupakan perbedaan bisnis atau hukum daripada perbedaan teknis. Misalnya, Windows terbuka dalam arti bahwa alat pengembangan tersedia secara luas, dan tidak ada penjaga gerbang yang mengontrol akses ke pasar game Windows. IPhone Apple adalah platform yang agak lebih terbatas karena semua aplikasi harus melewati proses sertifikasi dan kelas aplikasi tertentu dilarang langsung. Konsol adalah platform game yang paling ketat, di mana akses ke alat pengembangan dikontrol dengan ketat. Ini sedikit terbuka dengan diperkenalkannya pasar game konsol online, yang cenderung lebih terbuka. Contoh yang sangat menarik adalah layanan Game Komunitas Xbox LIVE Microsoft, di mana alat pengembangan tersedia secara bebas dan “gatekeeping” dilakukan terutama oleh peer review. Game yang didistribusikan melalui layanan ini harus menggunakan platform mesin virtual yang disediakan oleh Microsoft untuk alasan keamanan.

Platform game menentukan banyak elemen dari pengalaman game. Misalnya, gamer PC menggunakan keyboard dan mouse, sedangkan gamer konsol menggunakan pengontrol game khusus. Banyak game konsol yang mendukung beberapa pemain di konsol yang sama, baik berbagi layar atau menyediakan windows untuk setiap pemain. Karena kesulitan berbagi keyboard dan mouse, jenis permainan ini tidak ditemukan di PC. Sistem permainan genggam akan memiliki skema kontrol yang berbeda dari telepon layar sentuh, dll.

Meskipun platform permainan sangat bervariasi, beberapa tren umum dapat dilihat. Sebagian besar platform memiliki beberapa inti pemrosesan, dibagi antara tujuan umum (CPU) dan khusus grafis (GPU). Peningkatan kinerja dari waktu ke waktu sebagian besar disebabkan oleh peningkatan jumlah inti; keuntungan dalam kinerja inti individu sederhana. Saat inti GPU semakin umum, garis antara inti GPU dan CPU semakin kabur. Kapasitas penyimpanan cenderung meningkat pada tingkat yang lebih lambat daripada daya pemrosesan, dan bandwidth komunikasi (antara inti serta antara setiap inti dan penyimpanan) masih tumbuh dengan kecepatan yang lebih lambat.

25.2 SUMBER TERBATAS

Salah satu tantangan utama grafis game adalah kebutuhan untuk mengelola banyak kumpulan sumber daya yang terbatas. Setiap platform memberlakukan batasannya sendiri pada sumber daya perangkat keras seperti waktu pemrosesan, penyimpanan, dan bandwidth memori. Pada tingkat yang lebih tinggi, sumber daya pengembangan harus dikelola; ada tim programmer, seniman, dan desainer game berukuran tetap dengan waktu terbatas untuk menyelesaikan game, semoga tanpa bekerja terlalu banyak lembur! Ini perlu diperhitungkan ketika memutuskan teknik grafis mana yang akan diadopsi.

25.3 WAKTU PENGERJAAN

Pengembang game awal hanya perlu khawatir tentang penganggaran satu prosesor. Platform game saat ini berisi beberapa inti CPU dan GPU. Prosesor ini perlu disinkronkan dengan hati-hati untuk menghindari kebuntuan atau kemacetan yang berlebihan.

Karena waktu yang digunakan oleh satu perintah rendering sangat bervariasi, prosesor grafis dipisahkan dari sistem lainnya melalui buffer perintah. Buffer ini bertindak sebagai antrian; perintah disimpan di satu ujung dan GPU membaca perintah rendering dari ujung lainnya. Meningkatkan ukuran buffer ini mengurangi kemungkinan kelaparan GPU. Cukup umum bagi game untuk menyangga seluruh frame yang layak untuk merender perintah sebelum mengirimkannya ke GPU; ini menjamin bahwa kelaparan GPU tidak terjadi. Namun, pendekatan ini membutuhkan ruang penyimpanan yang cukup untuk dua perintah full frame (GPU bekerja pada satu, sementara CPU menyimpan perintah di yang lain). Ini juga meningkatkan latensi antara input pengguna dan tampilan, yang dapat menjadi masalah untuk game yang bergerak cepat.

Anggaran pemrosesan ditentukan oleh kecepatan bingkai, yang merupakan frekuensi di mana buffer bingkai disegarkan dengan rendering scene yang baru. Pada platform tetap (seperti konsol), kecepatan bingkai yang dialami oleh pengguna pada dasarnya sama dengan yang dilihat oleh pengembang game, sehingga batas kecepatan bingkai yang cukup ketat dapat diterapkan. Sebagian besar gim menargetkan kecepatan bingkai 30 bingkai per detik (fps); dalam gim di mana latensi respons sangat penting, targetnya sering kali 60 fps. Pada platform yang sangat bervariasi (seperti PC), anggaran frame-rate (sesuai kebutuhan) didefinisikan lebih longgar.

Frame rate yang dibutuhkan memberikan programmer grafis anggaran tetap per frame untuk bekerja dengan. Dalam kasus target 30fps, core CPU memiliki 33 milidetik untuk mengumpulkan input, memproses logika game, melakukan simulasi fisik apa pun, melintasi deskripsi scene, dan mengirim perintah rendering ke perangkat keras grafis. Secara paralel, tugas-tugas lain seperti pemrosesan audio dan jaringan harus ditangani, dengan waktu respons yang diperlukan sendiri. Saat ini terjadi, GPU biasanya menjalankan perintah grafis yang dikirimkan selama bingkai sebelumnya.

Dalam kebanyakan kasus, inti CPU adalah sumber daya yang homogen; semua inti adalah sama, dan salah satunya sama-sama cocok untuk beban kerja tertentu (ada beberapa pengecualian, seperti prosesor Sel yang digunakan di konsol PLAYSTATION3 Sony).

Sebaliknya, GPU berisi campuran sumber daya yang heterogen, masing-masing khusus untuk serangkaian tugas tertentu. Beberapa dari sumber daya ini terdiri dari perangkat keras dengan fungsi tetap (untuk rasterisasi segitiga, pencampuran alfa, dan pengambilan sampel tekstur), dan beberapa inti yang dapat diprogram. Pada GPU yang lebih lama, inti yang dapat diprogram selanjutnya dibedakan menjadi inti pemrosesan titik dan piksel; desain GPU yang lebih baru memiliki inti shader terpadu yang dapat mengeksekusi semua jenis shader yang dapat diprogram.

Sumber daya heterogen tersebut dianggarkan secara terpisah. Biasanya, pada titik mana pun, hanya satu jenis sumber daya yang akan menjadi hambatan, dan yang lainnya akan memiliki kapasitas berlebih. Di satu sisi, ini bagus, karena kapasitas ini dapat dimanfaatkan untuk meningkatkan kualitas visual tanpa menurunkan performa. Di sisi lain, meningkatkan kinerja menjadi lebih sulit, karena penurunan penggunaan sumber daya non-bottleneck tidak akan berpengaruh. Bahkan mengurangi penggunaan sumber daya bottleneck mungkin hanya sedikit meningkatkan kinerja, tergantung pada tingkat pemanfaatan "bottleneck berikutnya".

25.4 PENYIMPANAN

Platform game, seperti sistem komputasi modern lainnya, memiliki hierarki penyimpanan multi-tahap, dengan jenis memori yang lebih kecil dan lebih cepat di bagian atas dan penyimpanan yang lebih besar dan lebih lambat di bagian bawah. Pengaturan ini lahir dari kebutuhan rekayasa, meskipun hal itu mempersulit kehidupan pengembang. Sebagian besar platform menyertakan penyimpanan cakram optik, yang sangat lambat dan sebagian besar digunakan untuk pengiriman. Pada platform seperti Windows, proses instalasi yang lama dilakukan sekali untuk memindahkan semua data dari disk optik ke hard drive, yang secara signifikan lebih cepat. Disk optik tidak pernah digunakan lagi (kecuali sebagai tindakan anti-pembajakan). Pada platform konsol, hal ini kurang umum, meskipun terkadang terjadi ketika hard drive dijamin ada, seperti pada konsol PLAYSTATION 3 Sony. Lebih sering, hard drive (jika ada) hanya digunakan sebagai cache untuk disk optik.

Langkah selanjutnya dalam hierarki memori adalah RAM, yang pada banyak platform dibagi menjadi RAM sistem umum dan VRAM (RAM video) yang diuntungkan dari antarmuka berkecepatan tinggi ke perangkat keras grafis. Level game mungkin terlalu besar untuk dimasukkan ke dalam RAM, dalam hal ini pengembang game perlu mengatur pemindahan data masuk dan keluar dari RAM sesuai kebutuhan. Pada platform seperti Windows, memori virtual sering digunakan untuk ini. Pada platform konsol, streaming data khusus dan sistem caching biasanya digunakan. Terakhir, baik CPU maupun GPU memiliki berbagai jenis memori dan cache on-chip. Ini adalah standar yang sangat kecil yang biasanya dikelola oleh API grafis.

Sumber daya grafis menghabiskan banyak memori, sehingga menjadi fokus utama anggaran penyimpanan dalam pengembangan game. Tekstur biasanya merupakan konsumen memori terbesar, diikuti oleh geometri (data titik), dan akhirnya jenis data grafis lainnya seperti animasi. Tidak semua memori dapat digunakan untuk grafis—audio juga membutuhkan bit yang cukup tinggi, dan gamelogic dapat menggunakan struktur data yang cukup besar. Dalam hal waktu pemrosesan, penganggaran cenderung agak lebih longgar di Windows, di mana jumlah persis memori yang ada pada sistem pengguna tidak diketahui dan memori virtual mencakup banyak kesalahan. Sebaliknya, penganggaran memori pada platform konsol cukup ketat—seringkali pemrogram utama melacak memori pada spreadsheet dan pemrogram yang membutuhkan lebih banyak memori untuk sistem mereka perlu mengemis, meminjam, atau mencurinya dari orang lain.

Berbagai tingkat hierarki memori tidak hanya berbeda dalam ukuran, tetapi juga dalam kecepatan akses. Ini memiliki dua dimensi terpisah: latensi dan bandwidth. Latensi adalah waktu yang berlalu antara permintaan akses penyimpanan dan pemenuhan akhirnya. Ini bervariasi dari beberapa siklus clock (untuk cache chip) hingga jutaan siklus clock (untuk data yang berada di cakram optik). Latensi biasanya merupakan masalah untuk akses baca (walaupun latensi tulis juga dapat menjadi masalah jika hasilnya perlu dibaca kembali dari memori segera setelahnya). Dalam beberapa kasus, permintaan baca adalah pemblokiran, yang berarti bahwa inti prosesor yang mengirimkannya membaca tidak dapat melakukan hal lain hingga permintaan dipenuhi. Dalam kasus lain, pembacaan tidak memblokir; inti pemrosesan dapat mengirimkan permintaan baca, melakukan jenis pemrosesan lain, dan kemudian menggunakan hasil baca setelah tiba. Akses tekstur oleh GPU adalah contoh non-

blockingreads; aspek penting dari desain GPU adalah menemukan cara untuk "menyembunyikan" latensi baca tekstur dengan melakukan komputasi yang tidak terkait saat pembacaan tekstur dipenuhi.

Untuk penyembunyian latensi ini, harus ada jumlah komputasi yang cukup relatif terhadap akses tekstur. Ini merupakan pertimbangan penting bagi penulis shader; campuran optimal komputasi vs. akses tekstur terus berubah (mendukung lebih banyak komputasi) karena memori gagal untuk mengimbangi peningkatan daya pemrosesan. Bandwidth mengacu pada tingkat maksimum transfer ke dan dari penyimpanan. Biasanya diukur dalam gigabytesper detik.

Sumber Daya Pengembangan

Selain sumber daya perangkat keras, seperti daya pemrosesan dan ruang penyimpanan, pemrogram grafis game juga harus bersaing dengan sumber daya terbatas yang berbeda—waktu rekan satu timnya! Ketika memilih teknik grafis, sumber daya teknik yang diperlukan untuk mengimplementasikan setiap teknik harus diperhitungkan, serta semua alat yang diperlukan untuk menghitung data input (dalam banyak kasus, alat dapat memakan waktu lebih lama daripada menerapkan teknik itu sendiri). Mungkin yang paling penting, produktivitas seniman harus diperhitungkan. Sebagian besar teknik grafis menggunakan aset yang dibuat oleh seniman game, yang sejauh ini merupakan bagian terbesar dari sebagian besar tim game modern. Pemrogram grafis harus mendorong produktivitas dan kreativitas seniman, yang pada akhirnya akan menentukan kualitas visual permainan.

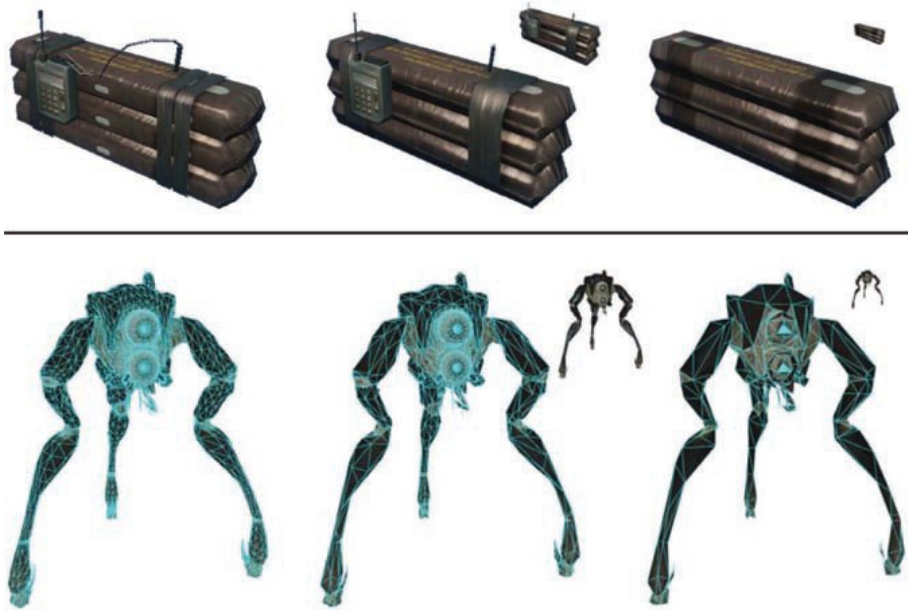
Teknik Optimisasi

Memanfaatkan sumber daya yang terbatas ini secara bijaksana adalah tantangan utama dari programmer grafis game. Untuk tujuan ini, berbagai teknik optimasi sering digunakan. Dalam banyak game, pemrosesan pixel shader adalah hambatan utama. Sebagian besar GPU berisi perangkat keras pemusnah kedalaman hierarkis yang dapat menghindari eksekusi pixel shadersonoccludedsurfaces. Untuk memanfaatkan perangkat keras ini, objek buram dapat ditampilkan dari belakang ke depan. Alternatifnya, penggunaan pemusnahan kedalaman yang optimal dapat dicapai dengan melakukan prepass kedalaman, yaitu, merender semua objek buram ke dalam buffer kedalaman (tanpa output warna atau shader piksel) sebelum merender scene secara normal. Ini memang menimbulkan beberapa overhead (karena kebutuhan untuk merender setiap objek dua kali), tetapi dalam banyak kasus, perolehan kinerja sepadan.

Cara tercepat untuk merender objek adalah dengan tidak merendernya sama sekali; dengan demikian metode apa pun untuk mengetahui sejak awal bahwa suatu objek tertutup dapat berguna. Ini tidak hanya menghemat pemrosesan piksel tetapi juga pemrosesan verteks dan bahkan waktu CPU yang akan dihabiskan untuk mengirimkan objek ke API grafis. View frustum culling (lihat Bagian 8.4.1) digunakan secara universal, tetapi dalam banyak permainan hal itu tidak cukup. Algoritma pemusnahan oklusi tingkat tinggi sering digunakan, memanfaatkan struktur data seperti pohon PVS (set yang berpotensi terlihat) atau BSP (partisi spasial biner) untuk dengan cepat mempersempit kumpulan objek yang berpotensi terlihat.

Bahkan jika suatu objek terlihat, mungkin pada jarak sedemikian rupa sehingga sebagian besar detailnya dapat dihilangkan tanpa efek yang nyata. Algoritma LOD (level-of-detail) membuat representasi objek yang berbeda berdasarkan jarak (atau faktor lain, seperti cakupan layar atau kepentingan). Ini dapat menghemat pemrosesan yang signifikan, khususnya pemrosesan verteks. Contohnya dapat dilihat pada Gambar 25.1.

Dalam banyak kasus, pemrosesan dapat dilakukan bahkan sebelum permainan dimulai. Hasil preprocessing tersebut dapat disimpan dan digunakan setiap frame, sehingga mempercepat permainan. Ini paling sering digunakan untuk penerangan, di mana algoritme penerangan global digunakan untuk menghitung penerangan di seluruh scene dan menyimpannya dalam peta cahaya dan struktur data lainnya untuk digunakan nanti.



Gambar 25.1. Dua contoh objek game pada berbagai tingkat detail. Gambar sisipan kecil menunjukkan ukuran relatif di mana model yang disederhanakan dapat digunakan. Baris atas gambar milik Crytek; baris bawah milik Valve Corp.

25.5 JENIS GAME

Karena persyaratan permainan sangat bervariasi, pemilihan teknik grafis didorong oleh jenis permainan yang sedang dikembangkan. Alokasi waktu pemrosesan sangat bergantung pada kecepatan bingkai. Saat ini, sebagian besar game konsol cenderung menargetkan 30 frame per detik, karena ini memungkinkan kualitas grafis yang jauh lebih tinggi. Namun, jenis game tertentu dengan gameplay yang cepat memerlukan latensi yang sangat rendah, dan game seperti itu biasanya menghasilkan 60 frame per detik. Ini termasuk permainan musik seperti Guitar Hero dan first-person shooters seperti Call of Duty.

Kecepatan bingkai menentukan waktu yang tersedia untuk merender scene. Komposisi scenenya sendiri juga sangat bervariasi dari game ke game. Sebagian besar game memiliki pembagian antara geometri latar belakang (scene, sebagian besar statis) dan geometri latar depan (karakter dan objek dinamis). Ini ditangani secara berbeda oleh mesin rendering. Misalnya, geometri latar belakang sering kali memiliki peta cahaya yang berisi pencahayaan yang telah dihitung sebelumnya, yang tidak layak untuk objek latar depan. Pencahayaan yang dihitung sebelumnya biasanya diterapkan pada objek latar depan melalui beberapa jenis representasi volumetrik yang dapat memperhitungkan perubahan posisi setiap objek dari waktu ke waktu.

Beberapa game memiliki lingkungan yang relatif tertutup, di mana sebagian besar kamera tetap di tempatnya. Contoh paling murni adalah game pertarungan seperti seri Street Fighter, tetapi ini juga berlaku untuk beberapa level untuk game seperti Devil MayCry dan God of War. Permainan ini memiliki kamera yang tidak berada di bawah kendali pemain langsung, dan permainan cenderung berpindah dari satu lingkungan tertutup ke yang lain, menghabiskan banyak waktu bermain di masing-masing. Hal ini memungkinkan pengembang game untuk mencurahkan sejumlah besar sumber daya (pemrosesan, penyimpanan, dan waktu artis) di setiap ruangan atau lingkungan tertutup, menghasilkan tingkat fidelitas grafis yang sangat tinggi.

Game lain memiliki dunia yang sangat besar, di mana pemain dapat bergerak dengan bebas. Ini paling benar untuk "permainan kotak pasir" seperti seri GrandTheftAuto dan permainan peran online seperti World of Warcraft. Permainan seperti itu menimbulkan

tantangan besar bagi pengembang grafis, karena alokasi sumber daya sangat sulit ketika selama setiap frame pemain dapat melihat sebagian besar dunia. Hal-hal rumit lainnya, pemain dapat dengan bebas pergi ke bagian dunia yang sebelumnya jauh dan mengamatinya dari dekat. Permainan seperti itu biasanya memiliki waktu yang berubah dalam sehari, yang membuat perhitungan awal pencahayaan menjadi sulit, jika bukan tidak mungkin.

Kebanyakan game, seperti first-person shooters, berada di antara dua ekstrem tersebut. Pemain dapat melihat cukup banyak scene setiap frame, tetapi pergerakan melalui dunia game agak dibatasi. Banyak game juga memiliki waktu tetap untuk setiap level game, untuk memudahkan perhitungan awal pencahayaan.

Jumlah objek latar depan yang dirender juga sangat bervariasi antar jenis game. Game strategi waktu nyata seperti seri Command and Conquer sering kali memiliki lusinan, jika bukan ratusan, unit yang terlihat di layar. Jenis permainan lain memiliki jumlah karakter yang terlihat lebih terbatas, dengan permainan pertarungan pada ekstrem yang berlawanan, di mana hanya dua karakter yang terlihat, masing-masing ditampilkan dengan detail yang sangat tinggi. Perbedaan harus dibuat antara jumlah karakter yang terlihat setiap saat (yang mempengaruhi penganggaran waktu pemrosesan) dan jumlah karakter unik yang berpotensi dapat terlihat dalam waktu singkat (yang mempengaruhi anggaran penyimpanan).

Gambar 25.2. Crysis mencontohkan grafis realistis dan detail yang diharapkan dari penembak orang pertama. Gambar milik Crytek.

Jenis atau genre permainan juga menentukan ekspektasi penonton terhadap grafis. Misalnya, penembak orang pertama secara historis memiliki tingkat kehalusan grafis yang sangat tinggi, dan harapan ini mendorong desain grafis ketika mengembangkan game baru dalam genre tersebut; lihat Gambar 25.2. Di sisi lain, game puzzle biasanya memiliki grafis yang relatif sederhana, sehingga sebagian besar pengembang game tidak akan menginvestasikan sejumlah besar sumber daya pemrograman atau seni untuk mengembangkan grafis fotorealistik untuk game semacam itu.

Meskipun sebagian besar game bertujuan untuk tampilan fotorealistik, beberapa memang mencoba rendering yang lebih bergaya. Salah satu contoh menarik dari hal ini adalah Okami, yang dapat dilihat pada Gambar 25.3.

Pengelolaan sumber daya pengembangan juga berbeda menurut jenis permainan. Sebagian besar game memiliki siklus pengembangan tertutup selama satu hingga dua tahun, yang berakhir setelah game dikirimkan. Baru-baru ini telah menjadi umum untuk memiliki konten yang dapat diunduh (DLC), yang dapat dibeli setelah game dikirimkan, jadi beberapa sumber daya pengembangan perlu dicadangkan untuk itu. Game online dunia persisten memiliki proses pengembangan tanpa akhir di mana konten baru terus dihasilkan, setidaknya selama game tersebut layak secara ekonomi (yang mungkin membutuhkan waktu beberapa dekade).

Gambar 25.3. Contoh rendering non-fotorealistik yang sangat bergaya dari game Okami. Gambar milik Capcom Entertainment, Inc.

Eksploitasi kreatif dari persyaratan dan batasan spesifik dari game tertentu adalah ciri khas programmer grafis game yang terampil. Contoh yang baik adalah game LittleBigPlanet, yang memiliki dunia game “dua setengah dimensi” yang terdiri dari sejumlah kecil lapisan dua dimensi, serta latar belakang noninteraktif. Kualitas grafis game ini sangat baik, didorong oleh penggunaan teknik rendering yang tidak biasa khusus untuk jenis lingkungan ini; lihat Gambar 25.4.

25.6 PROSES PRODUKSI GAME

Proses produksi game dimulai dengan desain atau konsep game dasar. Dalam beberapa kasus (seperti sekuel), gameplay dasar dan desain visualnya jelas, dan hanya perubahan bertahap yang dibuat. Dalam kasus jenis permainan baru, prototyping ekstensif diperlukan untuk menentukan gameplay dan desain. Sebagian besar kasus duduk di suatu tempat di tengah, di mana ada beberapa elemen gameplay baru dan desain visualnya agak terbuka. Setelah langkah ini mungkin ada tahap lampu hijau di mana beberapa demo atau konsep awal ditampilkan kepada penerbit game untuk mendapatkan persetujuan (dan pendanaan!) untuk game tersebut.

Gambar 25.4. Pengembang LittleBigPlanet berhati-hati dalam memilih teknik yang sesuai dengan batasan permainan, menggabungkan cara-cara yang tidak biasa untuk mencapai hasil yang menakjubkan. LittleBigPlanet ©2007 Sony Computer Entertainment Eropa. Dikembangkan oleh Media Molekul. LittleBigPlanet adalah merek dagang dari Sony Computer Entertainment Europe.

Langkah selanjutnya biasanya pra-produksi. Sementara tim lain bekerja untuk menyelesaikan game terakhir, tim inti kecil bekerja untuk membuat perubahan yang diperlukan pada mesin game dan rantai alat produksi, serta mengerjakan detail kasar dari setiap elemen gameplay baru. Tim inti ini bekerja di bawah tenggat waktu yang ketat. Setelah kapal permainan yang ada dan anggota tim lainnya kembali dari liburan yang memang layak, seluruh rantai alat dan mesin harus siap untuk mereka. Jika tim inti melewatkan tenggat waktu ini, beberapa lusin pengembang dapat dibiarkan menganggur—proposisi yang sangat mahal!

Produksi penuh adalah langkah berikutnya, dengan seluruh tim menciptakan aset seni, merancang level, mengubah gameplay, dan menerapkan perubahan lebih lanjut pada mesin game. Di dunia yang sempurna, semua yang dilakukan selama proses ini akan digunakan dalam game terakhir, tetapi pada kenyataannya ada sifat iteratif untuk pengembangan game yang akan mengakibatkan beberapa pekerjaan dibuang dan dikerjakan ulang. Tujuannya adalah untuk meminimalkan hal ini dengan perencanaan dan pembuatan prototipe yang cermat.

Ketika permainan selesai secara fungsional, tahap akhir dimulai. Istilah rilis alfa biasanya mengacu pada versi yang menandai dimulainya pengujian internal ekstensif, rilis beta untuk versi yang menandai awal pengujian eksternal ekstensif, dan rilis emas ke rilis final yang dikirimkan ke produsen konsol, tetapi perusahaan yang berbeda memiliki definisi istilah yang sedikit berbeda. Bagaimanapun, pengujian, atau jaminan kualitas (QA) adalah bagian penting dari fase ini, dan ini melibatkan penguji di studio pengembangan game, di penerbit, di produsen konsol, dan mungkin juga kontraktor QA eksternal. Berbagai putaran pengujian ini menghasilkan laporan bug yang dikirimkan kembali ke pengembang game dan dikerjakan hingga rilis berikutnya.

Setelah game dikirimkan, sebagian besar pengembang pergi berlibur untuk sementara waktu, tetapi tim kecil mungkin harus tetap bekerja pada tambalan atau konten yang dapat diunduh. Sementara itu, tim inti kecil telah mengerjakan pra-produksi untuk game berikutnya.

Penciptaan aset seni adalah aspek produksi game yang sangat relevan dengan pengembangan grafis, jadi saya akan membahasnya secara mendetail.

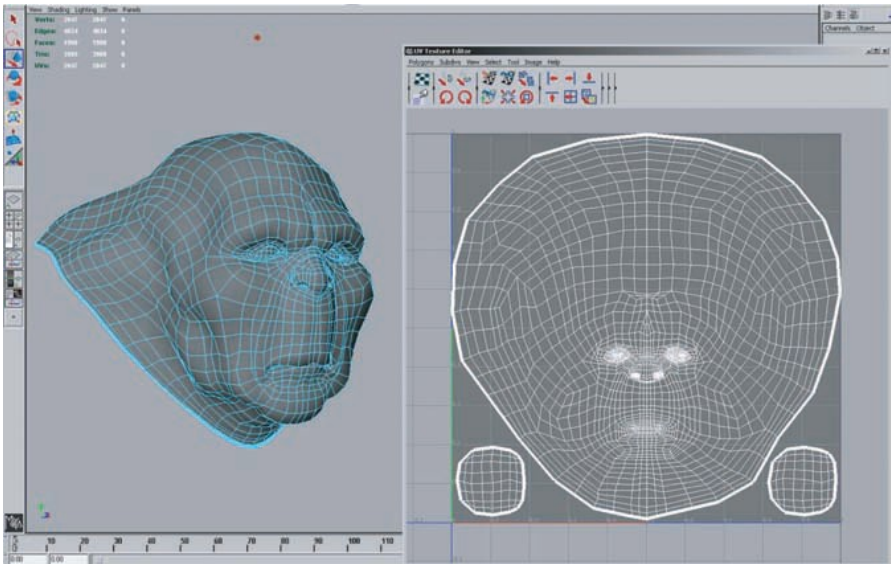
Penciptaan Aset

Sementara proses pembuatan set fartas yang tepat bervariasi dari satu game ke game lainnya, garis besar yang saya berikan di sini cukup representatif. Di masa lalu, seorang seniman tunggal akan menciptakan seluruh aset dari awal hingga akhir, tetapi proses ini sekarang jauh lebih terspesialisasi, melibatkan orang-orang dengan keahlian berbeda yang mengerjakan setiap aset pada waktu yang berbeda. Beberapa dari tahap ini memiliki ketergantungan yang jelas (misalnya, karakter tidak dapat dianimasikan sampai di-rigged dan tidak dapat dicurangi sebelum dimodelkan). Sebagian besar pengembang game memiliki proses persetujuan yang terdefinisi dengan baik, di mana art director atau artis utama menandatangani setiap tahap sebelum aset dikirim ke tahap berikutnya. Idealnya suatu aset melewati setiap tahap tepat satu kali, tetapi dalam praktiknya, perubahan dapat dilakukan yang memerlukan pengiriman ulang.

Modelling Awal

Biasanya proses pembuatan aset seni dimulai dengan memodelkan geometri objek. Langkah ini dilakukan dalam paket pemodelan tujuan umum seperti Maya, MAX atau Softimage. Geometri yang dimodelkan akan diteruskan langsung ke mesin game, jadi penting untuk meminimalkan jumlah simpul sambil mempertahankan siluet yang bagus. Jejaring karakter juga harus dibangun agar sesuai dengan animasi.

Pada tahap ini, parameterisasi permukaan dua dimensi untuk tekstur biasanya dibuat. Adalah penting bahwa parameterisasi ini menjadi sangat kontinu, karena diskontinuitas memerlukan duplikasi simpul dan dapat menyebabkan artefak penyaringan. Contoh mesh dengan parameterisasi tekstur yang terkait ditunjukkan pada Gambar 25.5.



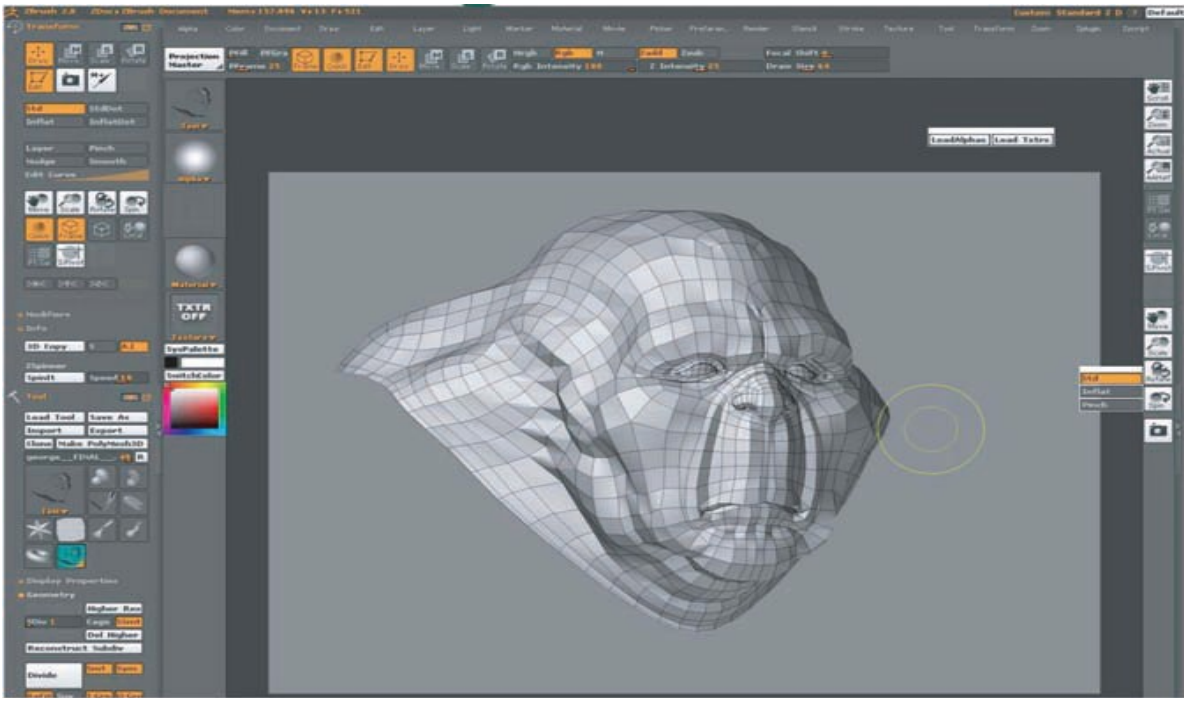
Gambar 25.5. Sebuah mesh dimodelkan dalam Maya, dengan parameterisasi tekstur terkait. Gambar milik Keith Bruns.

Texturing

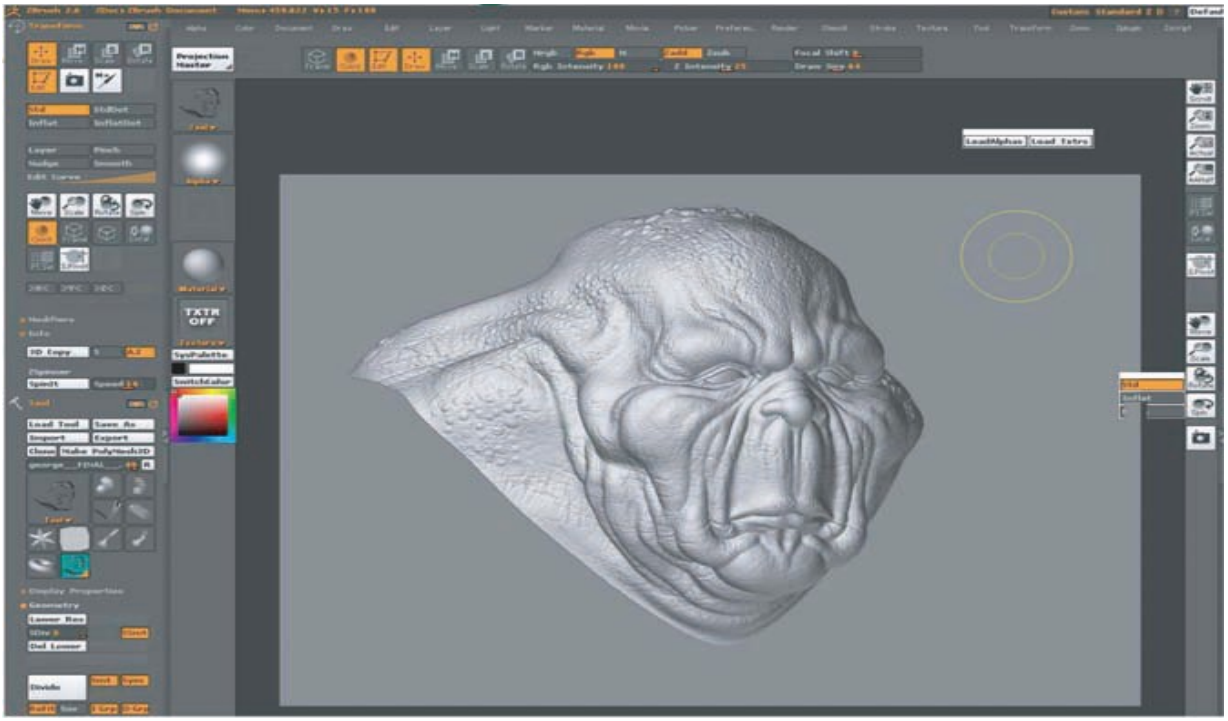
Di masa lalu, texturing adalah proses langsung melukis tekstur warna, biasanya di Photoshop. Sekarang, paket pemodelan detail khusus seperti ZBrush atau Mudbox biasanya digunakan untuk memahat detail permukaan yang halus. Gambar 25.6 dan 25.7 menunjukkan contoh proses ini.

Jika detail tambahan ini direpresentasikan dengan geometri sebenarnya, jutaan segitiga akan dibutuhkan. Sebaliknya, detail biasanya "dipanggang" ke dalam peta normal yang diterapkan pada aslinya, kasar, seperti yang ditunjukkan pada Gambar 25.8 dan 25.9.

Selain peta normal, beberapa tekstur yang berisi properti permukaan seperti warna difus, warna spekular, dan kehalusan (kekuatan spekular) juga dibuat. Ini baik dilukis langsung di permukaan dalam aplikasi pemodelan detail, atau dalam aplikasi dua dimensi seperti Photoshop. Semua peta tekstur ini menggunakan parameterisasi permukaan yang didefinisikan dalam fase pemodelan awal. Ketika tekstur dilukis dalam aplikasi lukisan dua dimensi, seniman harus sering beralih antara aplikasi lukisan dan beberapa aplikasi lain yang dapat menampilkan rendering tiga dimensi dari objek dengan tekstur yang diterapkan. Proses berulang ini diilustrasikan pada Gambar 25.10, 25.11, 25.12, dan 25.13.



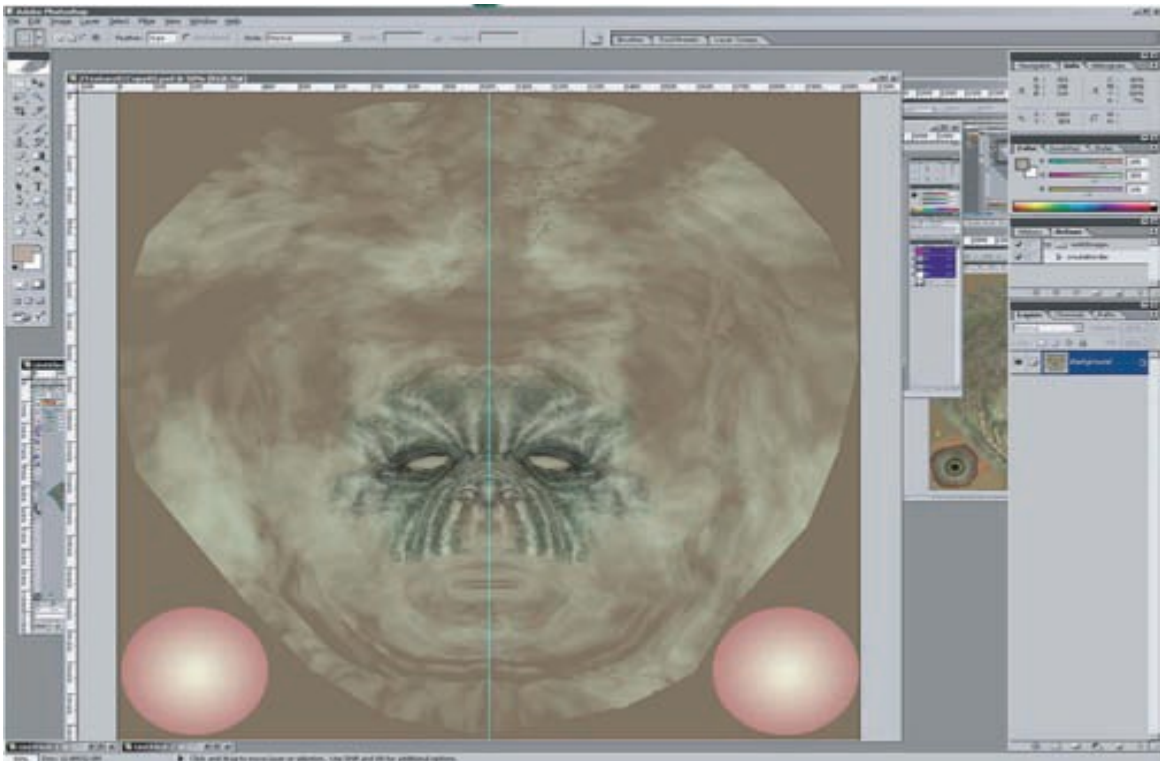
Gambar 25.6 Mesh dari Gambar 25.5 telah dibawa ke ZBrush untuk pemodelan detail. Gambar kesopanan Keith Bruns.



Gambar 25.7. Jaring dari Gambar 25.6, dengan detail halus ditambahkan di ZBrush. Gambar milik Keith Bruns.

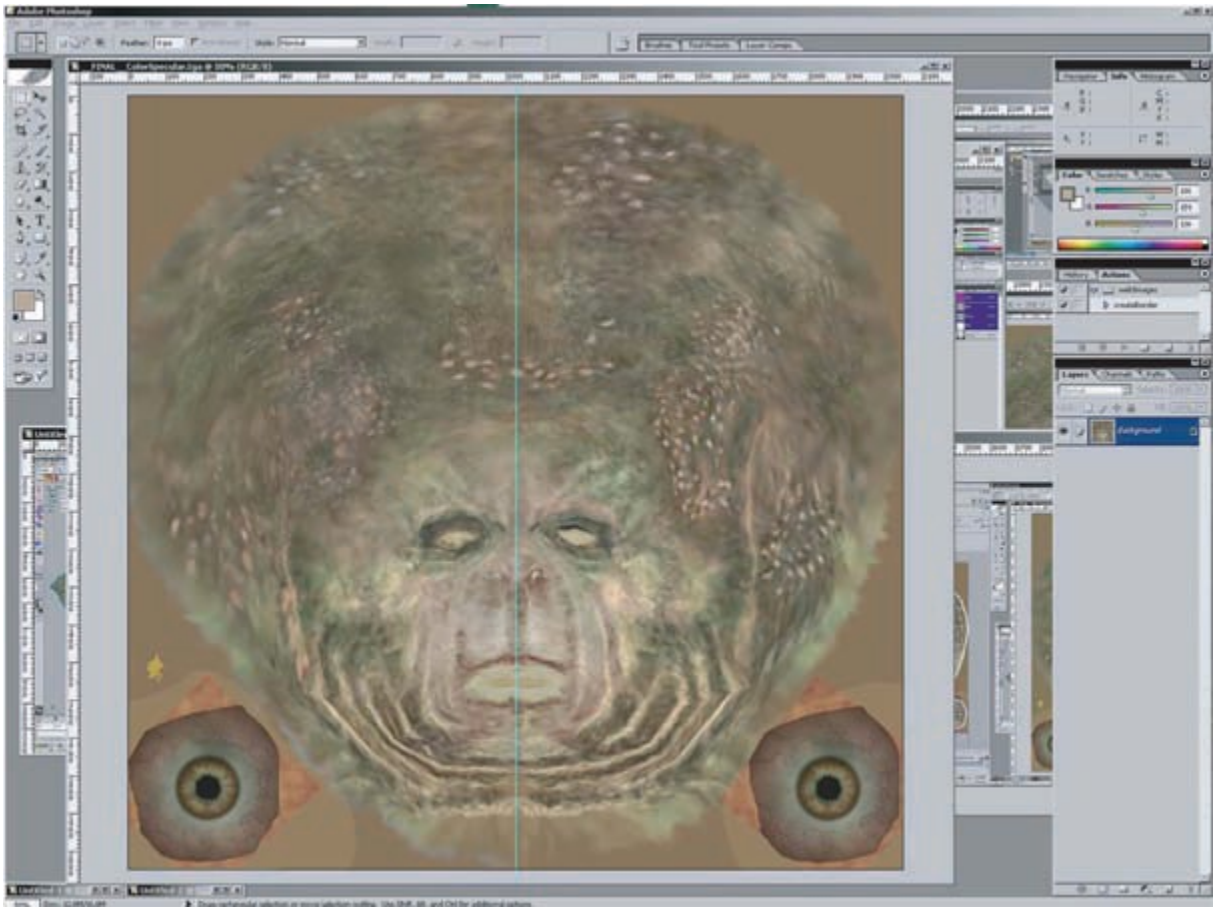
Gambar 25.8. Sebuah visualisasi (dalam ZBrush) mesh dari Gambar 25.6, dirender dengan peta normal yang berasal dari mesh rinci pada Gambar 25.7. Bagian bawah gambar menunjukkan antarmuka untuk alat "Zmapper" ZBrush, yang digunakan untuk menurunkan peta normal. Gambar milik Keith Bruns.

Gambar 25.9. Peta normal yang digunakan pada Gambar 25.8. Dalam gambar ini, saluran tekstur merah, hijau, dan biru berisi koordinat X, Y, dan Z dari normal permukaan. Gambar milik Keith Bruns.

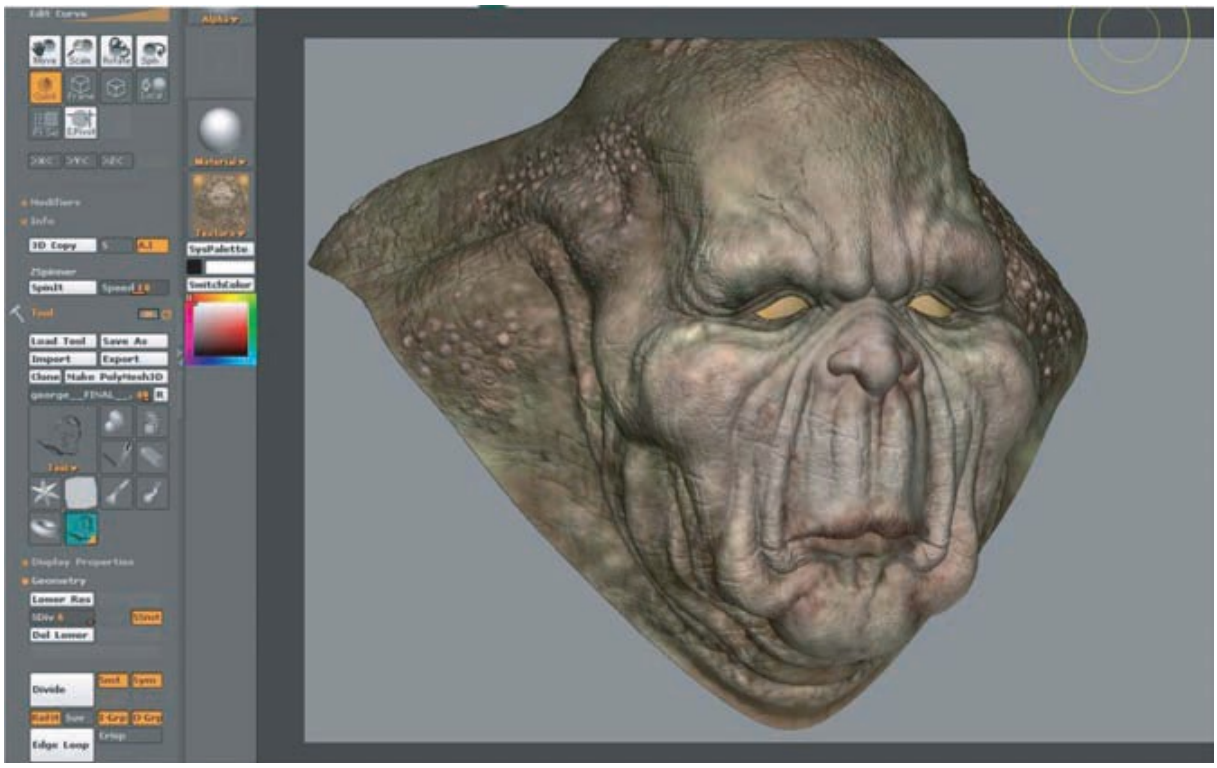


Gambar 25.10. Versi awal tekstur warna difus untuk mesh dari Gambar 25.8, ditampilkan di Photoshop. Gambar milik Keith Bruns.

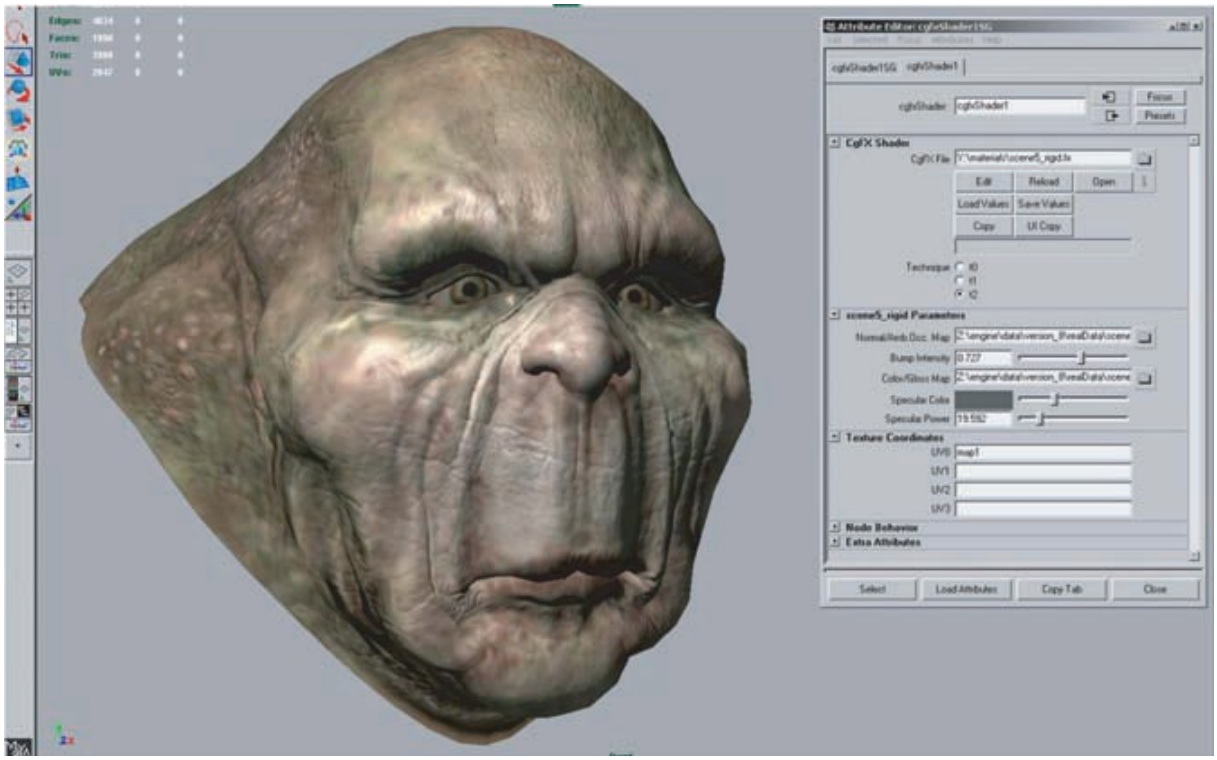
Gambar 25.11. Rendering (dalam ZBrush) mesh dengan peta normal dan tekstur warna difus awal (dari Gambar 25.10) diterapkan. Gambar milik Keith Bruns.



Gambar 25.12. Versi final tekstur warna dari Gambar 25.10. Gambar milik Keith Bruns.



Gambar 25.13. Rendering mesh dengan peta normal dan tekstur warna akhir (dari Gambar 25.12) diterapkan. Gambar milik Keith Bruns.



Gambar 25.14. Konfigurasi shader di Maya. Antarmuka di sebelah kanan digunakan untuk memilih shader, menetapkan tekstur ke input shader, dan mengatur nilai input shader non-tekstur (seperti slider “Specular Color” dan “Specular Power”). Render di sebelah kiri diperbarui secara dinamis sementara properti ini dimodifikasi, memungkinkan umpan balik visual langsung. Gambar milik Keith Bruns.

Shading

Shader biasanya diterapkan pada aplikasi yang sama dengan yang digunakan untuk pemodelan awal. Dalam proses ini, shader (dari set shader yang ditentukan untuk game itu) diterapkan ke mesh. Berbagai tekstur yang dihasilkan dari tahap pemodelan detail diterapkan sebagai input ke shader ini, menggunakan parameterisasi permukaan yang ditentukan selama pemodelan awal. Berbagai input shader lainnya diatur melalui eksperimen visual ("tweaking"); lihat Gambar 25.14.

Lighting

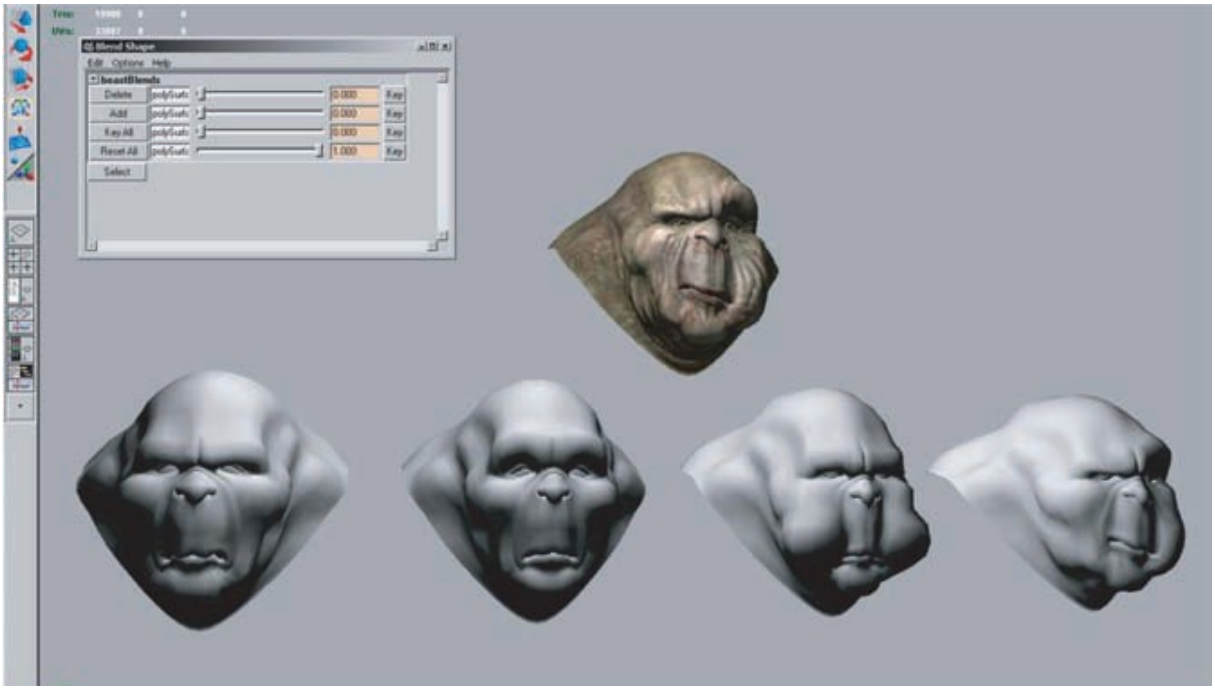
Dalam hal scene latar belakang, seniman pencahayaan biasanya akan memulai pekerjaan mereka setelah pemodelan, tekstur, dan bayangan selesai. Sumber cahaya ditempatkan dan efeknya dihitung dalam langkah pra-pemrosesan. Hasil dari proses ini disimpan dalam lightmaps untuk kemudian digunakan oleh mesin rendering.

Animasi

Jejaring karakter mengalami beberapa langkah tambahan yang terkait dengan animasi. Metode utama yang digunakan untuk menganimasikan karakter game adalah menguliti. Ini membutuhkan rig, yang terdiri dari hierarki node transformasi yang dilampirkan ke karakter, proses yang dikenal sebagai rigging. Area efek dari setiap node transformasi dilukis ke subset simpul mesh. Terakhir, animator membuat animasi yang menggerakkan, memutar, dan menskalakan node transformasi ini, "menyeret" jaring di belakangnya.

Karakter permainan yang khas akan memiliki banyak lusinan animasi, sesuai dengan mode gerakan yang berbeda (berjalan, berlari, berputar) serta berbagai tindakan seperti serangan. Dalam kasus karakter utama, jumlah animasi bisa mencapai ratusan. Transisi antara animasi yang berbeda juga perlu didefinisikan.

Untuk animasi wajah, teknik lain, yang disebut target morf, terkadang digunakan. Dalam teknik ini, simpul tema dimanipulasi secara langsung untuk merusak bentuk jala. Salinan yang berbeda dari mesh yang cacat disimpan (misalnya, untuk ekspresi wajah yang berbeda) dan digabungkan oleh mesin game saat runtime. Pembuatan morph targetsis ditunjukkan pada Gambar 25.15.



Gambar 25.15. Antarmuka target morf di Maya. Baris bawah menunjukkan empat target morf yang berbeda, dan model di atas menunjukkan efek menggabungkan beberapa target

morf secara bersamaan. Antarmuka di kiri atas digunakan untuk mengontrol sejauh mana setiap target morf diterapkan. Gambar milik Keith Bruns.

25.7 CATATAN

Ada banyak sekali informasi tentang rendering waktu-nyata dan pemrograman game yang tersedia, baik dalam buku maupun online. Berikut adalah beberapa sumber yang dapat saya rekomendasikan dari pengenalan pribadi.

GameDeveloperMagazine adalah sumber informasi yang baik tentang pengembangan game, seperti juga slide dari pembicaraan yang diberikan pada Konferensi Pengembang Game/*Game Developers Conference* (GDC) tahunan dan konferensi Microsoft's Gamefest. Seri buku GPU Gems dan ShaderX juga berisi informasi yang bagus—semua yang pertama dan dua yang pertama juga tersedia secara online.

Eric Lengyel's Mathematics for 3DGame Programming & Computer Graphics, sekarang dalam edisi kedua, adalah referensi yang baik untuk berbagai jenis matematika yang digunakan dalam grafis dan permainan. Area spesifik dari pemrograman game yang terkait erat dengan grafis adalah deteksi tabrakan, di mana Deteksi Tabrakan Waktu Nyata Christer Ericson adalah sumber daya definitifnya.

Sejak edisi pertama pada tahun 1999, Rendering RealTime Eric Haines dan Tomas Akenine-Mo¨ller telah berusaha untuk mencakup bidang yang berkembang pesat ini secara menyeluruh. Sebagai penggemar lama buku ini, saya senang mendapat kesempatan menjadi rekan penulis di edisi ketiga, yang keluar pada pertengahan 2008.

Membaca saja tidak cukup—pastikan Anda memainkan berbagai permainan secara teratur untuk mendapatkan gambaran yang baik tentang persyaratan berbagai jenis permainan, serta keadaan seni saat ini.

25.8 LATIHAN

1. Periksa visual dari dua game yang berbeda. Perbedaan apa yang dapat Anda simpulkan dalam persyaratan grafis dari kedua game tersebut? Analisis pengaruhnya pada waktu rendering, anggaran penyimpanan, dll.

BAB 26

VISUALISASI

Area aplikasi utama grafis komputer adalah visualisasi, di mana gambar yang dihasilkan komputer digunakan untuk membantu orang memahami data spasial dan nonspasial. Visualisasi digunakan ketika tujuannya adalah untuk meningkatkan kemampuan manusia dalam situasi di mana masalah tidak cukup didefinisikan dengan baik untuk komputer untuk menangani secara algoritmik. Jika solusi yang sepenuhnya otomatis dapat sepenuhnya menggantikan penilaian manusia, maka visualisasi biasanya tidak diperlukan. Visualisasi dapat digunakan untuk menghasilkan hipotesis baru ketika menjelajahi kumpulan data yang sama sekali tidak dikenal, untuk mengkonfirmasi hipotesis yang ada dalam kumpulan data yang dipahami sebagian, atau untuk menyajikan informasi tentang kumpulan data yang diketahui kepada audiens lain.

Visualisasi memungkinkan orang untuk memuat kognisi ke sistem persepsi, menggunakan gambar yang dirancang dengan cermat sebagai bentuk memori eksternal. Sistem visual manusia adalah saluran bandwidth yang sangat tinggi ke otak, dengan sejumlah besar pemrosesan yang terjadi secara paralel dan pada tingkat pra-sadar. Dengan demikian kita dapat menggunakan gambar eksternal sebagai pengganti untuk melacak hal-hal di dalam kepala kita sendiri. Sebagai contoh, mari kita pertimbangkan tugas untuk memahami hubungan antara subset topik dalam buku bagus Godel, Escher, Bach: The Eternal Golden Braid (Hofstadter, 1979); lihat Gambar 26.1.

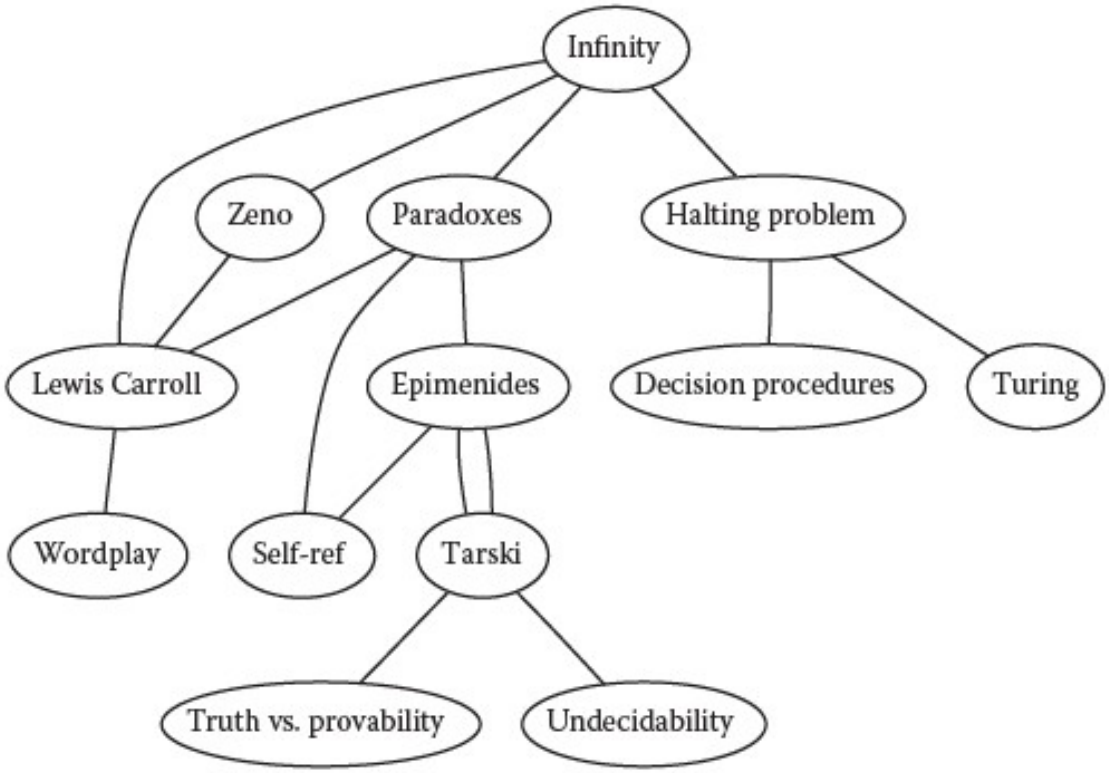
Ketika kita melihat kumpulan data sebagai daftar teks, pada tingkat rendah kita harus membaca kata-kata dan membandingkannya dengan ingatan dari kata-kata yang telah dibaca sebelumnya. Sulit untuk melacak hanya selusin topik ini menggunakan kognisi dan memori saja, apalagi ratusan topik dalam buku lengkap. Masalah tingkat yang lebih tinggi dalam mengidentifikasi lingkungan, misalnya menemukan semua topik yang jauh dari Paradoks topik target, sangat sulit.

Infinity - Lewis Carroll	Epimenides - Self-ref
Infinity - Zeno	Epimenides - Tarski
Infinity - Paradoxes	Tarski - Epimenides
Infinity - Halting problem	Halting problem - Decision procedures
Zeno - Lewis Carroll	Halting problem - Turing
Paradoxes - Lewis Carroll	Lewis Carroll - Wordplay
Paradoxes - Epimenides	Tarski - Truth vs. provability
Paradoxes - Self-ref	Tarski - Undecidability

Gambar 26.1. Melacak hubungan antar topik sulit dilakukan dengan menggunakan daftar teks.

Gambar 26.2 menunjukkan representasi visual eksternal dari dataset yang sama sebagai grafis node-link, di mana setiap topik adalah node dan hubungan antara dua topik ditunjukkan langsung dengan garis. Mengikuti garis dengan menggerakkan mata kita di sekitar gambar adalah operasi tingkat rendah yang cepat dengan beban kognitif minimal, sehingga penemuan lingkungan tingkat yang lebih tinggi menjadi mungkin. Penempatan node dan routing link antara mereka dibuat secara otomatis oleh program menggambar grafis titik (Gansner, Koutsofois, North, & Vo, 1993).

Kami menyebut mapping atribut kumpulan data ke representasi visual sebagai pengkodean visual. Salah satu masalah utama dalam visualisasi adalah memilih pengkodean yang tepat dari ruang besar kemungkinan representasi visual, dengan mempertimbangkan karakteristik sistem persepsi manusia, kumpulan data yang bersangkutan, dan tugas yang ada.



Gambar 26.2. Mengganti persepsi untuk kognisi dan memori memungkinkan kita untuk memahami hubungan antara topik buku dengan cepat.

26.1 BACKGROUND

Sejarah

Orang-orang memiliki sejarah panjang dalam menyampaikan makna melalui gambar statis, berasal dari lukisan gua tertua yang diketahui lebih dari tiga puluh ribu tahun yang lalu. Kami terus berkomunikasi secara visual hari ini dengan cara mulai dari sketsa kasar di belakang serbet hingga desain grafis iklan yang apik. Selama ribuan tahun, kartografer telah mempelajari masalah pembuatan peta yang mewakili beberapa aspek dunia di sekitar kita. Representasi visual pertama dari kumpulan data abstrak dan nonspasial dibuat pada abad ke-18 oleh William Playfair (Friendly, 2008).

Meskipun kami telah memiliki kekuatan untuk membuat gambar bergerak selama lebih dari seratus lima puluh tahun, membuat gambar dinamis secara interaktif adalah perkembangan yang lebih baru yang hanya dimungkinkan oleh ketersediaan luas perangkat keras dan algoritme grafis komputer yang cepat dalam beberapa dekade terakhir. Visualisasi statis kumpulan data kecil dapat dibuat dengan tangan, tetapi grafis komputer memungkinkan visualisasi interaktif kumpulan data besar.

26.2 KETERBATASAN SUMBER DAYA

Saat merancang sistem visualisasi, kita harus mempertimbangkan tiga jenis batasan yang berbeda: kapasitas komputasi, kapasitas persepsi dan kognitif manusia, dan kapasitas

tampilan. Seperti halnya aplikasi grafis komputer, waktu dan memori komputer adalah sumber daya yang terbatas dan kita sering mengalami kendala yang sulit. Jika sistem visualisasi perlu memberikan respon interaktif, maka harus menggunakan algoritma yang dapat berjalan dalam sepersekian detik daripada menit atau jam.

Di sisi manusia, memori dan perhatian harus dianggap sebagai sumber daya yang terbatas. Ingatan manusia sangat terbatas, baik untuk ingatan jangka panjang maupun untuk ingatan kerja jangka pendek. Kemudian dalam bab ini, kita membahas beberapa kekuatan dan keterbatasan mekanisme perhatian visual tingkat rendah yang melakukan pemrosesan paralel besar-besaran dari bidang visual. Kami menyimpan sedikit informasi secara internal dalam memori kerja visual, membuat kami rentan terhadap perubahan kebutaan, fenomena di mana bahkan perubahan yang sangat besar tidak diperhatikan jika kami memperhatikan sesuatu yang lain dalam pandangan kami (Simons, 2000). Selain itu, kewaspadaan juga merupakan sumber daya yang sangat terbatas; kemampuan kita untuk melakukan tugas pencarian visual menurun dengan cepat, dengan hasil yang jauh lebih buruk setelah beberapa jam daripada dalam beberapa menit pertama (Ware, 2000).

Kapasitas tampilan adalah jenis batasan ketiga yang perlu dipertimbangkan. Perancang visualisasi sering “kehabisan piksel”, di mana resolusi layar tidak cukup besar untuk menampilkan semua informasi yang diinginkan secara bersamaan. Kepadatan informasi dari bingkai tertentu adalah ukuran jumlah informasi yang dikodekan versus jumlah ruang yang tidak digunakan. Ada tradeoff antara manfaat menampilkan sebanyak mungkin sekaligus, untuk meminimalkan kebutuhan navigasi dan eksplorasi, dan biaya menampilkan terlalu banyak sekaligus, di mana pengguna diliputi oleh kekacauan visual.

26.3 JENIS DATA

Banyak aspek dari desain visualisasi didorong oleh jenis data yang perlu kita perhatikan. Misalnya, bilangan yang dapat diambil, rangkaian hubungan antar item, atau data spasial yang melekat seperti lokasi di permukaan bumi atau kumpulan dokumen?

Kita mulai dengan mempertimbangkan tabel data. Kami menyebut rowsitems data dan kolom adalah dimensi, juga dikenal sebagai atribut. Misalnya, baris dapat mewakili orang, dan kolom dapat berupa nama, usia, tinggi badan, ukuran baju, dan buah favorit.

Kami membedakan antara tiga jenis dimensi: kuantitatif, teratur, dan kategoris. Data kuantitatif, seperti usia atau tinggi badan, adalah numerik dan kita dapat melakukan aritmatika di atasnya. Misalnya, jumlah 68 inci dikurangi 42 inci adalah 26 inci. Dengan data terurut, seperti ukuran baju, kita tidak dapat melakukan aritmatika bermata penuh, tetapi ada pengurutan yang terdefinisi dengan baik. Misalnya, medium minus besar bukanlah konsep yang berarti, tetapi kita tahu bahwa medium berada di antara kecil dan besar. Data kategori, seperti buah atau nama favorit, tidak memiliki urutan implisit. Kita hanya bisa membedakan apakah dua hal itu sama (apel) atau berbeda (apel vs pisang).

Data relasional, orgraf, adalah tipe data lain di mana tidak ada desa yang terhubung melalui tautan. Salah satu jenis grafis tertentu adalah pohon, yang biasanya digunakan untuk data hierarkis. Baik node dan edge dapat memiliki atribut terkait. Grafis kata sayangnya kelebihan beban dalam visualisasi. Graf simpul-tautan yang kita bahas di sini, mengikuti terminologi penggambaran graf dan teori graf, bisa juga disebut jaringan. Dalam bidang grafis statistik, grafis sering digunakan untuk grafis, seperti pada grafis garis untuk data deret waktu yang ditunjukkan pada Gambar 26.10.

Beberapa data secara inheren spasial, seperti lokasi geografis atau bidang pengukuran pada posisi dalam ruang tiga dimensi dalam MRI atau CT scan yang digunakan oleh dokter untuk melihat struktur internal tubuh seseorang. Informasi yang terkait dengan setiap titik dalam ruang dapat berupa himpunan besaran skalar yang tidak terurut, atau vektor terindeks,

atau tensor. Sebaliknya, data nonspasial dapat dikodekan secara visual menggunakan posisi spasial, tetapi pengkodean tersebut dipilih oleh perancang daripada diberikan secara implisit dalam semantik dataset itu sendiri. Pilihan ini adalah salah satu masalah paling sentral dan sulit dari desain visualisasi.

26.4 DIMENSI DAN JUMLAH BARANG

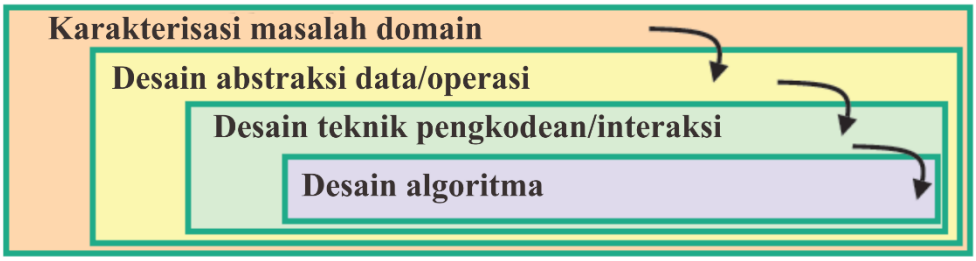
Jumlah dimensi data yang perlu dikodekan secara visual merupakan salah satu aspek paling mendasar dari masalah desain visualisasi. Teknik yang bekerja untuk kumpulan data berdimensi rendah dengan beberapa kolom akan sering gagal untuk kumpulan data berdimensi sangat tinggi dengan lusinan atau ratusan kolom. Dimensi ada mungkin memiliki struktur hierarki, misalnya dengan kumpulan data deret waktu di mana terdapat pola yang menarik pada beberapa skala temporal.

Jumlah item data juga penting: visualisasi yang berkinerja baik untuk beberapa ratus item sering kali tidak berskala hingga jutaan item. Dalam beberapa kasus, kesulitannya murni algoritmik, di mana perhitungan akan memakan waktu terlalu lama; di lain itu adalah masalah persepsi yang lebih dalam bahwa bahkan algoritma instan tidak bisa memecahkan, di mana kekacauan visual membuat representasi tidak dapat digunakan oleh seseorang. Rentang nilai yang mungkin dalam suatu dimensi mungkin juga relevan.

Transformasi Data dan Dimensi Turunan

Data sering diubah dari satu jenis ke jenis lainnya sebagai bagian dari saluran visualisasi untuk memecahkan masalah domain. Misalnya, dimensi data asli mungkin terdiri dari data kuantitatif: angka titik mengambang yang mewakili suhu. Untuk beberapa tugas, seperti menemukan anomali dalam pola cuaca lokal, data mentah dapat digunakan secara langsung. Untuk tugas lain, seperti memutuskan apakah air adalah suhu yang sesuai untuk mandi, data dapat diubah menjadi dimensi yang teratur: panas, hangat, atau dingin. Dalam transformasi ini, sebagian besar detail dikumpulkan. Dalam contoh ketiga, saat membuat roti panggang, transformasi yang lebih lossy menjadi dimensi kategoris mungkin cukup: dibakar atau tidak.

Prinsip mengubah data menjadi dimensi turunan, daripada hanya mengkodekan data secara visual dalam bentuk aslinya, adalah ide yang kuat. Pada Gambar 26.10, data asli adalah kumpulan terurut dari kurva deret waktu. Transformasi adalah mengelompokkan data, mengurangi jumlah informasi untuk dikodekan secara visual menjadi beberapa kurva yang sangat berarti.



Gambar 26.3. Empat lapisan validasi bersarang untuk visualisasi.

26.5 PROSES DESAIN HUMAN-CENTER

Proses desain visualisasi dapat dibagi menjadi satu set cascading lapisan, seperti yang ditunjukkan pada Gambar 26.3. Semua lapisan ini saling bergantung satu sama lain; output dari level di atas adalah input ke level di bawahnya.

Karakterisasi Tugas

Kumpulan data tertentu memiliki banyak kemungkinan pengkodean visual. Memilih pengkodean visual mana yang akan digunakan dapat dipandu oleh kebutuhan khusus dari beberapa pengguna yang dituju. Pertanyaan, atau tugas yang berbeda, memerlukan pengkodean visual yang sangat berbeda. Misalnya, pertimbangkan domain rekayasa perangkat lunak. Tugas memahami cakupan rangkaian tes didukung dengan baik oleh antarmuka Tarantula yang ditunjukkan pada Gambar 26.11. Namun, tugas memahami dekomposisi modular dari perangkat lunak saat refactoring kode mungkin lebih baik dilayani dengan menunjukkan struktur hirarkis lebih langsung sebagai node-linkgraph.

Memahami persyaratan beberapa audiens target adalah masalah yang rumit. Dalam pendekatan desain yang berpusat pada manusia, desainer visualisasi bekerja dengan sekelompok pengguna target dari waktu ke waktu (C. Lewis & Rieman, 1993). Dalam kebanyakan kasus, pengguna tahu bahwa mereka perlu entah bagaimana melihat data mereka tetapi tidak dapat secara langsung mengartikulasikan kebutuhan mereka sebagai tugas yang jelas dalam hal operasi pada tipe data. Proses desain berulang mencakup pengumpulan informasi dari pengguna target tentang masalah mereka melalui wawancara dan pengamatan mereka di tempat kerja, membuat prototipe, dan mengamati bagaimana pengguna berinteraksi dengan prototipe tersebut untuk melihat seberapa baik solusi yang diusulkan benar-benar bekerja. Metodologi rekayasa perangkat lunak analisis kebutuhan juga dapat berguna (Kovitz, 1999).

Abstraksi

Setelah masalah domain tertentu telah diidentifikasi di lapisan pertama, lapisan berikutnya membutuhkan abstraksi ke dalam representasi yang lebih umum sebagai operasi pada tipe data yang dibahas di bagian sebelumnya. Masalah dari domain yang sangat berbeda dapat dipetakan ke abstraksi visualisasi yang sama. Operasi generik ini termasuk penyortiran, penyaringan, karakterisasi tren dan distribusi, menemukan anomali dan outlier, dan menemukan korelasi (Amar, Eagan, & Stasko, 2005). Mereka juga mencakup operasi yang khusus untuk tipe data tertentu, misalnya mengikuti jalur untuk data relasional dalam bentuk grafis atau pohon.

Langkah abstraksi ini sering melibatkan transformasi data dari data mentah asli ke dalam dimensi turunan. Dimensi turunan ini sering kali bertipe berbeda dari data aslinya: graf dapat diubah menjadi pohon, data tabular dapat diubah menjadi graf dengan menggunakan ambang batas untuk memutuskan apakah tautan harus ada berdasarkan nilai bidang, dan seterusnya.

Teknik dan Desain Algoritma

Setelah abstraksi dipilih, lapisan berikutnya adalah merancang teknik pengkodean visual dan interaksi yang sesuai. Bagian 26.4 mencakup prinsip-prinsip pengkodean visual, dan kami membahas prinsip-prinsip interaksi di Bagian 26.5. Kami menyajikan teknik yang mempertimbangkan prinsip-prinsip ini di Bagian 26.6 dan 26.7. Sayangnya, pembahasan rinci tentang algoritma visualisasi berada di luar cakupan bab ini.

Validasi

Masing-masing dari empat lapisan memiliki persyaratan validasi yang berbeda. Lapisan pertama dirancang untuk menentukan apakah masalah dicirikan dengan benar: apakah benar-benar ada audiens target yang melakukan tugas tertentu yang akan mendapat manfaat dari alat yang diusulkan? Cara langsung untuk menguji asumsi dan dugaan adalah dengan mengamati atau mewawancarai anggota audiens target, untuk memastikan bahwa perancang visualisasi sepenuhnya memahami tugas mereka. Pengukuran yang tidak dapat dilakukan sampai alat dibangun dan digunakan adalah untuk memantau tingkat adopsi dalam

komunitas itu, meskipun tentu saja banyak faktor lain selain utilitas yang mempengaruhi adopsi.

Lapisan berikutnya digunakan untuk menentukan apakah abstraksi dari masalah domain ke dalam operasi pada tipe data tertentu benar-benar memecahkan masalah yang diinginkan. Setelah prototipe atau alat jadi telah digunakan, studi lapangan dapat dilakukan untuk mengamati apakah dan bagaimana itu digunakan oleh audiens yang dituju. Selain itu, gambar yang dihasilkan oleh sistem dapat dianalisis baik secara kualitatif maupun kuantitatif.

Tujuan dari lapisan ketiga adalah untuk memverifikasi bahwa pengkodean visual dan teknik interaksi yang dipilih oleh perancang secara efektif mengkomunikasikan abstraksi yang dipilih kepada pengguna. Tes langsung adalah untuk membenarkan bahwa pilihan desain individu tidak melanggar prinsip persepsi dan kognitif yang diketahui. Pembenaran seperti itu diperlukan tetapi tidak cukup, karena desain visualisasi melibatkan banyak pertukaran antara pilihan yang berinteraksi. Setelah sistem dibangun, dapat diuji melalui studi laboratorium formal di mana banyak orang diminta untuk melakukan tugas yang diberikan sehingga pengukuran waktu yang diperlukan bagi mereka untuk menyelesaikan tugas dan tingkat kesalahan mereka dapat dianalisis secara statistik.

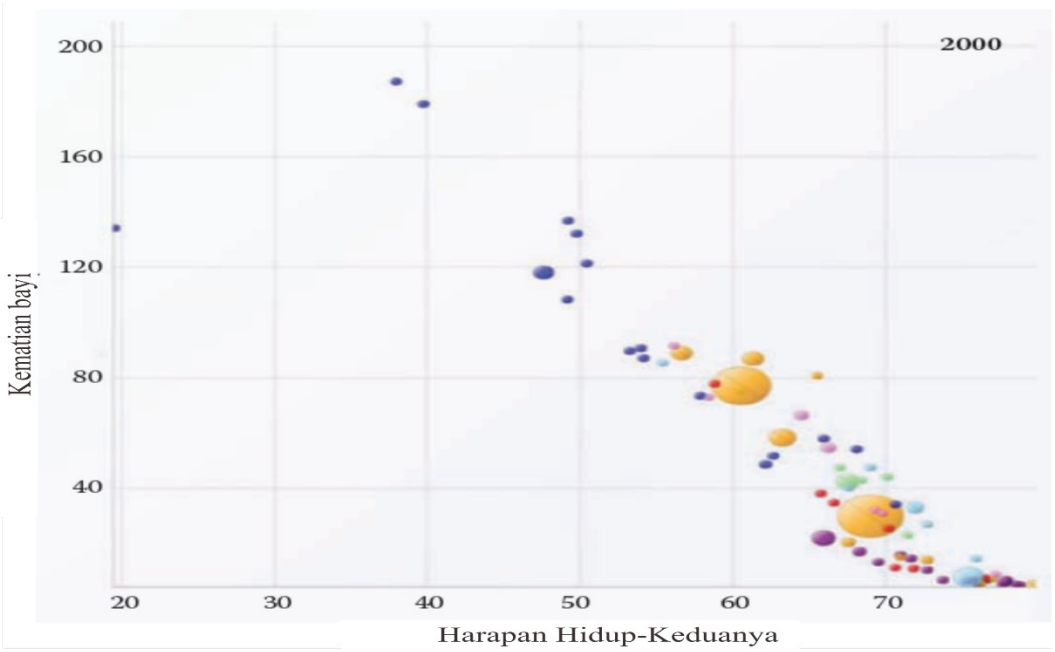
Lapisan keempat digunakan untuk memverifikasi bahwa algoritme yang dirancang untuk melakukan pengkodean dan pilihan interaksi lebih cepat atau membutuhkan lebih sedikit memori daripada algoritme sebelumnya. Tes langsung adalah menganalisis kompleksitas komputasi dari algoritma yang diusulkan. Setelah implementasi, kinerja waktu aktual dan penggunaan memori dari sistem dapat diukur secara langsung.

26.6 PRINSIP PENGKODEAN VISUAL

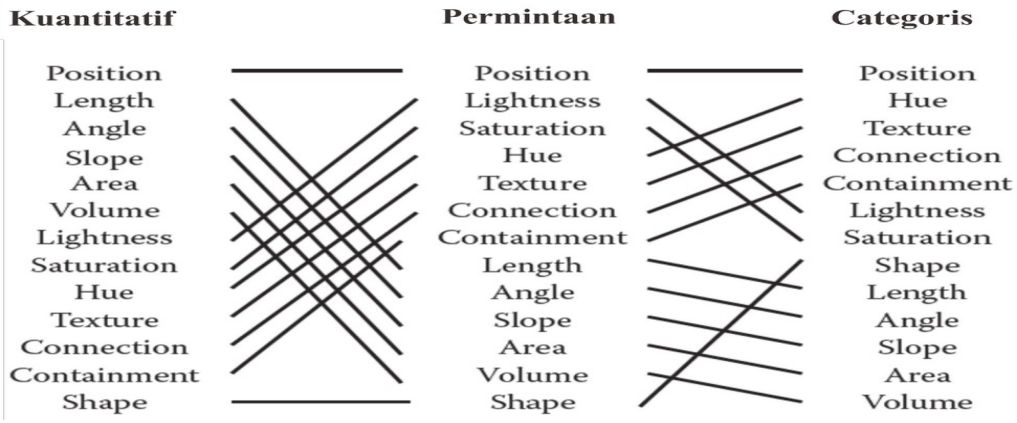
Kita dapat menggambarkan pengkodean visual sebagai elemen grafis, yang disebut tanda, yang menyampaikan informasi melalui saluran visual. Tanda berdimensi nol adalah titik, tanda satu dimensi adalah garis, tanda dua dimensi adalah luas, dan tanda tiga dimensi adalah volume. Banyak saluran visual dapat mengkodekan informasi, termasuk posisi spasial, warna, ukuran, bentuk, orientasi, dan arah gerakan. Beberapa saluran visual dapat digunakan untuk secara bersamaan mengkodekan dimensi data yang berbeda; misalnya, Gambar 26.4 menunjukkan penggunaan posisi spasial horizontal dan vertikal, warna, dan ukuran untuk menampilkan empat dimensi data. Lebih dari satu saluran dapat digunakan untuk mengkodekan dimensi yang sama secara berlebihan, untuk desain yang menampilkan lebih sedikit informasi tetapi menunjukkannya dengan lebih jelas.

Karakteristik Saluran Visual

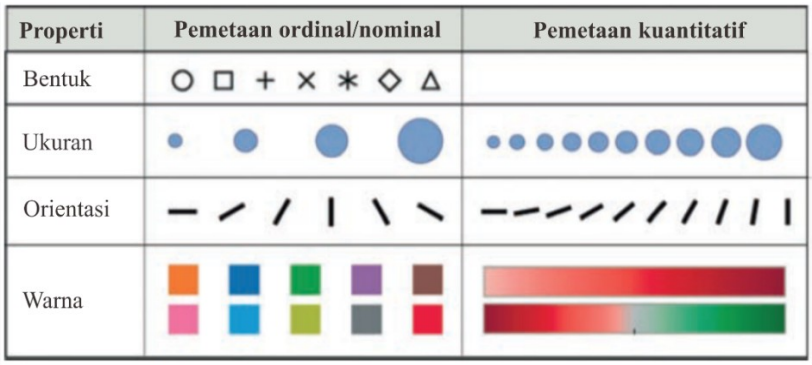
Karakteristik penting dari saluran visual adalah kemampuan membedakan, keterpisahan, dan popout. Saluran tidak semuanya sama-sama dapat dibedakan. Banyak eksperimen psikofisik telah dilakukan untuk mengukur kemampuan orang untuk membuat perbedaan yang tepat tentang informasi yang dikodekan oleh saluran visual yang berbeda. Kemampuan kita bergantung pada apakah tipe datanya kuantitatif, berurutan, atau kategoris. Gambar 26.5 menunjukkan peringkat saluran visual untuk ketiga tipe data. Gambar 26.6 menunjukkan beberapa mapping default untuk saluran visual dalam sistem Tableau/Polaris, yang memperhitungkan tipe data.



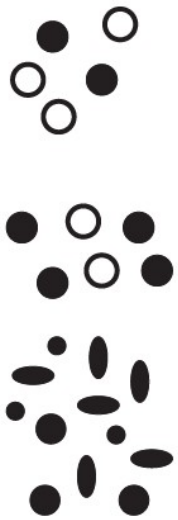
Gambar 26.4. Empat saluran visual posisi spasial horizontal dan vertikal, warna, dan ukuran digunakan untuk mengkodekan informasi dalam bagan sebar ini Gambar milik George Robertson (Robertson, Fernandez, Fisher, Lee, & Stasko, 2008), © IEEE 2008.



Gambar 26.5. Kemampuan kita untuk menerima informasi yang dikodekan oleh saluran visual tergantung pada jenis data yang digunakan, dari yang paling akurat di bagian atas hingga paling sedikit di bagian bawah. Digambar ulang dan diadaptasi dari (Mackinlay, 1986).



Gambar 26.6. Mapping default sistem Tableau/Polaris untuk empat saluran visual menurut tipe data. Gambar milik Chris Stolte (Stolte, Tang, & Hanrahan, 2008), © 2008 IEEE.



Gambar 26.7. Warna dan lokasi adalah saluran yang dapat dipisahkan yang cocok untuk mengkodekan dimensi data yang berbeda, tetapi saluran ukuran horizontal dan vertikal secara otomatis menyatu menjadi persepsi area yang terintegrasi. Digambar ulang setelahnya (Ware, 2000).

Posisi spasial adalah saluran visual yang paling akurat untuk ketiga jenis data, dan mendominasi persepsi kita tentang pengkodean visual. Dengan demikian, dua dimensi data yang paling penting sering dipetakan ke posisi spasial horizontal dan vertikal.

Namun, saluran lain sangat berbeda antara jenis. Saluran-saluran panjang dan sudut sangat dapat dibedakan untuk data kuantitatif tetapi buruk untuk keterurutan dan kategoris, sedangkan rona kontras sangat akurat untuk data kategorikal tetapi biasa-biasa saja untuk data kuantitatif.

Kami harus selalu mempertimbangkan apakah ada kecocokan yang baik antara rentang dinamis yang diperlukan untuk menunjukkan dimensi data dan rentang dinamis yang tersedia di saluran. Misalnya, pengkodean dengan lebar garis menggunakan tanda satu dimensi dan saluran ukuran. Ada sejumlah langkah lebar terbatas yang dapat kita gunakan dengan andal untuk menyandikan informasi secara visual: ketipisan minimum satu piksel ditentukan oleh resolusi layar (mengabaikan antialiasing untuk menyederhanakan diskusi ini), dan ada ketebalan maksimum yang melebihi batas yang akan dicapai objek. dianggap sebagai poligon daripada garis. Lebar garis dapat bekerja dengan sangat baik untuk memperlihatkan tiga atau empat nilai berbeda dalam dimensi data, tetapi itu akan menjadi pilihan yang buruk untuk lusinan atau ratusan nilai.

Beberapa saluran visual merupakan bagian integral, menyatu bersama pada tingkat pra-kesadaran, sehingga bukan merupakan pilihan yang baik untuk mengkodekan dimensi data yang berbeda secara visual. Lainnya dapat dipisahkan, tanpa interaksi di antara mereka selama pemrosesan visual, dan aman digunakan untuk pengkodean beberapa dimensi. Gambar 26.7 menunjukkan dua pasangan saluran. Warna dan posisi sangat dapat dipisahkan. Kita dapat melihat bahwa ukuran horizontal dan ukuran vertikal tidak begitu mudah untuk dipisahkan, karena sistem visual kita secara otomatis mengintegrasikannya ke dalam persepsi area yang seragam. Ukuran berinteraksi dengan banyak saluran: semakin kecil ukuran suatu objek, semakin sulit membedakan bentuk atau warnanya.

Kita dapat memilih saluran secara selektif sehingga item dari jenis tertentu "muncul" secara visual, seperti yang dibahas dalam Bagian 20.4.3. Contoh visual popout adalah ketika kita langsung melihat item merah di tengah lautan biru, atau membedakan lingkaran dari kotak. Popout visual sangat kuat dan skalabel karena muncul secara paralel, tanpa perlu

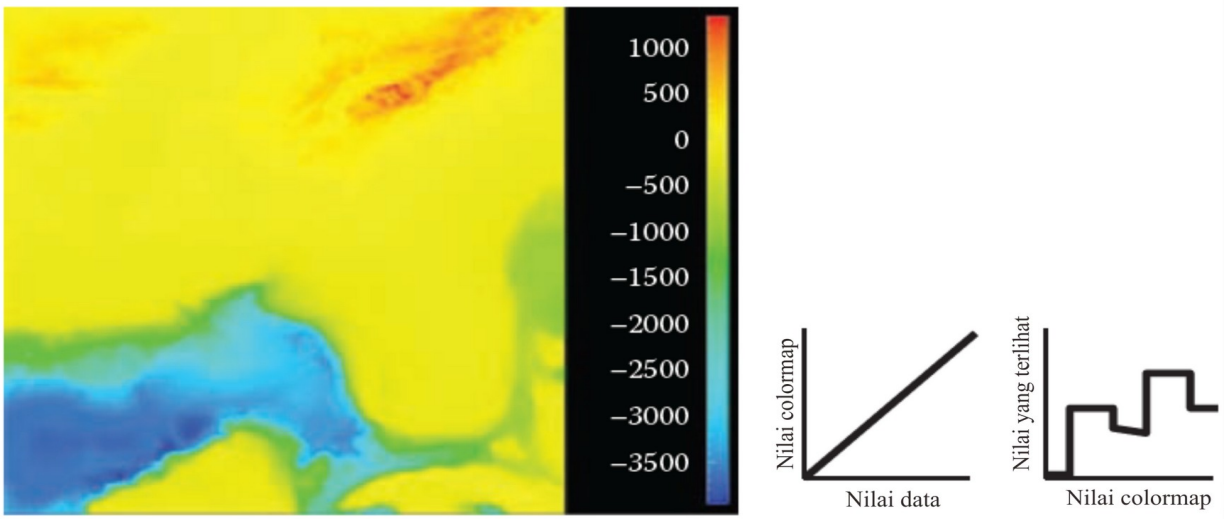
memproses item satu per satu secara sadar. Banyak saluran visual memiliki properti popout ini, termasuk tidak hanya daftar di atas tetapi juga kelengkungan, icker, kedalaman stereoskopik, dan bahkan arah pencahayaan. Namun, secara umum kami hanya dapat memanfaatkan popout untuk satu saluran dalam satu waktu. Misalnya, lingkaran putih tidak muncul dari sekelompok lingkaran dan kotak yang bisa berwarna putih atau hitam, seperti yang ditunjukkan pada Gambar 20.46. Ketika kita perlu mencari di lebih dari satu saluran secara bersamaan, lamanya waktu yang dibutuhkan untuk menemukan objek target tergantung secara linier pada jumlah objek di tempat kejadian.

26.7 WARNA

Warna bisa menjadi saluran yang sangat kuat, tetapi banyak orang tidak memahami sifat-sifatnya dan menggunakannya secara tidak benar. Seperti yang dibahas dalam Bagian 20.2.2, kita dapat mempertimbangkan warna dalam tiga saluran visual yang terpisah: rona, saturasi, dan kecerahan. Ukuran wilayah sangat mempengaruhi kemampuan kita untuk merasakan warna. Warna di daerah kecil relatif sulit untuk dilihat, dan desainer harus menggunakan warna-warna cerah dan sangat jenuh untuk memastikan bahwa kode warna dapat dibedakan. Situasi kebalikannya benar ketika daerah berwarna besar, seperti di latar belakang, di mana warna pastel saturasi rendah harus digunakan untuk menghindari menyilaukan penonton.

Hueisa sangat kuat untuk mengkodekan data kategorikal. Namun, rentang dinamis yang tersedia sangat terbatas. Orang hanya dapat membedakan sekitar selusin rona dengan andal saat area berwarna kecil dan tersebar di sekitar layar. Pedoman yang baik untuk pengkodean warna adalah menjaga jumlah kategori kurang dari delapan, dengan mengingat bahwa latar belakang dan warna objek netral juga dihitung secara total.

Untuk data terurut, lightness dan saturation efektif karena memiliki urutan persepsi implisit. Orang dapat dengan andal memesan dengan ringan, selalu menempatkan abu-abu di antara hitam dan putih. Dengan saturasi, orang dengan andal menempatkan warna merah muda yang kurang jenuh antara merah jenuh penuh dan putih saturasi nol. Namun, hue tidak sebaik saluran untuk data yang dipesan karena tidak memiliki pemesanan persepsi implisit. Ketika diminta untuk membuat urutan warna merah, biru, hijau, dan kuning, orang tidak semuanya memberikan jawaban yang sama. Orang dapat dan belajar konvensi, seperti hijau-kuning-merah untuk lampu lalu lintas, atau urutan warna dalam pelangi, tetapi konstruksi ini berada pada tingkat yang lebih tinggi daripada persepsi murni. Data yang dipesan biasanya ditampilkan dengan serangkaian nilai warna yang berbeda.



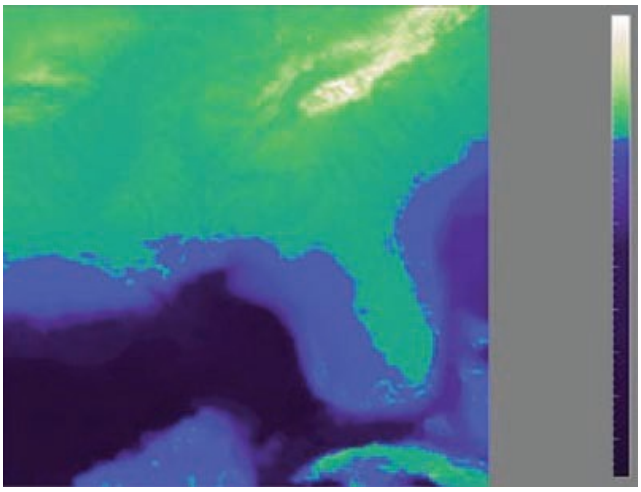
Gambar 26.8. Peta warna pelangi standar memiliki dua cacat: menggunakan rona untuk menunjukkan urutan, dan tidak isolinear secara persepsi. Gambar milik Bernice Rogowitz.

Data kuantitatif ditampilkan dengan peta warna, rentang nilai warna yang dapat kontinu atau diskrit. Sebuah default yang sangat disayangkan dalam banyak paket perangkat lunak adalah peta warna pelangi, seperti yang ditunjukkan pada Gambar 26.8. Skala pelangi standar mengalami tiga masalah. Pertama, hue digunakan untuk menunjukkan keteraturan. Pilihan yang lebih baik adalah menggunakan lightness karena memiliki urutan persepsi implisit. Bahkan yang lebih penting, mata manusia merespon paling kuat terhadap pencahayaan. Kedua, skalanya tidak linier secara persepsi: langkah yang sama dalam rentang kontinu tidak dianggap sebagai langkah yang sama oleh mata kita. Gambar 26.8 menunjukkan contoh, di mana peta warna pelangi mengaburkan data. Sementara rentang dari 2000 hingga 1000 memiliki tiga warna berbeda (sian, hijau, dan kuning), rentang dengan ukuran yang sama dari 1000 hingga0 hanya terlihat kuning di seluruh bagian. Grafis di sebelah kanan menunjukkan bahwa nilai yang dirasakan sangat terkait dengan luminansi, yang dicatat bahkan meningkat secara monoton dalam skala ini.

Sebaliknya, Gambar 26.9 menunjukkan data yang sama dengan peta warna yang lebih sesuai, di mana tingkat kecerahan meningkat secara monoton. Hue digunakan untuk membuat kategorisasi yang bermakna secara semantik: penonton dapat mendiskusikan struktur dalam kumpulan data, seperti laut biru tua, landas kontinen cyan, dataran rendah hijau, dan pegunungan putih.

Dalam kasus diskrit dan kontinu, peta warna harus memperhitungkan apakah data berurutan atau divergen. Aplikasi ColorBrewer (www.colorbrewer.org) merupakan sumber daya yang sangat baik untuk konstruksi peta warna (Brewer, 1999).

Masalah penting lainnya ketika pengkodean dengan warna adalah bahwa sebagian besar populasi, kira-kira 10% pria, kekurangan warna merah-hijau. Jika pengkodean menggunakan merah dan hijau dipilih karena konvensi di domain target, pengkodean lightness atau saturasi yang berlebihan selain hue adalah bijaksana. Alat-alat seperti situs web http://www.vischeck.com harus digunakan untuk memeriksa apakah skema warna dapat dibedakan dengan orang-orang dengan gangguan penglihatan warna.



Gambar 26.9. Struktur kumpulan data yang sama jauh lebih jelas dengan peta warna di mana peningkatan kecerahan secara monoton digunakan untuk menunjukkan pemesanan dan rona digunakan sebagai gantinya untuk mengelompokkan ke dalam wilayah kategoris. Gambar milik Bernice Rogowitz.

26.8 TATA LETAK SPASIAL 2D VS. 3D

Pertanyaan apakah menggunakan dua atau tiga saluran untuk posisi spasial telah dipelajari secara ekstensif. Ketika visualisasi berbasis komputer dimulai pada akhir 1980-an, dan grafis 3D interaktif adalah kemampuan baru, ada banyak antusiasme untuk representasi

3D. Sebagai bidang matang, peneliti mulai memahami biaya pendekatan 3D ketika digunakan untuk dataset abstrak (Ware, 2001).

Oklusi, di mana beberapa bagian dari kumpulan data tersembunyi di belakang yang lain, merupakan masalah utama dengan 3D. Meskipun algoritme penghilangan permukaan tersembunyi seperti zbuffers dan pohon BSP memungkinkan komputasi cepat dari gambar 2D yang benar, orang masih harus mensintesis banyak gambar ini ke dalam peta mental internal. Ketika orang melihat scene realistis yang dibuat dari objek yang familiar, biasanya mereka dapat dengan cepat memahami apa yang mereka lihat. Namun, ketika mereka melihat kumpulan data yang tidak dikenal, di mana pengkodean visual yang dipilih memetakan dimensi abstrak ke dalam posisi spasial, memahami detail yang sesuai dengan struktur 3D dapat menjadi tantangan bahkan ketika mereka dapat menggunakan kontrol navigasi interaktif untuk mengubah sudut pandang 3D mereka. Penyebabnya sekali lagi adalah keterbatasan kapasitas memori kerja manusia (Plumlee & Ware, 2006).

Masalah lain dengan 3D adalah distorsi perspektif. Meskipun objek dunia nyata memang tampak lebih kecil ketika mereka berada lebih jauh dari mata kita, pemendekan membuat perbandingan langsung dari ketinggian objek menjadi sulit (Tory, Kirkpatrick, Atkins, & Mo¨ller, 2006). Sekali lagi, meskipun kita sering kali dapat menilai ketinggian dari objek yang sama di dunia nyata berdasarkan pengkodean visual yang lengkap, kita tidak dapat selalu melakukannya dengan data yang benar-benar dapat dipahami berdasarkan pengalaman masa lalu. Misalnya, lebih sulit untuk menilai tinggi batang dalam bagan batang 3D daripada dalam beberapa bagan batang 2D yang disejajarkan secara horizontal.

Masalah lain dengan representasi 3D yang tidak dibatasi adalah bahwa teks pada orientasi arbitrer dalam ruang 3D jauh lebih sulit untuk dibaca daripada teks yang disejajarkan dalam bidang gambar 2D (Grossman, Wigdor, & Balakrishnan, 2007).

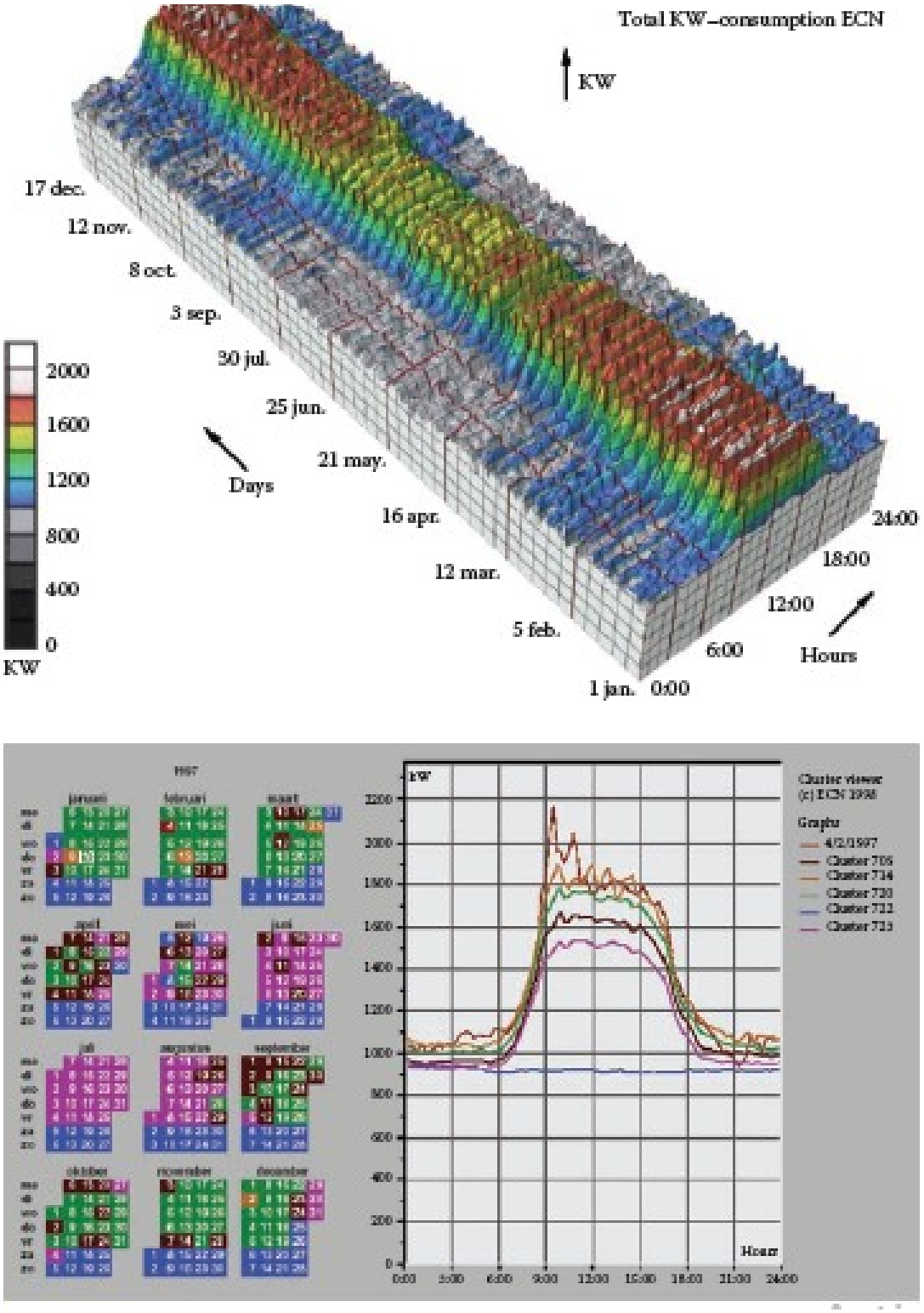
Gambar 26.10 mengilustrasikan bagaimana tampilan 2D yang dipilih dengan cermat dari kumpulan data abstrak dapat menghindari masalah dengan oklusi dan distorsi perspektif yang melekat pada tampilan 3D. Tampilan atas menunjukkan representasi 3D yang dibuat langsung dari data deret waktu asli, di mana setiap penampang adalah kurva deret waktu 2D yang menunjukkan konsumsi daya untuk satu hari, dengan satu kurva untuk setiap hari dalam setahun di sepanjang sumbu ketiga yang diekstrusi. Meskipun representasi ini mudah dibuat, kita hanya dapat melihat pola skala besar seperti konsumsi yang lebih tinggi selama jam kerja dan variasi musiman antara musim dingin dan musim panas. Untuk membuat tampilan tertaut 2D di bagian bawah, kurva dikelompokkan secara hierarkis, dan hanya kurva agregat yang mewakili kelompok atas yang digambar ditumpangkan dalam bingkai 2D yang sama. Perbandingan langsung antara ketinggian kurva setiap saat sepanjang hari adalah mudah karena tidak ada distorsi atau oklusi perspektif. Kode warna yang sama digunakan dalam tampilan kalender, yang sangat efektif untuk memahami pola temporal.

Sebaliknya, jika kumpulan data terdiri dari data spasial 3D, seperti menunjukkan aliran cairan di atas sayap pesawat terbang atau kumpulan data pencitraan medis dari pemindaian MRI, maka biaya tampilan 3D sebanding dengan manfaatnya dalam membantu pengguna membangun model struktur kumpulan data yang berguna.

26.9 LABEL TEKS

Teks dalam bentuk label dan legenda merupakan faktor yang sangat penting dalam menciptakan visualisasi yang bermanfaat daripada sekadar cantik. Kapak dan tanda centang harus diberi label. Legenda harus menunjukkan arti warna, apakah digunakan sebagai tambalan diskrit atau di landai warna berkelanjutan. Item individual dalam kumpulan data biasanya memiliki label teks bermakna yang terkait dengannya. Dalam banyak kasus, menampilkan semua label setiap saat akan menghasilkan terlalu banyak kekacauan visual,

sehingga label dapat ditampilkan untuk subset item menggunakan algoritme pemosisian label yang menampilkan label pada kepadatan yang diinginkan sambil menghindari tumpang tindih (Luboschik, Schumann, & Cords , 2008). Cara langsung untuk memilih label terbaik untuk mewakili sekelompok item adalah menggunakan algoritma serakah berdasarkan beberapa ukuran kepentingan label, tetapi mensintesis label baru berdasarkan karakteristik grup tetap menjadi masalah yang sulit. Pendekatan yang lebih berpusat pada interaksi adalah hanya menampilkan label untuk masing-masing item berdasarkan indikasi interaktif dari pengguna.



Gambar 26.10. Atas: Representasi 3D dari kumpulan data deret waktu ini memperkenalkan masalah oklusi dan distorsi perspektif. Bawah: Tampilan 2D terkait dari kurva agregat turunan dan kalender memungkinkan perbandingan langsung dan menunjukkan pola yang lebih halus. Gambar milik Jarke van Wijk (van Wijk & van Selow, 1999), © 1999 IEEE.

26.10 PRINSIP INTERAKSI

Beberapa prinsip interaksi penting ketika merancang visualisasi. Umpan balik visual latensi rendah memungkinkan pengguna untuk menjelajahi lebih lancar, misalnya dengan menunjukkan lebih banyak detail ketika kursor hanya mengarahkan kursor ke suatu objek daripada mengharuskan pengguna untuk mengklik secara eksplisit. Memilih item adalah operasi mendasar saat berinteraksi dengan kumpulan data besar, seperti yang secara visual menunjukkan kumpulan yang dipilih dengan penyorotan. Pengodean warna adalah bentuk penyorotan yang umum, tetapi saluran lain juga dapat digunakan.

Banyak bentuk interaksi yang dapat dipertimbangkan dalam aspek tampilan apa yang mereka ubah. Navigasi dapat dianggap sebagai perubahan viewport. Sortasi adalah perubahan urutan spasial; yaitu, mengubah cara data dipetakan ke saluran visual posisi spasial. Seluruh pengkodean visual juga dapat diubah.

Ikhtisar Pertama, Zoom dan Filter, Detail Sesuai Permintaan

Mantra yang berpengaruh "Tinjauan dulu, zoom dan filter, detail sesuai permintaan" (Shneiderman, 1996) menjelaskan peran interaksi dan navigasi dalam desain visualisasi. Ikhtisar membantu pengguna melihat daerah-daerah di mana penyelidikan lebih lanjut mungkin produktif, baik melalui navigasi spasial atau melalui penyaringan. Seperti yang akan kita bahas di bawah ini, detail dapat ditampilkan dalam banyak cara: dengan pop-up dari mengklik atau kursor melayang, di windows terpisah, dan dengan mengubah tata letak pada y untuk memberi ruang untuk menampilkan informasi tambahan.

26.11 BIAYA INTERAKTIVITAS

Interaktivitas memiliki kekuatan dan biaya. Manfaat interaksi adalah bahwa orang dapat menjelajahi ruang informasi yang lebih besar daripada yang dapat dipahami dalam satu gambar statis. Namun, biaya untuk interaksi adalah bahwa hal itu membutuhkan waktu dan perhatian manusia. Jika pengguna harus memeriksa setiap kemungkinan secara mendalam, penggunaan sistem visualisasi dapat berubah menjadi pencarian bertenaga manusia. Secara otomatis mendeteksi fitur yang menarik untuk secara eksplisit menarik perhatian pengguna melalui pengkodean visual adalah tujuan yang berguna bagi desainer visualisasi. Namun, jika tugas yang ada dapat diselesaikan sepenuhnya dengan cara otomatis, pertama-tama diperlukan visualisasi. Jadi, selalu ada kompromi antara menemukan aspek yang dapat diotomatisasi dan mengandalkan manusia dalam lingkaran untuk mendeteksi pola.

26.12 ANIMASI

Animasi menunjukkan perubahan menggunakan waktu. Kami membedakan animasi, di mana frame yang berurutan hanya dapat dimainkan, dijeda, atau dihentikan, dari kontrol interaktif yang sebenarnya. Ada banyak bukti bahwa transisi animasi bisa lebih efektif daripada potongan melompat, dengan membantu orang melacak perubahan posisi objek atau sudut pandang kamera (Heer & Robertson, 2007). Meskipun animasi bisa sangat efektif untuk narasi dan penceritaan, sering kali digunakan secara tidak efektif dalam konteks visualisasi (Tversky, Morrison, & Betrancourt, 2002). Mungkin tampak jelas untuk menunjukkan data yang berubah seiring waktu dengan menggunakan animasi, modalitas visual yang berubah seiring waktu. Namun, orang mengalami kesulitan dalam membuat perbandingan khusus antara frame individu yang tidak bersebelahan ketika mereka melihat animasi yang terdiri dari banyak frame. Kapasitas memori visual manusia yang sangat terbatas berarti lebih banyak menggunakan kursi untuk membandingkan kenangan dari hal-hal yang telah kita lihat di masa lalu daripada membandingkan hal-hal yang ada di bidang pandang kita saat ini. Untuk tugas-tugas yang membutuhkan perbandingan antara beberapa lusin frame, perbandingan

berdampingan seringkali lebih efektif daripada animasi. Selain itu, jika jumlah objek yang berubah antar frame besar, orang akan kesulitan melacak semua yang terjadi (Robertson et al., 2008). Animasi naratif dirancang dengan cermat untuk menghindari terlalu banyak tindakan yang terjadi secara bersamaan, sedangkan kumpulan data yang divisualisasikan tidak memiliki batasan seperti itu. Untuk kasus khusus hanya dua frame dengan jumlah perubahan yang terbatas, animasi yang sangat sederhana dari bolak-balik antara keduanya dapat menjadi cara yang berguna untuk mengidentifikasi perbedaan di antara keduanya.

Tampilan Gabungan dan Berdekatan

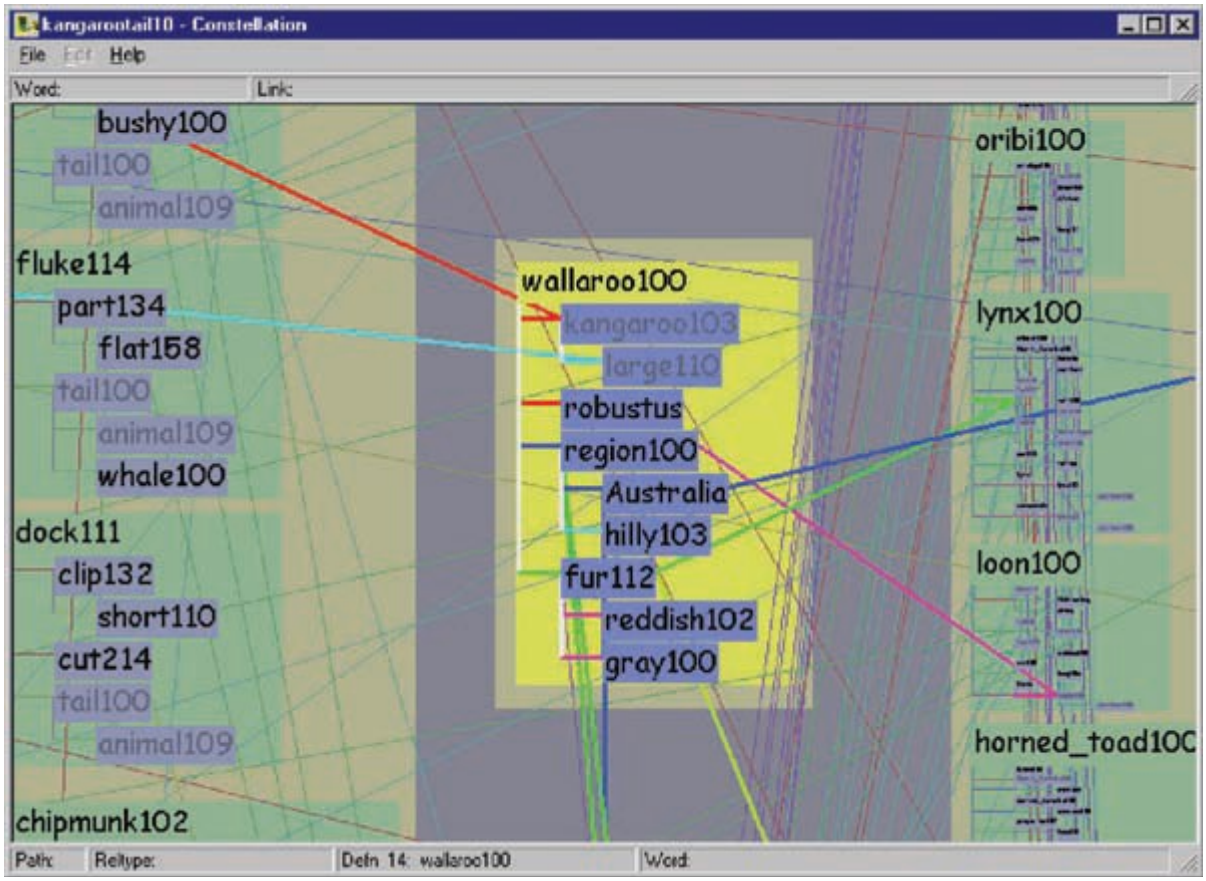
Pilihan pengkodean visual yang sangat mendasar adalah apakah memiliki tampilan komposit tunggal yang menunjukkan semuanya dalam bingkai atau windows yang sama, atau memiliki beberapa tampilan yang berdekatan satu sama lain.

26.13 GAMBAR TUNGGAL

Ketika hanya ada satu atau dua dimensi data yang akan dikodekan, maka posisi spasial horizontal dan vertikal adalah saluran visual yang jelas untuk digunakan, karena kita memahaminya dengan paling akurat dan posisi memiliki pengaruh paling kuat pada model internal kita dari kumpulan data. Tampilan grafis statistik tradisional dari diagram garis, diagram batang, dan plot sebar semuanya menggunakan urutan spasial dari tanda untuk mengkodekan informasi. Tampilan ini dapat ditambah dengan saluran visual tambahan, seperti warna dan ukuran dan bentuk, seperti pada scatterplot yang ditunjukkan pada Gambar 26.4.

Tanda paling sederhana yang mungkin adalah satu piksel. Dalam tampilan berorientasi piksel, tujuan adalah untuk memberikan gambaran umum tentang banyaknya item yang mungkin. Pendekatan ini menggunakan posisi spasial dan saluran warna pada kepadatan informasi yang tinggi, tetapi menghalangi penggunaan saluran ukuran dan bentuk. Gambar 26.11 menunjukkan alat visualisasi perangkat lunak Tarantula (Jones et al., 2002), di mana sebagian besar layar dikhususkan untuk ikhtisar source code menggunakan garis tinggi satu piksel (Eick, Steffen, & Sumner, 1992). Warna dan kecerahan setiap baris menunjukkan apakah garis tersebut lolos, gagal, atau memiliki hasil yang beragam saat menjalankan serangkaian kasus uji.

Gambar 26.11. Tarantula menunjukkan ikhtisar source code menggunakan kode warna garis satu piksel dengan status eksekusiofasoftware test suite. Gambar milik John Stasko (Jones,Harrold, & Stasko, 2002).



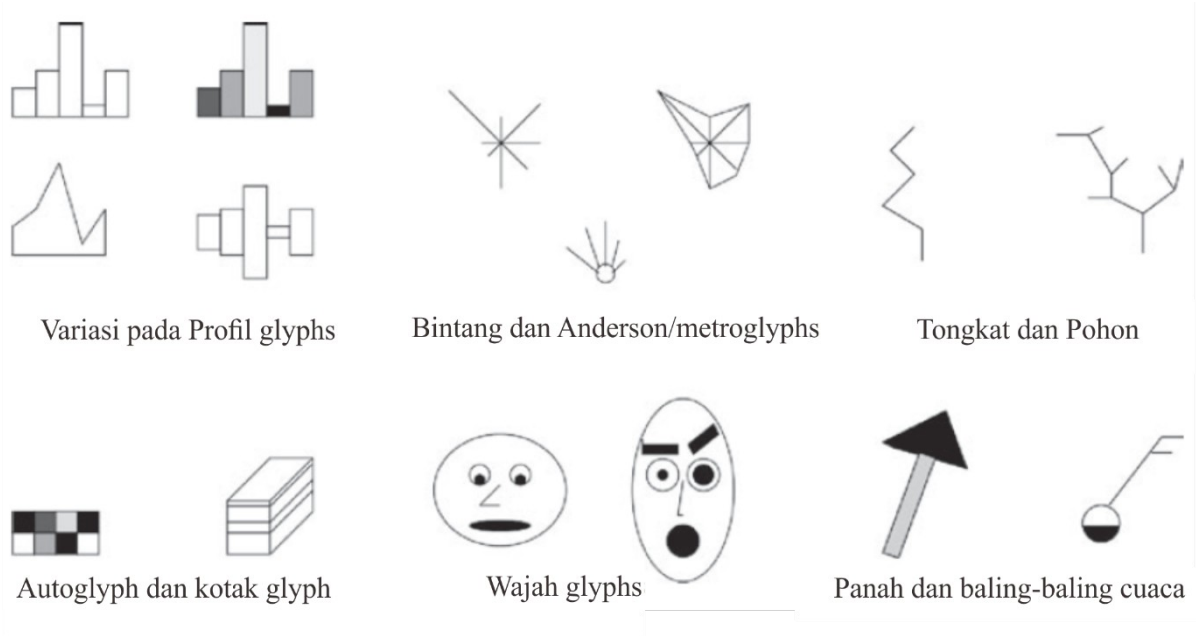
Gambar 26.12. Pelapisan visual dengan ukuran, saturasi, dan kecerahan dalam sistem Constellation (Munzner, 2000).

Melapisi dan Melapisi

Beberapa item dapat ditumpangkan dalam bingkai yang sama ketika posisi spasialnya kompatibel. Beberapa garis dapat ditampilkan dalam bagan garis yang sama, dan banyak titik dalam plot sebar yang sama, ketika sumbu dibagi di semua item. Salah satu manfaat dari satu tampilan bersama adalah membandingkan posisi item yang berbeda sangat mudah. Jika jumlah item dalam kumpulan data terbatas, maka satu tampilan saja sudah cukup. Pelapisan visual dapat memperluas kegunaan satu tampilan ketika ada cukup item yang membuat kekacauan visual menjadi perhatian. Gambar 26.12 menunjukkan kombinasi redundan dari saluran ukuran, saturasi, dan kecerahan berfungsi untuk membedakan lapisan latar depan dari lapisan latar belakang saat pengguna menggerakkan kursor di atas blok kata.

Glyphs

Kami telah mendiskusikan ide pengkodean visual menggunakan tanda sederhana, di mana tanda tunggal hanya dapat memiliki satu nilai untuk setiap saluran visual yang digunakan. Dengan tanda yang lebih kompleks, yang akan kita sebut mesin terbang, ada struktur internal di mana subkawasan memiliki pengkodean saluran visual yang berbeda.



Gambar 26.13. Tanda kompleks, yang kami sebut mesin terbang, memiliki subbagian yang secara visual mengkodekan dimensi data yang berbeda. Gambar milik Matt Ward (M. O. Ward, 2002).

Merancang glif yang tepat memiliki tantangan yang sama dengan merancang pengkodean visual. Gambar 26.13 menunjukkan berbagai mesin terbang, termasuk wajah terkenal yang awalnya diusulkan oleh Chernoff. Bahaya menggunakan wajah untuk menunjukkan dimensi data abstrak adalah bahwa respons persepsi dan emosional kita terhadap fitur wajah yang berbeda sangat nonlinier dengan cara yang tidak sepenuhnya dipahami, tetapi variabilitasnya lebih besar daripada di antara saluran visual yang telah kita bahas sejauh ini. Kita mungkin jauh lebih terbiasa dengan fitur yang menunjukkan keadaan emosional, seperti orientasi alis, daripada fitur lain, seperti ukuran hidung atau bentuk wajah.

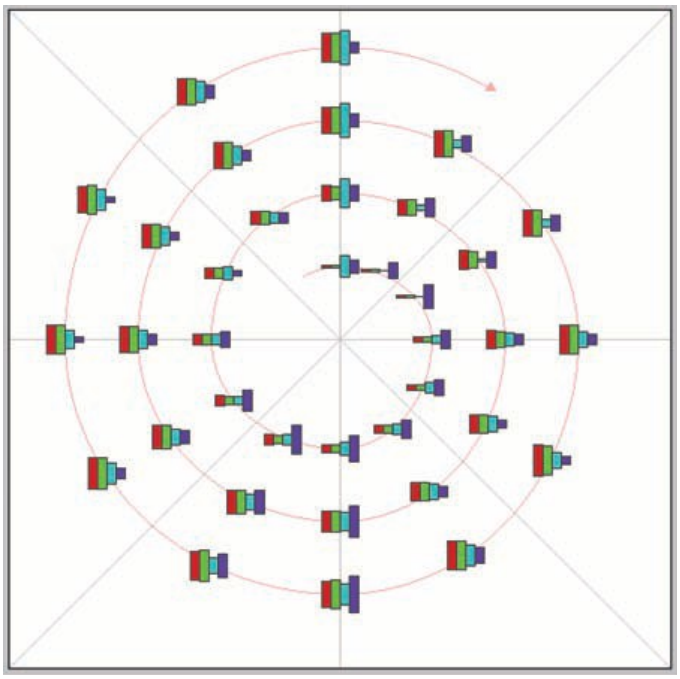
Mesin terbang kompleks membutuhkan area tampilan yang signifikan untuk setiap mesin terbang, seperti yang ditunjukkan pada Gambar 26.14 di mana diagram batang mini menunjukkan nilai empat dimensi berbeda di banyak titik di sepanjang jalur spiral. Glyph yang lebih sederhana dapat digunakan untuk membuat tekstur visual global, ukuran glyph sangat kecil sehingga nilai individual tidak dapat dibaca tanpa zoom, tetapi batas wilayah dapat dilihat dari tingkat ikhtisar. Gambar 26.15 menunjukkan contoh penggunaan figur tongkat di kanan atas pada Gambar 26.13. Mesin terbang dapat ditempatkan pada interval reguler, atau dalam posisi spasial berbasis data menggunakan dimensi data asli atau turunan.

Beberapa Tampilan

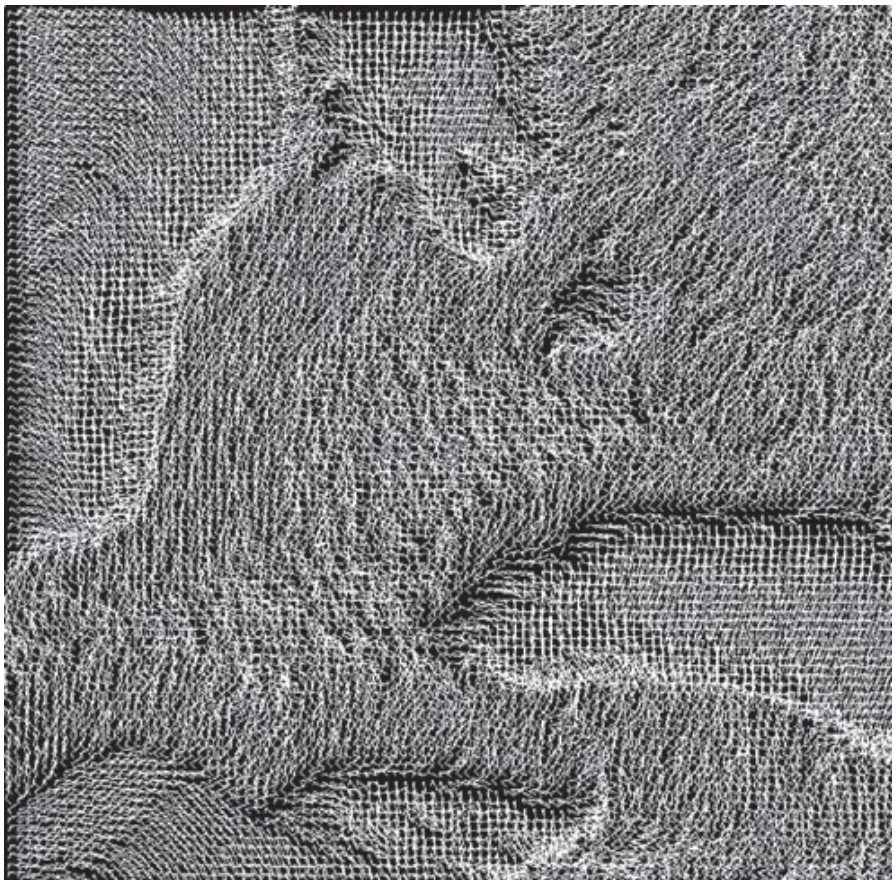
Kami sekarang beralih dari pendekatan dengan hanya satu bingkai ke pendekatan yang menggunakan banyak tampilan yang dihubungkan bersama. Bentuk tautan yang paling umum adalah penyorotan tertaut, di mana item yang dipilih dalam satu tampilan disorot di semua tampilan lainnya. Dalam navigasi tertaut, gerakan di satu tampilan memicu gerakan di tampilan lainnya.

Ada banyak jenis pendekatan multi-view. Dalam apa yang biasanya disebut pendekatan multi-tampilan, data yang sama ditampilkan dalam beberapa tampilan, yang masing-masing memiliki pengkodean visual yang berbeda yang menunjukkan aspek-aspek tertentu dari kumpulan data dengan paling jelas. Kekuatan penyorotan tertaut di beberapa penyandian visual adalah bahwa item yang termasuk dalam wilayah yang berdekatan dalam satu tampilan sering kali didistribusikan dengan sangat berbeda di tampilan lainnya. Dalam pendekatan kelipatan kecil, setiap tampilan memiliki penyandian visual yang sama untuk

kumpulan data yang berbeda, biasanya dengan sumbu bersama antara bingkai sehingga perbandingan posisi spasial di antara mereka bermakna. Perbandingan berdampingan dengan kelipatan kecil adalah alternatif untuk kekacauan visual dari melapiskan semua data dalam tampilan yang sama, dan keterbatasan memori manusia dalam mengingat bingkai yang dilihat sebelumnya dalam animasi yang berubah seiring waktu.



Gambar 26.14. Mesin terbang yang kompleks membutuhkan area tampilan yang signifikan sehingga informasi yang dikodekan dapat dibaca. Gambar milik Matt Ward, dibuat dengan perangkat lunak SpiralGlyphics (M. O. Ward, 2002).

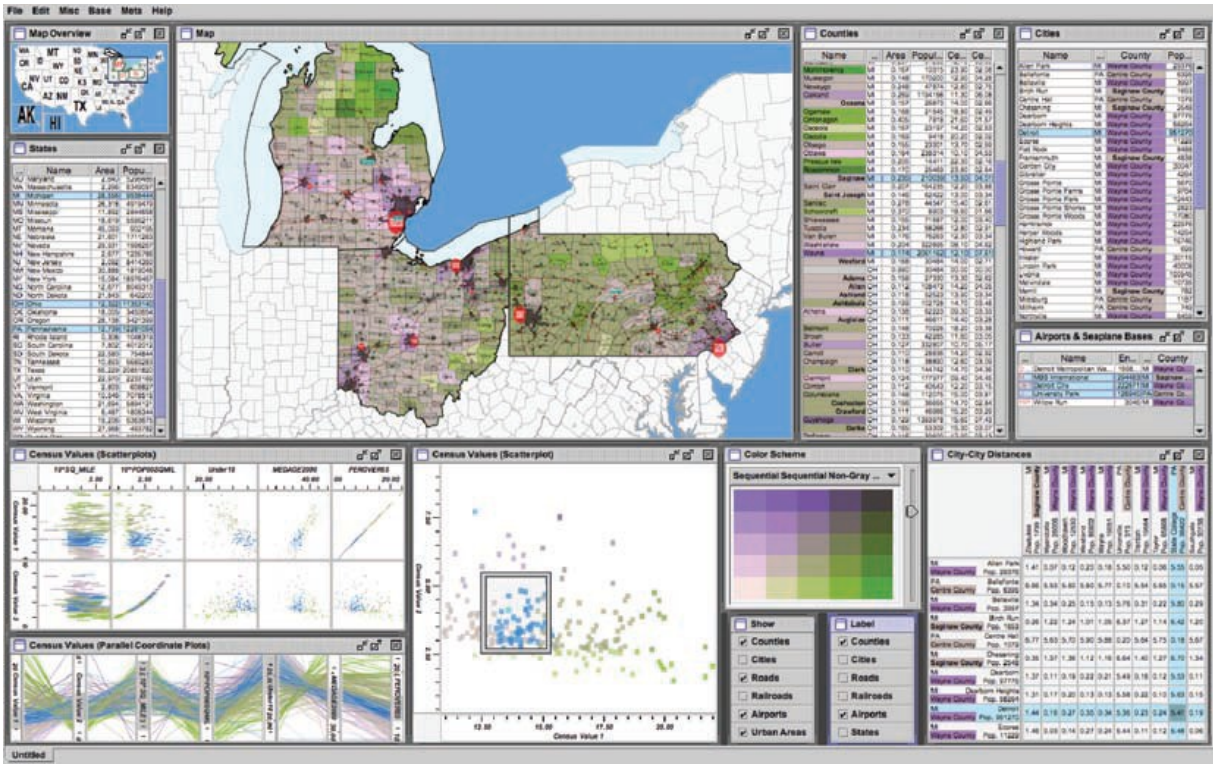


Gambar 26.15. Sebuah array padat mesin terbang sederhana. Gambar milik Georges Grinstein (S. Smith, Grinstein, & Bergeron, 1991), © 1991 IEEE.

Pendekatan ikhtisar-dan-detail harus memiliki data yang sama dan penyandian visual yang sama dalam dua tampilan, di mana satu-satunya perbedaan di antara keduanya adalah tingkat pembesaran. Dalam kebanyakan kasus, ikhtisar menggunakan lebih sedikit ruang tampilan daripada tampilan detail. Kombinasi tampilan ikhtisar dan detail umum di luar visualisasi di banyak alat mulai dari perangkat lunak mapping hingga pengeditan foto. Dengan pendekatan detail-on-demand, tampilan lain menunjukkan informasi lebih lanjut tentang beberapa item yang dipilih, baik sebagai windows pop-up di dekat kursor atau di windows permanen di bagian lain dari tampilan.

Menentukan posisi spasial yang paling tepat dari pandangan itu sendiri terhadap satu sama lain dapat menjadi masalah yang sama pentingnya dengan menentukan posisi spasial tanda dalam satu tampilan. Di beberapa sistem, lokasi tampilannya acak-acakan dan dibiarkan hingga sistem windows atau pengguna. Menyelaraskan pandangan memungkinkan perbandingan yang tepat di antara mereka, baik secara vertikal, horizontal, atau dengan larik untuk kedua arah. Sama seperti item yang dapat diurutkan dalam tampilan, tampilan dapat diurutkan dalam tampilan, biasanya sehubungan dengan variabel turunan yang mengukur beberapa aspek dari keseluruhan tampilan yang bertentangan dengan item individual di dalamnya.

Gambar 26.16 menunjukkan visualisasi data sensus yang menggunakan banyak tampilan. Selain informasi geografis, informasi demografis untuk setiap negara mencakup populasi, kepadatan, jenis kelamin, usia rata-rata, persentase perubahan sejak tahun 1990, dan proporsi kelompok etnis utama. Pengkodean visual yang digunakan meliputi tampilan geografis, scatterplot, koordinat paralel, tabular, dan matriks. Encoding warna yang sama digunakan di semua tampilan, dengan legenda di tengah bawah. Matriks scatterplot menunjukkan penyorotan terkait di semua tampilan, di mana item biru berdekatan di beberapa tampilan dan tersebar di yang lain. Peta di pojok kiri atas merupakan gambaran umum untuk peta detail besar di tengah. Tampilan tabular memungkinkan penyortiran langsung berdasarkan dan pemilihan dalam dimensi yang diinginkan.



Gambar 26.16. Toolkit Improvisasi digunakan untuk membuat visualisasi multi-tampilan ini. Gambar milik Chris Weaver.

26.14 REDUKSI DATA

Teknik pengkodean visual yang telah kita bahas sejauh ini menunjukkan semua item dalam kumpulan data. Namun, banyak kumpulan data yang begitu besar sehingga menampilkan semuanya secara bersamaan akan menghasilkan begitu banyak kekacauan visual sehingga representasi visual akan sulit atau tidak mungkin dipahami oleh pemirsa. Strategi utama untuk mengurangi jumlah data yang ditampilkan adalah tinjauan umum dan agregasi, penyaringan dan navigasi, teknik fokus+konteks, dan pengurangan dimensi.

Ikhtisar dan Agregasi

Dengan kumpulan data kecil, pengkodean visual dapat dengan mudah menampilkan semua dimensi data untuk semua item. Untuk kumpulan data berukuran sedang, gambaran umum yang menunjukkan informasi tentang semua item dapat dibuat dengan menampilkan lebih sedikit detail untuk setiap item. Banyak kumpulan data memiliki struktur internal atau turunan pada berbagai skala. Dalam kasus ini, representasi visual multiskala dapat memberikan banyak tingkat ikhtisar, daripada hanya satu tingkat. Ikhtisar biasanya digunakan sebagai titik awal untuk memberi petunjuk kepada pengguna tentang di mana harus menelusuri untuk memeriksa lebih detail.

Untuk kumpulan data yang lebih besar, membuat ikhtisar memerlukan semacam ringkasan visual. Salah satu pendekatan untuk reduksi data adalah dengan menggunakan representasi agregat di mana satu tanda visual dalam ikhtisar secara eksplisit mewakili banyak item.

Tantangan agregasi adalah untuk menghindari menghilangkan sinyal menarik dalam dataset dalam proses peringkasan. Dalam literatur kartografi, masalah membuat peta pada skala yang berbeda sambil mempertahankan karakteristik pembeda yang penting telah dipelajari secara ekstensif dengan nama generalisasi kartografi (Slocum, McMaster, Kessler, & Howard, 2008).

Pemfilteran dan Navigasi

Pendekatan lain untuk reduksi data adalah dengan menyaring data, hanya menunjukkan subset item. Pemfilteran sering dilakukan dengan memilih secara langsung rentang yang diinginkan dalam satu atau lebih dimensi data.

Navigasi adalah jenis penyaringan khusus berdasarkan posisi spasial, di mana mengubah sudut pandang akan mengubah kumpulan item yang terlihat. Zoom geometris dan nongeometrik digunakan dalam visualisasi. Dengan zoom geometris, posisi kamera dalam ruang 2D atau 3D dapat diubah dengan kontrol grafis komputer standar. Scene yang tidak realistis, item harus digambar dengan ukuran yang bergantung pada jaraknya dari kamera, dan hanya ukuran nyata yang berubah berdasarkan jarak tersebut. Namun, dalam pengkodean visual dari ruang abstrak, pembesaran nongeometrik dapat berguna. Dalam pembesaran semantik, tampilan visual suatu objek berubah secara dramatis berdasarkan jumlah piksel yang tersedia untuk menggambarnya. Misalnya, representasi visual abstrak dari file teks dapat berubah dari kotak kecil berkode warna tanpa label menjadi kotak berukuran sedang yang hanya berisi nama file sebagai label teks menjadi persegi panjang besar yang berisi ringkasan multi-baris dari file. isi. Dalam scene realistis, objek yang cukup jauh dari kamera tidak terlihat dalam gambar, misalnya, setelah mereka mengurangi kurang dari satu piksel area layar. Dengan visibilitas yang terjamin, salah satu dimensi data asli atau turunan digunakan sebagai ukuran kepentingan, dan objek yang cukup penting harus memiliki beberapa jenis representasi yang terlihat dalam bidang gambar setiap saat.

Fokus + Konteks

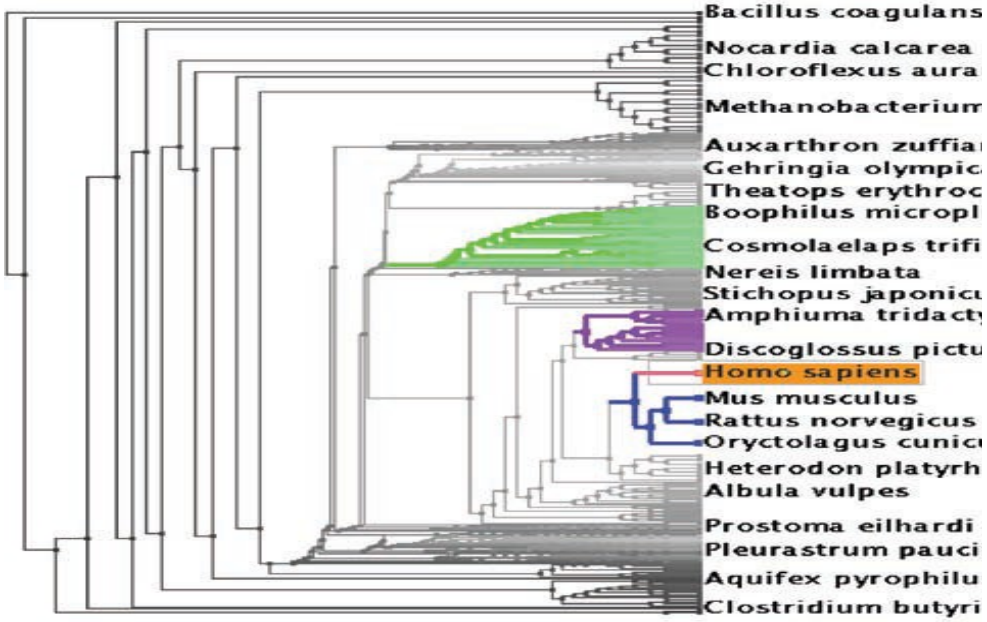
Fokus + teknik konteks adalah pendekatan lain untuk reduksi data. Sebuah subset dari item dataset secara interaktif dipilih oleh pengguna untuk menjadi fokus dan digambar secara

detail. Pengkodean visual juga mencakup informasi tentang beberapa atau semua sisa kumpulan data yang ditampilkan untuk konteks, terintegrasi ke dalam tampilan yang sama yang menunjukkan item fokus. Banyak dari teknik ini menggunakan distorsi yang dipilih dengan hati-hati untuk menggabungkan daerah fokus yang diperbesar dan daerah konteks yang diperkecil ke dalam tampilan terpadu.

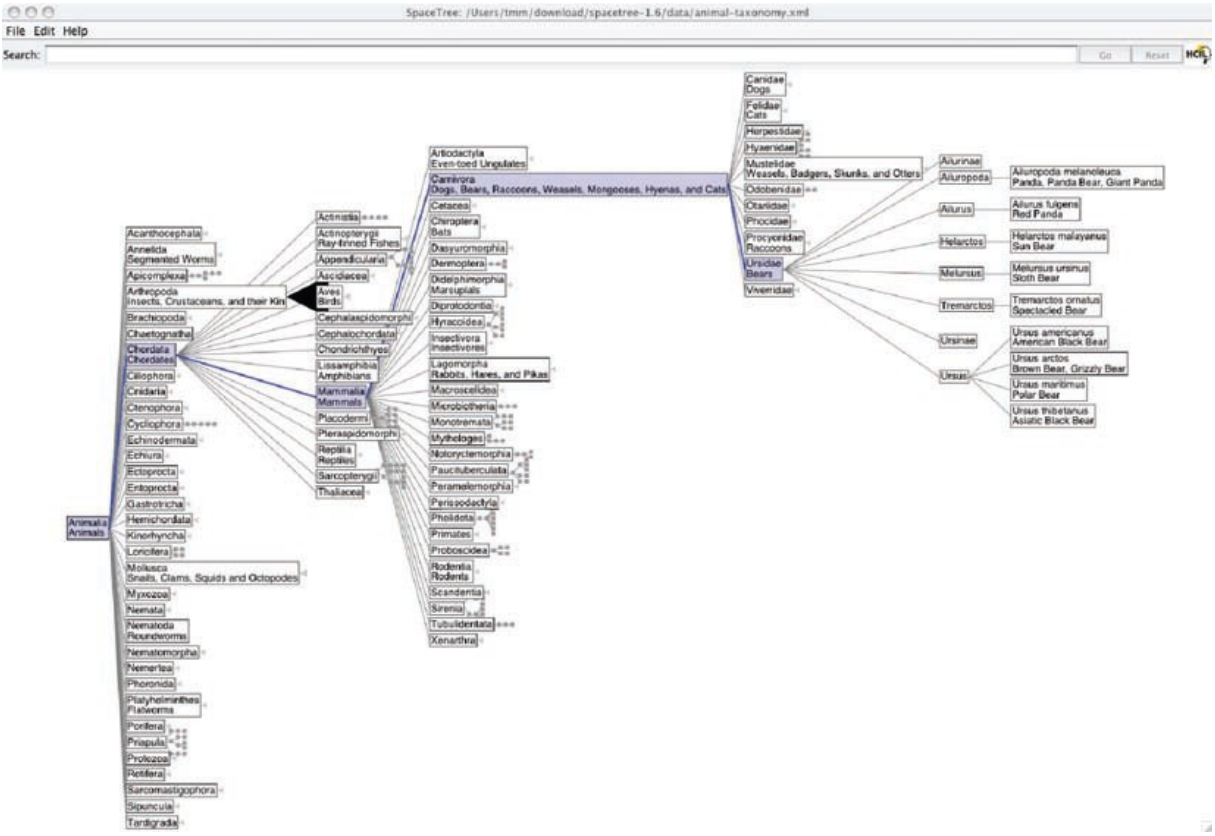
Salah satu metafora interaksi yang umum adalah lensa mata ikan yang dapat dipindahkan. Geometri hiperbolik memberikan kerangka matematika yang elegan untuk lensa radial tunggal yang memengaruhi semua objek dalam tampilan. Metafora interaksi lainnya adalah dengan menggunakan beberapa lensa dari berbagai bentuk dan tingkat pembesaran yang hanya mempengaruhi wilayah lokal. Navigasi regangan dan squish menggunakan metafora interaksi lembaran karet di mana meregangkan satu wilayah akan menekan sisanya, seperti yang ditunjukkan pada Gambar 26.17. Batas lembar tetap sehingga semua item berada dalam viewport, meskipun banyak item mungkin dikompresi ke ukuran subpiksel. Metafora mata ikan tidak terbatas pada lensa geometris yang digunakan setelah tata ruang; itu dapat digunakan secara langsung pada data terstruktur, seperti dokumen hierarkis di mana beberapa bagian diciutkan sementara yang lain dibiarkan diperluas.

Pendekatan berbasis distorsi ini adalah contoh lain dari navigasi nonliteral dalam semangat yang sama dengan zoom nongeometrik. Saat menavigasi dalam kumpulan data yang besar dan tidak dikenal dengan gerakan kamera yang realistis, pengguna dapat menjadi bingung pada tingkat zoom tinggi ketika mereka hanya dapat melihat wilayah lokal yang kecil. Pendekatan ini dirancang untuk memberikan lebih banyak informasi kontekstual daripada tampilan tunggal yang tidak terdistorsi, dengan harapan bahwa orang dapat tetap berorientasi jika landmark tetap dapat dikenali. Namun, jenis distorsi ini masih dapat membingungkan atau sulit diikuti oleh pengguna. Biaya dan manfaat distorsi, sebagai lawan dari beberapa pandangan atau satu pandangan realistis, belum sepenuhnya dipahami. Perspektif 3D standar adalah jenis distorsi yang sangat familiar dan secara eksplisit digunakan sebagai bentuk fokus+konteks dalam pekerjaan visualisasi awal. Namun, karena biaya tata ruang 3D yang dibahas dalam Bagian 26.4 menjadi lebih dipahami, pendekatan ini menjadi kurang populer.

Pendekatan lain untuk menyediakan konteks di sekitar item fokus tidak memerlukan distorsi. Misalnya, sistem SpaceTree yang ditunjukkan pada Gambar 26.18 menghilangkan sebagian besar node di pohon, menunjukkan jalur antara node fokus yang dipilih secara interaktif dan akar pohon untuk konteks.



Gambar 26.17. Sistem TreeJuxtaposer menampilkan navigasi regangan dan squish serta jaminan visibilitas wilayah yang ditandai dengan warna (Munzner, Guimbretie`re, Tasiran, Zhang, & Zhou, 2003).



Gambar 26.18. Sistem SpaceTree menunjukkan jalur antara root dan node fokus yang dipilih secara interaktif untuk menyediakan konteks (Grosjean, Plaisant, & Bederson, 2002

26.15 PENGURANGAN DIMENSI

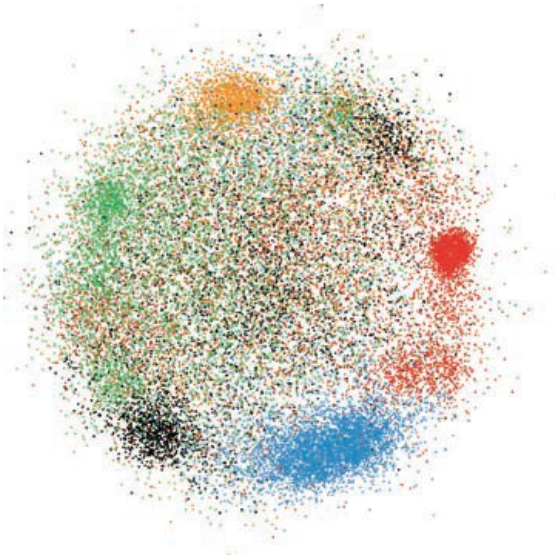
Pendekatan reduksi data yang dibahas sejauh ini mengurangi jumlah item untuk menarik. Ketika ada banyak dimensi data, pengurangan dimensi juga bisa efektif.

Dengan pengirisan, satu nilai dipilih dari dimensi untuk dihilangkan, dan hanya item yang cocok dengan nilai tersebut untuk dimensi yang diekstraksi untuk disertakan dalam irisan dimensi lebih rendah. Pemotongan sangat berguna dengan data spasial 3D, misalnya saat memeriksa irisan melalui CT scan kepala manusia pada ketinggian berbeda di sepanjang tengkorak. Slicing dapat digunakan untuk menghilangkan beberapa dimensi sekaligus.

Dengan proyeksi, tidak ada informasi tentang dimensi yang dihilangkan yang disimpan; nilai-nilai untuk dimensi tersebut dijatuhkan begitu saja, dan semua item masih ditampilkan. Bentuk proyeksi yang umum adalah transformasi perspektif grafis standar yang memproyeksikan dari 3D ke 2D, kehilangan informasi tentang kedalaman di sepanjang jalan. Dalam visualisasi matematis, struktur objek geometris berdimensi lebih tinggi dapat ditunjukkan dengan memproyeksikan dari 4D ke 3D sebelum proyeksi standar ke bidang gambar dan menggunakan warna untuk mengkodekan informasi dari dimensi jauh yang diproyeksikan. Teknik ini terkadang disebut pemfilteran dimensional bila digunakan untuk data nonspasial.

Dalam beberapa kumpulan data, mungkin ada struktur tersembunyi yang menarik di ruang dimensi yang jauh lebih rendah daripada jumlah dimensi data asli. Misalnya, kadang-kadang mengukur secara langsung variabel-variabel independen yang diminati itu sulit atau tidak mungkin, tetapi sejumlah besar variabel dependen atau tidak langsung tetap tersedia.

Tujuannya adalah untuk menemukan kumpulan kecil dimensi yang dengan setia mewakili sebagian besar struktur atau varian dalam kumpulan data. Dimensi ini mungkin yang asli, atau yang baru disintesis yang merupakan kombinasi linier atau nonlinier dari aslinya. Analisis komponen utama adalah metode linier yang cepat dan banyak digunakan. Banyak pendekatan nonlinier telah diusulkan, termasuk scalling multidimensi (MDS). Metode ini biasanya digunakan untuk menentukan apakah ada cluster skala besar dalam kumpulan data; struktur berbutir halus di plot berdimensi lebih rendah biasanya tidak dapat diandalkan karena informasi hilang dalam reduksi. Gambar 26.19 menunjukkan kumpulan dokumen dalam satu scatterplot. Ketika dimensi sebenarnya dari dataset jauh lebih tinggi dari dua, matriks scatterplot yang menunjukkan pasangan dimensi sintetis mungkin diperlukan.



Gambar 26.19. Pengurangan dimensi dengan pendekatan scalling multidimensi Glimmer menunjukkan cluster dalam kumpulan data dokumen (Ingram, Munzner, & Olano, 2009), © 2009 IEEE.

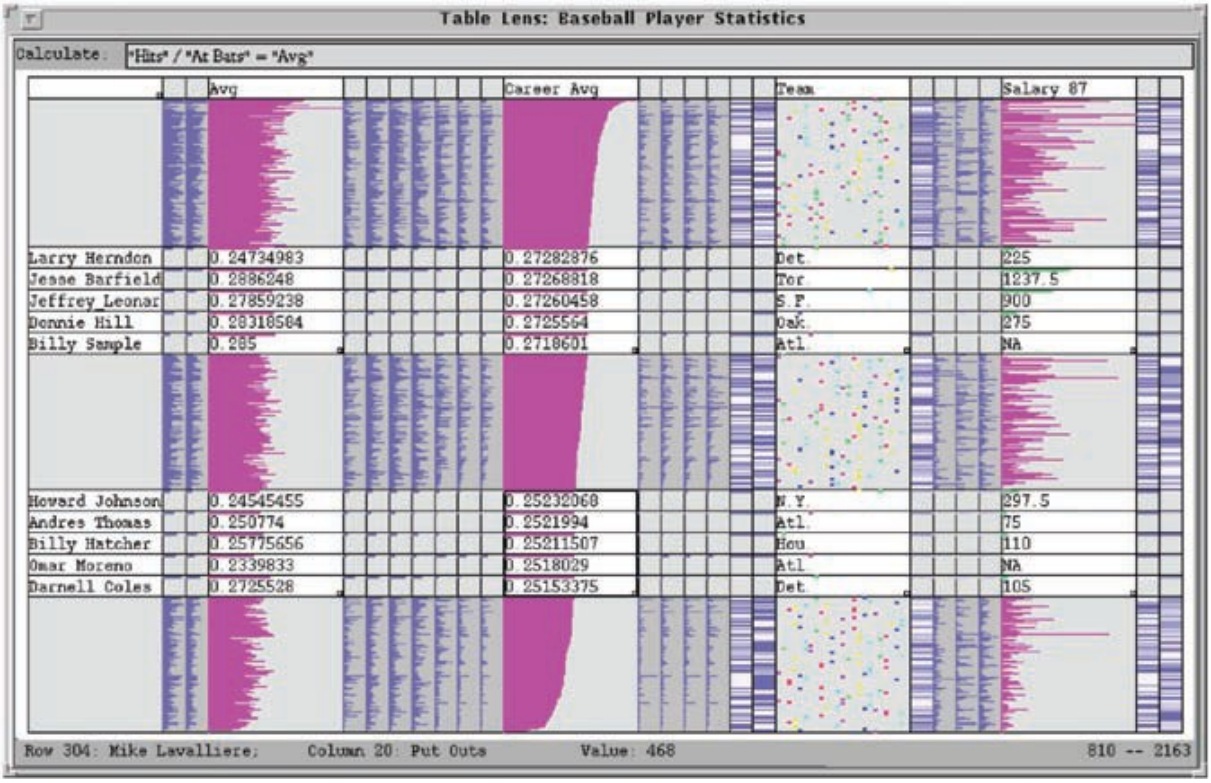
Contoh

Data tabular sangat umum, seperti yang diketahui semua pengguna spreadsheet. Tujuan dalam visualisasi adalah untuk mengkodekan informasi ini melalui saluran visual yang mudah dipahami daripada memaksa orang untuk membacanya sebagai angka dan teks. Gambar 26.20 menunjukkan Lensa Tabel, pendekatan fokus+konteks di mana nilai kuantitatif dikodekan sebagai panjang garis tinggi satu piksel di wilayah konteks, dan ditampilkan sebagai angka di wilayah fokus. Setiap dimensi kumpulan data ditampilkan sebagai kolom, dan baris item dapat dipilih sesuai dengan nilai di kolom tersebut dengan satu klik di headernya.

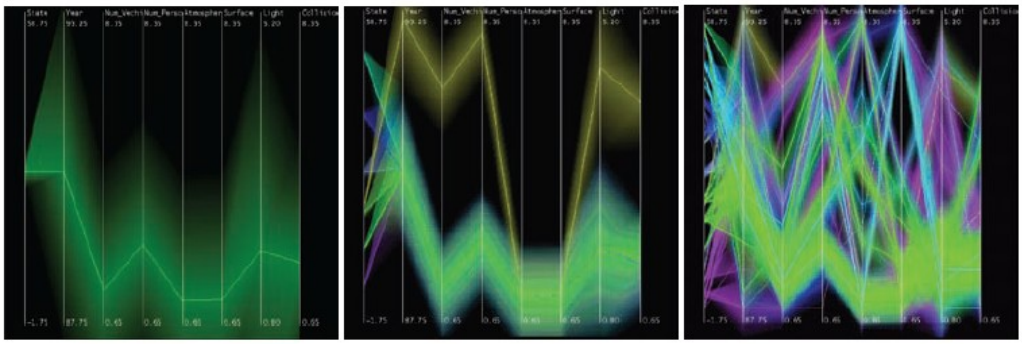
Tabel

Pendekatan Kartesius tradisional dari plot sebar, di mana item diplot sebagai titik sehubungan dengan sumbu tegak lurus, hanya dapat digunakan untuk data dua dan tiga dimensi. Banyak tabel berisi lebih dari tiga dimensi data, dan jumlah dimensi tambahan yang dapat dikodekan dengan saluran visual lainnya terbatas. Koordinat paralel adalah pendekatan untuk memvisualisasikan lebih banyak dimensi sekaligus menggunakan posisi spasial, di mana sumbunya sejajar daripada tegak lurus dan item n-dimensi ditampilkan sebagai polyline yang melintasi setiap sumbu n satu kali (Inselberg & Dimsdale, 1990; Wegman , 1990). Gambar 26.21 menunjukkan kumpulan data delapan dimensi dari 230.000 item pada berbagai tingkat detail (Fua, Ward, & Rundensteiner, 1999), dari tampilan tingkat tinggi di bagian atas hingga detail yang lebih halus di bagian bawah. Dengan koordinat paralel hierarkis, item dikelompokkan dan seluruh cluster item diwakili oleh pita dengan lebar dan opasitas yang

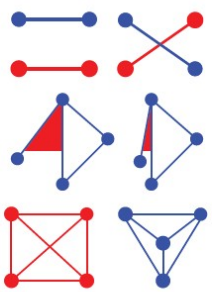
bervariasi, di mana mean berada di tengah dan lebar pada setiap sumbu bergantung pada nilai item dalam cluster di dimensi. Pewarnaan masing-masing band didasarkan pada kedekatan antar cluster sesuai dengan metrik kesamaan.



Gambar 26.20. Lensa Tabel menyediakan interaksi fokus+konteks dengan data tabular, yang segera dapat disusun ulang berdasarkan nilai di setiap kolom dimensi. Gambar milik Stuart Card(Rao &Card, 1994), © 1994 ACM, Inc. Disertakan di sini dengan izin



Gambar 26.21. Koordinat paralel hierarkis menunjukkan data berdimensi tinggi pada berbagai tingkat detail. Gambar milik Matt Ward (Fua et al., 1999), © 1999 IEEE.



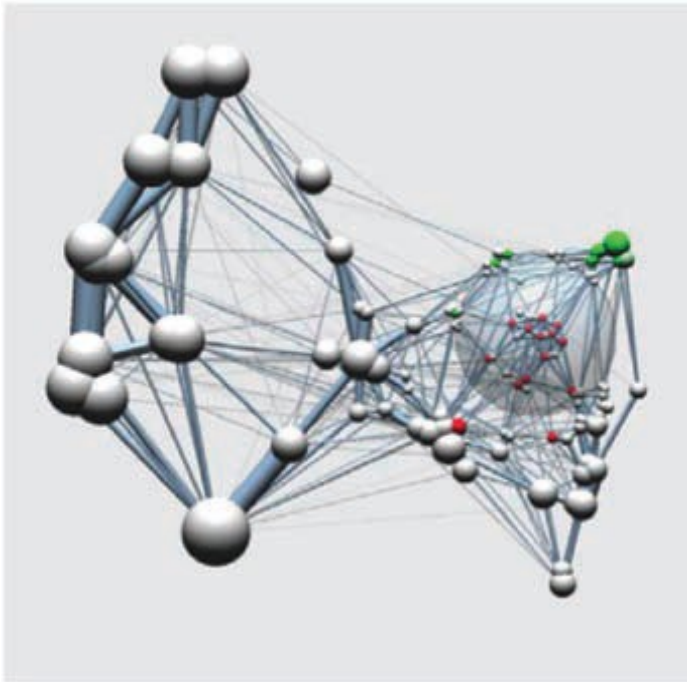
Gambar 26.22. Kriteria estetika tata letak grafis. Atas: Penyeberangan tepi harus diminimalkan. Tengah: Resolusi sudut harus dimaksimalkan. Bawah: Simetri dimaksimalkan di sebelah kiri, sedangkan penyeberangan diminimalkan di sebelah kanan, menunjukkan konflik antara kriteria NPhard individual

26.16 GRAFIS

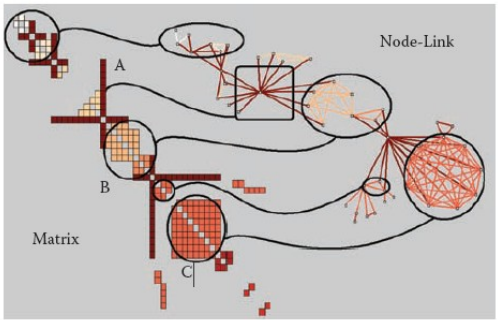
Bidang menggambar grafis berkaitan dengan menemukan posisi spasial untuk node dalam grafis dalam ruang 2D atau 3D dan routing tepi antara node tersebut (Di Battista, Eades, Tamassia, & Tollis, 1999). Dalam banyak kasus, masalah perutean tepi disederhanakan dengan hanya menggunakan tepi lurus, atau dengan hanya mengizinkan pembengkokan sudut siku-siku untuk kelas tata letak ortogonal, tetapi beberapa pendekatan menangani kurva sebenarnya. Jika graf memiliki tepi berarah, pendekatan berlapis dapat digunakan untuk menunjukkan struktur hierarki melalui urutan spasial horizontal atau vertikal dari node, seperti yang ditunjukkan pada Gambar 26.2.

Serangkaian kriteria estetika mengoperasionalkan penilaian manusia tentang grafis yang dapat dibaca sebagai metrik yang dapat dihitung pada tata letak yang diusulkan (Ware, Purchase, Colpys, & McGill, 2002). Gambar 26.22 menunjukkan beberapa contoh. Beberapa metrik harus diminimalkan, seperti jumlah persilangan tepi, luas total tata letak, dan jumlah tikungan atau kurva sudut kanan. Lainnya harus dimaksimalkan, seperti resolusi sudut atau simetri. Masalahnya sulit karena sebagian besar kriteria ini secara individual NP-hard, dan terlebih lagi mereka saling tidak kompatibel (Brandenburg, 1988).

Banyak pendekatan untuk menggambar grafis simpul-tautan menggunakan penempatan yang diarahkan oleh gaya, dimotivasi oleh metafora fisik intuitif dari gaya pegas di tepi yang menggambar bersama-sama partikel yang saling menolak pada derah. Meskipun pendekatan naif memiliki kompleksitas waktu yang tinggi dan cenderung terjebak dalam minimum lokal, banyak pekerjaan telah dilakukan untuk mengembangkan algoritme yang lebih canggih seperti GEM (Frick, Ludwig, & Mehldau, 1994) atau IPSep-CoLa (Dwyer, Koren, & Marriott, 2006). Gambar 26.23 menunjukkan sistem interaktif menggunakan model energi r-PolyLog, di mana tampilan fokus+konteks dari grafis berkerumun dibuat dengan mata ikan geometris dan semantik (van Ham & van Wijk, 2004).



Gambar26.23. Penempatan yang diarahkan secara paksa yang menunjukkan grafis kluster dengan mata ikan geometris dan semantik. Gambar milik Jarke van Wijk (van Ham & van Wijk, 2004), © 2004 IEEE.

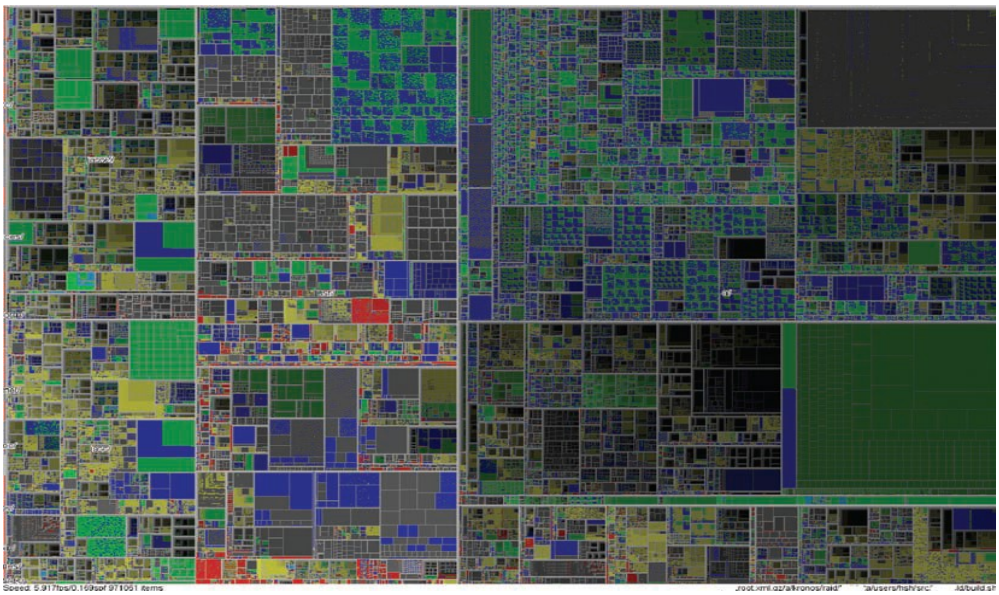


Gambar 26.24. Grafis dapat ditampilkan dengan tampilan matriks atau node-link. Gambar milik JeanDaniel Fekete (Henry & Fekete, 2006), © 2006 IEEE.

Grafis juga dapat dikodekan secara visual dengan menunjukkan matriks adjacency, di mana semua simpul ditempatkan di sepanjang setiap sumbu dan sel di antara dua simpul berwarna jika ada tepi di antara keduanya. Sistem MatrixExplorer menggunakan beberapa tampilan terkait untuk membantu peneliti ilmu sosial secara visual menganalisis jaringan sosial dengan representasi matriks dan node-link (Henry & Fekete, 2006). Gambar 26.24 menunjukkan pola visual berbeda yang dibuat oleh struktur grafis yang sama dalam dua tampilan ini: A mewakili aktor yang menghubungkan beberapa komunitas; B adalah komunitas; dan C adalah clique, atau sub-graf lengkap. Tampilan matriks tidak mengalami persilangan tepi yang berantakan, tetapi banyak tugas termasuk mengikuti jalur lebih sulit dengan pendekatan ini.

Pohon

Pohon adalah kasus khusus dari grafis yang begitu umum sehingga banyak penelitian visualisasi telah dilakukan untuk mereka. Algoritma lurus ke depan untuk meletakkan pohon di bidang dua dimensi bekerja dengan baik untuk pohon kecil (Reingold & Tilford, 1981), sementara pendekatan yang lebih kompleks tetapi skalabel berjalan dalam waktu linier (Buchheim, Ju¨nger, & Leipert, 2002). Gambar 26.17 dan 26.18 juga menunjukkan pohon dengan pendekatan tata ruang yang berbeda, tetapi keempat metode ini secara visual mengkodekan hubungan antara simpul induk dan simpul anak dengan menggambar tautan yang menghubungkannya.



Gambar 26.25. Treemap menunjukkan sistem file hampir satu juta file. Gambar milik JeanDaniel Fekete (Fekete & Plaisant, 2002), © 2002 IEEE.

Treemaps menggunakan penahanan daripada koneksi untuk menunjukkan hubungan hierarkis antara node induk dan anak dalam sebuah pohon (B. Johnson & Shneiderman, 1991). Artinya, peta pohon menunjukkan simpul anak yang bersarang di dalam garis besar simpul induk. Gambar 26.25 menunjukkan sistem file hierarkis dari hampir satu juta file, di mana ukuran file dikodekan oleh ukuran persegi panjang dan tipe file dikodekan oleh warna (Fekete & Plaisant, 2002). Ukuran simpul di daun pohon dapat mengkodekan dimensi data tambahan, tetapi ukuran simpul di bagian dalam tidak menunjukkan nilai dimensi itu; ditentukan oleh ukuran kumulatif turunannya. Meskipun tugas-tugas seperti memahami struktur topologi pohon atau menelusuri jalur melaluinya lebih sulit dengan peta pohon daripada dengan pendekatan nodelink, tugas-tugas yang melibatkan pemahaman atribut yang terkait dengan node daun didukung dengan baik. Treemaps adalah representasi yang mengisi ruang yang biasanya lebih kompak daripada pendekatan node-link.

26.17 GEOGRAFIS

Berbagai macam analisis seperti epidemiologi memerlukan pemahaman baik data geografis maupun nonspasial. Gambar 26.26 menunjukkan alat untuk analisis visual dari kumpulan data demografi kanker yang menggabungkan banyak ide yang dijelaskan dalam bab ini (MacEachren, Dai, Hardisty, Guo, & Lengerich, 2003). Matriks atas dari tampilan tertaut menampilkan kelipatan kecil dari tiga jenis pengkodean visual: peta geografis yang menunjukkan kabupaten Appalachian di kiri bawah, histogram melintasi diagonal matriks, dan plot sebar di kanan atas. Matriks 2 × 2 bagian bawah, yang menghubungkan plot sebar dengan peta, menyertakan legenda warna untuk keduanya. Kecerahan colormapha sekuensial bivariat diskrit meningkat secara berurutan untuk masing-masing dari dua rona komplementer dan efektif untuk orang yang kekurangan warna.

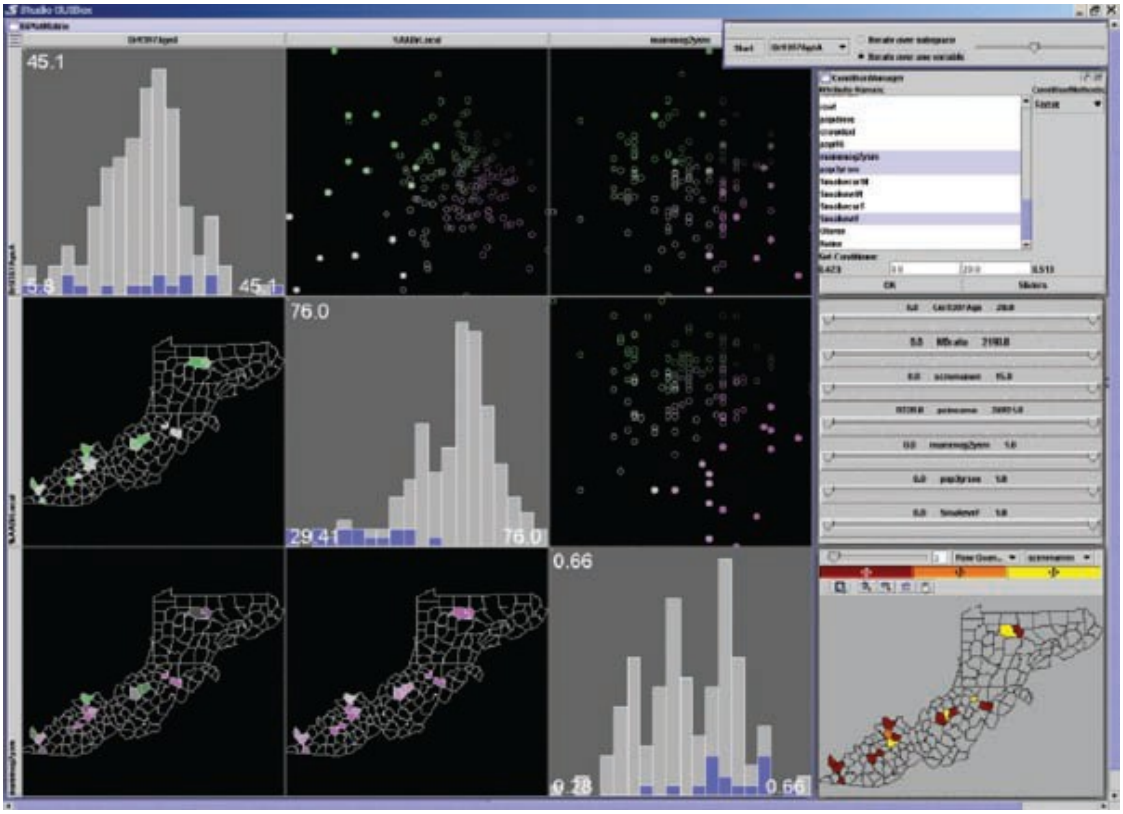
Bidang Tata Ruang

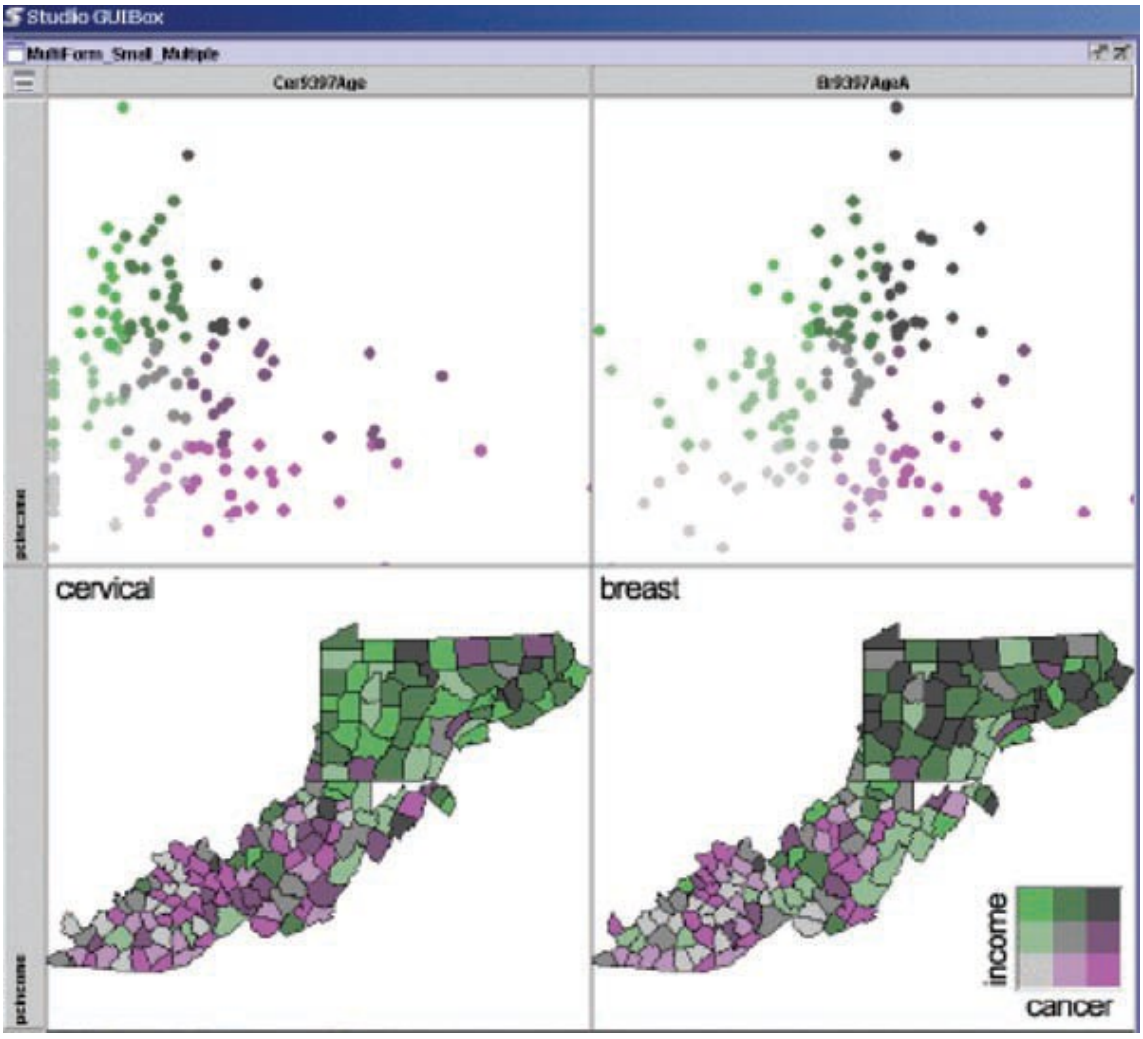
Sebagian besar data spasial nongeografis dimodelkan sebagai bidang, di mana ada satu atau lebih nilai yang terkait dengan setiap titik dalam ruang 2Dor3D. Bidang skalar, misalnya pemindaian pencitraan medis CT atau MRI, biasanya divisualisasikan dengan menemukan permukaan iso atau menggunakan rendering volume langsung. Bidang vektor, misalnya, aliran di air atau udara, sering divisualisasikan menggunakan panah, garis arus (McLouglin, Laramee, Peikert, Post, & Chen, 2009), dan konvolusi integral garis (LIC) (Laramee et al., 2004). Medan tensor, seperti yang menggambarkan difusi anisotropik molekul melalui otak manusia, sangat menantang untuk ditampilkan (Kindlmann, Weinstein, & Hart, 2000).

Pertanyaan yang Sering Diajukan

- ***Konferensi dan jurnal apa yang merupakan tempat yang baik untuk mencari informasi lebih lanjut tentang visualisasi?***

Konferensi IEEE VisWeek terdiri dari tiga subkonferensi: InfoVis (Visualisasi Informasi), Vis (Visualization), dan VAST (Visual Analytics Science and Technology). Ada juga konferensi EuroVis Eropa dan tempat Asian PacificVis. Jurnal yang relevan antara lain IEEE TVCG (Transactions on Visualization and Computer Graphics) dan Palgrave InformationVisualization.





Gambar 26.26. Dua matriks dari kelipatan kecil terkait yang menunjukkan data demografi kanker (MacEachren et al., 2003), © 2003 IEEE.

• Perangkat lunak dan perangkat apa yang tersedia untuk visualisasi? Toolkit paling populer untuk data spasial adalah vtk, basis kode C/C++ yang tersedia di www.vtk.org. Untuk data abstrak, prefuse berbasis Java (http://www.prefuse.org) dan toolkit Processing (processing.org) menjadi banyak digunakan. ManyEyessite dari IBM Research (www.many-eyes.com memungkinkan orang untuk mengunggah data mereka sendiri, membuat visualisasi interaktif dalam berbagai format, dan melakukan percakapan tentang analisis data visual.

Referensi

Adelson, E. H. (1999). Lightness Perception and Lightness Illusions. In M. S. Gazzaniga (Ed.), The New Cognitive Neurosciences (Second ed., pp. 339–351). Cambridge, MA: MIT Press.

Adzhiev, V., Cartwright, R., Fausett, E., Ossipov, A., Pasko, A., & Savchenko, V. (1999, Sep). Hyperfun Project: A Framework for Collaborative Multidimensional F-rep Modeling. In Implicit Surfaces ’99 (pp. 59–69). Aire-laville, Switzerland: Eurographics Association.

Akenine-Mo¨ller,T.,Haines,E.,&Hoffman,N.(2008). Real-Time Rendering (Thirded.). Wellesley, MA: A K Peters.

Akkouche, S., & Galin, E. (2001). Adaptive Implicit Surface Polygonization Using MarchingTriangles. Computer Graphics Forum, 20(2),67–80.

Akleman, E., & Chen, J. (1999). Generalized Distance Functions. In Proceedings of the International Conference on Shape Modeling and Applications (pp. 72–79). Washington, DC: IEEE Computer Society Press.

Amanatides, J., & Woo, A. (1987). A Fast Voxel Traversal Algorithm for Ray Tracing. In Proceedingsof Eurographics(pp. 1–10). Amsterdam: Elsevier Science Publishers.

Amar, R., Eagan, J., & Stasko, J. (2005). Low-Level Components of Analytic Activity in Information Visualization. In Proc. IEEE Symposium on Information Visualization (InfoVis) (pp. 111–117). Washington, DC: IEEE Computer Society Press.

American National Standard Institute. (1986). Nomenclature and Definitions for Illumination Engineering. ANSI Report (New York). (ANSI/IES RP-161986)

Angel, E. (2002). Interactive Computer Graphics: A Top-Down Approach with OpenGL (Third ed.). Reading, MA: Addison-Wesley.

Appel, A. (1968). Some Techniquesfor Shading Machine Renderings of Solids. In Proceedings of the AFIPS Spring Joint Computing Conference (Vol. 32, pp. 37–45).

AFIPS. Arvo, J. (1995). Analytic Methods for Simulated Light Transport (Unpublished doctoraldissertation).

Ashdown, I. (1994). Radiosity: A Programmer’s Perspective. New York: John Wiley & Sons.

Ashikhmin, M. (2002). A Tone Mapping Algorithm for High Contrast Images. In EGRW’02: Proceedings of the13th Eurographics Workshopon Rendering (pp. 145–155). Aire-la-Ville, Switzerland: EurographicsAssociation.

Ashikhmin, M., Premo˘ze, S., & Shirley, P. (2000). A Microfacet-Based BRDF Generator. In Proceedings of SIGGRAPH (pp. 65–74). Reading, MA: Addison-WesleyLongman.

Ashikhmin, M., & Shirley, P. (2000). An Anisotropic Phong BRDF Model. journal of graphics tools, 5(2),25–32.

Baerentzen, J., & Christensen, N. (2002, May). Volume Sculpting Using the Level-Set Method. In SMI ’02: Proceedings of Shape Modeling International 2002 (SMI ’02) (pp. 175–182). Washington, DC: IEEE Computer Society Press.

Barr, A. H. (1984). Global and Local Deformations of Solid Primitives. Proc. SIGGRAPH ’84 Computer Graphics, 18(3),21–30.

Bartels, R. H., Beatty, J. C., & Barsky, B. A. (1987). An Introduction to Splines for Use in Computer Graphics and Geometric Modeling. San Francisco, CA: MorganKaufmann.

Barthe,L.,Dodgson,N.A.,Sabin,M.A.,Wyvill,B.,&Gaildrat,V. (2003). Twodimensional Potential Fields for Advanced Implicit Modeling Operators. Computer Graphics Forum, 22(1),23–33.

Barthe, L., Mora, B., Dodgson, N. A., & Sabin, M. A. (2002). InteractiveImplicit Modelling based on C1 Reconstruction of Regular Grids. International Journal of Shape Modeling,8(2),99–117.

Baumgart, B. (1974, October). Geometric Modeling for Computer Vision (Tech. Rep. No. AIM-249). Palo Alto, CA: Stanford University AI Laboratory.

Beck, K., & Andres, C. (2004). Extreme Programming Explained: Embrace Change(Second ed.). Reading, MA: Addison-Wesley.

Berlin, B., & Kay, P. (1969). Basic Color Terms: Their Universality and Evolution. Berkeley, CA: University of California Press.

Berns, R. S. (2000). Billmeyer and Saltzman's Principles of Color Technology (3rd ed.). New York: John Wiley and Sons. Blinn, J. (1982). A Generalization of Algebraic Surface Drawing. ACM Transactions on Graphics, 1(3),235–258.

Blinn, J. (1996). Jim Blinn's Corner. San Francisco, CA: MorganKaufmann. Blinn, J. F. (1976). Textureand Reflection in Computer GeneratedImages. Communicationsof the ACM, 19(10), 542–547.

Bloomenthal, J. (1988). Polygonization of Implicit Surfaces. Computer Aided Geometric Design, 4(5),341–355.

Bloomenthal, J. (1990). Calculation of Reference Frames Along a Space Curve. In A. Glassner (Ed.), Graphics Gems (pp. 567–571). Boston: Academic Press.

Bloomenthal,J. (1995). Skeletal Design of Natural Forms (Unpublished doctoral dissertation). University of Calgary,Canada.

Bloomenthal, J. (1997). Bulge Elimination in Convolution Surfaces. Computer Graphics Forum, 16(1),31–41.

Bloomenthal, J., & Shoemake, K. (1991). Convolution Surfaces. Proc. SIGGRAPGH '91, Computer Graphics, 25(4),251–257.

Brandenburg, F. J. (1988). Nice Drawing of Graphs are Computationally Hard. In P. Gorney & M. J. Tauber (Eds.), Visualization in Human-Computer Interaction (pp. 1–15). Berlin: Springer-Verlag.

Bresenham, J. E. (1965). Algorithm for Computer Control of a Digital Plotter. IBM Systems Journal, 4(1),25–30.

Brewer, C. A. (1999). Color Use Guidelines for Data Representation. In Proc. Section on Statistical Graphics (pp. 55–60). Alexandria, VA: American Statistical Association.

Buchheim, C., Ju¨nger, M., & Leipert, S. (2002). Improving Walker's Algorithm to Run in Linear Time. InGD '02: Revised Papers from the 10th International Symposium on Graph Drawing (pp. 344–353). London: SpringerVerlag.

Campagna, S., Kobbelt, L., & Seidel, H.-P. (1998). Directed Edges—A Scalable RepresentationforTriangleMeshes. journalof graphicstools, 3(4),1–12.

Carr, J. C., Beatson, R. K., Cherrie, J. B., Mitchell, T. J., Fright, W. R., McCallum, B.C., & Evans, T. R. (2001). Reconstruction and representation of 3d objects with radial basis functions. In SIGGRAPH '01: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (pp. 67–76). New York: ACM Press.

Chiu, K., Herf, M., Shirley, P., Swamy, S., Wang, C., & Zimmerman, K. (1993). Spatially Nonuniform Scaling Functions for High Contrast Images. InProceedings of Graphics Interface ’93 (pp. 245–253). Wellesley, MA: A K Peters & Canadian Human-ComputerCommunicationsSociety.

Choudhury, P., & Tumblin, J. (2003). The Trilateral Filter for High Contrast Images and Meshes. In EGRW ’03: Proceedings of the 14th Eurographics Workshop on Rendering (pp. 186–196). Aire-la-Ville, Switzerland: EurographicsAssociation.

Cleary, J., Wyvill, B., Birtwistle, G., & Vatti, R. (1983). A Parallel Ray Tracing Computer. In Proceedings of the Association of Simula Users Conference (pp.77–80).

Cohen, E., Riesenfeld, R. F., & Elber, G. (2001). Geometric Modeling with Splines: An Introduction. Wellesley, MA: A K Peters.

Cohen, M. F., Chen, S. E., Wallace, J. R., & Greenberg,D. P. (1988). A Progressive Refinement Approach to Fast Radiosity Image Generation. In SIGGRAPH ’88: Proceedings of the 15th Annual Conference on Computer Graphicsand Interactive Techniques (pp.75–84). New York: ACM Press.

Cohen, M. F., & Wallace, J. R. (1993). Radiosity and Realistic Image Synthesis. Cambridge,MA: Academic Press, Inc.

Comaniciu,D., & Meer,P. (2002). MeanShift: ARobustApproachTowardFeature Space Analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence,24(5),603–619.

Cook, R. L., Carpenter, L., & Catmull, E. (1987). The Reyes Image Rendering Architecture. Proc. SIGGRAPH ’87 Computer Graphics, 21(4),95–102.

Cook, R. L., Porter, T., & Carpenter, L. (1984). Distributed Ray Tracing. Proc. SIGGRAPH ’84, Computer Graphics, 18(3),137–145.

Cook, R. L., & Torrance, K. E. (1982). A Reflectance Model for Computer Graphics. ACM Transactions on Graphics, 1(1),7–24.

Crow, F. C. (1978). The Use of Grayscale for Improved Raster DisplayofVectors and Characters. In SIGGRAPH ’78: Proceedings of the 5th Annual Conference on Computer Graphics and Interactive Techniques (pp. 1–5). New York: ACM Press.

Crowe, M. J. (1994). A History of Vector Analysis. Mineola, NY: Dover. da Vinci, L. (1970). The Notebooksof Leonardo da Vinci(Vol. 1). Mineola, NY: DoverPress.

Dachsbacher, C., Vogelgsang, C., & Stamminger, M. (2003). Sequential Point Trees. ACM Transactions on Graphics, (Proc. SIGGRAPH03), 22(3),657– 662.

Debevec, P. E., & Malik, J. (1997). Recovering High Dynamic Range Radiance Mapsfrom Photographs. In SIGGRAPH’97: Proceedingsof the 24th Annual Conferenceon Computer Graphics and Interactive Techniques (pp. 369–378). Reading, MA: Addison-Wesley.

De Boor, C. (1978). A Practical Guide to Splines. Berlin: Springer-Verlag. De Boor, C. (2001). A Practical Guide to Splines. Berlin: Springer-Verlag.

deGroot, E., & Wyvill, B. (2005). Rayskip: Faster Ray Tracing of Implicit Surface Animations. In GRAPHITE ’05: Proceedings of the 3rd International Conferenceon Computer Graphicsand InteractiveTechniquesin Australasia and South East Asia (pp. 31–36). New York: ACM Press.

DeRose, T. (1989). A Coordinate-Free Approach to Geometric Programming (Tech. Rep. No. 89-09-16). Seattle, WA: University of Washington.

Di Battista, G., Eades, P., Tamassia, R., & Tollis, I. G. (1999). Graph Drawing: Algorithms for the Visualization of Graphs. Englewood Cliffs, NJ : Prentice Hall.

Dinh, H., Slabaugh, G., & Turk, G. (2001). Reconstructing Surfaces Using Anisotropic Basis Functions. In International Conference on Computer Vision (ICCV) 2001(pp. 606–613). Washington, DC: IEEE.

Dobkin, D. P., & Mitchell, D. P. (1993). Random-Edge Discrepancy of Supersampling Patterns. In Proceedings of Graphics Interface (pp. 62–69). Wellesley, MA: A K Peters & Canadian Human-Computer Communications Society.

Dooley, D., & Cohen, M. F. (1990). Automatic Illustration of 3D Geometric Models: Lines. In SI3D'90: Proceedings of the 1990 Symposium on Interactive 3D Graphics(pp. 77–82). New York: ACM Press.

Doran,C., & Lasenby,A. (2003). Geometric Algebra for Physicists. Cambridge, UK: Cambridge University Press.

Drago, F., Myszkowski, K., Annen, T., & Chiba, N. (2003). Adaptive Logarithmic Mapping for Displaying High Contrast Scenes. Computer Graphics Forum, 22(3), 419–426.

Duchon, J. (1977). Constructive Theory of Functions of Several Variables. In (pp. 85–100). Berlin: Springer-Verlag.

Durand, F., & Dorsey, J. (2002). Fast Bilateral Filtering for the Display of High Dynamic-Range Images. ACM Transactionson Graphics, 21(3), 257–266.

Dutre´, P., Bala, K., & Bekaert, P.(2002). Advanced Global Illumination. Wellesley, MA: A K Peters.

Dwyer, T., Koren, Y., & Marriott, K. (2006). IPSep-CoLa: An Incremental Procedure for Separation Constraint Layout of Graphs. IEEETrans. Visualization and Computer Graphics (Proc. InfoVis 06), 12(5),821–828.

Eberly,D. (2000). 3D Game Engine Design: A Practical Approach to Real-Time Computer Graphics. San Francisco, CA: Morgan Kaufmann.

Eckman, P., & Friesen, W. V. (1978). Facial Action Coding System. Palo Alto, CA: Consulting PsychologistsPress.

Eick, S. G., Steffen, J. L., & Sumner, E. E., Jr. (1992). Seesoft-A Tool for Visualizing Line Oriented Software Statistics. IEEE Trans. Software Eng., 18(11),957–968.

Ershov, S., Kolchin, K., & Myszkowski, K. (2001). Rendering Pearlescent AppearanceBased on Paint-Composition Modelling. Computer Graphics Forum, 20(3),227–238.

Fairchild, M. D. (2005). Color Appearance Models (Second ed.). New York: John Wiley & Sons.

Fairchild, M. D., & Johnson, G. M. (2002). Meet iCAM: An Image Color Appearance Model. InIS&T/SID 10 th Color Imaging Conference (pp. 33 – 38). Springfield, VA: Society for Imaging Science & Technology.

Fairchild, M. D., & Johnson, G. M. (2004). The iCAM Framework for Image Appearance, Image Differences, and Image Quality. Journal of Electronic Imaging,13,126–138.

Farin, G. (2002). Curves and Surfaces for CAGD: A Practical Guide. San Francisco, CA: Morgan Kaufmann.

Farin, G., & Hansford, D. (2004). Practical Linear Algebra: A Geometry Toolbox. Wellesley, MA: A K Peters.

Farin, G., Hoschek, J., & Kim, M.-S. (Eds.). (2002). Handbook of Computer Aided Geometric Design. Amsterdam: Elsevier.

Fattal, R., Lischinski, D., & Werman, M. (2002). Gradient Domain High Dynamic Range Compression. ACM Transactions on Graphics, 21(3),249–256.

Fekete, J.-D., & Plaisant, C. (2002). Interactive Information Visualization of a MillionItems. InProc. IEEESymposiumon Information Visualization (InfoVis 02) (pp. 117–124). Washington, DC: IEEE Computer Society Press.

Ferwerda, J. A., Pattanaik, S., Shirley, P., & Greenberg, D. P. (1996). A Model of Visual Adaptation for Realistic Image Synthesis. In SIGGRAPH ’96: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques(pp. 249–258). New York: ACM Press.

Ferwerda, J. A., Shirley, P., Pattanaik, S. N., & Greenberg, D. P. (1997). A Model of Visual Masking for Computer Graphics. In SIGGRAPH ’97: Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques(pp. 143–152). Reading, MA: Addison-Wesley.

Foley, J. D., Van Dam, A., Feiner, S. K., & Hughes, J. F. (1990). Computer Graphics: Principles and Practice (Second ed.). Reading, MA: Addison Wesley.

Forsyth, D. A., & Ponce, J. (2002). Computer Vision: A Modern Approach. Englewoods Cliffs, NJ: Prentice Hall.

Francis S. Hill, J. (2000). Computer Graphics Using OpenGL (Second ed.). EnglewoodCliffs, NJ: Prentice Hall.

Frick, A., Ludwig, A., & Mehldau, H. (1994). A Fast Adaptive Layout Algorithm for Undirected Graphs. In GD ’94: Proceedings of the DIMACS International Workshop on Graph Drawing (pp. 388–403). London: SpringerVerlag.

Friendly, M. (2008). A Brief History of Data Visualization. In Handbook of Data Visualization (pp. 15–56). (Web document, http://www.math.yorku.ca/SCS/Gallery/milestone/.)

Frisken, S., Perry, R., Rockwood, A., & Jones, T. (2000). Adaptively Sampled Distance Fields. In Siggraph ’00: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (pp. 249–254). New York: ACM Press/Addison-Wesley Publishing Co.

Fua, Y.-H., Ward, M. O., & Rundensteiner, E. A. (1999). Hierarchical Parallel Coordinates for Exploration of Large Datasets. In Proc. IEEE Visualization Conference (Vis ’99) (pp. 43–50). Washington, DC: IEEE Computer Society Press.

Fujimoto, A., Tanaka, T., & Iwata, K. (1986). ARTSccelerated Ray-Tracing System. IEEE Computer Graphics & Applications,6(4),16–26.

Galin, E., & Akkouche, S. (1999). Incremental Polygonization of Implicit Surfaces. Graphical Models, 62(1),19–39.

Gansner, E. R., Koutsofois, E., North, S. C., & Vo, K.-P. (1993, March). A Technique for Drawing Directed Graphs. IEEE Transactions on Software Engineering,19(3),214–229.

Gascuel,M.-P. (1993,Aug). AnImplicit Formulation for Precise Contact Modeling Between Flexible Solids. In SIGGRAPH ’93: Proceedings of the 20th AnnualConferenceon ComputerGraphics and InteractiveTechniques(pp. 313–320). New York: ACM Press.

Gibson, J. J. (1950). The Perception of the Visual World. Cambridge, MA: RiversidePress. Gilchrist, A. L., Kossyfidis, C., Bonato, F., Agostini, T., Cataliotti, J., Li, X., ... Economou, E. (1999). An Anchoring Theory of Lightness Perception

PsychologicalReview, 106(4),795–834. Glassner, A. (1984). Space Subdivision for Fast Ray Tracing. IEEE Computer Graphics & Applications, 4(10),15–22.

Glassner, A. (1988). Spacetime Ray Tracing for Animation. IEEE Computer Graphics & Applications, 8(2),60–70.

Glassner, A. (Ed.). (1989). An Introduction to Ray Tracing. London: Academic Press. Glassner, A. (1995). Principles of Digital Image Synthesis. San Francisco, CA: MorganKaufmann.

Goldman, R. (1985). Illicit Expressions in Vector Algebra. ACM Transactions on Graphics, 4(3),223–243.

Goldsmith, J., & Salmon, J. (1987). Automatic Creation of Object Hierarchies for Ray Tracing. IEEE Computer Graphics & Applications,7(5),14–20.

Gooch, A., Gooch, B., Shirley, P., & Cohen, E. (1998). A Non-Photorealistic Lighting Model for Automatic Technical Illustration. In SIGGRAPH ’98: Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques(pp. 447–452). New York: ACM Press.

Goral, C. M., Torrance, K. E., Greenberg, D. P., & Battaile, B. (1984). Modeling the Interaction of Light between Diffuse Surfaces. Proc. SIGGRAPH ’84, Computer Graphics, 18(3),213–222.

Gouraud, H. (1971). Continuous Shading of Curved Surfaces. Communications of the ACM, 18(6),623–629.

Grassmann, H. (1853). Zur Theorie der Farbenmischung. Annalen der Physik und Chemie, 89, 69–84.

Gregory, R. L. (1997). Eye and Brain: The Psychology of Seeing (Fifth ed.). Princeton,NJ: Princeton University Press.

Grosjean, J., Plaisant, C., & Bederson, B. (2002). SpaceTree: SupportingExploration in Large Node Link Tree, Design Evolution and Empirical Evaluation. In Proc. IEEE Symposium on Information Visualization (InfoVis) (pp. 57–64). Washington, DC: IEEE Computer Society Press.

Grossman, T., Wigdor, D., & Balakrishnan, R. (2007). Exploring and Reducing the Effects of Orientation on Text Readability in Volumetric Displays. In Proc. ACM Conf. Human Factors in Computing Systems (CHI) (pp. 483– 492). New York: ACM Press.

Hammersley, J., & Handscomb, D. (1964). Monte-Carlo Methods. London: Methuen.

Hanrahan,P.,& Lawson,J. (1990). A Language for Shading and Lighting Calculations. In SIGGRAPH’90: Proceedings of the17th Annual Conferenceon Computer Graphics and Interactive Techniques (pp. 289–298). New York: ACM Press.

Hanson, A. J. (2005). Visualizing Quaternions. San Francisco, CA: Morgan Kaufmann.

Harris, M. J. (2004). Fast Fluid Dynamics Simulation on the GPU. In GPU Gems: ProgrammingTechniques,Tips, and Tricks for Real-TimeGraphics (chap. 38). Reading, MA: Addison-Wesley.

Hart, J. C., & Baker,B. (1996,Oct). Implicit Modeling of Tree Surfaces. In Proceedings of Implicit Surfaces’ 96 (pp. 143 – 152). Aire-la-Ville, Switzerland: Eurographics Association.

Hartmann, E. (1998). A Marching Method for the Triangulation of Surfaces. The Visual Computer, 14(3),95–108.

Hausner, M. (1998). A Vector Space Approach to Geometry. Mineola, NY: Dover. Havran, V. (2000). Heuristic Ray Shooting Algorithms (Unpublished doctoral dissertation). Czech Technical University in Prague.

He, X. D., Heynen, P. O., Phillips, R. L., Torrance, K. E., Salesin, D. H., & Greenberg, D. P. (1992). A Fast and Accurate Light Reflection Model. Proc. SIGGRAPH ’92, Computer Graphics, 26(2),253–254.

Hearn, D., & Baker, M. P. (1986). Computer Graphics. Englewood Cliffs, N.J.: Prentice Hall.

Heer, J., & Robertson, G. (2007). Animated Transitions in Statistical Data Graphics. IEEE Trans. on Visualization and Computer Graphics (Proc. InfoVis07), 13(6),1240–1247.

Heidrich, W., & Seidel, H.-P. (1998). Ray-Tracing Procedural Displacement Shaders. In Proceedings of Graphics Interface (pp. 8–16). Wellesley, MA: A K Peters & Canadian Human-Computer Communications Society.

Henry, N., & Fekete, J.-D. (2006). MatrixExplorer: a Dual-Representation System to Explore Social Networks. IEEE Trans. Visualization and Computer Graphics (Proc. InfoVis 06), 12(5),677–684.

Hoffmann,B. (1975). About Vectors. Mineola, NY: Dover.

Hofstadter, D. (1979). G¨odel, Escher, Bach: an Eternal Golden Braid. New York: Basic Books.

Hood, D. C., Finkelstein, M. A., & Buckingham, E. (1979). PsychophysicalTests of Modelsof the Response Function. Vision Research, 19, 401–406.

Hoppe, H. (1994). Surface Reconstruction from Unorganized Points (Unpublished doctoraldissertation). University of Washington.

Horn, B. K. P. (1974). Determining Lightness from an Image. CVGIP, 3, 277– 299.

Hughes, J. F., & Mo¨ller, T. (1999). Building an Orthonormal Basis from a Unit Vector. journal of graphics tools, 4(4),33–35.

Hunt, R. W. G. (2004). The Reproduction of Color (6th ed.). Chichester, UK: John Wiley and Sons.

IEEE Standards Association. (1985). IEEE Standard for Binary Floating-Point Arithmetic (Tech. Rep.). New York: IEEE Report. (ANSI/IEEE Std 7541985)

Igarashi, T., Matsuoka, S., & Tanaka, H. (1999). Teddy: A Sketching Interface for 3D FreeformDesign. In Siggraph ’99: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (pp. 409– 416). New York: ACM Press/Addison-Wesley Publishing Co.

Immel, D. S., Cohen, M. F., & Greenberg, D. P. (1986). A Radiosity Method for Non-Diffuse Environments. Proc. SIGGRAPH ’86, Computer Graphics, 20(4),133–142.

Ingram, S., Munzner, T., & Olano, M. (2009). Glimmer: Multilevel MDS on the GPU. IEEETrans. Visualization and Computer Graphics, 15(2), 249–261.

Inselberg, A., & Dimsdale, B. (1990). Parallel Coordinates: A Tool for Visualizing Multi-Dimensional Geometry. In Vis ’90: Proceedings of the 1st Conferenceon Visualization’90. LosAlamitos, CA:IEEEComputerSociety Press.

Jansen, F. W. (1986). Data Structures for Ray Tracing. In Proceedings of a Workshop Eurographics Seminars on Data Structures for Raster Graphics (pp. 57–73). New York: Springer-Verlag.

Jensen, H. W. (2001). Realistic ImageSynthesis Using Photon Mapping. Wellesley, MA: A K Peters.

Johansson, G. (1973). Visual Perception of Biological Motion and a Model for Its Analysis. Perception & Psychophysics, 14,201–211.

Johnson, B., & Shneiderman, B. (1991). Treemaps: A Space-filling Approach to the Visualization of Hierarchical Information. In VIS ’91: Proceedings of the 2nd Conference on Visualization ’91 (pp. 284–291). Los Alamitos, CA: IEEE Computer Society Press.

Johnson, G. M., & Fairchild, M. D. (2003). Rendering HDR Images. InIS&T/SID 11th Color Imaging Conference (pp. 36–41). Springfield, VA: Society for Imaging Science & Technology.

Jones, J. A., Harrold, M. J., & Stasko, J. (2002). VisualizationofTestInformation to Assist Fault Localization. In ICSE ’02: Proceedings of the 24th International Conference on Software Engineering (pp. 467–477). New York: ACM Press.

Judd, D. B. (1932). Chromaticity Sensibility to Stimulus Differences. Journal of the Optical Society of America, 22, 72–108.

Kainz, F., Bogart, R., & Hess, D. (2003). The Open EXR Image File Format. In SIGGRAPH Technical Sketches. New York: ACM Press. (see also: http://www.openexr.com/)

Kajiya, J. T. (1986). The Rendering Equation. Proc SIGGRAPH ’86 Computer Graphics, 20(4),143–150.

Kalos, M., & Whitlock, P. (1986). Monte Carlo Methods, Basics. New York: Wiley-Interscience.

Kalra, D., & Barr, A. (1989, July). Guaranteed Ray Intersections with Implicit Functions. Computer Graphics (Proc. SIGGRAPH 89), 23(3),297–306.

Kay, D. S., & Greenberg, D. (1979). Transparency for Computer Synthesized Images. Proc. SIGGRAPH ’79 Computer Graphics, 13(2),158–164.

Kernighan, B. W., & Pike, R. (1999). The Practice of Programming. Reading, MA: Addison-Wesley.

Kersten,D., Mamassian, P., &Knill, D. C. (1997). MovingCast ShadowsInduce ApparentMotion in Depth. Perception, 26(2),171–192.

Kindlmann, G., Weinstein, D., & Hart, D. (2000). Strategies for Direct Volume Rendering of Diffusion Tensor Fields. IEEE Transactions on Visualization and Computer Graphics, 6(2),124–138.

Kirk, D., & Arvo, J. (1988). The Ray Tracing Kernel. In Proceedings of Ausgraph. Melbourne,Australia: Australian Computer Graphics Association.

Klatzky, R. L. (1998). Allocentric and Egocentric Spatial Representations: Definitions, Distinctions, and Interconnections. In C. Freksa, C. Habel, & K. F. Wender (Eds.), Spatial Cognition—An Interdisciplinary Approach to Representation and Processing of Spatial Knowledge (Vol. 5, pp. 1–17). Berlin: Springer-Verlag.

Knill, D. C. (1998). Surface OrientationFrom Texture: IdealObservers,Generic Observers and the Information Content of Texture Cues. Vision Research, 38,1655–1682.

Kollig, T., & Keller, A. (2002). Efficient Multidimensional Sampling. Computer Graphics Forum, 21(3),557–564.

Kovitz, B. L. (1999). Practical Sofware Requirements: A Manual of Content & Style. New York: Manning.

Lafortune, E. P. F., Foo, S.-C., Torrance, K. E., & Greenberg, D. P. (1997). Non-Linear Approximation of Reflectance Functions. In Proceedings of SIGGRAPH ’97 (pp. 117–126). Reading, MA: Addison-Wesley.

Laramee, R. S., Hauser, H., Doleisch, H., Vrolijk, B., Post, F. H., & Weiskopf, D. (2004). The State of the Art in Flow Visualization: Dense and TextureBased Techniques. Computer Graphics Forum, 23(2),203–221.

Lasseter, J. (1987). Principles of Traditional Animation Applied to 3D Computer Animation. Proc. SIGGRAPH ’87, Computer Graphics, 21(4),35–44.

Lawrence, J., Rusinkiewicz, S., & Ramamoorthi, R. (2004). Efficient BRDF Importance Sampling Using a Factored Representation. ACM Transactions on Graphics (Proc. SIGGRAPH ’04),23(3),496–505.

Lee, D. N., & Reddish, P. (1981). Plummeting Gannets: A Paradigm of Ecological Optics. Nature, 293,293–294.

Leung, T., & Malik, J. (1997). On Perpendicular Texture: Why DoWe See More Flowers in the Distance? In Proc. IEEE Conference on Computer Vision and Pattern Recognition(pp. 807–813). Los Alamitos, CA: IEEE Press.

Lewis, C., & Rieman, J. (1993). Task-Centered User Interface Design: A Practical Introduction. http://hcibib.org/tcuid/.

Lewis, R. R. (1994). Making Shaders More Physically Plausible. Computer Graphics Forum, 13(2),109–120.

Loop, C. (2000). Managing Adjacency in Triangular Meshes (Tech. Rep. No. MSR-TR-2000-24). Bellingham, WA: Microsoft Research.

Lorensen, W., & Cline, H. (1987). Marching Cubes: A High Resolution 3D Surface Construction Algorithm. Computer Graphics (Proc. SIGGRAPH 87), 21(4),163–169.

Luboschik, M., Schumann, H., & Cords, H. (2008). Particle-Based Labeling: Fast Point-Feature Labeling without Obscuring Other Visual Features. IEEE Trans. on Visualization and Computer Graphics (Proc. InfoVis08), 14(6), 1237–1244.

MacEachren, A., Dai, X., Hardisty, F., Guo, D., & Lengerich, G. (2003). Exploring High-D Spaces with Multiform Matrices and Small Multiples. In Proc. IEEESymposiumon InformationVisualization (InfoVis) (pp.31–38). Washington, DC: IEEE Computer Society Press.

Mackinlay, J. (1986). Automating the Design of Graphical Presentations of RelationalInformation. ACM Trans. on Graphics (TOG), 5(2),110–141.

Malley, T. (1988). A Shading Method for Computer Generated Images (Unpublishedmaster’sthesis). Computer Science Department, University of Utah.

Marschner, S. R., & Lobb, R. J. (1994, Oct). An Evaluation of Reconstruction Filters for Volume Rendering. In VIS ’94: Proceedings of the Conference on Visualization’94(pp.100–107). Washington, DC : IEEE Computer Society Press.

Marshall, J. A., Burbeck, C. A., Arely, D., Rolland, J. P., & Martin, K. E. (1999). Occlusion Edge Blur: A Cue to Relative Visual Depth. Journal of the Optical Society of America A, 13,681–688.

Matusik, W., Pfister, H., Brand, M., & McMillan, L. (2003). A Data-Driven Reflectance Model. ACM Transactions on Graphics (Proc. SIGGRAPH ’03),22(3),759–769.

McLouglin,T., Laramee,R. S., Peikert, R., Post, F. H., & Chen, M. (2009). Over Two Decades of Integration-Based Geometric Flow Visualization. InProc. Eurographics 2009, State of the Art Reports. Aire-la-Ville, Switzerland: Eurographics Association.

Meyers, S. (1995). More Effective C++ : 35 New Ways to Improve Your Programs and Designs. Reading, MA: Addison-Wesley.

Meyers, S. (1997). Effective C++: 50 Specific Ways to Improve Your Programs and Designs (Second ed.). Reading, MA: Addison-Wesley.

Mitchell, D. P. (1990, May). Robust Ray Intersection with Interval Arithmetic. In Graphics interface ’90 (pp. 68–74). Wellesley, MA: Canadian Human Computer Communications Society & A K Peters.

Mitchell, D. P. (1996). Consequences of Stratified Sampling in Graphics. In SIGGRAPH’96: Proceedingsof the23rd Annual Conferenceon Computer Graphics and Interactive Techniques (pp. 277–280). New York: ACM Press.

Mitchell, D. P., & Netravali, A. N. (1988). Reconstruction Filters in Computer Graphics. Computer Graphics, 22(4),221–228.

Morse, B., Yoo, T., Rheingans, P., Chen, D., & Subramanian, K. (2001). Interpolating Implicit Surfaces from Scattered Surface Data Using Compactly Supported Radial Basis Functions. In Proceedings of Shape Modeling International(pp. 89–98). Washington, DC: IEEE Computer Society Press.

Mortenson,M. (1985). Geometric Modeling. New York: John Wiley & Sons. Munkres,J. (2000). Topology (Seconded.). Englewood Cliffs, NJ : Prentice Hall.

Munzner, T. (2000). Interactive Visualization of Large Graphs and Networks (Unpublished doctoral dissertation). Stanford University Department of ComputerScience.

Munzner, T., Guimbretie`re, F., Tasiran, S., Zhang, L., & Zhou, Y. (2003). TreeJuxtaposer: Scalable Tree ComparisonUsing Focus+Contextwith Guaranteed Visibility. ACM Transactions on Graphics (Proc. SIGGRAPH ’03), 22(3),453–462.

Museth, K., Breen, D., Whitaker, R., & Barr, A. (2002). LevelSetSurfaceEditing Operators. ACM Transactions on Graphics, 21(3),330–338.

Muuss, M. J. (1995). Towards Real-Time Ray-Tracing of Combinatorial Solid Geometric Models. In Proceedings of BRL-CAD Symposium.

Nicodemus, F. E., Richmond, J. C., Hsia, J. J., Ginsberg, I., & Limperis, T. (1977). Geometrical Considerations and Nomenclature for Reflectance (Tech. Rep. No. 160). Washington, D.C.: National Bureau of Standards.

Nishimura, H., Hirai, A., Kawai, T., Kawata, T., Shirikawa, I., & Omura, K. (1985). Object Modeling by Distribution Function and a Method of Image Generation. In J. Electronics Comm. Conf. ’85 (Vol. J69-D, pp. 718–725).

Ohtake, Y., Belyaev, A., & Pasko, A. (2003). Dynamic Mesh Optimization for Polygonized Implicit Surfaces with Sharp Features. The Visual Computer, 19(2),115–126.

Oppenheim, A. V., Schafer, R., & Stockham, T. (1968). Nonlinear Filtering of Multiplied and ConvolvedSignals. Proceedings of the IEEE, 56(8),1264– 1291.

Oren, M., & Nayar, S. K. (1994). Generalization of Lambert's Reflectance Model. In SIGGRAPH '94: Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques (pp. 239–246). New York: ACM Press.

Osher,S.,&Sethian,J.A. (1988). FrontsPropagatingwithCurvature-Dependent Speed: Algorithms Based on Hamilton–Jacobi Formulations. Journal of Computational Physics, 79(1),12–49.

Osterberg,G. (1935). Topographyof the Layer of Rods and Cones in the Human Retina. Acta Ophthalmologica,6(1),11–97.

Paeth, A. W. (1990). A Fast Algorithm for GeneralRaster Rotation. In Graphics Gems (pp. 179–195). Boston, MA: Academic Press.

Palmer, S. E. (1999). Vision Science—Photons to Phenomenology. Cambridge, MA: MIT Press.

Parker, S., Martin, W., Sloan, P., Shirley, P., Smits, B., & Hansen, C. (1999). Interactive Ray Tracing. In ACM Symposium on Interactive 3D Graphics (pp. 119–126). New York: ACM Press.

Parker, S. G., Bigler, J., Dietrich, A., Friedrich, H., Hoberock, J., Luebke, D., ... Stich, M. (2010, July). Optix: A General Purpose Ray Tracing Engine. ACM Trans. Graph., 29(4), 66:1–66:13. doi: 10.1145/1778765.1778803

Pascale, D. (2003). A review of RGB color spaces (Tech. Rep.). The Babel Color Company. (www.babelcolor.com.)

Pashler, H. E. (1998). The Psychology of Attention. Cambridge, MA : MITPress.

Pasko, A., Adzhiev, V., Sourin, A., & Savchenko,V. (1995). Function Representation in Geometric Modeling: Concepts, Implementation and Applications. The Visual Computer, 11(8),419–428.

Pasko, G., Pasko, A., Ikeda, M., & Kunii, T. (2002, May). Bounded Blending Operations. In Proceedings of the International Conference on Shape Modeling and Applications (SMI 2002) (pp. 95–103). Washington, DC: IEEE Computer Society Press.

Pattanaik, S. N., Ferwerda, J. A., Fairchild, M. D., & Greenberg, D. P. (1998). A Multiscale Model of Adaptation and Spatial Vision for Realistic Image Display. In SIGGRAPH '98: Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques (pp. 287–298). New York: ACM Press.

Pattanaik, S. N., & Yee, H. (2002). Adaptive Gain Control for High Dynamic Range Image Display. In SCCG '02: Proceedings of the 18th Spring Conference on Computer Graphics (pp. 83–87). New York: ACM Press.

Patterson, J., Hoggar, S., & Logie, J. (1991). Inverse Displacement Mapping. Computer Graphics Forum, 10(2),129–139.

Peachey, D. R. (1985). Solid Texturingof Complex Surfaces. Proc. SIGGRAPH '85, Computer Graphics, 19(3),279–286.

Penna, M., & Patterson, R. (1986). Projective Geometry and Its Applications to Computer Graphics. EnglewoodCliffs, NJ: Prentice Hall.

Perlin, K. (1985). An Image Synthesizer. Computer Graphics, 19(3), 287–296. (SIGGRAPH ’85) Perlin, K., & Hoffert, E. M. (1989). Hypertexture. Computer Graphics, 23(3), 253–262. (SIGGRAPH ’89)

Pharr, M., & Hanrahan,P. (1996). GeometryCaching for Ray-Tracing Displacement Maps. In Proceedings of the Eurographics Workshop on Rendering Techniques’96 (pp. 31–40). London,UK: Springer-Verlag.

Pharr, M., & Humphreys, G. (2004). Physically Based Rendering. SanFrancisco, CA: MorganKaufmann.

Pharr, M., Kolb, C., Gershbein, R., & Hanrahan, P. (1997). Rendering Complex Sceneswith Memory-Coherent Ray Tracing. In SIGGRAPH’97: Proceed ings of the 24th Annual Conference on Computer Graphics and Interactive Techniques(pp. 101–108). Reading, MA: Addison-Wesley.

Phong,B.-T. (1975). Illumination for Computer Generated Images. Communications of the ACM, 18(6),311–317.

Pineda, J. (1988). A Parallel Algorithm for Polygon Rasterization. Proc. SIGGRAPH ’88, Computer Graphics, 22(4),17–20.

Pitteway, M. L. V. (1967). Algorithm for Drawing Ellipses or Hyperbolae with a Digital Plotter. Computer Journal, 10(3),282–289.

Plauger, P. J. (1991). The Standard C Library. Englewood Cliffs, NJ: Prentice Hall.

Plumlee, M., & Ware, C. (2006). Zooming Versus Multiple Window Interfaces: Cognitive Costs of Visual Comparisons. Proc. ACM Trans. on Computer Human Interaction (ToCHI), 13(2),179–209.

Porter, T., & Duff, T. (1984). Compositing Digital Images. In SIGGRAPH ’84: Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques(pp. 253–259). New York: ACM Press.

Poynton, C. (2003). Digital Video and HDTV: Algorithms and Interfaces. San Francisco: Morgan Kaufmann Publishers.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (1992). Numerical Recipes in C: The Art of Scientific Computing (Second ed.). Cambridge, UK: Cambridge University Press.

Purcell, T. J., Buck, I., Mark, W. R., & Hanrahan, P. (2002). Ray Tracing on Programmable Graphics Hardware. ACM Transactionson Graphics(Proc. SIGGRAPH ’02),21(3),703–712.

Rahman, Z., Jobson, D. J., & Woodell, G. A. (1996). A Multiscale Retinex for Color Rendition and Dynamic Range Compression. In SPIE Proceedings: Applications of Digital Image Processing XIX (Vol. 2847). Bellingham, WA: SPIE.

Rao, R., & Card, S. K. (1994). TheTableLens: MergingGraphicalandSymbolic Representations in an Interactive Focus+Context Visualization for Tabular Information. In Proc. ACM Human Factors in Computing Systems (CHI) (pp. 318–322). New York: ACM Press.

Reeves, W. T. (1983). Particle Systems—A Technique for Modeling a Class of Fuzzy Objects. ACM Transactions on Graphics, 2(2),91–108.

Reingold, E. M., & Tilford, J. S. (1981). Tidier Drawings of Trees. IEEE Trans. Software Engineering,7(2),223–228.

Reinhard, E. (2003). Parameter Estimation for Photographic Tone Reproduction. journal of graphics tools, 7(1),45–51.

Reinhard, E., Ashikhmin, M., Gooch, B., & Shirley, P. (2001). Color Transfer Between Images. IEEE Computer Graphics and Applications,21,34–41.

Reinhard, E., & Devlin, K. (2005). Dynamic Range Reduction Inspired by Photoreceptor Physiology. IEEE Transactions on Visualization and Computer Graphics, 11(1),13–24.

Reinhard, E., Khan, E. A., Akyu¨z, A. O., & Johnson, G. M. (2008). Color Imaging: Fundamentalsand Applications. Wellesley: A K Peters.

Reinhard, E., Stark, M., Shirley, P., & Ferwerda, J. (2002). Photographic Tone Reproduction for Digital Images. ACM Transactions on Graphics (Proc. SIGGRAPH'02), 21(3),267–276.

Reinhard, E., Ward, G., Debevec, P., & Pattanaik, S. (2005). High Dynamic Range Imaging. San Francisco: MorganKaufmann.

Requicha, A. A. G. (1980). Representations for Rigid Solids: Theory, Methods and Systems. ComputingSurveys, 12(4),437–464.

Reuter, P. (2003). Reconstruction and Rendering of Implicit Surfaces from Large Unorganized Point Sets (Unpublished doctoral dissertation). LABRI – Universite Bordeaux.

Reynolds, C. W. (1987). Flocks, Herds and Schools: A Distributed Behavioral Model. Proc. SIGGRAPH '87, Computer Graphics, 21(4),25–34.

Ricci,A. (1973,May). Constructive Geometry for Computer Graphics. Computer Journal, 16(2), 157–160.

Riesenfeld, R. F. (1981, January). Homogeneous Coordinates and Projective Planes in Computer Graphics. IEEE Computer Graphics & Applications, 1(1),50–55.

Roberts, L. (1965,May). Homogenous Matrix Representation and Manipulation of N-Dimensional Constructs (Tech. Rep. No. MS-1505). Lexington, MA: MIT Lincoln Laboratory.

Robertson, G., Fernandez,R., Fisher, D., Lee, B., & Stasko, J. (2008). Effectiveness of Animation in Trend Visualization. IEEE Trans. on Visualization and Computer Graphics (Proc. InfoVis08),14(6),1325–1332.

Rogers, D. F. (1985). Procedural Elements for Computer Graphics. New York: McGrawHill.

Rogers,D.F. (1989). Mathematical Elements for Computer Graphics. NewYork: McGrawHill.

Rogers, D. F. (2000). An Introduction to NURBS: With Historical Perspective. San Francisco, CA: Morgan Kaufmann.

Rogers, S. (1995). PerceivingPictorial Space. In W. Epstein & S. Rogers (Eds.), Perception of Space and Motion (Vol. 5, pp. 119–163).

San Diego: Academic Press. Roth, S. (1982). Ray Casting for Modeling Solids. Computer Graphics and Image Processing, 18(2),109–144.

Rtiling, S. M., & Whitted, T. (1980). A 3-Dimensional Representation for Fast Rendering of Complex Scenes. Proc.SIGGRAPH'80, Computer Graphics, 14(3),110–116.

Rusinkiewicz, S., & Levoy, M. (2000). QSplat: A Multiresolution Point Rendering System for Large Meshes. In SIGGRAPH'00: Proceedingsof the 27th Annual Conferenceon Computer Graphics and Interactive Techniques(pp. 343–352). Reading, MA: Addison-Wesley.

Rvachev,V. L. (1963). On the Analytic Descriptionof Some Geometric Objects. Reports of the Ukrainian Acadameyof Sciences, 153,765–767.

Saito, T., & Takahashi, T. (1990). Comprehensible Rendering of 3-D Shapes. Proc. SIGGRAPH ’90, Computer Graphics, 24(4),197–206.

Salomon, D. (1999). Computer Graphics and Geometric Modeling. New York: Springer-Verlag.

Savchenko, V., Pasko, A., Okunev,O., & Kunii, T. (1995). Function Representation of Solids Reconstructed from Scattered Surface Points and Contours. Computer Graphics Forum, 14(4),181–188.

Savchenko, V. V., Pasko, A. A., Sourin, A. I., & Kunii, T. L. (1998). Volume Modelling: Representations and Advanced Operations. In CGI ’98: Proceedings of the Computer Graphics International 1 (pp. 4–13). Washington,DC: IEEE Computer Society Press.

Sbert, M. (1997). The Use of Global Random Directions to Compute Radiosity. Global Monte Carlo Techniques (PhD. Thesis). Universitat Polite`nica de Catalunya.

Schlick, C. (1994a). An Inexpensive BRDF Model for Physically-Based Rendering. Computer Graphics Forum, 13(3),233–246.

Schlick, C. (1994b). Quantization Techniques for the Visualization of High Dynamic Range Pictures.

In P. Shirley, G. Sakas, & S. Mu¨ller (Eds.), Photorealistic Rendering Techniques(pp. 7–20). Berlin: Springer-Verlag.

Schmidt, R., Grimm, C., & Wyvill, B. (2006, July). Interactive Decal Compositing with Discrete ExponentialMaps. ACM Transactions on Graphics, 25(3),605–613.

Schmidt, R., Wyvill, B., & Galin, E. (2005). Interactive Implicit Modeling with Hierarchical Spatial Caching. In SMI ’05: Proceedings of the International Conference on Shape Modeling and Applications 2005 (pp. 104– 113). Washington, DC: IEEE Computer Society Press.

Schmidt, R., Wyvill, B., Sousa, M. C., & Jorge, J. A. (2005). Shapeshop: Sketch-Based Solid Modeling with BlobTrees. In Proceedings of the 2nd Eurographics Workshop on Sketch-Based Interfaces and Modeling. Airela-ville, Switzerland: EurographicsAssociation.

Sederberg, T. W., & Parry, S. R. (1986). Free-Form Deformation of Solid Geometric Models. Proc. SIGGRAPH ’86, Computer Graphics, 20(4), 151– 160.

Seetzen, H., Heidrich, W., Stuerzlinger, W., Ward, G., Whitehead, L., Trentacoste, M., ... Vorozcovs, A. (2004). High Dynamic Range Display Systems. ACM Transactions on Graphics (Proc. SIGGRAPH ’04), 23(3),760–768.

Seetzen, H., Whitehead, L. A., & Ward, G. (2003). A High Dynamic Range Display Using Low and High Resolution Modulators. In The Society for Information Display International Symposium. San Jose, CA: Society for InformationDisplay.

Segal, M., Korobkin,C., van Widenfelt,R., Foran, J., & Haeberli, P. (1992). Fast Shadows and Lighting Effects Using Texture Mapping. Proc. SIGGRAPH ’92, Computer Graphics, 26(2),249–252.

Shannon,C. E., & Weaver, W. (1964). The MathematicalTheory of Communication. Urbana, IL: University of Illinois Press. Shapiro, V. (1988). Theory of R-Functions and Applications: A Primer (Tech. Rep. No. CPA88-3). Ithaca, NY: Cornell University.

Shapiro,V. (1994). RealFunctionsforRepresentationofRigid Solids. Computer Aided Geometric Design, 11,153–175. Shene, C.-K. (2003). CS 3621 Introduction to Computing

with Geometry Notes. Available from World Wide Web. (http://www.cs.mtu.edu/shene/COURSES/cs3621/NOTES/notes.html)

Sherstyuk, A. (1999). Interactive Shape Design with Convolution Surfaces. In SMI ’99: Proceedings of the International Conference on Shape Modeling and Applications (pp. 56–65). Washington, DC: IEEE Computer Society Press.

Shirley, P. (1991). Physically Based Lighting Calculations for Computer Graphics (Unpublished doctoral dissertation). University of Illinois, UrbanaChampaign.

Shirley, P., Smits, B., Hu, H., & Lafortune, E. (1997). A Practitioners’ Assessment of Light Reflection Models. In PG ’97: Proceedings of the 5th Pacific Conference on Computer Graphics and Applications (pp. 40–49).

Los Alamitos, CA: IEEE ComputerSociety Press. Shirley, P., Wang, C., & Zimmerman, K. (1996). Monte Carlo Techniques for Direct Lighting Calculations. ACM Transactionson Graphics,15(1),1–36.

Shneiderman, B. (1996). The Eyes Have It: A Task by Data Type Taxonomy for InformationVisualizations. In Proc. IEEE Visual Languages(pp.336– 343). Washington, DC: IEEE Computer Society Press.

Shreiner, D., Neider, J., Woo, M., & Davis, T. (2004). OpenGL Programming Guide (Fourth ed.). Reading, MA: Addison-Wesley.

Sillion, F. X., & Puech, C. (1994). Radiosity and Global Illumination. San Francisco, California: Morgan Kaufmann Publishers, Inc.

Simons,D.J.(2000). Current Approaches to Change Blindness. Visual Cognition, 7(1/2/3),1–15.

Slocum, T. A., McMaster, R. B., Kessler, F. C., & Howard, H. H. (2008). Thematic Cartography and Geovisualization (3rd ed.). Englewood Cliffs, NJ: Prentice Hall.

Smith, A. R. (1995). A Pixel is Not a Little Square! (Technical Memo No. 6). Bellingham, WA: Microsoft Research.

Smith, S., Grinstein, G., & Bergeron, R. D. (1991). Interactive Data Exploration with a Supercomputer. In VIS ’91: Proceedings of the 2nd Conference on Visualization ’91 (pp. 248–254). Los Alamitos, CA: IEEE Computer Society Press.

Smits, B. E., Shirley,P., & Stark, M. M. (2000). DirectRay Tracingof Displacement Mapped Triangles. In Proceedings of the Eurographics Workshop on Rendering Techniques 2000 (pp. 307–318). London, UK: Springer-Verlag.

Snyder, J. M., & Barr, A. H. (1987). Ray Tracing Complex Models Containing Surface Tessellations. Proc. SIGGRAPH ’87, Computer Graphics, 21(4), 119–128.

Sobel, I., Stone, J., & Messer, R. (1975). The Monte Carlo Method. Chicago, IL: University of Chicago Press. Solomon,H. (1978). Geometric Probability. Philadelphia, PA: SIAM Press.

Stam, J. (1999). Diffraction Shaders. In SIGGRAPH’99: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (pp. 101–110). Reading, MA: Addison-Wesley.

Stark, M. M., Arvo, J., & Smits, B. (2005). Barycentric Parameterizations for Isotropic BRDFs. IEEE Transactions on Visualization and Computer Graphics, 11(2),126–138.

Stockham, T. (1972). Image Processing in the Context of a Visual Model. Proceedings of the IEEE, 60(7),828–842.

Stolte, C., Tang, D., & Hanrahan, P. (2008). Polaris: A System for Query, Analysis, and Visualization of Multidimensional Databases. Commun. ACM, 51(11),75–84.

Stone, M. C. (2003). A Field Guide to Digital Color. Natick, MA: A K Peters. Strang, G. (1988). Linear Algebra and Its Applications (Third ed.). Florence, KY: Brooks Cole.

Sutherland,I. E., Sproull,R. F., & Schumacker,R. A. (1974). A Characterization of Ten Hidden-Surface Algorithms. ACM Computing Surveys, 6(1),1–55.

Thompson, W. B., & Pong, T. C. (1990). Detecting Moving Objects. International Journal of Computer Vision, 4(1),39–57.

Thompson, W. B., Shirley, P., & Ferwerda, J. (2002). A Spatial Post-Processing Algorithm for Images of Night Scenes. journal of graphics tools, 7(1), 1–12.

Tomasi, C., & Manduchi, R. (1998). Bilateral Filtering for Gray and Color Images. In Proc. IEEE International Conference on Computer Vision (pp. 836–846). Washington, DC: IEEE Computer Society Press.

Tory, M., Kirkpatrick, A. E., Atkins, M. S., & Mo¨ller, T. (2006). Visualization Task Performance with 2D, 3D, and Combination Displays. IEEE Trans. Visualization and Computer Graphics (TVCG), 12(1),2–13.

Tumblin, J., & Turk, G. (1999). LCIS: A Boundary Hierarchy for DetailPreserving Contrast Reduction. In A. Rockwood (Ed.), SIGGRAPH ’99: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (pp. 83–90). Reading, MA: Addison Wesley Longman.

Turk,G.,& O’Brien,J. (1999). ShapeTransformationUsing VariationalImplicit Functions. InSIGGRAPH’99: Proceedingsofthe26th AnnualConference on Computer Graphics and Interactive Techniques (pp. 335–342). New York: ACM Press/Addison-Wesley Publishing Co.

Turk, G., & O’Brien, J. F. (2002). Modellingwith ImplicitSurfacesthat Interpolate. ACM Transactions on Graphics, 21(4),855–873.

Turkowski,K. (1990). Propertiesof Surface-NormalTransformations. InGraphics Gems (pp. 539–547). Boston: Academic Press.

Tversky, B., Morrison, J., & Betrancourt, M. (2002). Animation: Can It Facilitate? InternationalJournal of Human Computer Studies, 57(4),247–262.

Upstill, S. (1985). The Realistic Presentation of Synthetic Images: Image Processing in Computer Graphics (Unpublished doctoral dissertation). University of California at Berkeley.

van Aken, J., & Novak, M. (1985). Curve-Drawing Algorithms for Raster Displays. ACM Transactions on Graphics, 4(2),147–169.

van Ham, F., & van Wijk, J. J. (2004). Interactive Visualization of Small World Graphs. InINFOVIS’04: Proceedings of the IEEE Symposium on Information Visualization (pp. 199–206). Washington, DC: IEEE Computer Society Press.

van Wijk, J. J., & van Selow, E. R. (1999). Cluster and Calendar-Based Visualization of Time Series Data. In INFOVIS ’99: Proceedings of the 1999 IEEESymposiumon Information Visualization(pp.4–9). Washington,DC: IEEE Computer Society Press.

van Overveld, K., & Wyvill, B. (2004). Shrinkwrap: An Efficient Adaptive Algorithm for Triangulating an Iso-Surface. The Visual Computer, 20(6), 362–369.

Veach, E., & Guibas, L. J. (1997). Metropolis Light Transport. In SIGGRAPH ’97: Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques(pp. 65–76). Reading, MA: Addison-Wesley.

Wald, I., Slusallek, P., Benthin, C., & Wagner, M. (2001). InteractiveDistributed Ray Tracing of Highly Complex Models. In Proceedings of the 12th EurographicsWorkshop on Rendering Techniques(pp. 277–288). London,UK: Springer-Verlag.

Walter, B., Hubbard, P. M., Shirley, P., & Greenberg, D. F. (1997). Global Illumination Using Local Linear Density Estimation. ACM Transactions on Graphics, 16(3),217–259.

Wandell, B. A. (1995). Foundations of Vision. Sunderland, MA: Sinauer Associates.

Wann, J. P., Rushton, S., & Mon-Williams, M. (1995). Natural Problems for Stereoscopic Depth Perception in Virtual Environments. Vision Research, 35(19),2731–2736.

Ward, G., & Simmons, M. (2004). Subband Encoding of High Dynamic Range Imagery. In First ACM Symposium on Applied Perception in Graphics and Visualization (APGV) (pp. 83–90). NY: ACM Press.

Ward, G. J. (1992). Measuring and Modeling Anisotropic Reflection. Proc. SIGGRAPH ’92, Computer Graphics, 26(2),265–272.

Ward,G.J. (1994). The RADIANCE Lighting Simulation and Rendering System. In A. Glassner (Ed.), SIGGRAPH’94: Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques (pp. 459–472). New York: ACM Press.

Ward, M. O. (2002,December). A Taxonomy of Glyph Placement Strategies for Multidimensional Data Visualization. Information Visualization Journal, 1(3–4),194–210.

Ward Larson, G., Rushmeier, H., & Piatko, C. (1997).AVisibilityMatchingTone Reproduction Operator for High Dynamic Range Scenes. IEEE Transactions on Visualization and Computer Graphics, 3(4),291–306.

Ward Larson, G., & Shakespeare, R. A. (1998). Rendering with Radiance. San Francisco, CA: Morgan Kaufmann Publishers.

Ware, C. (2000). InformationVisualization: PerceptionforDesign. Boston,MA: Morgan Kaufmann/Academic Press.

Ware,C. (2001). DesigningWitha21/2DAttitude. InformationDesignJournal, 10(3),255–262.

Ware, C., Purchase, H., Colpys, L., & McGill, M. (2002). Cognitive Measures of Graph Aesthetics. Information Visualization,1(2),103–110.

Warn, D. R. (1983). Lighting Controls for Synthetic Images. Proc. SIGGRAPH ’83, Computer Graphics, 17(3),13–21.

Watt, A. (1991). AdvancedAnimationandRenderingTechniques. Reading,MA: Addison-Wesley.

Watt, A. (1993). 3D Computer Graphics. Reading, MA: Addison-Wesley.

Wegman, E. J. (1990, Sep). Hyperdimensional Data Analysis Using Parallel Coordinates. Journ. American Statistical Association, 85(411),664–675.

Whitted, T. (1980). An ImprovedIllumination Model for Shaded Display. Communicationsof the ACM, 23(6),343–349.

Williams, A., Barrus, S., Morley, R. K., & Shirley, P. (2005). An Efficient and Robust Ray-Box Intersection Algorithm. journal of graphics tools, 10(1), 49–54.

Williams, L. (1991). Shading in Two Dimensions. In Proceedings of Graphics Interface (pp. 143–151). Wellesley, MA: A K Peters & Canadian Human Computer Communications Society.

Wyszecki, G., & Stiles, W. (2000). Color Science: Concepts and Methods, Quantitative Data and Formulae (Second ed.). New York: Wiley.

Wyvill, B., Galin, E., & Guy, A. (1999). Extending the CSG Tree: Warping, Blending and Boolean Operationsin an Implicit Surface Modeling System. Computer Graphics Forum, 18(2),149–158.

Wyvill, B., McPheeters, C., & Wyvill, G. (1986). Data Structures for Soft Objects. The Visual Computer, 2(4),227–234.

Yantis, S. (Ed.). (2000). Visual Perception: Essential Readings. London, UK: Taylor & Francis Group.

Yessios, C. I. (1979). Computer Drafting of Stones, Wood, Plant and Ground Materials. Proc. SIGGRAPH ’79, Computer Graphics, 13(2),190–198.

Yonas, A., Goldsmith, L. T., & Hallstrom, J. L. (1978). The Development of Sensitivity to Information from Cast Shadows in Pictures. Perception, 7, 333–342.

Dr. Mars Caroline Wibowo. S.T., M.Mm.Tech

DASAR DESAIN GRAFIS JILID 2

Bio Data Penulis

Penulis lahir di Semarang pada tanggal 1 Maret 1983. Penulis menempuh pendidikan Sarjana Teknik Elektro di Universitas Kristen Satya Wacana (UKSW), lulus tahun 2004, kemudian tahun 2005 melanjutkan studi pada Magister Desain pada Fakultas Seni Rupa dan Desain, Institut Teknologi Bandung (ITB), dan melanjutkan studi pada program studi Teknologi Multimedia pada Swinburne University of Technology Australia.

Penulis sejak tahun 2010, menjadi dosen pada program studi Desain Grafis Universitas Sains dan Teknologi Komputer (Universitas STEKOM), memiliki jabatan fungsional Lektor 300 dan sedang proses mengajukan kenaikan jabatan fungsional menjadi Lektor Kepala. Penulis juga seorang wirausaha di bidang toko online yang berhasil di kota Semarang dan juga aktif sebagai freelancer dalam bidang fotografi, web design dan multimedia.

YAYASAN PRIMA AGUS TEKNIK

PENERBIT :
YAYASAN PRIMA AGUS TEKNIK
Jl. Majapahit No. 605 Semarang
Telp. (024) 6723456. Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

ISBN 978-623-5734-47-7 (jil.2 PDF)

9 786235 734477

Dr. Mars Caroline Wibowo, ST, M.Mm.Tech.

DASAR DESAIN GRAFIS

JILID 2

YAYASAN PRIMA AGUS TEKNIK

PENERBIT :
YAYASAN PRIMA AGUS TEKNIK
Jl. Majapahit No. 605 Semarang
Telp. (024) 6723456. Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id