Coding Kreatif dalam Desain Grafis Visual

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

Coding Kreatif dalam Desain Grafis Visual

Dr. Mars Caroline Wibowo. S.T.. M.Mm.Tech

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 978-634-7227-10-2 (PDF)
9 786347 227102

Coding Kreatif dalam Desain Grafis Visual

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

ISBN : 978-634-7227-10-2

Editor :
Dr. Ir. Agus Wibowo, M.Kom, M.Si, M.M.

Penyunting :
Dr. Joseph Teguh Santoso, 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)

Anggota IKAPI No: 279 / ALB / JTE / 2023

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 ke hadirat Tuhan Yang Maha Esa atas limpahan rahmat dan karunia-Nya, sehingga buku ***"Coding Kreatif dalam Desain Grafis Visual"*** ini dapat terselesaikan dengan baik dan siap untuk dibagikan kepada para pembaca. Di era digital saat ini, batas antara seni dan teknologi semakin kabur. Desain grafis tidak lagi hanya soal estetika visual, tetapi juga melibatkan pemahaman teknologi dan pemrograman sebagai alat ekspresi kreatif. Buku ini hadir sebagai jembatan yang menghubungkan dunia desain dengan dunia kode, memberikan wawasan dan keterampilan bagi para desainer, seniman digital, dan pengembang kreatif untuk mengeksplorasi potensi tak terbatas dari coding dalam menciptakan karya visual yang inovatif dan dinamis.

Buku ini tersusun dari beberapa bab yang membahas berbagai aspek penting dalam penggabungan desain grafis dan coding. Bab 1 mengulas hubungan antara desainer dan kode serta bagaimana memulai proses kreatif menggunakan komputer sebagai medium ekspresi. Bab 2 dan 3 mengeksplorasi teknik menggambar dengan angka serta konsep pertumbuhan dan bentuk organik yang terinspirasi dari alam, memberikan dasar pemahaman tentang pola dan struktur visual yang kompleks. Selanjutnya, Bab 4 memfokuskan pada tipografi dinamis dan interaktif sebagai elemen krusial dalam desain visual yang responsif terhadap pengguna. Bab 5 mengajak pembaca untuk memahami cara melihat dan berinteraksi dengan ruang digital, memperluas perspektif visual dalam konteks teknologi modern. Terakhir, Bab 6 memperkenalkan penggunaan data eksternal yang besar dan langsung sebagai sumber inspirasi serta alat visualisasi, memungkinkan desain yang lebih kontekstual dan informatif. Keseluruhan bab ini dirancang untuk membekali pembaca dengan pemahaman dan keterampilan yang menyeluruh dalam menciptakan karya desain grafis yang inovatif dan berbasis kode.

Setiap bab dirancang untuk memberikan pemahaman mendalam sekaligus praktik langsung yang dapat memperkuat kemampuan coding sekaligus kreativitas desain. Dengan pendekatan ini, diharapkan pembaca tidak hanya mampu menguasai teknik, tetapi juga mampu mengembangkan gaya dan bahasa visualnya sendiri yang unik dan relevan dengan perkembangan teknologi saat ini.

Penulis berharap buku ini dapat menjadi sumber inspirasi dan referensi penting bagi mahasiswa, praktisi desain, programmer kreatif, maupun siapa saja yang ingin memperluas cakrawala kreativitasnya melalui perpaduan desain grafis dan pemrograman. Semoga buku ini dapat membuka peluang baru dalam berkarya dan berinovasi di dunia seni digital.

Akhir kata, kami mengucapkan terima kasih kepada semua pihak yang telah memberikan dukungan dan kontribusi dalam proses penulisan dan penerbitan buku ini. Semoga karya ini memberikan manfaat dan menjadi bagian dari perjalanan kreatif Anda.

Semarang, Mei 2025
Penulis

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

DAFTAR ISI

BAB 1
DESAINER DAN KODE

"Cobalah untuk mendapatkan hasil maksimal dari materi Anda, tetapi selalu dengan cara yang paling menghargainya."
William Morris

1.1 KOMPUTER SEBAGAI LINGKUNGAN KREATIF

Pengenalan proses komputer telah mengubah alat dan metode sehari-hari dalam proses desain grafis dari lingkungan fisik berupa tinta dan kertas ke lingkungan digital berupa piksel dan layar. Desainer kini dapat mengedit, menyalin, menyempurnakan, memanipulasi, mengubah, dan menggabungkan visual dengan cara yang tidak terbayangkan menggunakan proses tradisional.

Namun, computer tak hanya sekadar kanvas digital, kamar gelap, atau perangkat penyuntingan. Dengan melihat di luar batasan alat perangkat lunak, komputer dapat dilihat sebagai media kreatif tersendiri, dengan serangkaian properti, karakteristik, dan atribut uniknya sendiri yang membuka kemungkinan baru untuk praktik kreatif. Sama seperti bahan fisik, kertas, tinta, dan cat yang masing-masing memiliki kualitas tersendiri untuk dipertimbangkan dan dimanipulasi, demikian pula komputer memiliki kemampuan dan karakteristik uniknya sendiri untuk dieksplorasi secara kreatif.

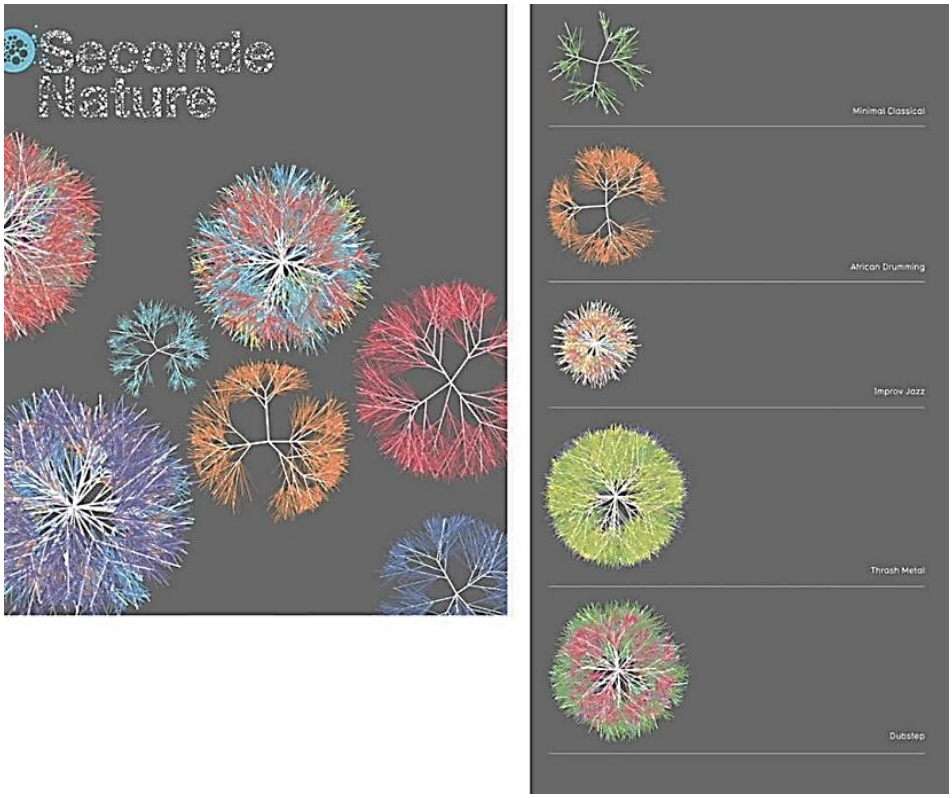
Melihat dan menggunakan potensi bawaan komputer sebagai perangkat yang dapat mengambil dan menggunakan banyak jenis masukan data yang berbeda (misalnya, dari tetikus, kamera video, atau mikrofon) untuk menghasilkan visual membuka cara berpikir baru, cara kerja baru, dan jenis baru gambar yang dibuat secara komputasional.

Grafik *"Reaktif"*

Komputer adalah perangkat pemrosesan dan penghitungan data yang fantastis, yang dirancang untuk memanipulasi dan memproses sejumlah besar informasi dengan sangat cepat. Kemampuan komputer untuk mengambil, menggunakan, dan menggabungkan media berarti bahwa ia dapat menghubungkan visual dan data dengan cepat dan dengan cara yang menarik. Masukan informasi melalui sumber data yang terhubung (file web, suara, video, atau teks) dapat digunakan untuk menghasilkan berbagai macam gambar, objek, atau lingkungan digital.

Kemampuan pemrosesan komputer yang sangat besar untuk menggunakan dan mengubah media (gambar, teks, video, atau suara) dalam format langsung atau interaktif kemudian membentuk sumber inovasi dan kemungkinan kreatif yang hebat. Sumber suara digital misalnya, dapat digunakan untuk menghasilkan gambar; sepotong teks dapat diterjemahkan secara visual menjadi gambar grafis; Data dari pergerakan orang yang lewat di depan jendela toko dapat digunakan untuk membuat gambar dan suara yang diproyeksikan. Visual yang dibuat dengan cara ini melampaui grafik tradisional dan menjadi jenis grafik reaktif, interaktif, atau generatif baru yang dibuat sebagai respons terhadap input data. Jenis grafik ini unik, responsif terhadap lingkungannya, dan hanya mungkin dilakukan dengan memanfaatkan kemampuan pemrosesan komputer.

Oleh karena itu, mengeksplorasi kemungkinan komputer sebagai perangkat untuk menghasilkan grafik reaktif membuka kemungkinan baru untuk memikirkan kembali objek digital dan melampaui batasan perangkat lunak penyuntingan gambar tradisional dan grafik digital.



Seconde Nature, Universal Everything

Generator identitas audio-reaktif untuk tempat pertunjukan musik di Aix-En-Provence, Prancis. Pohon reaktif komputer tumbuh sebagai respons terhadap pemutaran berbagai genre musik. File vektor diproduksi untuk digunakan oleh tim komunikasi tempat pertunjukan.

1.2 MEMANIPULASI MEDIA

Mengeksplorasi potensi kreatif komputer sebagai mesin pengolah data berarti membuat program khusus (perangkat lunak) untuk memanipulasi dan menyortir informasi digital (teks, gambar, dll.), menggunakan dan mengubahnya untuk menciptakan hasil visual baru. Di lingkungan fisik, tangan manusia memanipulasi bahan, seperti pena di atas kertas atau cat di atas kanvas. Di lingkungan komputasi, data secara langsung dimanipulasi menjadi objek digital dengan menulis baris kode pemrograman. Setiap objek yang ada di lingkungan digital dibuat dan dikendalikan oleh kode. Setiap alat digital yang Anda gunakan (pengolah kata, penyunting gambar, peramban web) dibuat menggunakan kode pemrograman.

Saat Anda menulis buku, mengirim email, atau mengedit foto, di balik menu drag-and-drop dan antarmuka pengguna yang mengilap, komputer sedang sibuk memproses instruksi pemrograman untuk melakukan tugas-tugas yang diperlukan. Kode dapat digunakan untuk berbagai tugas dan tujuan; dapat digunakan untuk membuat satu halaman web atau untuk membuat aplikasi perangkat lunak berskala besar. Penggunaan kode menyediakan cara yang alami dan langsung untuk membuat dan memanipulasi lingkungan digital. Bahasa pemrograman hadir dalam berbagai bentuk dan digunakan dalam banyak konteks yang berbeda. Setiap bahasa dibuat untuk tujuan tertentu, dibuat untuk bekerja dengan serangkaian media tertentu atau melakukan serangkaian tugas tertentu. Setiap bahasa pemrograman memiliki kosakata dan tata bahasanya sendiri, yang menentukan jangkauan dan cakupan bahasa dan menentukan jenis tugas yang dapat dilakukannya.

Bahasa-bahasa umum meliputi Java, JavaScript, dan C#. Sama seperti mempelajari

bahasa asing baru memungkinkan pembicara untuk terhubung dengan orang-orang baru dan membuka peluang untuk komunikasi, demikian pula mempelajari bahasa pemrograman memungkinkan programmer untuk berkomunikasi langsung dengan komputer dan membuka peluang baru untuk menciptakan objek dan lingkungan digital.

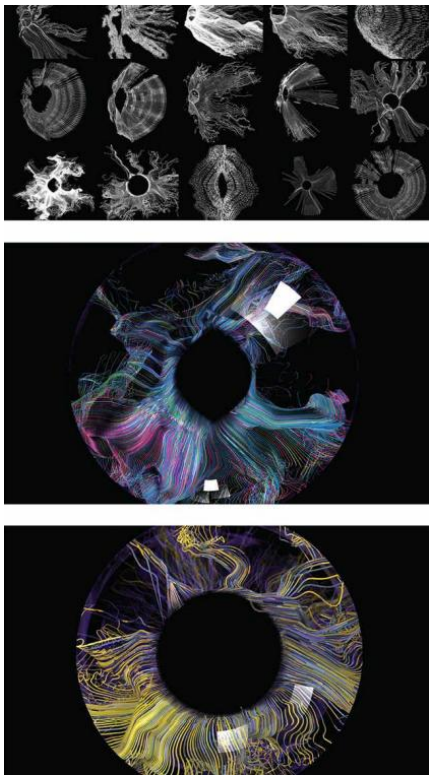
Seperti bahasa alami, bahasa pemrograman memiliki kosakata dan tata bahasa. Tidak seperti bahasa alami, kata-kata dalam bahasa pemrograman sering kali terbatas pada instruksi dan perintah, dan aturan penggunaannya ketat dan terdefinisi dengan baik. Instruksi harus ditulis dengan tepat; jika tidak, komputer tidak tahu apa yang harus dilakukan dan program tidak akan berfungsi. Hanya ada sedikit ruang untuk menjadi deskriptif dengan bahasa pemrograman tidak ada puisi, hanya perintah.

1.3 KODE DAN KREATIVITAS

Semua desainer, apa pun medianya, perlu memiliki apresiasi dan pemahaman terhadap materi mereka. Pengalaman tentang proses, karakteristik, keterbatasan, dan kemampuan media mereka memainkan peran penting dalam menciptakan solusi desain yang sukses dan praktis, yang memanfaatkan materi dengan sebaik-baiknya dan paling tepat.

Seniman dan desainer yang ingin mengeksplorasi kualitas, karakteristik, dan kemungkinan kreatif lingkungan komputer melakukannya dengan menggunakan kode; sama seperti desainer berbasis cetak yang memiliki pemahaman tentang bobot kertas, proses cetak, pemisahan warna, dll. menggunakan pengalaman ini untuk menginformasikan pekerjaan desain mereka, demikian pula, desainer yang bekerja untuk hasil berbasis layar yang memiliki pemahaman tentang kode dapat menggunakan pengetahuan ini untuk mengembangkan solusi desain yang kreatif dan berwawasan desain. Memahami cara kerja kode dapat membuka cara baru dalam berpikir tentang dan mendesain dengan dan untuk lingkungan digital; menyajikan peluang baru untuk pemikiran kreatif, dan memberikan peluang untuk mendesain visual dan pengalaman digital yang menakjubkan.

Contoh karya desain yang memanfaatkan proses berbasis data secara inovatif. Sistem yang dibuat untuk biro iklan didasarkan pada gambar digital generatif yang menyerupai iris mata manusia. Visualnya bersifat adaptif, fleksibel, dan didukung data, yang berubah tergantung pada konteks dan aplikasinya, sehingga menciptakan desain unik untuk setiap aplikasi. Mata dapat mencerminkan klien yang berkunjung dan dapat diaktifkan oleh data klien, besar atau kecil. Elemen fleksibel dari identitas merek visual menciptakan pendekatan visual kohesif yang fleksibel yang dapat diterapkan di berbagai jenis media.



Mata Besar, Ilustrasi Identitas, FIELD dan SomeOne

Kode adalah alat kreatif yang berguna dan canggih yang digunakan seniman dan desainer untuk mengeksplorasi kemungkinan kreatif dari lingkungan digital dan untuk menciptakan objek, desain, dan karya seni digital asli. Dengan menulis kode, desainer dapat membuat permainan interaktif, menghasilkan visual yang merespons masukan pengguna atau lingkungan fisik, menggambar grafik generatif, atau memanipulasi tipografi. Menulis kode membuka peluang baru untuk menciptakan objek dan lingkungan

digital yang unik dan individual serta perangkat lunak aplikasi dan sistem.

Pemrograman adalah proses kreatif yang tidak seperti yang lain; ini adalah cara yang fleksibel dan alami untuk terlibat dengan dan memanfaatkan kekuatan pemrosesan digital komputer untuk keluaran visual yang kreatif. Ini memberikan kemungkinan baru untuk membuat objek dan solusi yang tidak terpikirkan sebelumnya. Daripada membatasi diri pada alat menggambar dan mengedit digital, seniman dan desainer dapat menggunakan kode untuk membuat dan menciptakan alat menggambar dan aplikasi mereka sendiri.

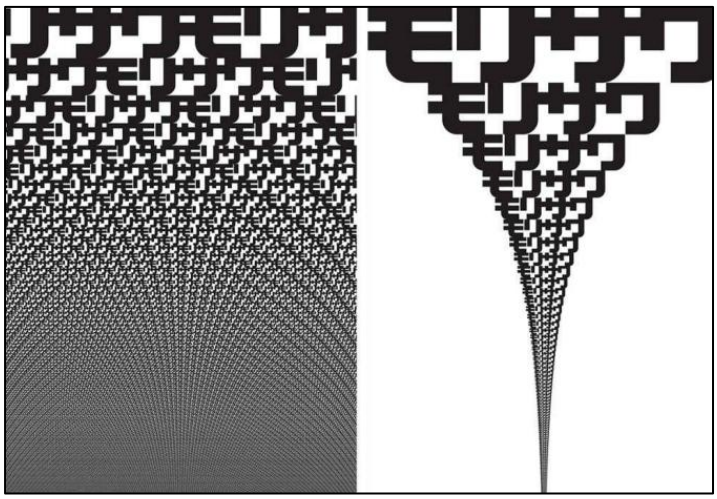
CONTOH

John Maeda

John Maeda adalah seorang desainer grafis dan ilmuwan komputer yang telah menjadi sosok yang sangat berpengaruh di bidang desain dan teknologi dengan minat khusus pada cara kedua bidang ini menyatu dan bersinggungan. Sebagai seorang desainer, karya awal Maeda menggabungkan kode komputer dengan estetika desain kombinasi yang mendefinisikan ulang penggunaan kode sebagai alat untuk ekspresi visual, dan membuka jalan bagi generasi desainer media interaktif di masa mendatang.

Design By Numbers (DBN) adalah bahasa pemrograman dan lingkungan yang diciptakan oleh John Maeda pada tahun 1999 sebagai bagian dari Kelompok Estetika dan Komputasinya untuk mengajarkan pengodean kepada para desainer dan seniman. Tujuan dari proyek ini adalah untuk memperkenalkan kode sebagai cara untuk berpikir tentang pembuatan desain untuk layar dan untuk menyediakan alat yang berguna untuk memperkenalkan kode kepada komunitas yang melek visual. Bahasa ini menyediakan antarmuka terpadu untuk menulis dan menjalankan program sederhana. Dengan menggunakan kisi layar minimal 100 x 100 piksel dan serangkaian perintah dan fungsi yang terbatas, lingkungan pemrograman ini menyaring struktur dasar, konsep, dan perintah komputasi, yang mendorong desainer untuk berpikir hati-hati tentang perancangan dalam batasan ini.

Buku DBN yang terkait sangat bagus, dan buku ini merupakan diskusi tentang konsep kode sekaligus buku petunjuk untuk bahasa tersebut. Meskipun bahasa pemrograman itu sendiri tidak lagi digunakan, proyek ini terbukti sangat berpengaruh; warisannya terus terasa. DBN memiliki pengaruh besar pada pengembangan bahasa pemrograman Processing yang diciptakan oleh dua murid Maeda, Ben Fry dan Casey Reas. Fry dan Reas mengembangkan Processing untuk melanjutkan ide asli DBN: menciptakan lingkungan pemrograman untuk para desainer.



Desain Berdasarkan Angka John Maeda

Karya desain grafis John Maeda menggabungkan estetika khas Jepang dengan proses komputasi yang menghasilkan karya desain yang elegan, baik untuk media cetak maupun layar.

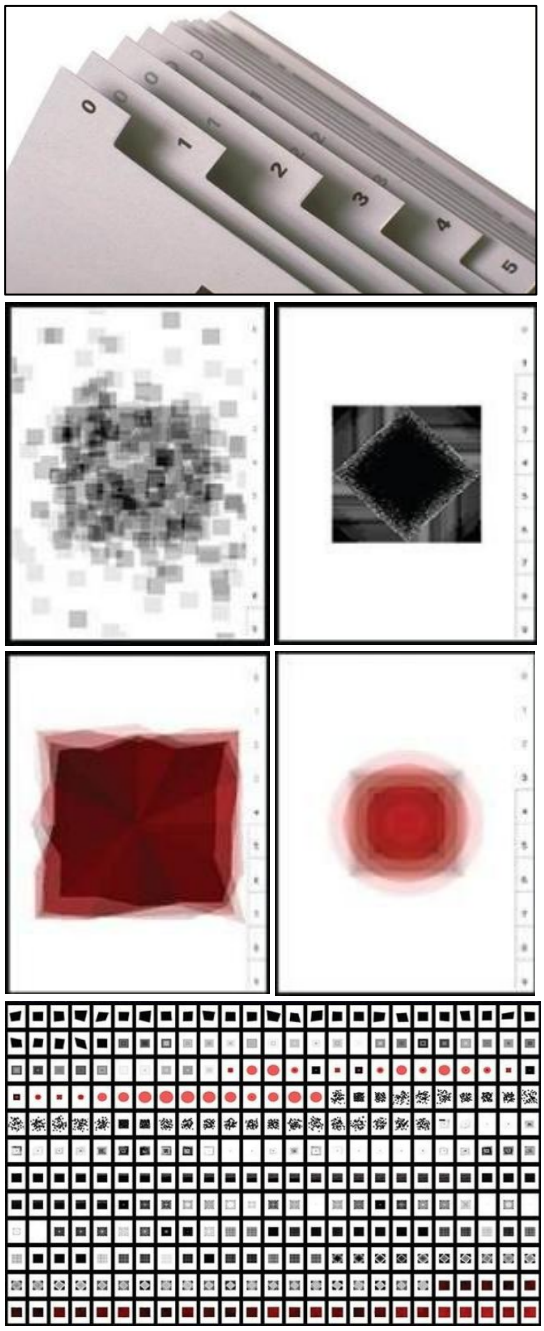
Design by Numbers (DBN)

Reactive Square (Reactive Books)

Ketertarikan awal John Maeda dalam penggunaan kode komputer sebagai bahan desain yang dapat digunakan sekreatif kuas atau pena telah menginspirasi dan memengaruhi generasi desainer digital.

Karya dan tulisannya membentuk gagasan bahwa kode dapat digunakan untuk menciptakan jenis desain visual baru. Ia meletakkan dasar bagi bidang desain yang kini sering disebut sebagai *"pengodean kreatif"*.

Eksperimen awalnya dalam aplikasi kreatif dan visual pemrograman komputer mungkin terlihat sederhana sekarang, tetapi eksperimen tersebut merupakan (dan masih) bagian unik dari desain digital yang menyiapkan landasan bagi banyak grafis interaktif yang kita lihat saat ini. Gagasannya tentang *"grafik reaktif"*, yang digunakan untuk menggambarkan bentuk dan jenis visual yang bergerak dan merespons jenis input komputer dan pengguna tertentu, masih bergema.



Seri Kotak Reaktif John Maeda

Seri *"Kotak Reaktif"* Maeda merupakan eksperimen awal yang penting dalam mengeksplorasi kemungkinan visual komputer sebagai lingkungan untuk menciptakan jenis desain baru. Dengan melihat melampaui batasan perangkat lunak, Maeda mengeksplorasi komputer sebagai media kreatif yang dapat menciptakan grafik yang responsif terhadap masukan pengguna, seperti suara.

Antara tahun 1994 dan 1999, Maeda menciptakan serangkaian buku digital untuk mengeksplorasi dan mengembangkan gagasan grafik reaktif. Setiap buku mengeksplorasi satu jenis sumber masukan untuk menghasilkan grafik: mikrofon, tetikus, waktu, dan papan ketik. Buku pertama adalah Kotak Reaktif: sepuluh kotak yang merespons audio dari mikrofon. Buku kedua adalah Flying Letters, yang menggunakan tetikus sebagai cara bermain dengan bentuk huruf digital.

Buku ketiga adalah *"dua belas jam lucu" yang* disebut *"jam 12"* yang menampilkan waktu dalam format digital tertentu. Buku terakhir *"Tap, Type, Write",* merupakan interaksi digital yang terinspirasi oleh mesin ketik; buku ini menggunakan papan ketik sebagai masukan. Sumber masukan tunggal yang digunakan untuk setiap buku eksperimen, bersama dengan kesederhanaan visual dan keceriaan setiap bagian, menciptakan serangkaian karya menarik yang memperkenalkan cara baru dalam memandang dan memikirkan desain digital bagi generasi baru seniman dan desainer. Buku-buku ini sekarang sulit dilihat karena memerlukan Mac dengan perangkat lunak pra-OS X, tetapi ide dan konsepnya tetap hidup seperti sebelumnya.

12 o'clocks John Maeda

Masing-masing buku reaktif merupakan eksperimen menyenangkan dalam estetika digital yang disajikan sebagai CD-ROM berbasis buku. Proyek tipografi 12 o'clocks tahun 1996, yang terlihat di sini, merupakan cikal bakal media interaktif yang sejak saat itu mengalami pertumbuhan pesat, membuka jalan bagi pengembangan desain digital yang lebih dinamis dan interaktif, serta menunjukkan potensi kreatif dari perpaduan teknologi dan seni. Karya ini tidak hanya memperluas batasan desain tradisional, tetapi juga memungkinkan pengguna untuk berinteraksi langsung dengan elemen visual, menciptakan pengalaman yang lebih imersif dan personal. Dengan demikian, 12 o'clocks menjadi contoh penting dalam sejarah desain digital, menginspirasi generasi desainer dan seniman untuk mengeksplorasi kemungkinan baru dalam media interaktif.

1.4 MEMULAI DESAIN DAN KODE

Salah satu daya tarik menjelajahi kode adalah aksesibilitasnya. Tidak seperti membeli perangkat lunak desain versi terbaru, memulai dengan bahasa pemrograman tidak memerlukan pengeluaran finansial yang besar. Selama Anda memiliki akses ke komputer yang terhubung ke Internet, Anda dapat mengunduh dan mengakses sumber daya secara gratis dan segera mulai bereksperimen dan menjelajahi proses dan ide bahasa tersebut.

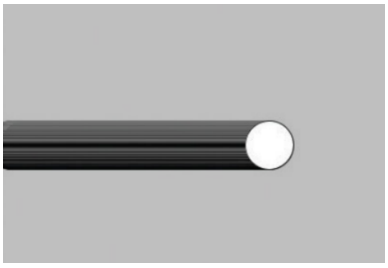
Sebagian besar bahasa pemrograman dibuat dan dikirimkan sebagai proyek sumber terbuka, yang berarti bahwa bahasa tersebut gratis untuk digunakan dan dikembangkan. Komunitas sumber terbuka yang besar dan terus berkembang yang terdiri dari desainer, programmer, seniman, pakar, dan pemula sama-sama berbagi ide, proyek, kode sumber, dan sumber daya untuk memperluas kemampuan setiap bahasa. Saat mulai menjelajahi kemungkinan kreatif dari kode, persyaratan utama untuk memulai perjalanan Anda adalah rasa ingin tahu yang besar (keinginan untuk memahami cara kerja berbagai hal dan melihat di balik "*wajah*" komputer), minat untuk bereksperimen dengan ide-ide baru, dan kemauan untuk membuat kesalahan.

Alat dan teknologi

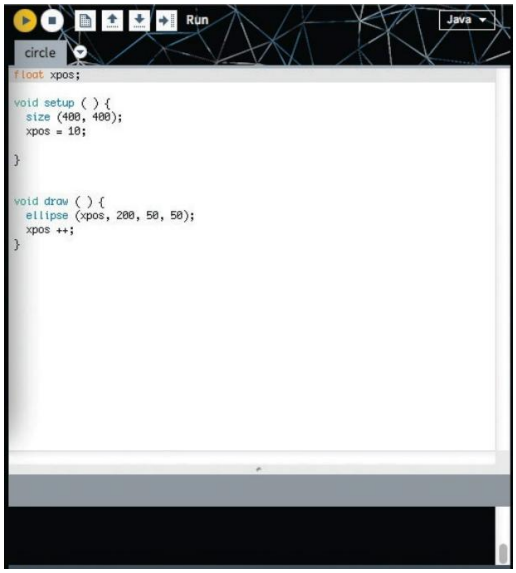
Ada banyak bahasa pemrograman dan lingkungan yang tersedia bagi desainer untuk digunakan. Setiap bahasa dan lingkungan memiliki kelebihan khusus yang membuatnya sesuai untuk jenis pekerjaan tertentu (misalnya, menangani video, data daring, dsb.). Berikut ini adalah ikhtisar singkat dari beberapa lingkungan pemrograman yang paling umum digunakan oleh pembuat kode, seniman, dan desainer kreatif. Dengan memahami karakteristik dan

kemampuan masing-masing alat, desainer dapat memilih teknologi yang paling tepat untuk mewujudkan visi kreatif mereka dan mencapai hasil yang optimal. Pemilihan alat yang tepat juga memungkinkan desainer untuk bekerja lebih efisien dan efektif dalam proyek-proyek yang kompleks dan beragam.

Processing:
www.processing.org



Processing

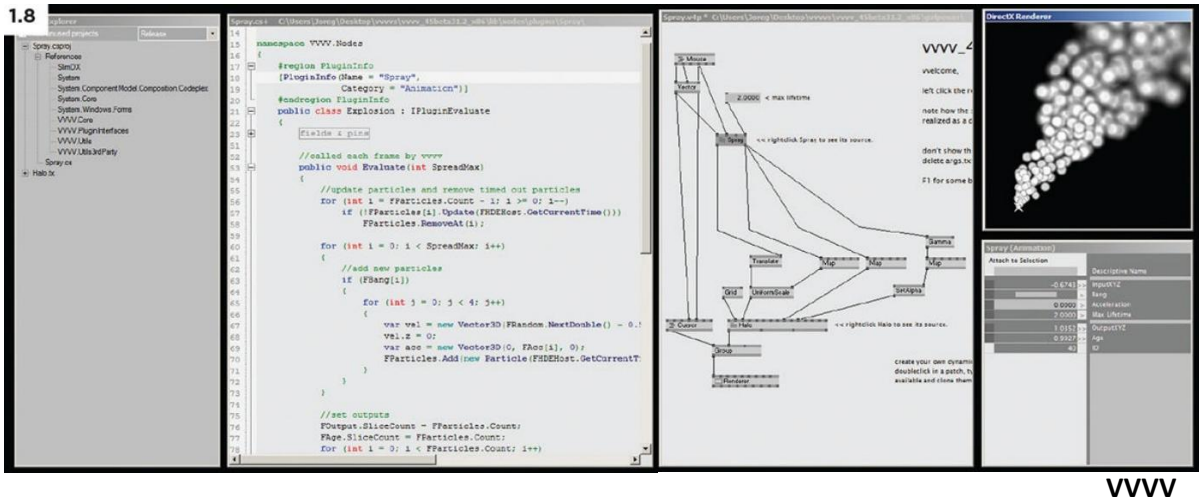


Processing adalah bahasa pemrograman yang dibuat untuk memberi seniman dan desainer cara sederhana untuk membuat gambar dan grafik interaktif serta mengajarkan dasar-dasar kode pemrograman. Awalnya dibuat sebagai alat untuk mengajarkan literasi perangkat lunak kepada komunitas seni visual dan desain, program ini telah berkembang menjadi bahasa yang digunakan oleh mahasiswa dan profesional.

Cuplikan layer diatas menunjukkan lingkungan pengembangan Processing. Kode ditulis langsung ke editor teks. Jendela tampilan menunjukkan hasil kode saat dijalankan.

Lingkungan Processing adalah versi *"tertutup"* dari bahasa Java (yang jauh lebih besar); bahasa ini memungkinkan pengguna untuk dengan cepat mulai membuat visual yang dihasilkan kode. Kode ditulis sebagai sketsa yang dapat diuji dengan cepat dan mudah. Selama bertahun-tahun, bahasa dan komunitas ini telah berkembang sedemikian rupa sehingga sekarang menjadi alat desain dan pembuatan prototipe lengkap yang digunakan untuk pekerjaan instalasi skala besar, grafik gerak, dan visualisasi data. Pemrosesan akan digunakan sebagai lingkungan pemrograman untuk contoh dan tutorial yang digunakan dalam buku ini.

VVVV: vvvv.org



VVVV

VVVV (juga disebut *"v4"* atau *"v-four"*) adalah
lingkungan pemrograman dengan penekanan khusus pada pekerjaan video waktu nyata, dan digunakan untuk membuat lingkungan media berskala besar dengan antarmuka fisik dan grafik gerak waktu nyata, audio, dan video. Meskipun merupakan lingkungan pemrograman, VVVV menggunakan editor alat pemrograman grafis yang memungkinkan interaksi dan lingkungan

dibuat secara visual, tanpa harus menulis baris kode pemrograman sebagai teks. Instruksi dan fungsi individual direpresentasikan sebagai kotak/simpul, yang dihubungkan oleh pengguna untuk meneruskan data di antara keduanya. Seluruh struktur yang terdiri dari simpul dan tautan disebut *"patch."* Meskipun VVVV menyediakan cara visual untuk merepresentasikan struktur program, antarmukanya tidak intuitif, dan program yang rumit dapat dengan cepat membuat jaringan simpul dan tautan yang rumit.

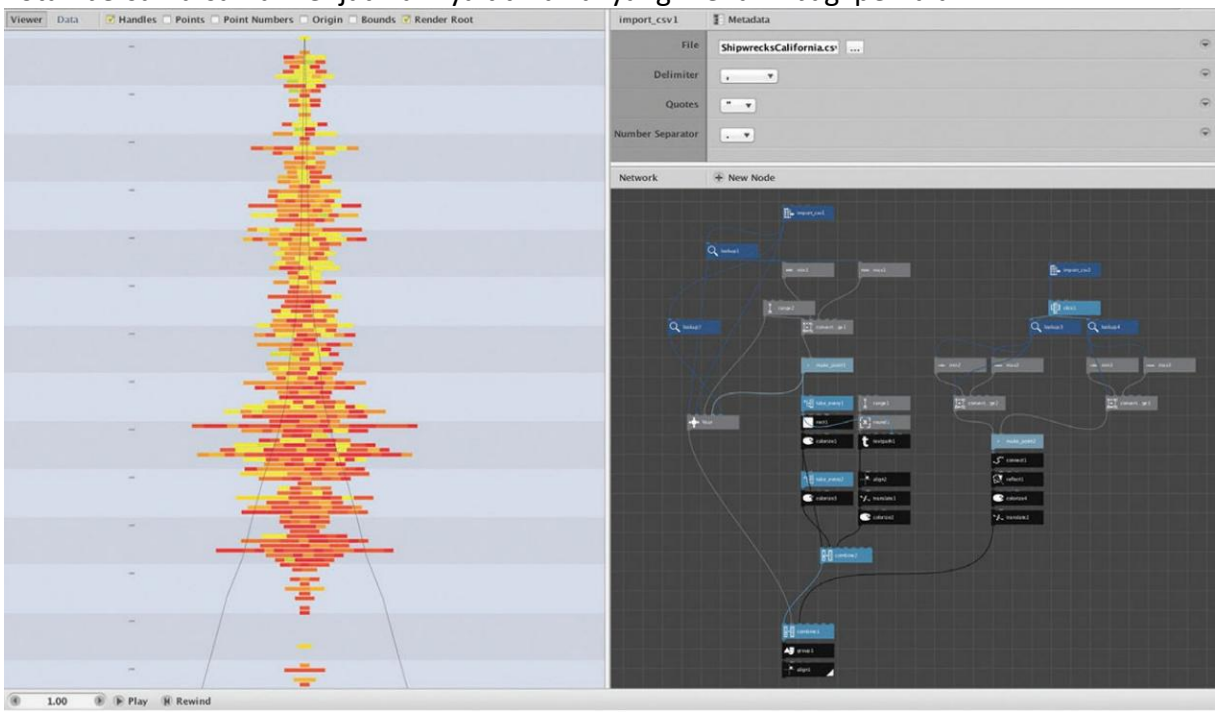
openFrameworks: www.openframeworks.cc

OpenFrameworks adalah pustaka perangkat lunak yang dirancang untuk pengodean kreatif. Pustaka ini dibuat untuk memungkinkan desainer dan seniman membuat karya yang menggabungkan elemen media (grafik, suara, video, dll.) dengan cara yang interaktif. Ide di balik openFrameworks sangat mirip dengan Processing.

Namun, tidak seperti Processing, openFrameworks bukanlah bahasa pemrograman; melainkan perangkat perekat yang menyatukan banyak pustaka dan aset pemrograman yang berbeda ke dalam satu kerangka kerja. Hal ini membuat openFrameworks lebih fleksibel dan canggih, terutama saat membuat grafik 3D dan manipulasi video secara real-time. Namun, struktur pustaka dan kerangka kerja terkadang membuatnya membingungkan atau sulit bagi programmer pemula untuk menavigasinya.

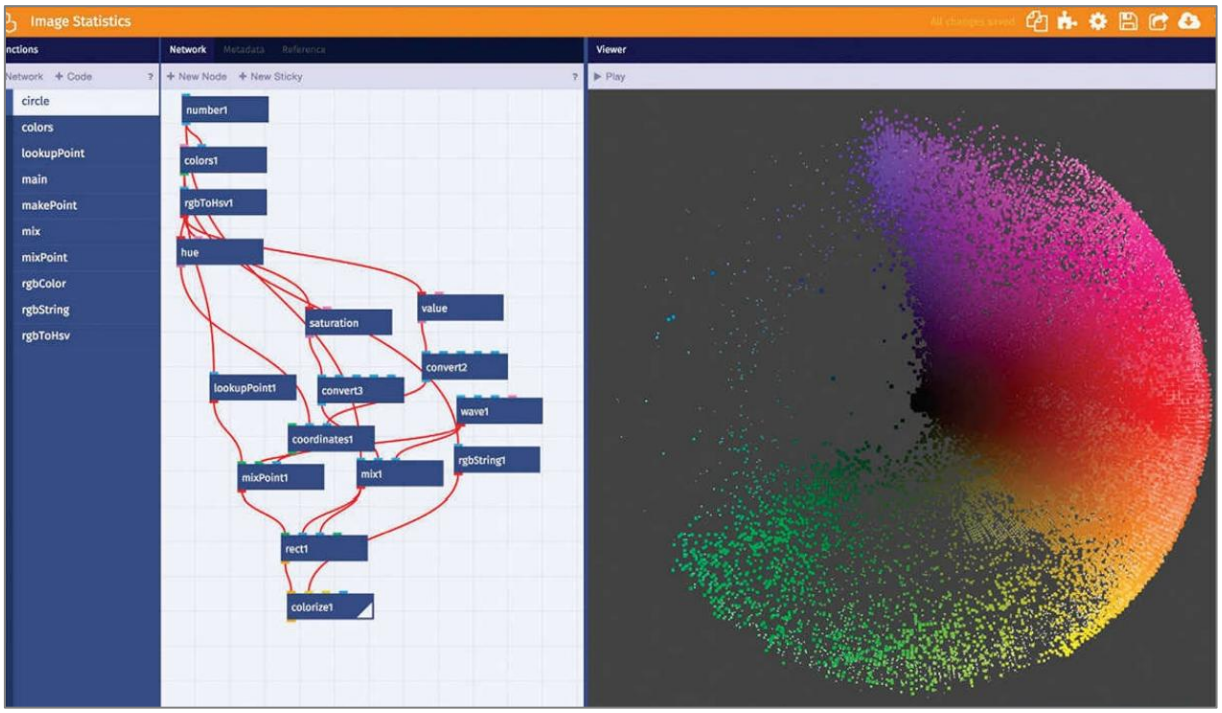
NodeBox3: www.nodebox.net/node/

NodeBox 3 adalah alat pemrograman visual yang dirancang untuk membuat visualisasi data dengan cepat dan mudah. Program dalam NodeBox 3 dibuat secara visual dengan menyeret kotak (node) ke layar dan menautkannya bersama-sama. Setiap node melakukan tugas (fungsi) tertentu, yang detailnya dapat diedit dengan memilih dari item menu di dalam setiap node. Data eksternal dapat dimasukkan ke dalam node untuk mengatur dan mengubah visual di layar. Kemampuan untuk membuat program secara visual dengan menautkan kotak-kotak bersama-sama menjadikannya titik awal yang menarik bagi pemula.



Meskipun node individual dapat dibuat dan diedit secara khusus menggunakan Python, lingkungan pemrogramannya tidak seluas dan ekspresif seperti beberapa contoh lainnya. Kekurangan dari antarmuka visual ini adalah fleksibilitas dan kontrolnya yang lebih sedikit. Metode ini bagus untuk dieksplorasi untuk bereksperimen dengan visualisasi data.

Versi baru, NodeBox Live, berjalan di browser dan akan mendukung pemrograman visual (melalui node) dan pemrograman tekstual (menggunakan JavaScript).



NodeBox Live F. De Bleser, 2014 Gambar divisualisasikan sebagai lingkaran

HTML, CSS, dan JavaScript

HTML (HyperText Markup Language) adalah bahasa markup yang digunakan untuk membuat struktur dan konten halaman web, sedangkan *CSS* (Cascading Style Sheet) digunakan untuk mengontrol tampilan dan layout halaman web. Sementara itu, JavaScript adalah bahasa pemrograman yang memungkinkan penambahan interaktivitas dan dinamika pada halaman web, sehingga memungkinkan pengembangan aplikasi web yang lebih kompleks dan menarik. Meskipun berbeda dalam fungsi dan tujuan, ketiga teknologi ini saling melengkapi dalam pengembangan web.

Selain lingkungan kode kreatif ini, ada bahasa pemrograman dan skrip arus utama lain yang digunakan di web. HTML dan CSS adalah bahasa *"mark up"* khusus yang digunakan untuk membuat konten dan tampilan halaman web. Dan JavaScript adalah bahasa pemrograman yang digunakan untuk membuat interaktivitas dan permainan di web. Meskipun buku ini tidak akan membahas JavaScript secara khusus, prinsip inti pemrograman yang digunakan dalam banyak contoh ini digunakan dalam banyak bahasa, termasuk JavaScript.

Kode tidak hanya digunakan sebagai bahasa yang berdiri sendiri; melainkan kode juga dapat digunakan dalam perangkat lunak untuk meningkatkan atau menambahkan fungsionalitas atau memperkenalkan elemen interaktivitas. Hal ini dapat dilakukan untuk menghemat waktu dan mengotomatiskan proses yang berulang atau untuk menghasilkan pekerjaan secara dinamis. Misalnya, perangkat lunak grafik gerak yang populer, Adobe After Effects, memiliki lingkungan pemrograman bawaannya sendiri yang disebut *"Expressions"* yang dapat digunakan untuk membuat animasi dinamis tanpa menambahkan banyak bingkai utama secara manual. Skrip Expressions didasarkan pada sistem bahasa dan struktur JavaScript.

Demikian pula, alat animasi Adobe Flash memiliki bahasa pemrograman bawaannya

sendiri yang disebut actionScript. ActionScript telah berevolusi dengan perangkat lunak tersebut; ia telah berkembang dari serangkaian fungsi sederhana menjadi bahasa yang mampu menambahkan berbagai tingkat interaksi dan berkembang dari penekanan tombol dan item menu sederhana menjadi animasi yang sepenuhnya dinamis yang sepenuhnya dihasilkan oleh kode. Kemampuan untuk menambahkan interaksi ke animasi menggunakan actionScript telah menjadikannya alat yang populer bagi para desainer, meskipun sekarang lebih umum digunakan untuk membuat game daring.

Pemrosesan

Contoh-contoh dalam buku ini menggunakan pemrosesan sebagai bahasa default. Pemrosesan adalah lingkungan pemrograman serba guna yang bagus dan tersedia secara gratis, mudah untuk dijalankan, dan didukung dengan baik dengan komunitas daring yang besar dan memiliki banyak contoh kode. Bahasa Pemrosesan menggunakan konsep pemrograman yang umum untuk bahasa dan lingkungan lain, sehingga menyediakan bahasa pemula yang baik untuk mempelajari konsep dasar pemrograman, yang dapat diterjemahkan ke dalam lingkungan lain. Meskipun contoh kode yang diberikan dalam buku ini secara khusus dibuat dengan Pemrosesan, sebagian besar konsepnya umum untuk semua struktur pemrograman, dan dengan demikian memberikan gambaran umum yang baik tentang dasar-dasar kode.

1.5 KOSAKATA DAN PETUNJUK

Bagian ini akan dimulai dengan memperkenalkan beberapa elemen utama kode yang umum untuk banyak bahasa pemrograman. Elemen-elemen tersebut sangat penting dan membentuk prinsip dasar yang akan menjadi dasar pembuatan contoh-contoh lainnya dalam buku ini.

Fungsi

Setiap bahasa pemrograman terdiri dari serangkaian instruksi pra-tertulisnya sendiri, yang memberi tahu program apa yang harus dilakukan. Ini disebut fungsi, dan mengarahkan kode untuk melakukan tugas-tugas tertentu. Setiap bahasa pemrograman memiliki fungsinya sendiri, yang biasanya tercantum dalam panduan referensi. Fungsi ditulis berdasarkan nama, diikuti oleh serangkaian tanda kurung() dan titik koma (;) untuk mengakhiri baris.

```
doSomething();
```

Dalam Processing, misalnya, fungsi untuk *"memperhalus"* tampilan grafik di layar adalah: // fungsi untuk menggambar grafik dengan tepi anti-alias halus();

Panduan referensi Processing online memberikan rincian setiap fungsi dan menunjukkan argumen mana (dan berapa banyak) yang dibutuhkan fungsi tersebut.

Tanda kurung digunakan untuk informasi tambahan (disebut *"argumen"*) untuk menentukan cara kerja fungsi. Misalnya, fungsi untuk mengatur lebar garis mencakup satu argumen, angka, untuk menunjukkan lebar dalam piksel:

```
// set the stroke to 10 pixels wide
strokeWeight(10);
// set the stroke to 5 pixels wide
StrokeWeight(5);
```

Argumen dapat ditampilkan sebagai angka atau sebagai tipe data lain, seperti teks.

Fungsi println(), yang digunakan untuk mencetak teks di jendela output, menentukan teks yang akan ditampilkan dalam tanda kurung.

```
// Outputs the world “hello” to the output window
Println(“hello”)
```

Fungsi sering kali memerlukan lebih dari satu argumen agar dapat berfungsi. Argumen tambahan dipisahkan menggunakan koma:

```
// sets the width and height of the screen size
size(200,300);
```

Argumen dari tipe data yang berbeda dapat digunakan bersama-sama. Misalnya, fungsi untuk menulis teks di layar perlu mengetahui teks apa itu dan juga lokasi numerik x, y dari kata-kata:

```
teks (“hello”, 10, 20);
```

Fungsi harus diketik dengan cara yang benar. Kesalahan pada huruf kecil, huruf besar, atau ejaan akan menyebabkan kode menampilkan pesan kesalahan.

Tata Bahasa dan Sintaksis

Sama seperti bahasa tulis yang menggunakan tanda baca (seperti koma dan titik) dan paragraf untuk menyusun tulisan dan dengan demikian membantu pembaca memahami isinya, bahasa pemrograman juga memiliki serangkaian tanda baca sendiri yang digunakan untuk menyusun kode, yang membantu komputer membaca instruksi dengan benar.

Berikut ini adalah daftar singkat elemen tata bahasa umum yang digunakan dalam pemrograman untuk menentukan struktur kode.

Titik koma (;): Ini adalah bagian kode yang kecil namun penting yang digunakan untuk mengakhiri baris, mirip dengan titik penuh pada titik. Setiap perintah atau instruksi diakhiri dengan titik koma sehingga komputer mengetahui di mana akhir perintah tersebut. Tanpa ini, beberapa baris kode bergabung menjadi satu dan menimbulkan kesalahan.

```
//incorrect and will result in an error
Size (200,200)
Size (200,300); //correct
```

Tanda kurung kurawal ({}): Ini menunjukkan blok kode, mirip dengan paragraf. Bagian besar dari sebuah program sering digambarkan sebagai satu blok kode. Tanda kurung kurawal menunjukkan di mana blok kode dimulai dan berakhir. Tanda kurung harus dipasangkan; setiap tanda kurung buka harus memiliki tanda kurung tutup yang sesuai.

```
{
 a line of code;
Another line of code;
}
```

Blok kode dapat *"disusun"* di dalam satu sama lain. Elemen tertentu, seperti *"pernyataan jika,"* disusun menggunakan tanda kurung kurawal.

Komentar (//): Ini adalah catatan dan anotasi yang ditulis dalam kode yang diabaikan oleh komputer. Semua bahasa pemrograman menyertakan kemampuan untuk menambahkan komentar, yang dapat membuat kode lebih mudah dibaca oleh Anda dan orang lain. Dalam Pemrosesan, komentar ditambahkan ke dalam kode menggunakan dua simbol garis miring (//).

```
// this is a comment
smooth( );
// and this is a comment also
```

Banyak contoh kode dalam buku ini akan menyertakan komentar di samping kode untuk membantu menjelaskannya.

Variabel

Variabel merupakan inti dari banyak konsep pemrograman dan menyediakan landasan untuk menciptakan grafik yang dinamis dan responsif secara digital. Variabel hanyalah wadah bernama, yang mewakili sepotong informasi yang tersimpan. Nama wadah tetap sama, tetapi informasi yang disimpannya dapat diubah atau diambil saat program berjalan.

Variabel dapat digunakan untuk melacak informasi seperti skor permainan pemain, lokasi objek di layar, atau nama pengguna. Variabel dibuat, biasanya di awal program, dengan menentukan nama (tanpa spasi) dan *"tipe"* data (sesuatu yang menunjukkan tipe data yang disimpan). Variabel sering diberi nilai awal. Variabel biasanya diberi nama dengan cara yang mencerminkan cara penggunaannya.

```
// creates a number variable called score
int score = 0;
// creates a number variable called x_pos
float x_pos = 120.5;
// creates a variable to store a user name
String userName = “Bill”;
```

Setelah variabel dibuat, variabel tersebut ada di dalam program dan dapat digunakan di tempat lain dalam kode sebagai pengganti nilai data tetap.

```
String name = “Andrew”; // creates a variable
// prints out the variable name (“Andrew”)
println (name);
float thickness = 10;
strokeWeight (thickness);
```

Nilai dari konten suatu variabel dapat diperbarui dan diubah saat program berjalan.

```
String name = “Bill”;
println (name); // prints “Bill”
name = “Bob”;
println (name); // prints “Bob”
```

Variabel angka dapat diubah secara matematis atau melalui input lain, seperti interaksi

pengguna (misalnya, gerakan mouse).

Perhitungan matematika sederhana sering digunakan untuk *"menambah"* (increment) atau *"mengurangi"* (decrement) variabel numerik.

```
float xpos = 100;
float xpos = 100;
xpos = xpos + 10; // increase by 10 (to 110)
xpos = xpos – 20; // decrease by 20 (to 90)
// scale (multiply) by 10 (to 900)
xpos = xpos * 10;
xpos = xpos / 2; // divide by 2 (to 450)
```

Dalam bahasa pemrograman, simbol matematika dasar adalah:

Penambahan: +
Pengurangan: –
Perkalian: *
Pembagian: /

Berikut ini adalah cara singkat untuk menulis perhitungan yang sama:

```
xpos += 10;
xpos –= 20;
xpos *= 10;
xpos /= 2;
```

Mengubah variabel angka sering digunakan dalam fungsi menggambar, yang mengatur lokasi bentuk yang digambar di layar. Oleh karena itu, memperbarui variabel akan mengubah lokasi dan memindahkan bentuk.

Tipe Data Variabel

Variabel dapat digunakan untuk menyimpan banyak jenis data yang berbeda, (angka, teks, warna). Saat variabel dibuat, variabel tersebut diberi *"tipe data"* untuk menentukan tipe informasi yang akan disimpannya. Tipe data ditulis sebelum nama variabel saat dibuat:

```
int x = 10; // a variable with an “int” data type
```

Tipe data dasar (yang akan digunakan dalam teks ini) adalah sebagai berikut:

Integer (int): Tipe data int digunakan untuk menyimpan bilangan bulat (integer), yang umumnya digunakan untuk nilai yang akan selalu berupa bilangan bulat (misalnya, skor pemain dalam permainan).

```
int score = 50;
```

Float: Nilai float juga merupakan angka, tetapi dapat menyimpan nilai desimal, atau *"angka mengambang".* Nilai ini biasanya digunakan untuk nilai yang mungkin perlu didefinisikan atau diubah secara tepat misalnya, saat menghitung perubahan sudut atau gerakan objek di layar.

```
float angle = 45,5;
```

String: String adalah bagian data yang menyimpan urutan *("sebuah senar")* karakter, sering kali huruf, di dalam tanda kutip. String biasanya digunakan untuk menyimpan teks literal, seperti nama pengguna.

```
String name = "Andrew";
```

Boolean: Tipe data Boolean adalah tipe data yang dapat memiliki nilai benar atau salah. Tipe data ini berguna untuk membuat objek yang dapat digunakan sebagai tipe "saklar" dalam kode yang dapat dihidupkan atau dimatikan. Nilai ini sering digunakan dalam pernyataan kondisional (lihat di bawah) untuk menentukan alur program.

```
Boolean visible = true;
```

Alur Program

Mengendalikan “alur” program merupakan inti dari setiap proses interaktif atau generatif. Hal ini memungkinkan hasil program berubah sesuai dengan berbagai kondisi dan mencegah program menghasilkan hasil yang sama setiap kali dijalankan.

Pernyataan If

Pernyataan “if” umum digunakan dalam banyak bahasa pemrograman. Juga dikenal sebagai *“pernyataan bersyarat,”* konsep dasarnya sederhana; blok kode dibuat yang akan dijalankan hanya jika suatu kondisi benar. Struktur pernyataan “if” adalah sebagai berikut: if (kondisi)

```
{
do this if condition is true;
}
```

Kondisi yang akan diuji ditulis di dalam tanda kurung. Jika benar, maka instruksi di dalam kurung kurawal akan dieksekusi; jika tidak, maka instruksi tersebut diabaikan begitu saja. Pernyataan berikut memeriksa apakah nilai Boolean dari suatu variabel *("visible")* benar dan, jika benar, memberikan instruksi untuk menampilkan beberapa grafik.

```
if (visible == true) {
if (visible == true) {
// show graphics;
}
}
```

Pernyataan *“if”* juga dapat digunakan untuk memeriksa nilai suatu angka dan membandingkannya dengan angka lain. Simbol matematika standar digunakan untuk menilai apakah suatu angka lebih besar dari (>), lebih kecil dari (<), atau sama dengan (==) angka lain. Contoh berikut digunakan untuk memeriksa apakah skor pemain di bawah 20 atau di atas 50 dan menampilkan pesan yang sesuai.

```
if (score < 20) {
println (“bad luck”);
}
```

```
if (score > 50) {
println (“well done”);
}
```

Kondisi matematika yang digunakan dalam pernyataan “jika”:

Lebih Besar Dari: >

Kurang Dari: <

Sama Dengan: ==

Lebih Besar Dari atau Sama Dengan: >=

Kurang Dari atau Sama Dengan: <=

Pernyataan *“if”* juga dapat menyertakan bagian tambahan untuk menentukan apa yang terjadi jika kondisi awal tidak terpenuhi. Dengan kata lain, *“jika ini benar”* maka lakukan sesuatu; jika tidak, *(“jika tidak benar”)* lakukan sesuatu yang lain. Menambahkan pernyataan *“else”* akan menghasilkan dua hasil yang berbeda. Dalam pernyataan berikut, salah satu dari dua pernyataan dicetak, berdasarkan apakah nilai *“age”* lebih besar dari 60 atau tidak:

```
if (age > 60)
}
println (“you are over 60”);
} else {
println (“you are 60 or under”);
}
```

Perulangan

Elemen perulangan dalam program menyediakan cara yang sangat cepat dan berguna untuk menghitung, dengan mengerjakan daftar, dan untuk membuat urutan terstruktur dan teratur. Jenis perulangan umum yang digunakan dalam pemrograman adalah perulangan *"for".* Perulangan *"for"* menggunakan *"penghitung"* variabel dan terdiri dari tiga elemen: nilai awal variabel; kondisi pengujian, yang diperiksa untuk melihat apakah akan tetap berada dalam atau keluar dari perulangan; dan pernyataan *"pembaruan",* yang mengubah variabel di akhir setiap perulangan (biasanya menambahnya satu). Perulangan *“for”* standar berikut ini membuat sebuah variabel (i), yang dimulai dari nol (int i=0), berulang ketika i kurang dari 10 (i<10), dan menambah i sebanyak satu di akhir setiap perulangan (i++).

```
for (int i=0; i<10; i++) {
println (i); // outputs “0 1 2 3 4 5 6 7 8 9”
}
```

Setiap iterasi loop memanggil instruksi println, yang menghasilkan nilai variabel. Fungsi println() diulang 10 kali, menghasilkan nilai *“i”* setiap kali, menghasilkan urutan angka 0 hingga 9.

Fungsi yang Ditentukan Pengguna

Programmer dapat menggunakan fungsi bawaan yang telah ditulis sebelumnya, seperti `smooth()` dan `println()`, yang tersedia dalam bahasa pemrograman untuk membuat fungsi tambahan, yang merupakan blok kode yang dapat digunakan kembali. Fungsi adalah pembungkus bernama untuk sekelompok instruksi. Setelah dibuat, memanggil nama fungsi

secara otomatis menjalankan instruksi. Membungkus instruksi ke dalam fungsi membuat kode lebih dapat digunakan dan diulang.

Membuat fungsi dasar itu mudah. Buat fungsi dengan memberinya nama; *"tipe pengembalian"* (default adalah *"void"*); serangkaian tanda kurung, tempat variabel dapat ditambahkan; dan satu set kurung kurawal, di antaranya instruksi ditulis. Misalnya: \

```
void myFunction()
{
// do something
}
```

Dalam contoh ini, sebuah fungsi, yang disebut calcValue(), menghitung dan mencetak nilai kuadrat dari sebuah angka.

```
void calcValue() {
float number = 3;
println (number * number); // outputs 9
}
```

Setelah dibuat, suatu fungsi dapat digunakan seperti menggunakan instruksi kode yang telah ditulis sebelumnya: dengan memanggil namanya diikuti tanda kurung. Setiap kali calcValue() dipanggil, fungsi tersebut melakukan kalkulasi di dalam fungsi dan mencetak nilai penjumlahannya.

```
calcValue(); // calls the function
```

Sama seperti fungsi yang telah dibuat sebelumnya yang sering menggunakan nilai (argumen) agar dapat berfungsi, fungsi yang ditentukan pengguna juga dapat menyertakan variabel agar lebih bermanfaat. Versi yang lebih praktis dari fungsi contoh sebelumnya adalah menghitung dan mencetak nilai kuadrat dari angka yang diberikan. Untuk mencapai hal ini, variabel (num) ditambahkan di antara tanda kurung dan digunakan di dalam fungsi untuk memengaruhi kalkulasi.

```
void calcValue (float num) {
float number = num;
println (number * number);
}
```

Nilai *"num"* ditetapkan saat fungsi dipanggil.

Satu argumen, angka, diletakkan di antara tanda kurung untuk menetapkan nilai *"num"* dan menghasilkan hasil baru setiap kali:

```
calcValue (4); // num=4, prints "16"
calcValue (12); // num=12, prints "144"
```

Argumen tambahan dapat ditambahkan ke fungsi, yang harus didefinisikan saat dipanggil:

```
// a function now takes 2 number values (num1, num2)
```

```
void calcValue (float num1, float num2) {
float number1 = num1;
float number2 = num2;
println (number1 * number2);
}
}
//when the function is called num1 and num2 are multiplied together
calcValue (10, 2); // prints 20
calcValue (12, 5); // prints 60
```

Jika suatu fungsi dipanggil dengan argumen yang salah tempat atau hilang, maka fungsi tersebut akan mengembalikan kesalahan:

```
// wrong number of arguments (missing one
calcValue (10);
//right number of arguments, wrong type
calcValue (“three,” 2);
```

Fungsi merupakan bagian kode yang sangat penting; fungsi menciptakan cara yang berguna untuk menciptakan elemen yang dapat digunakan kembali dalam suatu program.

Perulangan dalam Struktur: setup() dan draw()

Lingkungan pemrograman umumnya memiliki dua blok struktural utama yang menentukan bagaimana dan kapan instruksi diproses. Yang pertama menciptakan kondisi awal untuk program, tempat yang *"menginisialisasi"* program sehingga semuanya diatur dengan benar saat program dimulai. Yang kedua menentukan apa yang terjadi saat program berjalan: bagian perulangan dari program tempat instruksi dieksekusi dan diproses berulang kali. Blok kode ini, fungsi inisialisasi dan fungsi perulangan, mungkin memiliki nama yang berbeda dalam bahasa yang berbeda; dalam Pemrosesan, keduanya disebut sebagai *"setup()"* dan *"draw()."*

Fungsi setup() dipanggil sekali saat program dimulai. Fungsi ini digunakan untuk menentukan properti awal, seperti ukuran layar. Kode di dalam setup() dipanggil hanya sekali.

```
void setup () {
//code in here is run only once
}
```

Fungsi draw() terus-menerus melakukan pengulangan saat program sedang berjalan. Semua instruksi atau baris kode di dalam fungsi draw terus-menerus diulang.

```
void draw() {
//code in here is repeated
}
```

Fungsi setup() dan draw() akan digunakan sebagai struktur dasar untuk contoh-contoh yang digunakan di seluruh buku ini.

Daftar dan Array

Array adalah daftar data, seperti daftar belanja, yang umum digunakan oleh bahasa pemrograman dan berguna untuk menyimpan banyak nilai variabel, seperti nilai angka (int,

float) atau data teks (String). Array dibuat dengan mendefinisikan jenis data yang akan disimpan dan jumlah elemen yang dapat ditambahkan ke daftar. Contoh berikut membuat daftar yang disebut shoppingList, yang memiliki 4 elemen, atau spasi, yang dapat menampung data teks (String):

```
String [] shoppingList = new String [4];
```

Setelah dibuat, daftar array diisi dengan meletakkan nilai data ke setiap elemen (setiap spasi) dalam array. Tanda kurung siku digunakan untuk mengakses item dalam daftar. Elemen dalam array diberi nomor mulai dari 0 (bukan 1). Data ditetapkan ke item pertama (elemen nol) dalam shoppingList sebagai berikut:

```
// the first item in the list is “apples”
shoppingList[0] = "appels";
```

Sisa daftar dapat diisi dengan cara yang sama, menggunakan angka untuk merujuk ke posisi data dalam daftar:

```
// the 2nd item in the shopping list
shoppingList[1] = “milk”;
// the 3rd item in the shopping list
shoppingList[2] = “eggs”;
// the 4th item in the shopping list
shoppingList[3] = “bread”;
```

Setelah diisi, item dalam array dapat ditemukan dengan merujuk ke nomor elemennya (yaitu, di mana mereka berada dalam daftar). Perlu diingat bahwa item pertama berada pada elemen nol, jadi item terakhir dalam daftar yang berisi 4 item akan berada pada posisi elemen 3.

```
println (shoppingList[0]); //prints out “apples”
println (shoppingList[1]); //prints out “milk”
println (shoppingList[2]); //prints out “eggs”
println (shoppingList[3]); //prints out “bread”
```

Setiap array memiliki variabel yang disebut “panjang”; ini adalah angka yang mengembalikan jumlah total elemen dalam array.

```
println (shoppingList.length); //prints out “4”
```

Perulangan *"For"* menyediakan cara cepat untuk menelusuri daftar guna mengakses atau mengubah data. Perulangan *"for"* berfungsi sebagai penghitung yang dapat mengakses setiap elemen array dengan cepat, satu per satu. Contoh berikut ini menelusuri dan mencetak setiap item dalam array shoppingList. Saat nilai *"i"* berubah dalam perulangan *"for"*, maka setiap elemen dalam array ditemukan secara bergantian.

```
for (int i=0; i<shoppingList.length; i++) {
// prints one element in the array, each time
println (shoppingList[i]); }
```

BAB 2
MENGGAMBAR DENGAN ANGKA

"Segala sesuatu adalah Angka"
Pythagoras

2.1 PETUNJUK MENGGAMBAR

Menggambar menggunakan kode memberi desainer serangkaian peluang dan alat unik untuk membuat grafik yang dinamis secara digital; ini adalah proses yang memerlukan pendekatan dan cara berpikir yang berbeda tentang bagaimana gambar dibuat, dan ini membuka jalan baru untuk ekspresi visual. Tidak seperti proses fisik membuat tanda dengan pena, kuas, atau bahkan tablet digital, gambar komputasi dibuat dengan daftar instruksi pemrograman tertulis berdasarkan aturan dan prosedur yang ditulis oleh programmer; komputer menerjemahkan dan menjalankan instruksi menjadi tanda, garis, dan bentuk individual di layar. Sebagian besar bahasa pemrograman menyertakan serangkaian instruksi khusus yang menghasilkan bentuk geometris sederhana (lingkaran, kotak, dan garis) dan yang memetakan dan memposisikan bentuk tersebut di layar dengan cara yang mirip dengan cara kita memetakan dan menggambar bentuk pada selembar kertas grafik.

Dalam lingkungan pemrograman, bentuk geometris sederhana ini adalah unit visual paling dasar yang darinya berbagai macam grafik yang rumit secara visual dapat dihasilkan. Meskipun sederhana, ketika digabungkan dan diulang ratusan atau bahkan ribuan kali dalam suatu program, bentuk-bentuk ini dapat menciptakan bentuk-bentuk baru yang lebih kompleks secara visual. Mempelajari cara menggunakan kode untuk menciptakan bentuk-bentuk paling dasar merupakan langkah awal yang penting untuk mengembangkan grafik berbasis data yang kompleks. Meskipun gambar tradisional dimulai dengan satu tanda di atas kertas, gambar berbasis kode (dinamis) dimulai dengan menulis instruksi untuk memplot garis atau titik di layar.

Bagian penting dari kekuatan unik menggambar kode terletak pada kemampuannya untuk memanfaatkan kapasitas pemrosesan komputer yang sangat besar. Cara komputer dapat menjalankan serangkaian instruksi menggambar berulang-ulang, ratusan kali per detik, berarti bahwa grafik dan pola yang rumit dapat dihasilkan bahkan dari beberapa baris kode saja. Ketika digabungkan dan diulang, instruksi individual bertindak bersama sebagai semacam "*mesin gambar*" yang dapat menghasilkan serangkaian visual yang padat dan kompleks.

Selain itu, proses ini biasanya mencakup elemen yang memungkinkan program untuk menghasilkan berbagai bentuk dan garis berdasarkan informasi data internal atau eksternal yang dimasukkan ke dalamnya. Programmer mendefinisikan serangkaian parameter gambar generik; komputer kemudian menggunakan dan menafsirkannya untuk menghasilkan visual. Oleh karena itu, serangkaian instruksi pemrograman yang sama dapat membuat berbagai bentuk, grafik, dan bentuk yang unik dan padat secara visual yang berkembang dan berubah di layar. Menggunakan komputer dengan cara ini membangun kemitraan kreatif antara programmer dan komputer yang menghasilkan gambar dan grafik yang tidak mungkin dilakukan dengan menggunakan proses tradisional apa pun.

Angka dan Urutan Angka

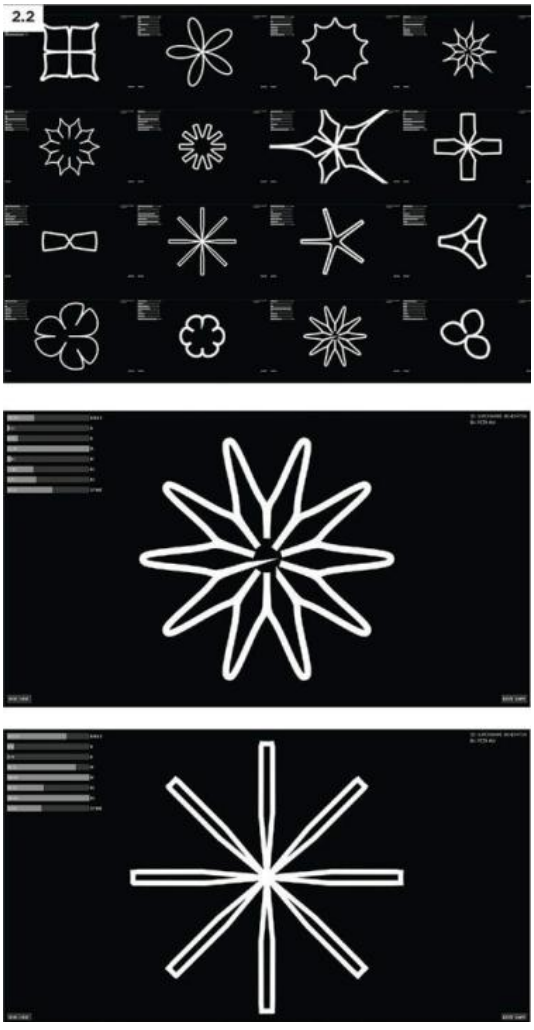
Angka adalah elemen kunci dalam grafis digital berbasis layar; mereka sangat penting dalam cara gambar digital dipahami dan diproses. Di balik *tampilan antarmuka pengguna*

grafis (GUI) dari program lukisan atau gambar digital, terdapat lingkungan numerik tersembunyi yang menentukan detail ukuran, bentuk, posisi, warna, dan fitur lainnya dari setiap garis dan bentuk digital yang digambar di layar. Kita melihat hal ini dalam praktek setiap kali perangkat lunak memberikan pengguna kemampuan untuk mengubah atau menentukan detail bentuk dengan memasukkan nilai angka yang menentukan atribut visualnya, seperti lebar, tinggi, rotasi, lokasi, atau warna. Adobe Illustrator, misalnya, memungkinkan pengguna untuk melihat dan mengatur ukuran atau posisi bentuk yang dipilih dengan mengubah nilai angka di jendela Transformasi.

Meskipun sebagian besar tersembunyi di balik antarmuka GUI perangkat lunak, pentingnya dan banyaknya angka yang diperlukan untuk membuat grafis digital menjadi tidak terhindarkan ketika melakukan pemrograman. Nilai angka individual sangat penting untuk sebagian besar aspek lingkungan pemrograman; mereka adalah unit dasar, titik awal, untuk semua gambar komputasi, dan mereka digunakan secara luas dalam semua kode. Angka sangat penting untuk menentukan, menggambarkan, membuat, memindahkan, mengubah, dan menggabungkan bentuk, garis, animasi, gambar, dan kata-kata di layar. Setiap fungsi pemrograman digunakan untuk membuat garis atau bentuk sederhana yang membutuhkan satu set angka untuk menentukan posisi, ukuran, bentuk, dan bahkan warnanya.

2D SuperShapes adalah aplikasi yang dibuat oleh Reza Ali yang memungkinkan pengguna bermain dengan parameter numerik untuk menghasilkan berbagai bentuk organik. Manipulasi angka menciptakan berbagai bentuk geometris yang menarik dan tak terduga.

Menggunakan kode untuk menggambar mengungkap apa yang terjadi di balik layar perangkat lunak grafis umum seperti Photoshop dan Illustrator. Dengan menggunakan kode pemrograman, seniman dan desainer memiliki kesempatan untuk menggali di bawah permukaan perangkat lunak dan langsung bermain dengan data numerik yang biasanya tersembunyi untuk membuat, mengulang, dan menggabungkan bentuk dari grafik dan hal ini menghasilkan berbagai macam gambar dinamis yang generatif dan berbasis data.



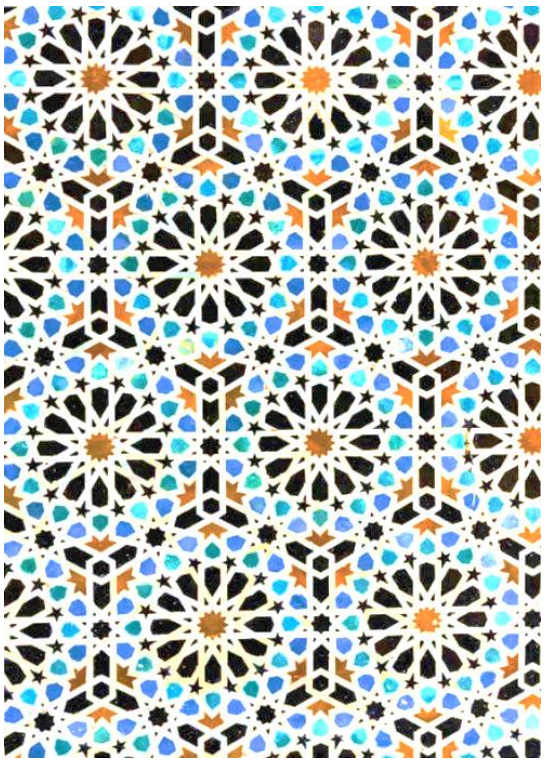
2D SuperShapes. Reza Ali

Pengaruh dan penggunaan nilai angka pada dasarnya penting sebagai sarana untuk menghasilkan grafik dan visual. Hubungan langsung antara nilai angka dan properti visual yang tersedia di lingkungan pemrograman membuka berbagai kemungkinan visual dan konseptual untuk menghasilkan grafik digital yang dinamis. Data angka dapat bersumber dari kalkulasi matematika internal atau dari sumber data eksternal dan digunakan untuk mengubah tampilan visual grafik secara dinamis (misalnya, bentuk, ukuran, warna, gerakan).

2.2 POLA ANGKA

Cara paling sederhana untuk membuat angka guna menentukan keluaran visual adalah dengan menggunakan pola angka yang dibuat dari perhitungan ulang nilai matematis yang berurutan dan berulang. Urutan angka dapat digunakan untuk membuat kisi-kisi yang dapat diulang dan struktur lain yang memberikan tatanan dan harmoni visual pada sebuah karya seni atau desain. Pola ini telah menginspirasi banyak generasi seniman dan desainer dari berbagai praktik kreatif (misalnya, seni, musik, dan desain) dan telah diterapkan pada berbagai konteks visual kreatif misalnya, visual logis dan fungsional dari desain modernis Swiss; pola geometris dekorasi ornamen; dan struktur komposisi musik Philip Glass. Konsistensi dan prediktabilitas mekanis dari nilai-nilai ini, ketika diterapkan pada halaman, layar, atau kanvas, dapat menciptakan rasa ritme, keseimbangan, dan harmoni yang jelas.

Penempatan ulang dan penggabungan bentuk geometris sederhana yang berulang dapat, misalnya, digunakan untuk menghasilkan pola rumit yang luar biasa. Pola kompleks muncul yang mengandung harmoni ritme matematis dan visual yang kuat. Ornamen yang ditemukan dalam pola Islam adalah contoh yang baik dari pengulangan bentuk geometris sederhana yang terinspirasi oleh angka yang menciptakan banyak pola yang sangat dekoratif. Dalam lingkungan pemrograman, menghasilkan urutan angka adalah cara langsung untuk menghasilkan visual yang dinamis dan menarik, memungkinkan seniman dan desainer untuk mengeksplorasi kemungkinan kreatif baru dan menciptakan karya yang unik dan inovatif. Dengan demikian, pola angka menjadi alat yang powerful bagi seniman dan desainer untuk menciptakan karya yang tidak hanya estetis, tetapi juga memiliki makna dan struktur yang kuat. Melalui penggunaan pola angka, seniman dan desainer dapat menciptakan karya yang memadukan keindahan visual dengan keakuratan matematis, sehingga menghasilkan pengalaman yang lebih mendalam dan menarik bagi pengguna.



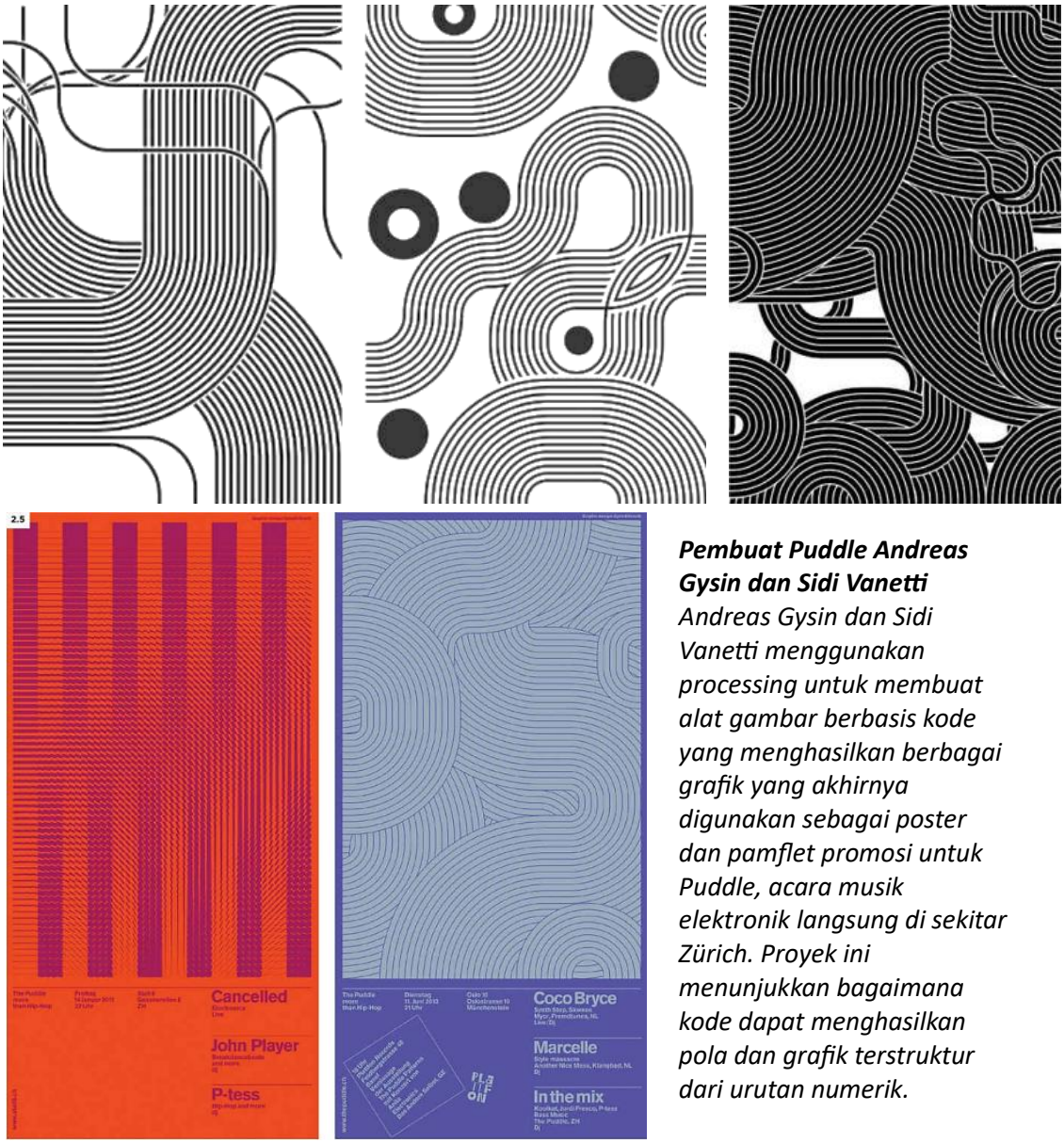
Seni Dekoratif Islam

Ubin dan desain dekoratif yang digunakan dalam seni Islam dicirikan oleh pola geometris yang sangat terstruktur yang merupakan representasi spiritualitas pandangan dunia Islam dan bertindak sebagai simbol konkret dari yang tak terbatas. Jenis pola geometris ini memiliki banyak kesamaan dengan grafik yang dihasilkan oleh kode. Kemampuan alami pemrograman komputer untuk mengulang, menggambar ulang, mengubah skala, dan mengubah nilai angka tanpa batas dapat menghasilkan berbagai macam pola dan bentuk geometris.

Setelah hubungan antara pola angka dan atribut visual dibuat, setiap penyesuaian pada perhitungannya, baik sengaja maupun tidak sengaja akan mengubah, atau mengganggu hasil visual. Sedikit perubahan pada perhitungan yang digunakan untuk menentukan sudut bentuk akan mengubah atau mengganggu pola rotasi ritmisnya, menciptakan bantuk baru. Bermain dengan nilai angka - mengubah cara mereka dihitung dan bagaimana mereka diterapkan sehingga memiliki efek langsung dan sering mengejutkan pada hasil visual akhir. Urutan angka sederhana dapat menghasilkan bentuk dan

pola yang dapat diulang dan digunakan kembali dalam banyak cara yang berbeda.

Perhitungan matematika sederhana (misalnya, menambah atau mengurangi nilai) menciptakan urutan angka yang dapat diterapkan untuk menentukan atribut visual suatu bentuk (ukurannya, gerakannya, lokasinya, sudutnya, warnanya) dan, oleh karena itu, digunakan untuk menghasilkan urutan ritmis dari pola yang dihasilkan secara komputasional. Kemampuan komputer untuk memproses ratusan perhitungan dengan sangat cepat, menerapkannya pada sejumlah besar grafik, membuat pengulangan perhitungan numerik menjadi sesuatu yang dapat dengan cepat menghasilkan pola visual yang kompleks.



Pembuat Puddle Andreas Gysin dan Sidi Vanetti
Andreas Gysin dan Sidi Vanetti menggunakan processing untuk membuat alat gambar berbasis kode yang menghasilkan berbagai grafik yang akhirnya digunakan sebagai poster dan pamflet promosi untuk Puddle, acara musik elektronik langsung di sekitar Zürich. Proyek ini menunjukkan bagaimana kode dapat menghasilkan pola dan grafik terstruktur dari urutan numerik.

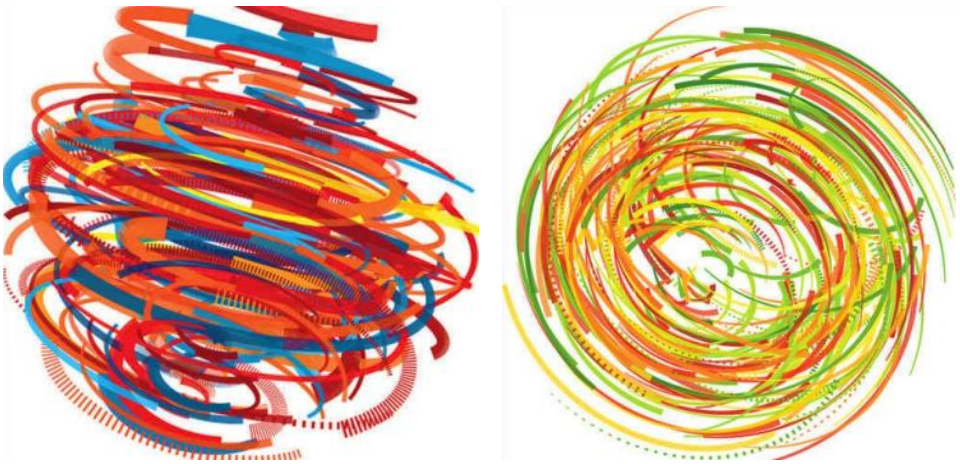
Setelah tautan antara pola angka dan atribut visual dibuat, penyesuaian apa pun pada perhitungannya, baik yang disengaja maupun tidak disengaja akan mendistorsi, mengganggu, atau mengubah hasil visual. Perubahan kecil pada perhitungan yang, misalnya, digunakan untuk mengatur sudut suatu bentuk akan mengubah atau mengganggu pola rotasi ritmisnya, sehingga menciptakan bentuk dan rupa versi baru. Bermain-main dengan nilai angka mengubah cara perhitungan dan penerapannya memiliki efek langsung, dan sering kali

mengejutkan, pada hasil visual akhir. Oleh karena itu, urutan angka sederhana dapat menghasilkan bentuk dan pola yang dapat diulang yang dapat digunakan dan digunakan kembali dalam berbagai cara.

CONTOH

Marius Watz

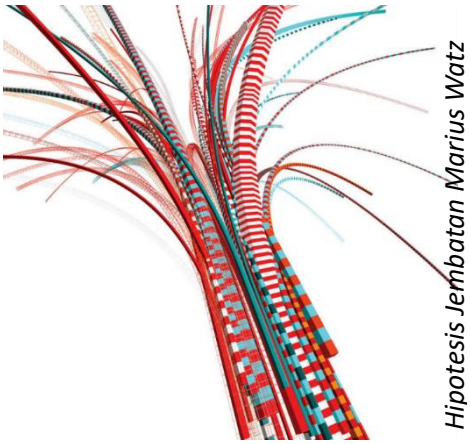
Marius Watz adalah seniman yang karyanya secara ekstensif mengeksplorasi penggunaan proses berbasis kode dan matematika untuk menghasilkan gambar komputasi yang memukau. Algoritma, kode yang digunakan untuk menghasilkan pola dan urutan angka, penting bagi sebagian besar karyanya saat ia menyelidiki cara-cara menghasilkan artefak unik secara visual dari kode. Karya tersebut dicirikan oleh bentuk geometris yang terdefinisi dengan baik dan warna-warna yang hidup, dan jangkauannya luas mulai dari karya perangkat lunak murni dan proyeksi hingga objek fisik yang diproduksi dengan teknologi fabrikasi digital.



Iluminasi B Marius Watz

Gambar-gambar ini diambil dari 100 rangkaian gambar vektor yang dihasilkan oleh sistem perangkat lunak yang ditulis secara khusus. Proyeksi gambar secara "real-time" digunakan bersama dengan versi cetak.

Gambar diam yang diambil dari animasi bentuk geometris yang dihasilkan secara komputasional yang mencerminkan struktur jembatan dan jalan yang kokoh. Watz merancang gambar asli agar dapat dilihat sebagai proyeksi skala besar.



Hipotesis Jembatan Marius Watz

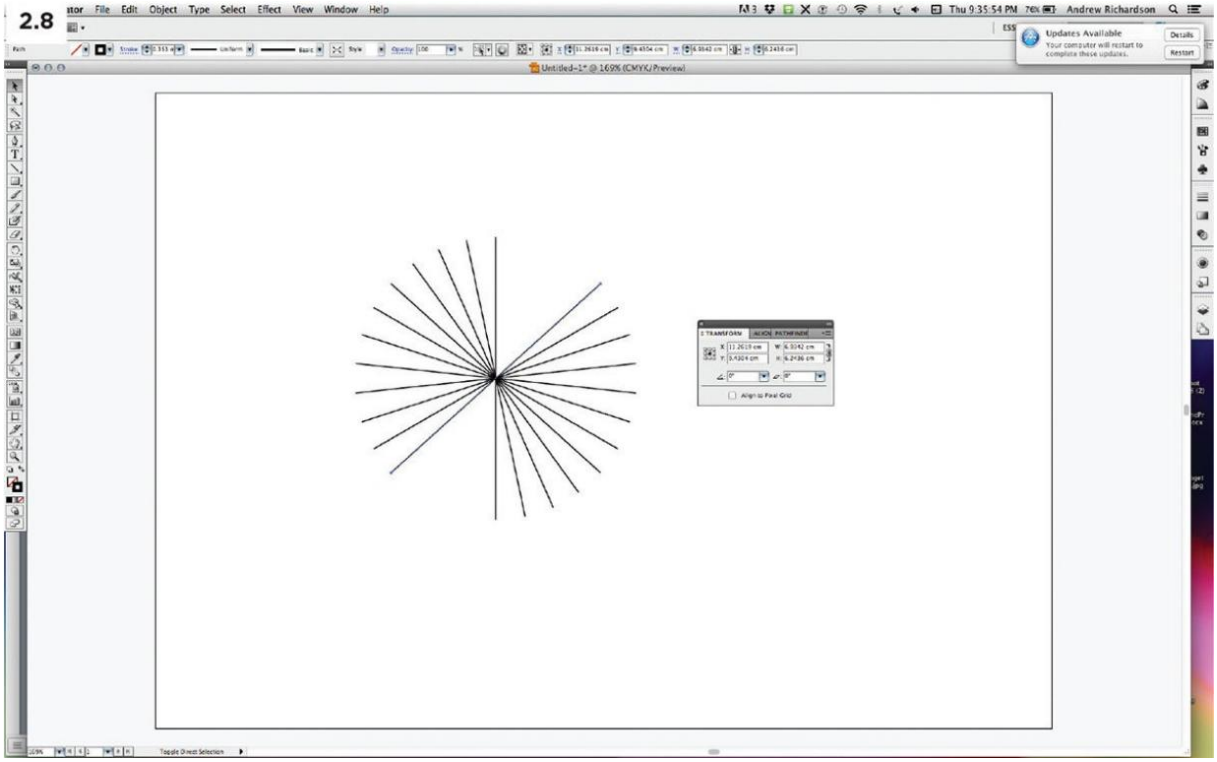
2.3 PENGULANGAN GAMBAR SISTEMATIS

Pengulangan adalah konsep inti dari banyak proses desain digital. Prosedur salin-tempel atau langkah dan ulangi umum di antara banyak perangkat lunak alat gambar. Prosedur ini memungkinkan bentuk untuk direproduksi dan diubah secara berulang dan akurat (diskalakan, diputar, dipindahkan, dll.) untuk membuat desain berulang yang rumit. Hasil dari jenis proses ini terorganisir dan terkendali, tetapi menghasilkan variasi. Penskalaan dan pemindahan berulang bahkan bentuk yang sederhana dapat menciptakan susunan bentuk dan pola yang beraneka ragam.

Ide pengulangan juga merupakan bagian yang sangat penting dari penulisan kode komputer. Kode dapat dengan cepat dan akurat mengulang instruksi atau kalkulasi atau memilah sekumpulan besar data, dan kemampuan utama ini merupakan inti dari banyak tugas yang harus dilakukan. Setiap bahasa pemrograman mencakup berbagai cara berbeda untuk mengulangi) instruksi atau fungsi; semuanya merupakan bagian dari struktur dan struktur

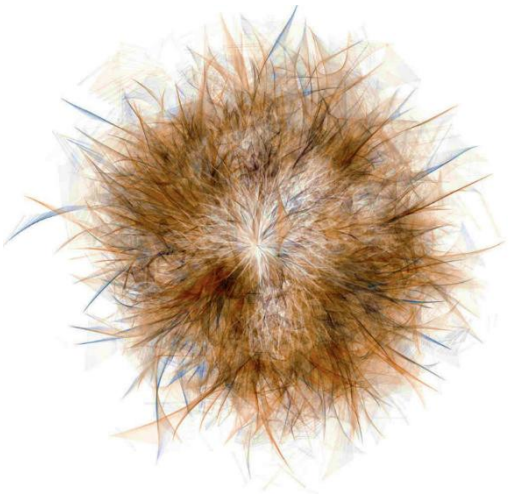
lingkungan pemrograman. Fungsi inti bawaan misalnya, draw() digunakan untuk terus-menerus mengeksekusi baris kode, berulang kali mengulang daftar instruksi selama program berjalan (berpotensi tak terbatas atau hingga program dimatikan).

Elemen pemrograman lain secara khusus dirancang untuk mengulang (mengulangi) serangkaian instruksi sejumlah waktu tertentu; jenis pengulangan ini (misalnya, pengulangan *"for"* atau pengulangan *"while"*) dapat digunakan untuk menghitung seluruh daftar item atau mengubah dan menggambar ulang bentuk sejumlah waktu tertentu. Menggunakan fungsi pengulangan (iteratif) untuk menghasilkan hasil visual menghasilkan grafik dengan struktur visual yang jelas. Fungsi pengulangan *"for"* dapat digunakan, misalnya, untuk menskalakan, memindahkan, atau memutar bentuk secara bertahap; fungsi ini menghasilkan hasil visual yang rapi, teratur, dan terkontrol dengan baik.



Tangkapan layar Illustrator

Contoh hasil visual dari proses langkah dan ulangi di Adobe Illustrator, yang dapat digunakan untuk menghasilkan bentuk dan rupa geometris sederhana.



Proses 6 Casey Reas

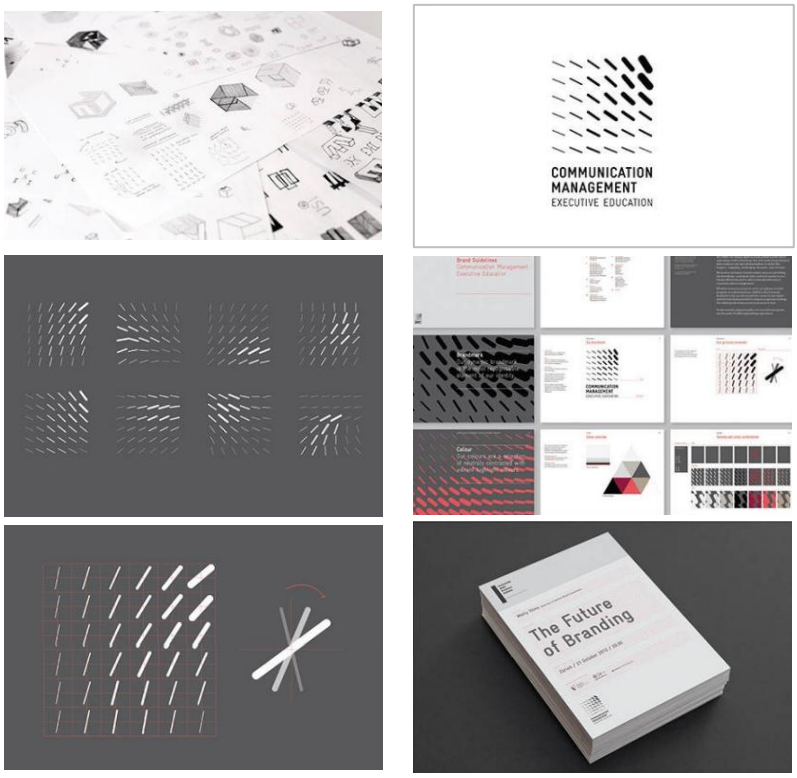
Karya "proses" Casey Reas adalah serangkaian gambar dan lukisan komputasional eksperimental di mana deskripsi tertulis formal dari suatu proses diterjemahkan ke dalam instruksi komputer untuk menghasilkan gambar. Setiap proses menghasilkan serangkaian gambar komputasional yang unik.

CONTOH

Moving Brands: Logo untuk EMScom

Moving Brands adalah agensi desain yang terkenal akan inovasi dan eksperimennya dalam bidang desain untuk media digital baru. Desain merek baru dari mereka untuk logo EMScom menyoroti pendekatan inovatif dan eksperimental terhadap desain grafis yang mencakup berbagai disiplin ilmu dalam bidang desain grafis dan menunjukkan cara-cara di mana desain pemrograman digital dapat menginformasikan desain grafis tradisional.

Identitas merek EMScom dikembangkan dari visualisasi identitas merek dan selanjutnya digunakan di seluruh media cetak dan layar. Kisi garis *"reaktif"* sederhana diprogram dalam Processing sebagai representasi visual interaktif dari karakteristik identitas inti perusahaan. Setiap garis individual dalam kisi diprogram agar reaktif terhadap klik mouse pengguna dan responsif terhadap kualitas garis di sekitarnya, sehingga setiap garis mengambil orientasi dan bobotnya dari garis-garis di sekitarnya.



EMScom Moving Brands

Jaringan garis organik interaktif, yang digunakan sebagai dasar identitas visual merek EMScom, dibuat menggunakan kode Pemrosesan. Struktur visual berfungsi sebagai dasar yang kuat dan fleksibel tempat berbagai hasil cetak dihasilkan.

Logo Generatif/Dinamis

Identitas perusahaan dibuat sebagai serangkaian kisi visual, aturan, dan struktur terperinci yang menguraikan keadaan dan aturan penggunaannya. Dengan menerapkan aturan dan instruksi pemrograman untuk mengatur dan memanipulasi atribut visualnya, identitas merek dapat dibuat sebagai logo yang *"dinamis"* dan reaktif tidak hanya dianimasikan tetapi juga responsif terhadap berbagai jenis masukan data. Logo *"dinamis"* ini menjadi objek yang fleksibel, generatif, dan hidup yang mengekspresikan nilai dan karakteristik suatu merek dalam berbagai cara yang menyenangkan, mudah dibentuk, dan sangat dapat disesuaikan.

Banyak contoh telah dibuat untuk mengeksplorasi dan bereksperimen dengan gagasan menerjemahkan (menciptakan) parameter angka dinamis untuk menghasilkan variasi dan sistem visual untuk desain logo.

2.4 KERUMITAN DARI KESEDERHANAAN

Kemampuan komputasional (penghitungan angka) komputer yang sangat tinggi berarti bahwa jumlah iterasi atau putaran yang dapat dilakukan benar-benar melampaui apa yang

mungkin dilakukan dengan cara lain. Ratusan atau bahkan ribuan instruksi dan proses dapat terus-menerus (dan tanpa usaha) diputar atau diulang beberapa kali per detik saat program berjalan. Jadi, menggunakan lingkungan ini untuk menggambar dapat memulai proses yang dapat berulang dan berjalan setidaknya secara teoritis selamanya.

Kemampuan program komputer dapat mengulang kalkulasi dan proses secara terus-menerus, sehingga memungkinkan pembuatan gambar dan grafik yang sangat rumit dan detail. Dengan mengulang dan menggabungkan bentuk geometris sederhana secara komputasional, kita dapat menciptakan bentuk digital yang kompleks dan menarik. Hal ini memungkinkan seniman dan desainer untuk menciptakan karya yang sangat detail dan rumit dengan lebih mudah dan efisien.

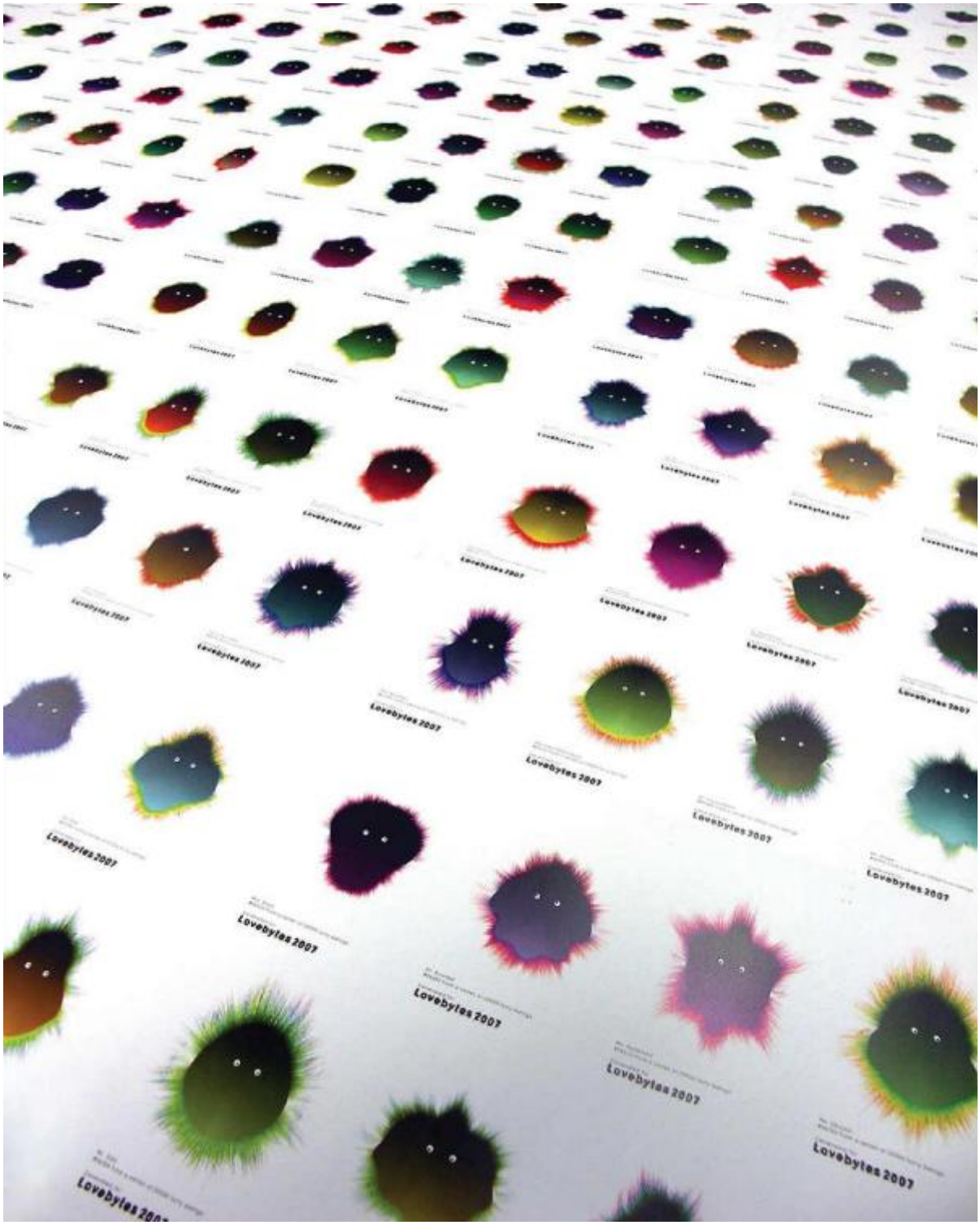
Ketika bentuk komputasional dimasukkan ke dalam program yang memanfaatkan kemampuan pemrosesan komputer yang luas untuk menggambar dan mengubah angka dan grafik, variasi dan kombinasi bentuk yang tak terbatas menjadi mungkin. Kombinasi lingkaran, garis, dan poligon dapat dibuat ulang dengan cepat untuk membuat berbagai macam bentuk grafis. Instruksi pemrograman dapat digunakan untuk menentukan aturan dan parameter awal, dan proses yang berulang memungkinkan gambar dan desain tumbuh dan berkembang secara bertahap. Perubahan kecil pada fungsi menggambar, jika diulang akan menghasilkan perubahan pada gambar garis, bentuk, dan grafik.

Oleh karena itu, loop pemrograman memberi desainer kemampuan untuk membuat gambar kompleks dari beberapa instruksi sederhana sehingga daftar perintah menggambar yang panjang tidak diperlukan. Beberapa instruksi menggambar dapat dieksekusi berulang kali ratusan atau ribuan kali. Aturan desain awal menyediakan kondisi dan parameter awal untuk proses tersebut. Kompleksitas visual yang dapat muncul dari beberapa instruksi awal saja memberikan kode dan gambar semacam keanggunan komputasional, menghasilkan kompleksitas visual dari kesederhanaan yang diprogram. Kombinasi kompleksitas dan kesederhanaan merupakan ide penting dan hebat yang sering digunakan oleh seniman dan desainer saat mengeksplorasi kapasitas kode pemrograman.

CONTOH

Universal Everything: Logo Lovebytes

Universal Everything menggunakan kode pemrograman untuk menghasilkan materi publisitas bagi Lovebytes, sebuah organisasi seni digital yang berpusat di Sheffield, Inggris. Solusi desain didasarkan pada program gambar generatif yang ditulis khusus yang menghasilkan 20.000 variasi unik dari satu karakter dasar amoeba yang kemudian digunakan sebagai gambar untuk 20.000 kartu pos dan undangan unik. Kode tersebut mengulang perintah untuk menyesuaikan atribut spesifik dari setiap amoeba (bentuk, tekstur, warna, posisi mata, dll.). Meskipun semuanya itu dibuat dari satu template komputasi generik, setiap gambar yang dihasilkan bersifat unik dan berbeda antara satu dengan yang lain. Proyek seperti ini menyoroti potensi kode komputer yang bekerja dengan proses cetak untuk menghasilkan grafik dengan variasi tak terbatas.



Lovebytes Universal Everything

Penerapan proses komputasi acak secara berulang-ulang (dan berpotensi hampir tak terbatas) pada proyek berbasis cetak menghasilkan serangkaian kartu pos generatif yang unik.

Moving Brands: IO

Pada tahun 2013, Moving Brands mengembangkan identitas visual dan pencitraan merek untuk aplikasi komunikasi mandiri Swisscom. Meskipun hasil akhirnya mencakup banyak jenis media cetak dan layar tradisional, keterlibatan eksperimental dengan kode generatif digunakan sebagai bagian utama dari proses dan solusi desain akhir. Proyek ini

merupakan contoh menarik tentang bagaimana cara kode dapat digunakan untuk membuat *"sketsa"* pemrograman visual yang sederhana namun fleksibel sehingga dapat menghasilkan serangkaian aset merek yang dinamis.

Dengan menggunakan Processing sebagai lingkungan pengodean, tim desain menulis aplikasi sederhana yang mengulang bentuk lengkung dasar dalam berbagai formasi spiral yang berbeda. Nilai angka yang digunakan untuk mengatur detail visual (lokasi, sudut, warna, dll.) dari setiap pola berulang dapat diubah melalui serangkaian penggeser di layar yang menggeser tampilan bentuk dan rupa secara keseluruhan.

Bahkan penyesuaian kecil pada nilai-nilai tersebut menghasilkan perubahan visual yang besar, dan mendorong desainer untuk *"bermain"* dengan sistem pemrograman untuk menghasilkan berbagai hasil yang berbeda-beda. Oleh karena itu, program yang menghasilkan visual tersebut mampu menghasilkan berbagai bentuk dan rupa yang hampir tak terbatas. Visual terpilih yang dihasilkan oleh program tersebut disimpan dan digunakan sebagai tekstur statis dan bergerak di seluruh merek, termasuk antarmuka aplikasi, elemen situs web, karya seni cetak, dan video promosi.

iO Moving Brands

Alat Pemrosesan yang dibuat khusus memungkinkan para desainer identitas visual untuk iO untuk mengembangkan serangkaian tekstur statis dan bergerak yang dibuat secara generatif yang membantu menginformasikan semua aspek solusi desain akhir.

2.5 GAMBAR ACAK

Jika urutan matematika sederhana menghasilkan prediktabilitas visual, maka angka acak menghasilkan ketidakpastian dan gangguan visual. Konsep *"keacakan"* sebagai bagian dari ekspresi visual sangat penting. Konsep ini telah dieksplorasi dan diujicobakan dengan berbagai cara oleh para seniman dan desainer. Contoh yang paling menonjol mungkin berasal dari gerakan ekspresionis abstrak dari Amerika yang dilambangkan oleh Jackson Pollock di dalam lukisan *"tetesannya" yang* telah menjadi simbol peluang artistik yang terkenal.

Memperkenalkan elemen visual yang tidak terkendali dari sebuah peluang atau kecelakaan memungkinkan seniman untuk melangkah keluar dari batasan upaya sadar yang biasa, bergerak melampaui prediktabilitas visual dan melangkah ke area yang diatur oleh kekuatan eksternal, tidak terkendali atau bawah sadar lainnya. Hasilnya adalah karya satu kali yang mengandung elemen yang tidak terduga. Elemen ketidakpastian menghadirkan keindahan visualnya sendiri, yang sering dikaitkan dengan proses buatan tangan atau non-mekanistik, dan bahkan telah diperkenalkan ke dalam lingkungan perangkat lunak grafis. Alat kuas khusus di Illustrator dan Photoshop menggunakan elemen keacakan untuk mensimulasikan efek alat lukis di kehidupan nyata.

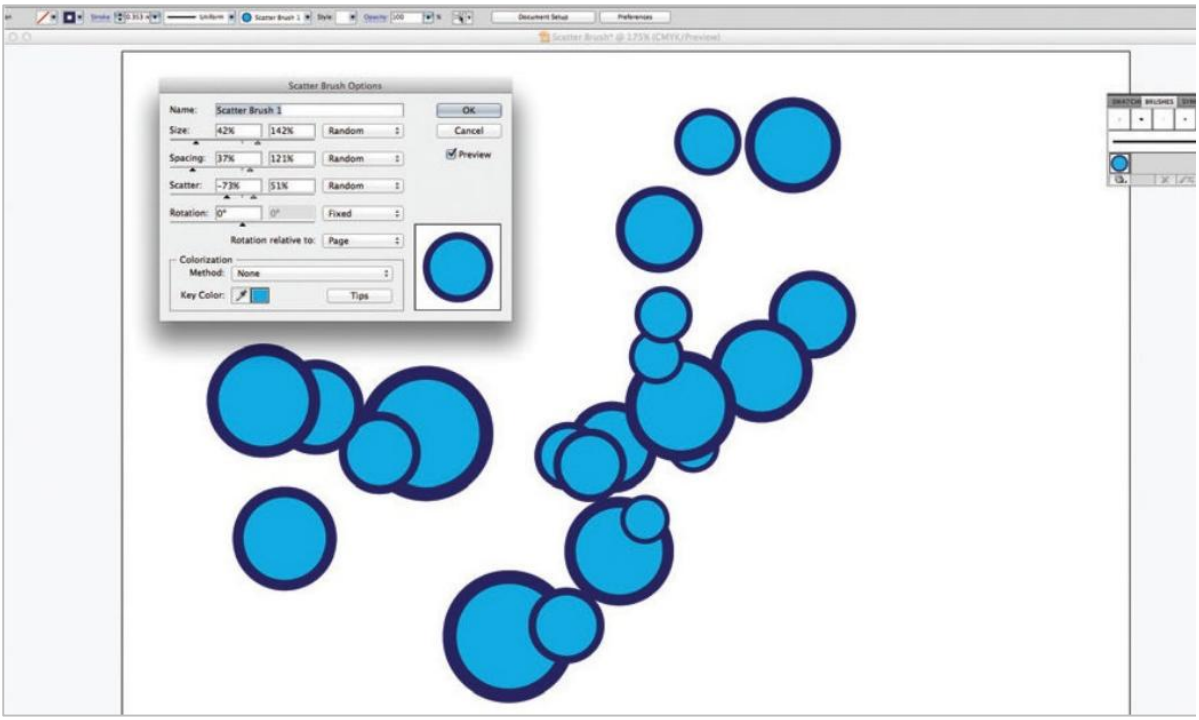
Pengenalan kesempatan dan kecelakaan *("keacakan")* tidak berarti bahwa karya tersebut sepenuhnya tanpa pengaruh seniman. Bahkan dalam proses acak, beberapa parameter dan atribut visual harus diatur dan didefinisikan oleh seniman atau desainer. Misalnya dalam lukisan *"drip"* Pollock, seniman memilih cat berwarna dan gerakan fisiknya berkontribusi pada penciptaan garis-garis proses *"acak".* Oleh karena itu melibatkan keseimbangan antara yang terkendali dengan yang tidak terkendali yang di mana seniman menetapkan batasan dan membuat penilaian akhir yaitu, berapa banyak *"keacakan"* yang diterapkan dan kapan karya tersebut selesai. Keacakan total menghasilkan kebisingan visual; namun, ketika diterapkan dengan hati-hati dan digunakan dalam parameter visual yang didefinisikan. Misalnya, palet warna terbatas), sifat hasil yang tidak terduga dapat menjadi menarik.

Contoh salah satu lukisan gaya "tetesan" Jackson Pollock yang terkenal.

Keacakan visual dapat digunakan untuk menciptakan tekstur yang unik dan menarik pada sebuah karya seni atau desain. Dengan menggunakan pengaturan kuas pencar yang tepat, seniman dan desainer dapat menciptakan efek yang mirip dengan sapuan kuas yang tidak teratur atau percikan cat yang acak. Hal ini dapat menambahkan dimensi baru pada karya seni atau desain, membuatnya terlihat lebih hidup dan dinamis. Selain itu, keacakan visual juga dapat digunakan untuk menciptakan efek yang lebih abstrak dan ekspresif. Dengan menggunakan pengaturan kuas pencar yang lebih ekstrem, seniman dan desainer dapat menciptakan karya yang lebih ekspresif dan emosional. Hal ini dapat digunakan untuk menciptakan suasana hati atau atmosfer tertentu pada sebuah karya seni atau desain.

Penggunaan keacakan visual juga dapat membantu seniman dan desainer untuk menciptakan karya yang lebih organik dan alami. Dengan menambahkan elemen keacakan yang terkontrol, seniman dan desainer dapat menciptakan karya yang terlihat lebih alami dan tidak terlalu sempurna. Hal ini dapat digunakan untuk menciptakan karya yang lebih realistis

dan dapat dihubungkan dengan alam. Dalam prakteknya, penggunaan keacakan visual memerlukan keseimbangan yang tepat antara kontrol dan keacakan. Seniman dan desainer perlu memastikan bahwa keacakan visual tidak terlalu dominan sehingga karya seni atau desain menjadi terlalu kacau atau tidak terbaca. Dengan demikian, seniman dan desainer dapat menciptakan karya yang unik dan menarik dengan menggunakan keacakan visual secara efektif.



Tangkapan layar Illustrator

Elemen "keacakan" visual sering kali menjadi bagian yang diinginkan dari sebuah gambar atau desain. Illustrator, seperti banyak perangkat lunak menggambar lainnya, memiliki kemampuan untuk mengacak bentuk dan tanda di layar dengan menyesuaikan pengaturan kuas "pencar". Dengan demikian, seniman dan desainer dapat menciptakan karya yang unik dan menarik dengan menambahkan elemen keacakan yang terkontrol.

Keacakan visual juga dapat digunakan untuk menciptakan efek yang lebih dinamis dan interaktif pada sebuah karya seni atau desain. Dengan menggunakan pengaturan kuas pencar yang tepat, seniman dan desainer dapat menciptakan karya yang terlihat lebih hidup dan bergerak. Hal ini dapat digunakan untuk menciptakan pengalaman yang lebih imersif dan menarik bagi penonton. Penggunaan keacakan visual juga dapat membantu seniman dan desainer untuk menciptakan karya yang lebih unik dan orisinal. Dengan menambahkan elemen keacakan yang terkontrol, seniman dan desainer dapat menciptakan karya yang tidak dapat diprediksi dan memiliki karakter yang unik. Hal ini dapat digunakan untuk menciptakan karya yang lebih kreatif dan inovatif.

Dalam keseluruhan, keacakan visual merupakan elemen yang penting dalam menciptakan karya seni atau desain yang unik dan menarik. Dengan menggunakan pengaturan kuas pencar yang tepat dalam menambahkan elemen keacakan yang terkontrol, seniman dan desainer dapat menciptakan karya yang lebih hidup, dinamis, dan kreatif. Hal ini dapat digunakan untuk menciptakan pengalaman yang lebih imersif dan menarik bagi penonton, serta menciptakan karya yang lebih unik dan orisinal.

Bahkan dalam proses acak, beberapa parameter dan atribut visual harus ditetapkan dan didefinisikan oleh seniman atau desainer. Misalnya, dalam lukisan "tetes" Pollock, seniman memilih cat berwarna dan gerakan fisiknya berkontribusi pada penciptaan garis. Oleh karena itu, proses "acak" melibatkan keseimbangan antara yang terkendali dan yang tidak terkendali di mana seniman menetapkan batasan dan membuat penilaian akhir yaitu, seberapa banyak "keacakan" yang diterapkan dan kapan karya tersebut selesai. Keacakan total menghasilkan gangguan visual; namun, ketika diterapkan dengan hati-hati dan digunakan dalam parameter visual yang ditentukan misalnya, palet warna terbatas, sifat hasil yang tidak terduga dapat menjadi menarik.

Dalam lingkungan kode, keacakan visual dicapai secara numerik: Fungsi pemrograman khusus menghasilkan nilai angka acak yang kemudian dapat diterapkan ke satu atau banyak elemen visual gambar. Bentuk, ukuran, warna, ketebalan garis, dan arah garis gambar dapat dihasilkan secara acak, menghasilkan hasil dan keluaran visual yang berbeda setiap kali program dijalankan. Hubungan langsung antara kode, angka, dan visual memungkinkan tautan langsung dibuat antara nilai angka yang dipilih secara acak dan elemen visual yang berubah secara acak. Menambahkan keacakan ke dalam gambar kode adalah cara yang sederhana tetapi ampuh untuk menghasilkan serangkaian desain visual yang tidak dapat diprediksi yang bergantung pada peluang dan pemilihan acak.

Proses 16 Casey Reas
Meskipun didefinisikan oleh serangkaian instruksi tertulis dan komputasional, hasil dari lukisan "proses" komputasional Casey Reas memiliki kecenderungan ke arah keacakan yang merupakan gema dari lukisan "tetesan" Jackson Pollock.

Desainer mendefinisikan "seberapa banyak" keacakan yang akan digunakan (tingkat "kebetulan" dalam karya) dengan menentukan rentang dari mana angka-angka dipilih dan menentukan atribut mana yang dihasilkan dengan cara ini. Ini penting karena hasil keacakan murni (komputasional), di mana semua warna, bentuk, dan tanda dihasilkan secara acak, menghasilkan hasil yang berantakan secara seragam: padanan visual dari "white noise".

Ketika diterapkan dengan hati-hati, elemen keacakan dan ketidakpastian dapat menawarkan nuansa perubahan dan varian yang menghasilkan variasi halus dari perubahan visual. Meskipun gagasan keacakan mungkin tampak bertentangan dengan keinginan seniman dan desainer untuk mempertahankan kontrol visual, mereka menghargainya; itu memberikan nuansa pada grafik yang dihasilkan dalam lingkungan komputasional yang dapat diprediksi dan berpotensi formulais. Saat menggunakan kode untuk membuat grafis, desainer harus menerima gagasan ketidakpastian visual. Melepaskan sebagian kendali atas detail visual akhir

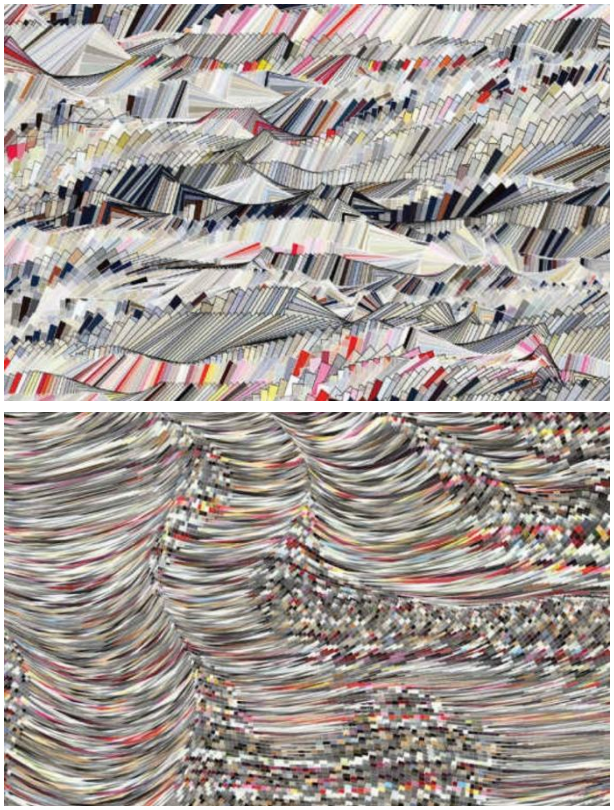
merupakan bagian penting dari proses kreatif.

CONTOH

Holger Lippmann: Perlin Scape 1

Holger Lippmann adalah seniman asal Jerman yang menggunakan pemrograman (terutama Pemrosesan) sebagai bagian penting dari praktik kreatifnya. Seri Perlin Scape didasarkan pada algoritma "perlin noise": fungsi yang menghasilkan serangkaian nilai angka acak yang memiliki urutan yang lebih naturalistik daripada angka "murni" acak.

Urutan angka perlin yang dihasilkan secara acak digunakan untuk menentukan atribut visual misalnya, warna dan sudut dari serangkaian persegi panjang. Hasil dari proses ini adalah komposisi yang harmonis: lukisan digital dari gelombang bentuk berwarna yang mengalir. Penggunaan nilai angka acak yang dihasilkan secara komputasional memberikan komposisi tersebut nuansa naturalistik dan berarti bahwa setiap versi baru dari karya seni tersebut bersifat unik.



Perlin Scape Holger Lippmann
Keacakan terkontrol dari nilai "Perlin noise" memberikan aliran visual yang harmonis dan naturalistis pada gambar.

Perlin Noise

"Perlin noise" adalah algoritma yang diciptakan oleh Ken Perlin pada tahun 1980an yang menghasilkan urutan angka acak yang "naturalistik". Algoritma ini sering digunakan untuk menciptakan tekstur, medan, dan bentuk yang tampak organik di lingkungan komputer.

10.000 Lukisan Digital

Studio desain inovatif Field menciptakan serangkaian 10.000 ilustrasi digital unik untuk digunakan pada cetakan promosi untuk mempublikasikan karya produsen kertas independen G. F. Smith. Studio tersebut menggunakan proses generatif acak untuk membuat patung digital organik; setiap fragmen individual darinya menjadi satu dari 10.000 gambar. Hasilnya adalah serangkaian lukisan yang hidup dan berwarna-warni yang penggunaan gambar yang dihasilkan secara dinamis mendorong batasan dan kemungkinan cetak digital.

10.000 Lukisan Digital FIELD
Serangkaian 10.000 ilustrasi unik yang dibuat melalui proses kode dan dikembangkan untuk digunakan sebagai sampul brosur kertas G. F. Smith. Solusi desain akhir menunjukkan kemungkinan besar dari desain cetak digital dan desain generatif.

2.6 GAMBAR DINAMIS/GENERATIF

Tidak seperti sketsa yang ditulis di atas kertas, atau animasi digital yang ditulis dalam waktu, grafik berbasis kode bersifat dinamis yaitu, dapat berubah dan bervariasi. Hasil gambar yang dibuat dengan kode dapat berubah dari informasi data eksternal dan dapat menghasilkan berbagai hasil yang mengejutkan dan tidak terduga. Bagian kode yang sama yang dijalankan seratus kali dapat menghasilkan seratus hasil yang sedikit berbeda karena angka yang menentukan atribut grafik bersifat dinamis dan dapat berubah, berubah setiap kali kode dijalankan.

Bertindak sebagai semacam visualisasi berbasis data, gambar dan desain komputasional dapat dibuat dari serangkaian sumber data numerik, yang membuka kemungkinan kreatif baru untuk menggambar dan memvisualisasikan informasi grafik. Gambar yang dibuat dengan kode dapat dibuat dari nilai angka yang dibuat secara acak atau matematis seperti yang dibahas, dari gerakan tetikus, gerakan dan interaksi pengguna, data waktu, umpan audio, atau dari banyak set data bersumber eksternal lainnya.

Dalam terminologi pemrograman, nilai data yang dapat diubah disebut sebagai "variabel". Variabel adalah wadah yang diberi nama untuk mengubah nilai dan umumnya digunakan untuk mengganti angka tetap yang digunakan untuk menentukan atribut visual suatu gambar. Variabel merupakan inti dari banyak konsep pemrograman dan menyediakan landasan untuk menciptakan gambar yang dinamis dan responsif secara digital. Setiap kali program dijalankan, serangkaian hasil visual baru dihasilkan. Hasilnya dapat mencakup berbagai eksperimen visual; kecelakaan yang menyenangkan dan kombinasi visual yang tak terduga muncul, dengan berbagai penggunaan dan aplikasi grafis.

Seorang desainer akan membuat struktur dan parameter menyeluruh dari kode tetapi mengesampingkan kontrol total dari hasil visual dengan memungkinkan pengaruh variabel, data eksternal misalnya, angka acak, data yang dibuat pengguna, atau data eksternal untuk

membuat elemen tertentu dari keluaran visual. Beberapa ide awal yang membayangkan bagaimana variabel dan visual dapat dihubungkan bersama membantu menciptakan konsep untuk menentukan parameter gambar. Konsep awal dapat sangat sederhana ("menggambar lingkaran pada posisi mouse") atau lebih kompleks; misalnya, dapat menghubungkan nilai warna ke angka yang dimasukkan dari sumber data eksternal.

Setelah tautan antara nilai angka dinamis dan parameter visual gambar dibuat, proses kreatif menjadi proses "menyempurnakan" hubungan antara angka dan visual untuk menghasilkan serangkaian hasil yang paling memuaskan. Siklus percobaan coba-coba dialog kreatif antara desainer dan program berkembang di mana keseimbangan dan keselarasan antara angka dan kode "keras" dan grafik "lunak" dikembangkan. Proses kreatif adalah "kemitraan" antara desainer, yang mendefinisikan konsep dan struktur, dan kode, yang mengalokasikan dan menggunakan nilai data tertentu.

Menghubungkan nilai variabel ke atribut visual memungkinkan gambar menjadi dapat diubah "secara dinamis". Berbagai macam sumber data numerik yang berbeda dapat digunakan untuk menghasilkan grafik yang dapat digunakan dalam berbagai konteks dan hasil visual. Bahkan hubungan sederhana antara visual dan nilai angka yang dapat diubah secara dinamis dapat membentuk hasil visual yang memuaskan dan menarik. Posisi tetikus di layar, misalnya, ditangkap oleh dua nilai angka koordinat sederhana (mouseX, mouseY), yang berubah saat pengguna menggerakkan tetikus.

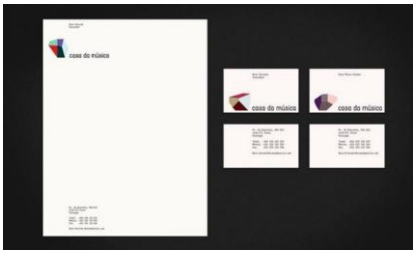
Pergerakan tetikus dapat dikaitkan dengan beberapa atribut visual dan digunakan untuk mengubah posisi, ukuran, atau warna suatu bentuk. Menghubungkan nilai "dinamis" ke properti visual suatu grafik menciptakan hubungan yang sederhana, tetapi langsung, antara pengguna dan grafik di layar, dan dapat menghasilkan banyak hasil visual yang menarik.

CONTOH

Sagmeister dan Walsh: Identitas Casa da Musica

Dibuat oleh Sagmeister dan Walsh, identitas visual untuk Casa Da Musica, sebuah pusat musik di Porto, Portugal, adalah contoh bagus tentang bagaimana perangkat lunak yang dibuat khusus dapat digunakan untuk menginformasikan identitas grafis yang fleksibel dan dinamis secara visual.

Bentuk bangunan yang dirancang Rem Koolhass yang unik dan khas digunakan sebagai sumber visual untuk menghasilkan logo 3D, yang dapat dilihat dari berbagai sudut pandang dan perspektif yang berbeda. Versi baru dari bentuk logo dibuat dari tampilan bangunan secara individual. Program



Casa da Musica Identitas Sagmeister & Walsh

Perangkat lunak pembuat logo diciptakan untuk mengekstrak warna dari gambar dan kemudian menerapkannya ke berbagai versi logo. Versi logo yang dibuat memiliki hubungan visual dengan gambar.

perangkat lunak yang dibuat khusus ditulis sebagai "generator logo" untuk membuat versi logo yang unik yang diinformasikan oleh nilai warna dari gambar yang dipilih (misalnya, foto musisi atau staf dari tempat tersebut).

Nilai warna diambil dari tujuh belas titik tertentu pada gambar dan digunakan untuk menentukan tujuh belas sisi berwarna dari bentuk logo 3D. Oleh karena itu, setiap gambar menghasilkan serangkaian identitas visual berwarna yang unik berdasarkan bentuk logo inti yang sesuai dengan warna gambar asli. Dengan cara ini, serangkaian identitas visual yang fleksibel dan dinamis tercipta yang memiliki hubungan visual yang jelas dengan gambar asli. Setiap rangkaian logo baru dapat digunakan pada berbagai hasil grafis yang terkait dengan tempat penyelenggaraan untuk menciptakan poster acara promosi yang unik atau kartu nama yang dipersonalisasi.

Print Screen

Saat menggunakan kode untuk membuat grafik, gambar dihasilkan pada resolusi "layar" rendah yang terlihat bagus di monitor tetapi tidak bagus untuk dicetak dalam skala besar (misalnya, untuk digunakan sebagai grafik pada poster).

Beberapa bahasa pemrograman menyertakan kemampuan untuk mengekspor grafik yang dihasilkan layar sebagai gambar skala besar atau bahkan sebagai file pdf, yang dapat dibuka dan diubah ukurannya pada resolusi yang mudah digunakan oleh printer. Bahasa Processing, misalnya, menyertakan pustaka khusus yang memungkinkan untuk menulis grafik yang dihasilkan kode sebagai file pdf, dan ini merupakan cara yang berguna untuk menghasilkan grafik yang dapat diskalakan, diubah ukurannya, dan dikeluarkan pada resolusi yang sangat tinggi.

IO: Here to There

Proyek Here to There mengeksplorasi proses yang menggabungkan algoritma pemrograman, proses, dan konsep di samping grafik yang diilustrasikan dengan tangan untuk menciptakan serangkaian ilustrasi imajinatif yang dibuat secara digital untuk anak-anak.

Terinspirasi oleh kenangan masa kecil akan karya seni di dinding kamar tidur mereka, desainer Emily Gobeille dan Theo Watson membangun rangkaian perangkat lunak mereka sendiri (menggunakan openFrameworks), yang masing-masing mengeksplorasi konsep komputasional seperti algoritma dan sebab akibat. Grafik yang dihasilkan setiap program menjadi blok bangunan dasar poster; ini dikombinasikan dengan grafik makhluk dan karakter yang diilustrasikan dengan tangan untuk menciptakan cerita hibrida dari dua dunia: "Kota" dan "Hutan."

Berbagai macam sumber data ditambang sebagai informasi numerik untuk menghasilkan gambar misalnya, data elevasi dari gunung berapi Hawaii dan dari permukaan bulan, serta data yang diambil dari bentuk gelombang suara para seniman. Ilustrasi yang dihasilkan adalah penggambaran yang menyenangkan dari fantasi dunia yang aneh; mereka memadukan citra yang dikembangkan secara komputasional dengan desain karakter, narasi, dan ilustrasi untuk menciptakan kisah dua lingkungan yang memukau secara visual. Karya seni terakhir dirilis sebagai cetakan terbatas.

Oleh karena itu, gambar komputasional dapat menghasilkan representasi visual data

yang menarik (gerakan, waktu, interaksi pengguna, dll.), yang menghasilkan asosiasi visual baru yang menarik yang diarahkan oleh aturan dan parameter pemrograman. Hubungan antara data dinamis dan hasil grafis dieksplorasi secara lebih rinci di seluruh bagian buku ini.

Di Sini ke Sana Desain IO (Here to ThereDesign IO)
Ilustrasi untuk poster anak-anak yang dibuat dari campuran grafis yang dihasilkan kode dan ilustrasi yang digambar tangan.

2.7 FUNGSI MENGGAMBAR

Banyak bahasa pemrograman memiliki kosakata fungsi menggambar bawaan yang memungkinkan programmer untuk menentukan dan membuat bentuk, titik, dan garis sederhana. Meskipun mendasar, elemen-elemen sederhana ini membentuk dasar dari banyak gambar komputasional, dan ketika diulang, berlapis, dan digabungkan dapat menciptakan bentuk visual yang rumit. Berikut ini adalah garis besar prosedur menggambar dasar yang digunakan dalam Pemrosesan.

Membuat Kanvas

Adalah berguna untuk menganggap "kanvas" di layar sebagai jenis kertas grafik digital yang setiap pikselnya merupakan satu kotak grafik. Ukuran area gambar (kanvas) ditentukan oleh fungsi "size", yang mengatur lebar dan tinggi piksel area gambar di layar.

```
size (width, height);
// creates a drawing area of 640px by 480px
size (640, 480);
```

Pemrosesan mencakup berbagai fungsi menggambar yang dapat digunakan untuk menggambar titik, garis, lingkaran, dan persegi. Masing-masing dibuat oleh fungsi

pemrograman sederhana yang memplot bentuk pada kisi koordinat x dan y kanvas.

Bentuk yang paling sederhana adalah titik piksel tunggal, yang dapat digambar ke layar menggunakan fungsi point. Fungsi tersebut menggunakan dua nilai angka, yang mengatur lokasi koordinat x dan y titik tersebut.

```
point (x, y); //the point function
// draws a dot at the coordinate point x:100px, y:150px`
point (100, 150);
```

Garis lurus dibuat dengan menghubungkan dua titik bersama-sama. Fungsi line mengambil koordinat x, y dari dua titik dan menggabungkannya menjadi garis lurus;

```
// sample syntax
line (x1, y1, x2, y2);
// draws a line from x:100, y:150 to x:300, y:250
line (100, 150, 300, 250);
```

Bentuk 2D sederhana lainnya dapat dibuat dengan memplot dan menghubungkan lebih banyak titik. Segitiga, misalnya, dibuat oleh fungsi triangle, yang menghubungkan tiga set titik koordinat x, y.

```
// 3 sets of coordinate points create a triangle
triangle (x1, y1, x2, y2, x3, y3);
// connect the points (100, 50), (25, 120), (150, 140) into a
triangle triangle (100, 50, 25, 120, 150, 140);
```

Elips dan persegi panjang digambar dengan menetapkan satu koordinat sebagai lokasi awal bentuk (misalnya, titik pusat elips atau titik sudut persegi panjang) dan menggunakan dua nilai lain untuk menentukan lebar dan tinggi bentuk.
Fungsi ellipse() menggambar elips dan lingkaran:

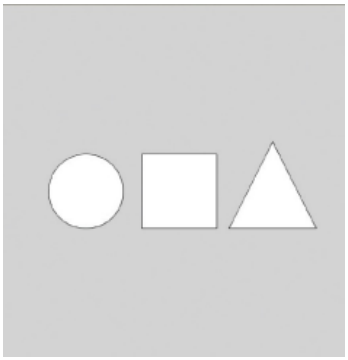
```
// set the x, y, width and height values
ellipse (x, y, w, h);
// draw an ellipse at x:70, y:60, width:40, height:30
ellipse (70, 60, 40, 30);
```

Fungsi rect() menggambar persegi panjang dan persegi:

```
// set the x, y, width and height values of the rectangle
rect (x, y, w, h);
rect (150, 200, 50, 70);
rect (150, 200, 100, 100); // a 100 pixel square
```

Lingkaran dan persegi digambar menggunakan angka lebar dan tinggi yang sama dengan fungsi

ellipse dan rect.



Contoh tipe dasar bentuk yang dapat dibuat oleh desainer dengan fungsi menggambar bentuk. Tipe bentuk ini adalah unit visual dasar dari gambar komputasional.

Warna: Isian dan Goresan

Warna, seperti banyak hal dalam lingkungan komputasional, didefinisikan secara tepat sebagai nilai angka. Kumpulan nilai warna yang paling sederhana adalah 256 corak abu-abu.

Nilai skala abu-abu, yang bergerak dari hitam ke putih, didefinisikan oleh satu rentang nilai angka dari 0 hingga 255, di mana 0 adalah “hitam,” 255 adalah “putih.” Semua nilai di antaranya adalah corak abu-abu, yang bergerak dari abu-abu gelap (angka terendah) ke abu-abu muda (nilai tertinggi).

Angka skala abu-abu dapat digunakan untuk mengubah warna elemen di layar menjadi corak hitam dan putih tertentu. Warna yang digunakan sebagai isian dan garis bentuk diatur dengan fungsi fill dan stroke. Warna latar belakang kanvas diatur menggunakan fungsi background.

```
// sets the background of the page to white
Background (255);
// sets the outline (stroke) color to black
Stroke (0);
fill (125); // sets the fill color to mid-gray
```

Rentang warna dan corak yang jauh lebih luas tersedia dengan menggunakan rentang warna RGB, yang dibuat dengan menggunakan tiga nilai angka, masing-masing antara 0 dan 255, untuk mewakili jumlah warna merah, hijau, dan biru. Seperti nilai skala abu-abu, nilai RGB dapat diterapkan untuk mengubah warna halaman, isian, atau garis luar.

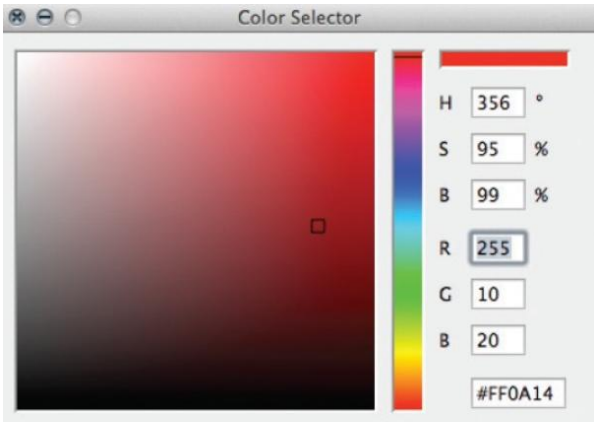
```
// syntax examples:
background (r, g, b);
fill (r, g, b);
stroke (r, g, b);
```

Dengan menggunakan angka untuk "mencampur" jumlah warna merah, hijau, dan biru secara digital, seorang desainer dapat membuat corak warna apa pun dalam spektrum RGB. Merah terang, misalnya, didefinisikan oleh nilai RGB (255, 0, 0), yang mencampur jumlah warna merah "terbanyak" (255) tanpa warna hijau atau biru (0, 0).

Warna lain dalam spektrum RGB dapat dicampur menggunakan kombinasi nilai RGB ini. Pemrosesan mencakup pemilih warna untuk membantu mengatur angka RGB untuk warna apa pun. Setelah nilai isian atau goresan ditetapkan, nilai tersebut diterapkan ke semua bentuk dan garis berikutnya, kecuali jika diubah lebih jauh ke bawah.

```
fill (13, 184, 216); // select blue as fill
ellipse (100, 200, 40, 40); // draw blue circle
fill (214, 104, 13); // change fill to orange
rect (300, 200, 50, 50); // draw orange square
```

Nilai alfa adalah nilai persentase (0 hingga 100) yang dapat ditambahkan di akhir nilai RGB sebagai angka keempat untuk mengubah tingkat transparansi warna.



Pemilih Warna dalam Pemrosesan memungkinkan nilai RGB (merah, hijau, biru) dari warna apa pun ditemukan. Ada alat serupa yang tersedia di sebagian besar perangkat lunak penyuntingan gambar.

```
// Syntax example:
fill (red, green, blue, ALPHA);

// bright red with a 50% alpha value
fill (255, 0, 0, 50);
```

Atribut visual lain dari suatu bentuk dapat disesuaikan, termasuk lebar garis atau garis luar:

```
strokeWeight (5); // mengatur ketebalan garis luar
noStroke; // menghapus semua garis luar
noFill; // menghapus warna isian
```

Menambahkan variabel

Setiap contoh gambar sejauh ini telah menggunakan nilai angka tetap dalam fungsi gambar ellipse() atau rect() untuk mengatur lokasi dan ukuran setiap bentuk. Namun, alih-alih nilai angka tetap, menggunakan variabel angka dalam fungsi gambar memberikan bentuk lebih banyak potensi untuk menjadi fleksibel dan dapat diubah. Setelah variabel digunakan dalam suatu fungsi untuk menggambar bentuk, mengubah variabel tersebut akan secara otomatis mengubah bentuk tersebut. Setelah variabel angka dibuat, variabel tersebut dapat digunakan sebagai pengganti angka tetap dalam fungsi bentuk.

```
float xpos = 20;
float ypos = 10;
ellipse (xpos, ypos, 10, 10);
```

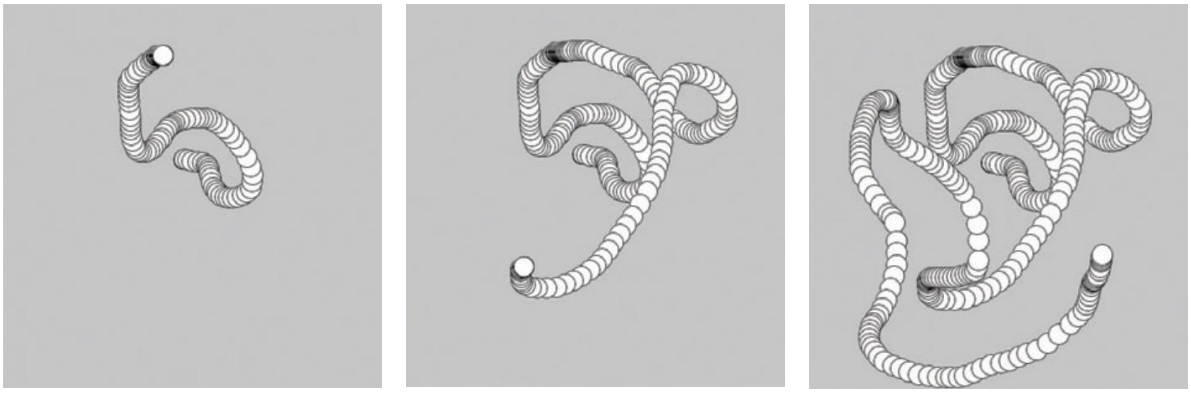
Mengubah atau menghitung ulang nilai variabel akan mengubah fungsi gambar yang digunakan.

Menulis kode untuk mengubah variabel dan menggambar bentuk di dalam elemen draw yang berulang dari suatu program akan menggambar ulang bentuk yang dihasilkan, dan penggambaran ulang yang berulang tersebut dapat membuat gerakan animasi sederhana. Misalnya, terus-menerus mengubah variabel yang digunakan untuk mengatur posisi x suatu bentuk akan memungkinkan bentuk tersebut bergerak secara horizontal melintasi layar.

```
float xpos = 10;
void draw {
ellipse (xpos, 50, 5, 5);
xpos += 2;
}
Mouse Position: mouseX, mouseY
```

Nilai angka variabel dapat diubah oleh data dari sumber lain seperti input pengguna. Variabel sistem mouseX dan mouseY adalah variabel yang sangat sederhana dan berguna yang mengambil lokasi layar mouse saat bergerak melintasi layar. Variabel tersebut dapat digunakan untuk membuat objek yang merespons gerakan mouse pengguna. Contoh berikut akan berulang kali menggambar lingkaran di lokasi mouse.

```
void draw {
elips (mouseX,mouseY, 50, 50);
}
```



Bentuk yang digambar menggunakan variabel sistem mouseX dan mouseY untuk mengatur lokasi x dan y akan mengikuti gerakan mouse saat pengguna menggerakkannya di layar.

Pengulangan dan Menggambar dengan Perulangan "For"

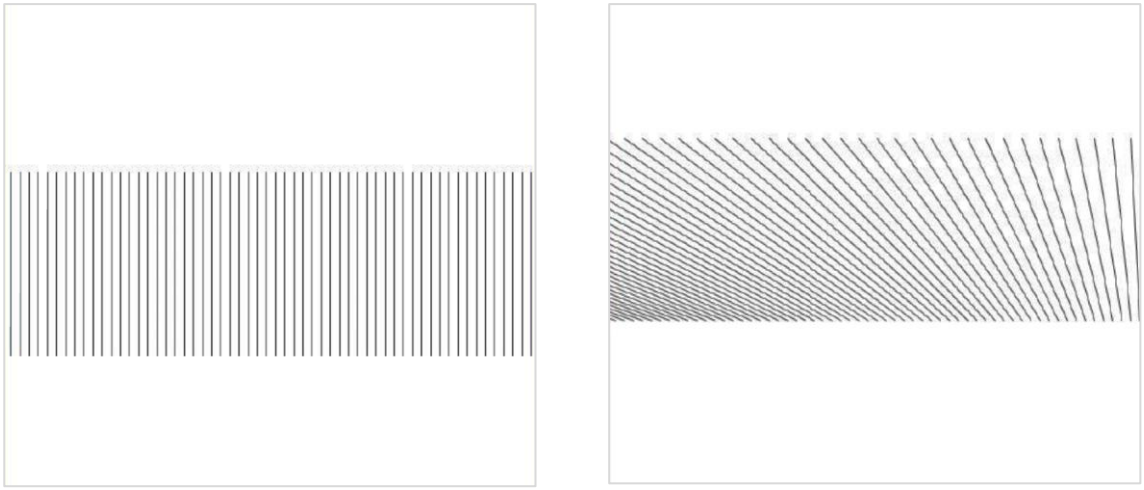
Perulangan "For" mengulang baris kode dan membuat urutan angka yang dapat diprediksi yang dapat digunakan untuk mengulang grafik dengan cara yang terstruktur dan teratur serta membuat bentuk dan pola yang berulang. Fungsi menggambar tunggal di dalam perulangan "for" akan diulang beberapa kali dan membuat beberapa gambar.

Dalam contoh ini, variabel "counter" (i) digunakan untuk menggambar 40 garis, menggunakan nilai "i" untuk mengubah posisi x setiap garis secara berurutan. Karena nilai "i" dihitung dari 0 hingga 40, maka posisi x setiap garis juga bertambah dari 0 hingga 40.

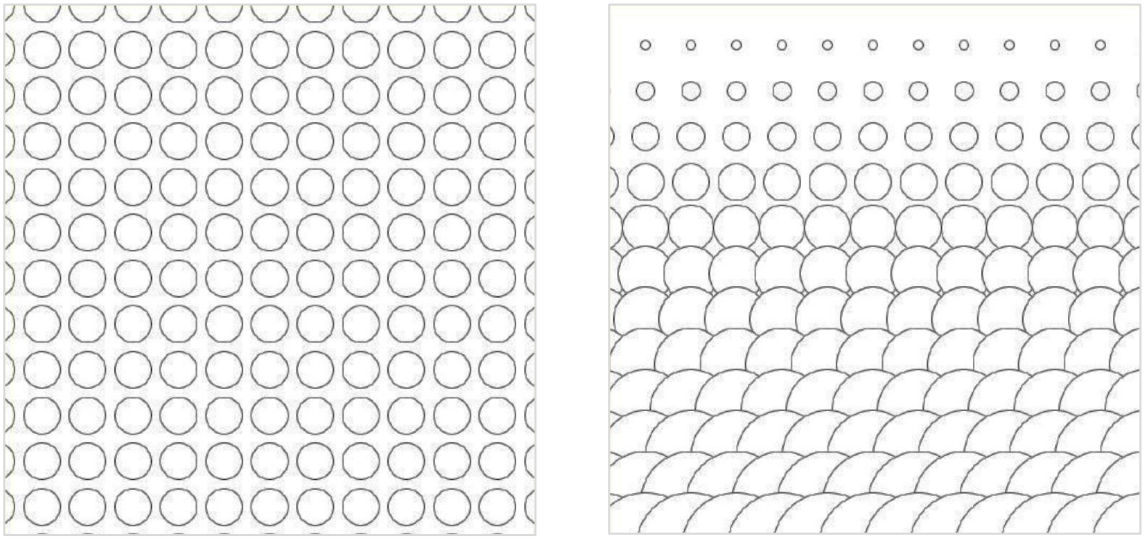
```
//as i changes, so does the x position of the both ends of the line
for (int i=0; i<40; i++)
{
  line (i, 10, i, 100);
}
```

Penyesuaian kecil pada kode fungsi baris akan mengubah jarak antar baris.

```
// draw each line 10 pixels apart
for (int i=0; i<40; i++)
{
  line (i*10, 10, i*10, 100);
}
```



Menggambar bentuk dengan perulangan "for" menciptakan urutan bentuk yang teratur. Dalam hal ini, garis dibuat dalam urutan yang teratur. Mengubah salah satu nilai angka misalnya, lokasi x di bagian atas garis menciptakan pola ritmis yang sederhana.



Menyusun dua loop "for", satu di dalam yang lain, digunakan untuk membuat "kotak" bentuk. Penyesuaian sederhana pada ukuran bentuk mengubah pola bentuk keseluruhan

Pola Acak

Berbeda dengan urutan angka yang dibuat oleh loop “for”, fungsi pemrograman lain dapat digunakan untuk menghasilkan urutan angka yang kurang dapat diprediksi yang dapat diterapkan untuk menciptakan variasi visual.

Dalam Processing, fungsi random() digunakan untuk menghasilkan angka acak dari dalam rentang tertentu. Rentang angka dinyatakan dalam tanda kurung baik sebagai nilai tunggal (maksimum) atau sebagai dua angka (minimum dan maksimum). Jika satu angka digunakan, 0 digunakan sebagai angka minimum default.

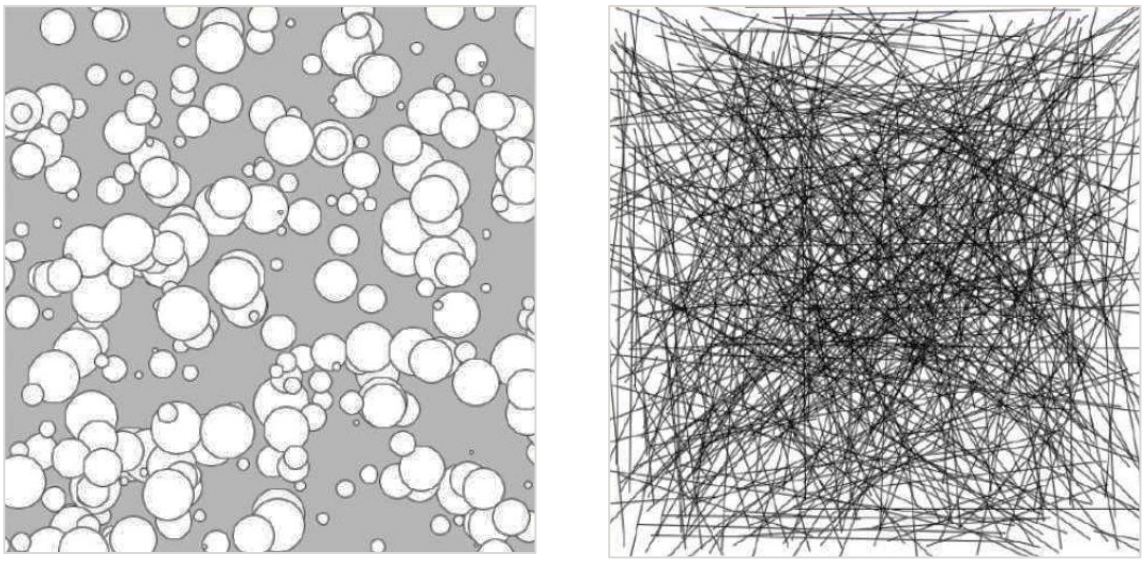
```
// generates a number between 0 and “max”
random(max);
// generates a number between 0 and 20
random (20);
random (min, max);
// generates a number between 10 and 20
random (10, 20);
// generates a number between –40 and 40
random (–40, 40);
```

Setiap kali fungsi random() dipanggil, fungsi tersebut menghasilkan nilai baru. Menerapkan angka acak ke variabel memungkinkan nilai variabel tersebut menghasilkan nilai angka yang tidak dapat diprediksi. Nilai acak dapat digunakan untuk menghasilkan/mengubah properti numerik apa pun (misalnya, ukuran, posisi, warna, dll.). Contoh berikut menggambar lingkaran di lokasi x, y acak setiap kali dijalankan.

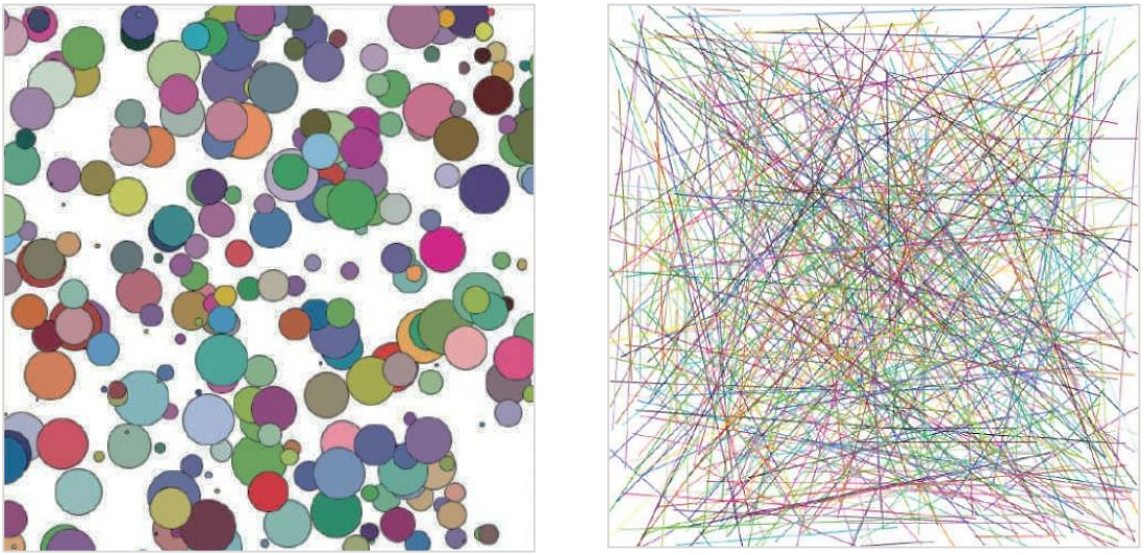
```
float x = random (400); // create a random x value
float y = random (200); // create a random y value
// draw a circle at the x, y location
ellipse (x, y, 20, 20);
```

Fungsi acak yang dikombinasikan dengan loop “for” dapat digunakan untuk menghasilkan banyak bentuk acak. Contoh berikut menghasilkan 100 lingkaran yang digambar di lokasi acak dengan ukuran acak. Setiap iterasi loop “for” menghasilkan nilai baru dan menggambar bentuk baru dengan ukuran dan lokasi baru.

```
// use a "for" loop to repeat the random drawing of a circle
for (int i = 0; i < 100; i++) {
    float x = random (300);
    float y = random (400);
    float w = random (5, 50);
    ellipse (x, y, w, w);
}
```



Garis dan bentuk acak dapat dibuat dengan cepat dengan mengulangi fungsi random untuk menggambar bentuk di lokasi baru dan dalam ukuran berbeda.



Fungsi random() menciptakan angka-angka yang tidak dapat diprediksi, yang dapat diterapkan pada atribut bentuk apa pun yang terlihat, termasuk warna menggunakan angka acak untuk nilai merah, hijau, dan biru.

Terjemahkan dan Putar

Memplot bentuk pada kisi layar dapat dilakukan dengan cara biasa. Misalnya

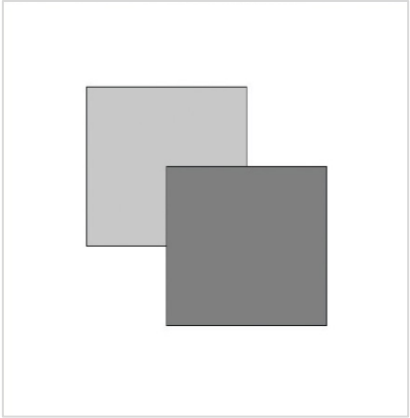
```
// draw a shape at x:10 and y:20
ellipse (10, 20, 5, 5);
```

Namun, gambar dalam pemrosesan juga dapat "diterjemahkan." Ini berarti bahwa alih-alih memindahkan bentuk ke koordinat layar tertentu, penerjemahan memindahkan kisi koordinat layar. Fungsi translate() memiliki efek menggeser titik asal (0,0) ke tempat baru. Efek "memindahkan kertas grafik" ini berarti bahwa posisi semua bentuk pada kisi digeser. Fungsi translate() menggunakan dua nilai untuk menggeser kisi koordinat layar.

```
translate(x, y); // contoh sintaks
```

Contoh berikut menggambar bentuk sebelum kisi layar diterjemahkan dan setelahnya, mengilustrasikan cara kerja fungsi:

```
// draw a shape
ellipse (10, 20, 5, 5);
// translate the grid of the screen
translate (50, 50);
// draw a shape on the translated screen
ellipse (10, 20, 5, 5);
```



Translasi adalah cara memindahkan kisi koordinat panggung, yang berarti bentuk yang digambar setelah translasi terjadi akan digeser ke tempat baru di layar.

Meskipun kode untuk menggambar kedua bentuk itu sama, bentuk kedua digambar setelah kisi gambar digeser 50 piksel ke atas dan ke bawah. Ini memiliki efek visual menggambar bentuk kedua lebih jauh ke bawah layar, pada koordinat 60, 70.

Fungsi translasi sering digunakan dengan fungsi pushMatrix() dan popMatrix(). Fungsi ini digunakan untuk menyimpan dan menyetel ulang kisi koordinat yang bergeser, sehingga area gambar disetel ulang setelah translasi terjadi.

Dalam contoh berikut, elips digambar pada kisi yang "ditranslasi". Fungsi pushMatrix() dan popMatrix() digunakan untuk menyimpan dan menyetel ulang kisi, sehingga persegi panjang digambar pada kisi koordinat normal.

```
rect (10, 20, 5, 5);
// save the current grid position
pushMatrix ();
translate (50, 50);
// draw a shape with translation
ellipse (10, 20, 5, 5);
// return back to the normal grid
popMatrix ();
// draw a shape on the "normal" grid
rect (30, 30, 5, 5);
```

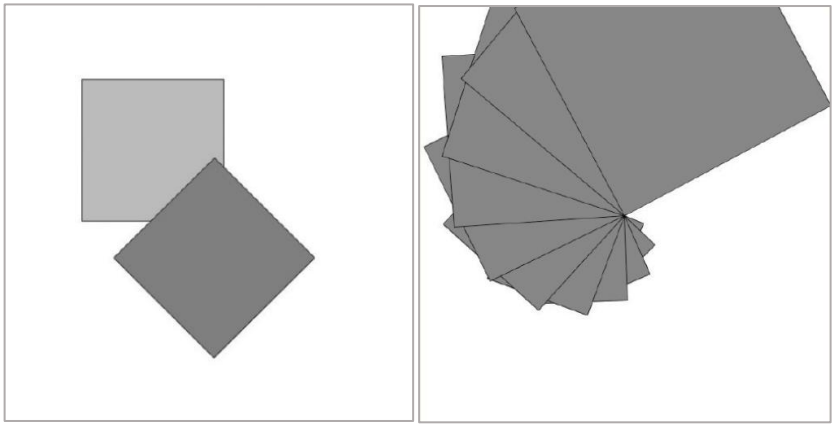
Penggunaan translate() paling berguna saat memutar bentuk dalam Processing. Memutar bentuk dilakukan dengan menggunakan perintah rotate lalu menggambar bentuk:

```
rotate (radians (45));
// putar kisi sebesar 45 derajat
```

```
rect (0, 0, 20, 20); // menggambar bentuk
```

Fungsi rotate memutar kisi tempat bentuk digambar, jadi saat bentuk diputar, ia melakukannya di sekitar titik asal kisi layar x:0, y:0. Untuk menempatkan bentuk di tengah layar dan membuatnya berputar di sekitar titik tengahnya, layar harus diterjemahkan terlebih dahulu. Urutan perintahnya adalah: 1) menerjemahkan kisi koordinat; 2) memutar kisi; 3) menggambar bentuk.
Berikut ini adalah contoh cara kerjanya.

```
pushMatrix();
translate (50, 50); // move the grid
rotate (radians (45)); // rotate the grid
rect (0, 0, 10, 10); // draw the shape
popMatrix();
```



Fungsi terjemahan menggeser kisi panggung, yang memungkinkan bentuk diputar di sekitar titik tengah. Serangkaian bentuk yang diputar dapat membuat spiral sederhana.

Perhatikan bagaimana fungsi bentuk memplotnya pada 0, 0. Karena layar telah diterjemahkan, 0, 0 sekarang tampak berada di tengah layar pada (50, 50). Ini semua dilakukan di dalam fungsi pushMatrix() dan popMatrix() sehingga gambar berikutnya akan ditempatkan pada kisi seperti biasa.

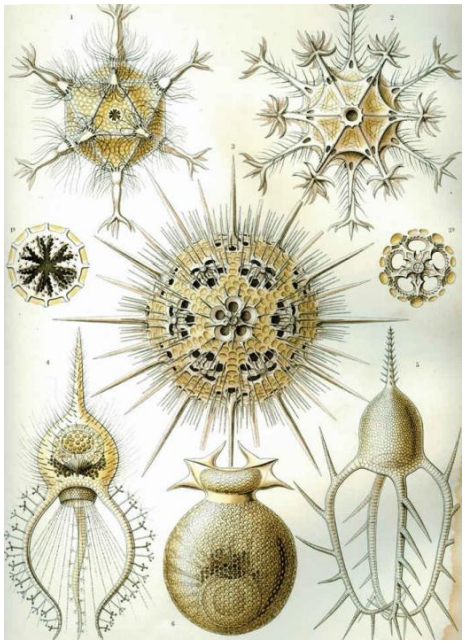
BAB 3
PERTUMBUHAN DAN BENTUK

"Keharmonisan dunia terwujud dalam Bentuk dan Angka."
D'Arcy Thompson

3.1 ALAM SEBAGAI INSPIRASI

Lingkungan dunia fisik alami dan lingkungan dunia digital interaktif memiliki banyak kesamaan. Banyak konsep dasar yang mengatur pertumbuhan dan perkembangan kehidupan organik menginspirasi bentuk dan gerakan lingkungan "virtual". Kode memberi "kehidupan" pada objek digital, menambahkan perilaku dan karakteristik individual pada gambar yang tidak bernyawa.

Bunga Matahari dan Kawanan Burung
Bentuk dan bentuk yang ditemukan di alam misalnya, dalam spiral bunga matahari atau formasi kawanan burung mengandung jenis ritme dan keindahan organiknya sendiri yang telah menjadi sumber inspirasi penting bagi para seniman dan desainer dari generasi ke generasi.



Ilustrasi Biologi Ernst Haeckel
Ernst Haeckel adalah seorang ahli biologi, naturalis, dan seniman Jerman abad ke-19 yang mempelajari asal usul silsilah banyak bentuk kehidupan. Ilustrasi terperinci dalam bukunya Art Forms of Nature (1899) menjembatani kesenjangan antara sains dan seni dan terbukti menjadi sumber yang sangat berpengaruh dalam arsitektur dan desain.

Desainer dan programmer menggunakan kode untuk menciptakan "bentuk kehidupan" digital mereka sendiri: Garis dan bentuk muncul dalam batas-batas dunia yang dihasilkan secara komputasional. Simulasi virtual dari kekuatan lingkungan kehidupan nyata pertumbuhan, gerakan, interaksi, dan pembusukan mendorong bentuk-bentuk tersebut. Hubungan antara dunia virtual dan digital sedemikian rupa sehingga desainer biasanya

meminjam bahasa dunia alami untuk menggambarkan atribut dan karakteristik kreasi interaktif mereka. Layar adalah “lingkungan”; objek sering disebut memiliki “kehidupan” atau “kesehatan” atau kemampuan untuk “menelurkan” bentuk baru. Objek interaktif terprogram diberi "perilaku"; objek tersebut "tumbuh", memiliki "struktur", dan menikmati atribut gerakan dan kehidupan digital.

Oleh karena itu, desainer komputasional sering kali melihat melampaui lanskap desain grafis tradisional dan beralih ke alam sebagai sumber inspirasi dan referensi kreatif. Perilaku, gerakan, dan tekstur yang terjadi secara alami memberikan banyak informasi visual yang memicu ide. Pengamatan dan dokumentasinya dapat menjadi bagian penting dari proses penelitian dan pengembangan visual dan konseptual. Bentuk dan struktur yang ditemukan pada tanaman, pohon, pakis, dan jamur; lengkungan dan lekukan daun dan kerang; dan gerakan yang diamati pada kawanan burung atau sekumpulan ikan dapat memberikan batu loncatan kreatif yang darinya konsep digital interaktif berkembang.

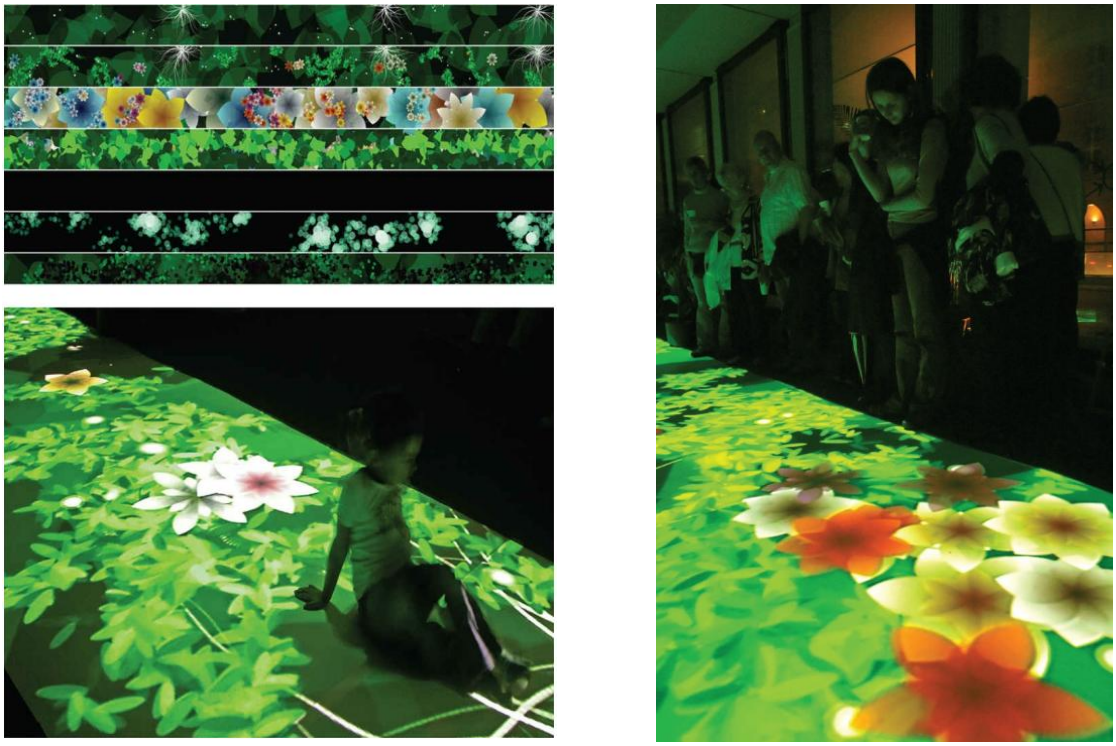
Pengamatan pribadi bukanlah satu-satunya titik awal yang memungkinkan; diagram ilmiah, foto, dan ilustrasi juga memberikan inspirasi untuk ide-ide interaktif dan generatif. Uraian para ilmuwan historis dan kontemporer yang berusaha mengatur bentuk-bentuk materi botani dan biologis menyediakan sumber informasi visual yang kaya yang terus menginspirasi dan menginformasikan karya banyak seniman digital. Gambar-gambar ahli biologi Jerman Ernst Haeckel dalam bukunya Art Forms of Nature (1899) dan ide-ide, diagram, dan pengamatan ahli biologi D'Arcy Wentworth Thompson dalam karyanya On Growth and Form (1917), misalnya, telah mengilhami karya-karya desainer berbasis kode kontemporer.

Dunia alam tidak hanya menyediakan sumber inspirasi visual yang penting untuk desain interaktif; tetapi juga memberikan inspirasi sebagai sistem pertumbuhan yang terus berkembang dan terorganisir yang menghasilkan bentuk, pola, dan perilaku yang kompleks. Alam, seperti kode, bekerja menurut serangkaian pola dan aturan yang rumit dan terdefinisi dengan baik yang mengatur cara tanaman, bunga, pohon, burung, ikan, serangga, dan lainnya berkembang. Memahami aturan, struktur, dan pola pertumbuhan kehidupan organik dapat menginspirasi dan memberi informasi kepada desainer digital dan komputasional dengan menunjukkan bagaimana "templat" aturan dan instruksi sederhana yang dapat diulang dapat diterapkan untuk menghasilkan bentuk yang rumit.

Sama seperti benih tanaman yang mengandung kode genetik untuk menentukan bentuk, pola, dan struktur setiap spesies (yaitu, jumlah, ukuran, dan frekuensi cabang, daun, dan bunga), demikian pula serangkaian instruksi pemrograman bertindak sebagai jenis "benih digital" yang digunakan untuk menentukan aturan komputasional yang darinya bentuk-bentuk digital muncul dan tumbuh. Satu jenis benih tunggal menghasilkan serangkaian tanaman yang, meskipun memiliki kesamaan generik, tumbuh dengan serangkaian karakteristik dan variasi individualnya sendiri yang halus; tidak ada dua bunga atau pohon yang persis identik.

Demikian pula, variasi yang tertanam dalam kode digital berarti bahwa setiap versi kode dapat menghasilkan hasil yang sangat bervariasi. Konsep sistem pertumbuhan yang berulang dan menghasilkan sendiri yang darinya berbagai macam gambar dapat "tumbuh" dan muncul memberikan sumber inspirasi yang penting dan kaya bagi para seniman dan desainer

yang menggunakan kode untuk menggambar.



Studio ART+COM Taman Ganda

Pertumbuhan "taman" digital berskala besar yang berisi tanaman dan bunga diatur oleh instruksi komputasi yang meniru bentuk dan pola organik bunga, daun, dan daun.

3.2 MENGGAMBAR SEBAGAI PERTUMBUHAN

Gambar yang dihasilkan kode, seperti kehidupan organik, tidak muncul sekaligus, tetapi berkembang secara bertahap seiring waktu. Prosesnya mirip dengan menanam benih dan melihatnya tumbuh: garis, warna, dan bentuk muncul secara bertahap. Mereka menyebar di kanvas layar seperti bentuk organik digital. Evolusi gambar yang digerakkan kode merupakan bagian penting dari pengalaman dan proses penggunaan kode, yang membuka cara-cara baru yang menarik dalam menghasilkan karya.

Alat gambar perangkat lunak tradisional memungkinkan desainer untuk membuat keputusan desain yang dipertimbangkan dengan cermat, di mana detail komposisi dapat dikontrol dan diedit dengan cermat. Tidak seperti karya digital konvensional, gambar yang dibuat oleh kode tunduk pada tingkat ketidakpastian yang jauh lebih luas. Aturan komputasi menentukan kondisi keseluruhan untuk pengembangan gambar tanpa secara tepat mendefinisikan hasil yang tepat. Atribut visual individual (misalnya, arah, gerakan, bentuk, warna, atau ukuran garis tertentu) dapat dibuat sebagai elemen variabel, yang nilainya dihasilkan dan diubah saat program berjalan dan gambar tumbuh. Kode menciptakan visual yang berkembang dengan cara yang tak terduga, menghasilkan hasil yang dapat mengejutkan baik bagi pemirsa maupun programmer.

Sama seperti di alam tidak ada dua bentuk yang identik, kode memungkinkan desainer untuk menghasilkan puluhan atau ratusan variasi di sekitar bentuk yang sama. Daripada

memiliki visi desain atau hasil yang tetap dalam pikiran, desainer/programmer dapat memulai dengan konsep yang merupakan "benih": instruksi yang membentuk dasar program. Konsep tersebut dapat didasarkan pada pengamatan bentuk, ritme, atau pola alami (misalnya, "garis bercabang menumbuhkan dua garis lagi") atau dapat lebih abstrak.

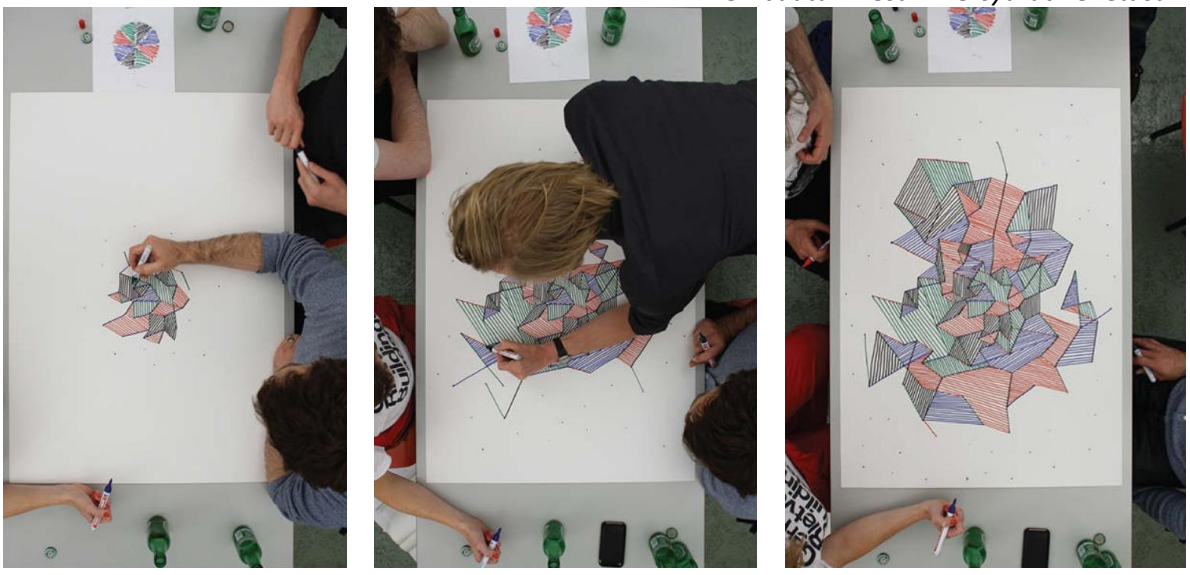
Dengan mengulang instruksi menggambar berulang kali, bentuk organik dapat tumbuh. Elemen variabel kecil atau "acak" dapat ditambahkan ke dalam kalkulasi yang secara halus mengubah kualitas setiap garis atau bentuk, memberikan masing-masing nuansa naturalistiknya sendiri. Mengubah, atau menambahkan variasi ke, konsep dasar akan menghasilkan berbagai hasil yang menarik dan mengejutkan di sekitar tema inti. Desainer dapat memodifikasi aturan tanpa sepenuhnya mengetahui efek perubahan yang mungkin terjadi pada hasil saat kode dijalankan; program menggambar yang sama dapat menghasilkan berbagai hasil visual baru dan tak terduga setiap saat.

CONTOH

Kelompok Desain Bersyarat

Kelompok Desain Bersyarat adalah kolektif desain grafis eksperimental berbasis penelitian yang mengeksplorasi cara membuat gambar dan artefak dengan menggunakan kondisi dan aturan main yang ditetapkan sendiri untuk menciptakan hasil visual organik. Hanya menggunakan bahan nondigital (pena, cat, tanah liat, selotip, dll.), kelompok peserta terlibat dalam aktivitas bermain, bergiliran menambahkan sesuatu ke gambar sesuai dengan serangkaian "aturan" yang telah ditetapkan sebelumnya. Aturan suatu aktivitas memberikan batasan yang jelas pada proses desain tetapi juga memungkinkan peserta untuk sepenuhnya dan bebas mengeksplorasi kemungkinan visual dalam batasan permainan tersebut. Hasilnya adalah serangkaian desain "generatif" yang muncul dari proses peluang, waktu, dan instruksi bersama.

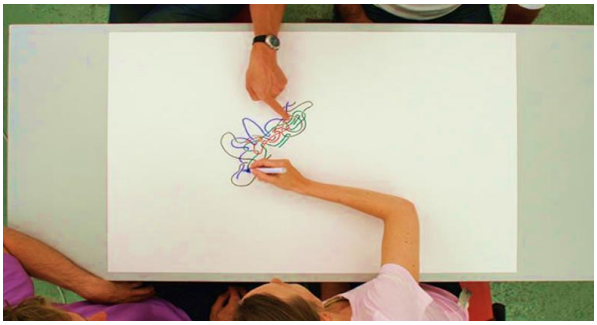
Pembuatan Desain Bersyarat Penetasan

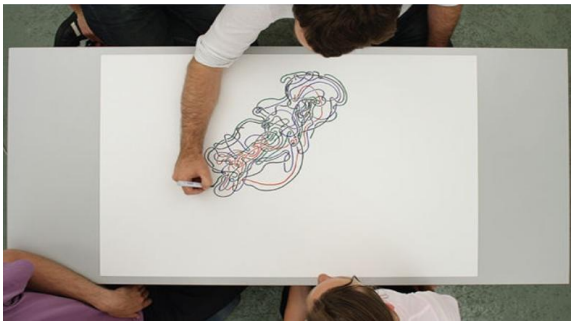


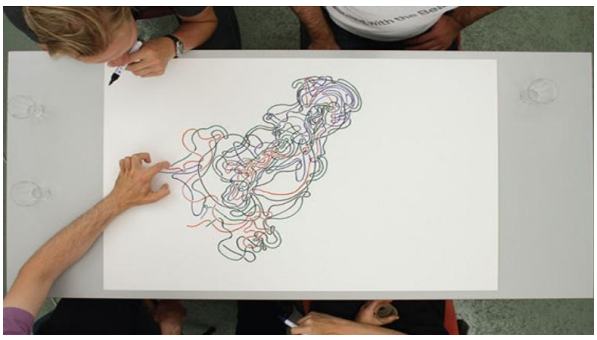
Pengembangan gambar "generatif" semacam ini, yang diatur oleh aturan dan juga "terbuka" untuk permainan, keacakan, eksplorasi, dan interpretasi, mencerminkan proses, konsep, dan prosedur yang digunakan dalam kode pemrograman. Meskipun bahannya fisik

dan bukan digital, dan garis-garisnya digambar dengan tangan, bukan oleh komputer, kemiripannya jelas. Aturan tertulis bertindak sebagai "program" dan peserta bertindak sebagai komputer, menerjemahkan dan menjalankan instruksi.

Memberi nomor pada instruksi, "bergantian," dan memberikan parameter yang tepat untuk gambar (misalnya, menentukan satu warna per orang atau menyatakan bahwa lingkaran harus berukuran antara 2 cm dan 5 cm) memastikan bahwa tindakan kreatif terjadi dalam parameter yang ditentukan dan dalam urutan yang teratur seperti yang terjadi dalam program komputer. Kreativitas individu peserta yang menggambar "dalam aturan" aktivitas mensimulasikan elemen variabilitas dan peluang yang tertanam dalam proses kode. Proses tersebut menetapkan aturan yang dilakukan dengan waktu dan peluang sebagai variabel. Hasil dari proses tersebut sangat bervariasi dan acak; mereka menerjemahkan konsep program komputer ke dunia fisik spidol, pena, dan kertas.







Pembuatan Simpul Desain Bersyarat
Sistem menggambar berbasis aturan yang dirancang oleh Desain Bersyarat memungkinkan peserta untuk terlibat dalam proses pembuatan gambar di mana gambar muncul sebagai pengalaman bermain bersama. Dengan menggabungkan aturan yang ditetapkan dengan pilihan individu, sistem ini mencerminkan proses sistem komputasional tempat bentuk muncul dan berkembang.

3.3 BENTUK ORGANIK SPIRAL DAN GELOMBANG

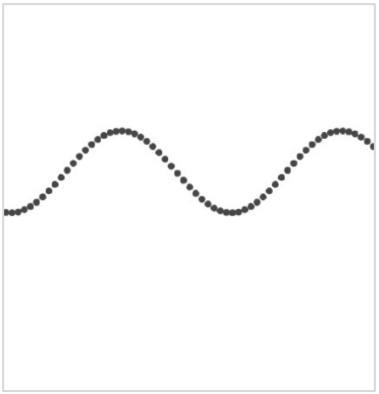
Pola organik tidak kacau atau acak; bahkan bentuk organik yang tampak paling rumit pun mengandung urutan pola dan struktur yang berulang secara teratur.

Pergerakan burung, bentuk daun, atau spiral cangkang siput dapat dijelaskan secara matematis melalui urutan angka. Hubungan antara alam dan matematika telah memberi informasi dan menginspirasi banyak generasi seniman dan desainer, dan telah lama dieksplorasi oleh matematikawan, ilmuwan, dan filsuf yang ingin menggambarkan dan mengungkap keindahan dan kompleksitas numerik di balik bentuk-bentuk alam. Contoh pola angka organik yang sangat terkenal adalah urutan Fibonacci.

Pola angka Fibonacci yang diamati oleh seorang matematikawan pada abad ketiga

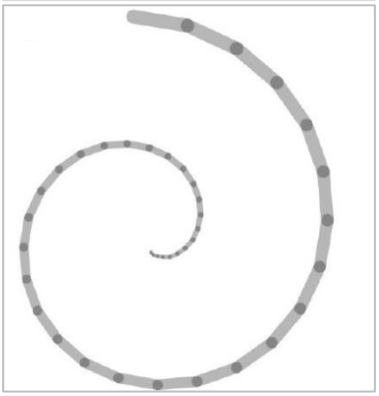
belas dihasilkan dari perhitungan numerik di mana 2 angka terakhir dari urutan tersebut ditambahkan untuk memperoleh angka berikutnya: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55 . . . dan seterusnya. Hubungan antara masing-masing angka yang tampaknya acak ini digunakan untuk menghasilkan Spiral Fibonacci (dengan "rasio emas"), sebuah pola yang dapat diamati dalam banyak bentuk organik yang terjadi secara alami, termasuk kerang laut, tanaman bercabang, biji, daun, dan susunan kelopak bunga.

Spiral Kerang
Bentuk spiral yang ditemukan pada kerang merupakan contoh pola angka Fibonacci yang terjadi secara alami.



Gelombang Sinus

Gelombang sinus adalah bentuk yang dihasilkan secara komputasional. Nilai osilasi yang "memantul" antara angka minimum dan maksimum menghasilkan bentuk gelombang yang teratur dan mengalir. Jenis gelombang ini dapat diterapkan untuk menggambar garis atau membuat gerakan**.**



Contoh Bentuk Spiral yang Dihasilkan Dengan Menghitung Ulang Sudut dan Panjang Garis**.**

Urutan angka lain yang dihasilkan oleh fungsi matematika formal serupa dapat digunakan untuk menghasilkan bentuk dan garis yang menarik secara estetika yang mencerminkan gerakan yang ditemukan di lingkungan alami. Contohnya adalah fungsi "sinus" matematika. Perhitungan sinus, yang awalnya digunakan dalam perhitungan Pythagoras untuk menghitung sifat-sifat segitiga, juga menghasilkan pola nilai angka (berosilasi) yang, ketika diplot sepanjang garis, menghasilkan bentuk dan wujud gelombang yang tampak organik.

Program komputasi sering kali menggambar dengan mengulangi prosedur untuk menggerakkan objek (sebagai "pena" virtual) melintasi layar; "jejak" jalur gerakan meninggalkan garis di layar. Perhitungan matematika berulang yang digunakan untuk memindahkan dan memposisikan ulang lokasi "pena" menciptakan lengkungan, kurva, atau

pusaran yang tampak organik. Fungsi menggambar yang, misalnya, berulang kali meningkatkan sudut untuk setiap lokasi "pena" akan menggambar garis yang secara bertahap berputar ke dalam. Demikian pula, angka yang dihasilkan oleh perhitungan matematika seperti deret Fibonacci atau fungsi sinus dapat digunakan untuk membuat gambar garis mengalir yang mencerminkan keanggunan anggun bentuk daun atau kelopak. Pola angka yang diterapkan pada grafik yang dihasilkan kode menciptakan ritme dan pola yang menyenangkan secara visual yang mencerminkan bentuk dan gerakan lingkungan alam.

3.4 MODEL MATEMATIKA KOMPLEKS

Baru-baru ini, sistem matematika berbasis komputer telah dikembangkan yang secara akurat dan realistis mensimulasikan pola berulang dari pertumbuhan botani yang kompleks (misalnya, tanaman, gulma, ragi, jamur). Sistem ini diterapkan untuk studi ilmiah dalam biologi sistem dan juga digunakan secara luas dalam teknologi permainan komputer untuk menghasilkan lanskap dan medan lingkungan yang realistis.

Meskipun bersifat teknis dan agak khusus, sistem pemodelan komputer menyoroti prinsip-prinsip penting pertumbuhan organik yang dapat direplikasi secara komputasi, memberikan ide dan inspirasi bagi para desainer. Berikut ini adalah ikhtisar singkat dari beberapa konsep di balik sistem utama untuk pertumbuhan organik yang dihasilkan komputer. Meskipun tidak perlu diikuti secara tepat, mereka mengungkapkan beberapa fitur penting dan menarik.

Rekursi

Struktur pemrograman dan struktur organik memiliki kecenderungan yang sama terhadap rekursi. Rekursi adalah proses yang merujuk diri sendiri dan mengulangi diri sendiri; bagian dari proses tersebut mencakup instruksi kembali ke aslinya, yang menciptakan lingkaran melingkar (yang berpotensi tak terbatas). Ada lelucon terkenal tentang definisi kamus yang menggambarkan konsep tersebut dalam tindakan: Rekursi (kata benda) Lihat “Rekursi.”

Struktur yang menampilkan sifat rekursi yang terlihat adalah struktur yang menunjukkan karakteristik “kemiripan diri”; yaitu, setiap bagian kecil dari bentuknya sama persis atau kurang lebih sama dengan bentuk keseluruhannya. Oleh karena itu, struktur rekursif paling sering muncul dalam bentuk yang berulang sendiri dari bentuk organik, seperti pohon dan batu. Pakis adalah contoh bentuk rekursif yang sangat bagus; setiap daun pakis yang kecil adalah versi yang mirip dan lebih kecil dari bentuk pakis secara keseluruhan.

Oleh karena itu, fungsi menggambar rekursif yang ditulis dalam kode adalah fungsi yang berisi instruksi untuk menggambar ulang dirinya sendiri, sehingga menciptakan pola yang berulang sendiri (tanpa batas). Ada hubungan visual bersama antara elemen organik terstruktur dari dunia alami dan grafik yang dibuat dari proses pemrograman rekursif (berulang). Keduanya menampilkan kesamaan diri secara visual.

Bentuk rekursif sering ditemukan di alam, terutama pada tumbuhan. Misalnya, pada tumbuhan pakis, keseluruhan bentuknya merupakan replika besar dari setiap bagian kecilnya.

Bunga Robert Hodgin

Proses rekursif dapat digunakan dalam kode untuk membuat struktur percabangan dengan bentuk seperti tanaman. Struktur percabangan pohon dibuat dengan aturan "rekursif" yang mengulang instruksi sederhana untuk setiap cabang guna membuat sejumlah cabang yang lebih kecil. Dimulai dengan satu garis (cabang), aturan diulang, dan bentuk pohon pun dibuat. Ini adalah jenis geometri "fraktal". Perubahan kecil pada cara cabang muncul mengubah tampilan tanaman secara keseluruhan, yang memungkinkan sistem menghasilkan berbagai bentuk.

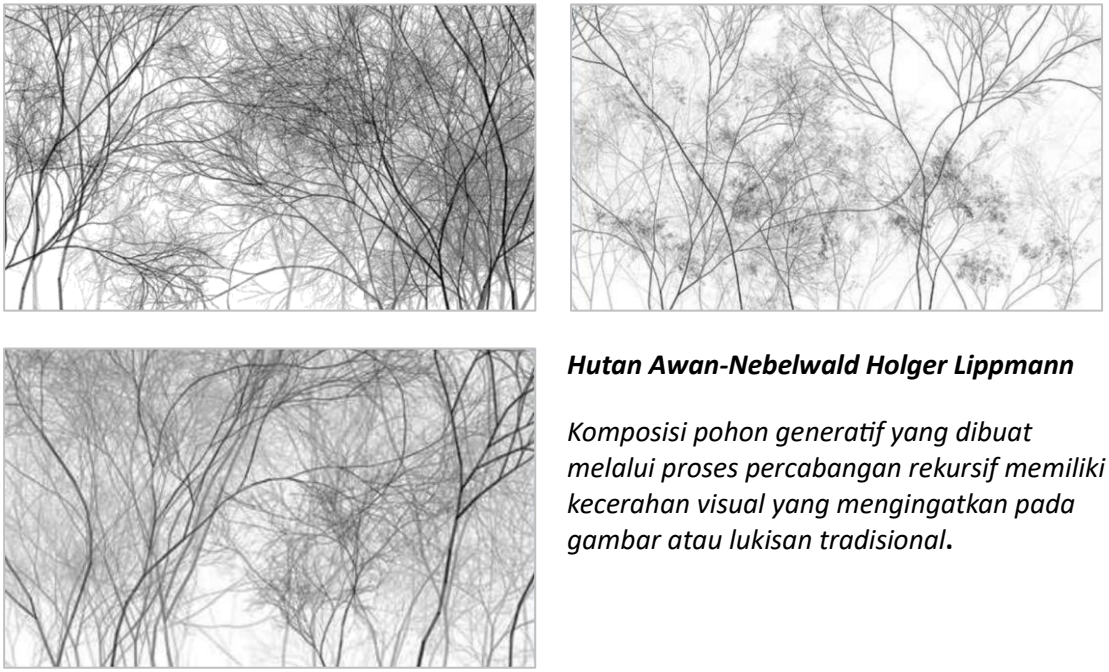
CONTOH

Holger Lippmann: Cloud Forest

Cloud Forest adalah serangkaian pohon generatif yang dibuat oleh programmer dan seniman digital Holger Lippmann. Lippmann menggunakan kode generatif rekursif (Pemrosesan) untuk membuat serangkaian gambar digital yang kompleks dan sangat rumit. Tingkat detail, kehalusan, dan kompleksitas dalam setiap gambar memberikan setiap bagian kualitas seperti lukisan, bukan gambar yang dibuat komputer. Inti dari karya ini adalah algoritma percabangan pohon (rekursif) yang sederhana.

Seniman tersebut telah dengan hati-hati mengubah dan menyesuaikan parameter dan

"pengaturan" algoritma gambar, mengubah parameter yang menentukan ukuran, posisi, dan transparansi untuk mencapai hasil yang paling mendekati visinya untuk gambar akhir. Bekerja dengan cara ini tidak seperti menjadi seorang programmer, tetapi lebih seperti menjadi seorang seniman atau pelukis bereksperimen dengan suatu proses, menyempurnakan material (kode), dan terus-menerus membuat penyesuaian dan penilaian estetika untuk mencapai hasil visual yang diinginkan. Lippmann sendiri menggambarkan proses penggunaan kode ini mirip dengan menari atau improvisasi musik, yang menjadi dasar pengembangan karya akhir.



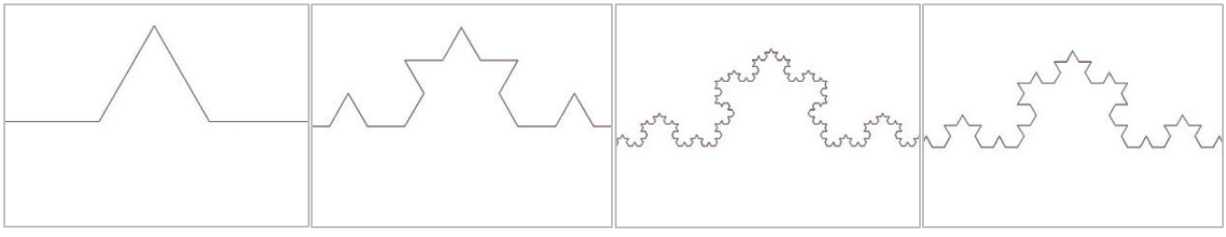
Hutan Awan-Nebelwald Holger Lippmann

*Komposisi pohon generatif yang dibuat melalui proses percabangan rekursif memiliki kecerahan visual yang mengingatkan pada gambar atau lukisan tradisional***.**

Kepingan Salju Koch

Fungsi rekursif memungkinkan desainer untuk mengulang dan menggambar ulang bentuk sederhana dengan cara yang meningkatkan kompleksitas. Kepingan salju Koch adalah bentuk kompleks yang dikembangkan dengan mengulang serangkaian instruksi menggambar sederhana. Ini adalah contoh bentuk fraktal yaitu, bentuk matematika yang berulang pada skala yang berbeda. Kepingan salju Koch dibuat dengan memulai dengan segitiga sama sisi dan menambahkan segitiga secara rekursif ke setiap sisinya. Pengulangan proses ini menciptakan bentuk seperti kepingan salju yang semakin kompleks. Variasi bentuk dapat dibuat dengan menerapkan proses yang sama ke berbagai jenis bentuk yang berbeda.

Contoh Kepingan Salju Koch
Dengan mengulang instruksi menggambar garis, bentuk fraktal secara bertahap menjadi lebih kompleks.



Sistem L

Sistem L (sistem Lindenmayer) adalah seperangkat aturan tertulis yang mensimulasikan pertumbuhan dengan merepresentasikan struktur tanaman sebagai serangkaian huruf yang berkembang. Sistem L menggambarkan pertumbuhan biologis sebagai struktur tata bahasa formal.

Huruf (biasanya A atau B) merepresentasikan bagian tanaman (misalnya, cabang kiri atau kanan). Dimulai dengan satu huruf, beberapa aturan sederhana digunakan untuk menghasilkan kembali huruf awal menjadi urutan yang semakin panjang, dengan menambahkan "cabang" baru setiap putaran. Program komputer kemudian dapat menginterpretasikan setiap huruf sebagai baris atau cabang baru, sehingga menciptakan keseluruhan struktur tanaman yang bercabang sesuai dengan urutan huruf yang berkembang. Misalnya, jika kita mulai dengan huruf "A" dan aturan bahwa untuk setiap putaran, "A" menjadi "AB" dan "B" menjadi "A," maka urutan berikut dibuat: aturan 1: A menjadi AB aturan 2: B menjadi A

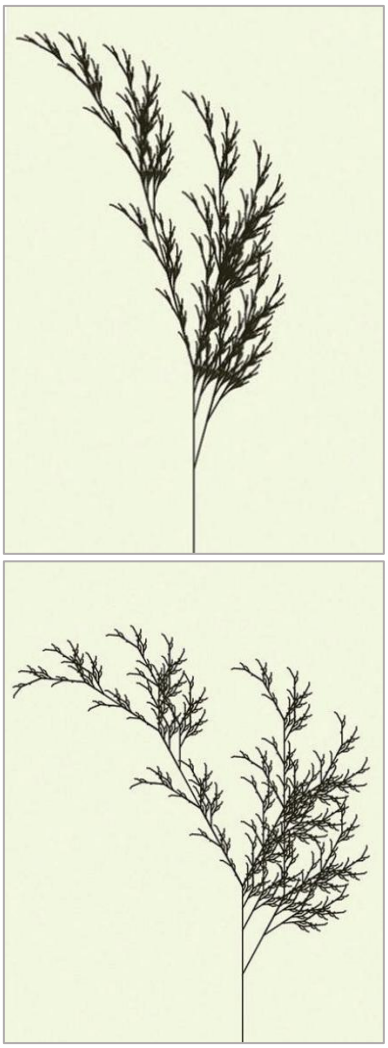
putaran 1: A

putaran 2: AB

putaran 3: ABA

putaran 4: ABAAB

Setelah setiap generasi tanaman, aturan diterapkan lagi dan urutan ditambahkan. Hasilnya adalah urutan huruf yang berkembang yang secara visual dapat diartikan sebagai pertumbuhan bentuk alami.



Contoh Sistem L
Saat diinterpretasikan oleh mesin gambar, rangkaian karakter yang dihasilkan oleh Sistem L menciptakan serangkaian bentuk dan rupa seperti tanaman.

3.5 EKOSISTEM DIGITAL

Lingkungan pemrograman adalah ekosistem digital yang berdiri sendiri: lingkungan abstrak tempat programmer menentukan aturan pergerakan, pertumbuhan, dan interaksi. Sama seperti lingkungan permainan komputer yang memiliki karakteristiknya sendiri yang menentukan bagaimana objek dan karakter bergerak dan berinteraksi, setiap lingkungan pemrograman juga memiliki aturannya sendiri, yang ditetapkan oleh programmer, yang mengatur cara objek di layar bergerak, tumbuh, dan berinteraksi. Aturan-aturan ini dapat bersifat alami, yang mencerminkan perilaku dunia fisik, atau sepenuhnya diciptakan dari imajinasi kreator. Layar adalah "kanvas kosong" tempat desainer dapat "berperan sebagai dewa," yang menentukan karakteristik khusus dari lingkungan baru.

Oleh karena itu, menentukan "aturan main" untuk lingkungan merupakan bagian penting dari proses kreatif, yang memberikan fokus dan fondasi pada karya tersebut.

Lingkungan digital dapat mengambil aturannya dari aturan yang memandu evolusi sekumpulan mikroorganisme, pergerakan gelembung, atau perilaku kawanan burung.

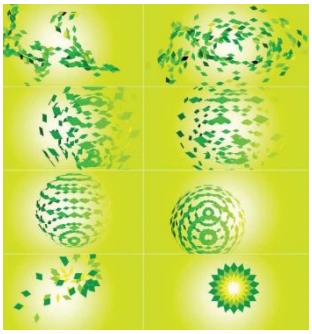
CONTOH

Simon Heijdens: Lightweeds

Proyek Simon Heijden "Lightweeds" adalah contoh menarik tentang bagaimana lingkungan digital dapat menjadi inspirasi dan refleksi dari dunia alam. Dengan menerapkan konsep yang terinspirasi oleh perubahan ritme pertumbuhan, kehidupan, dan pembusukan yang ada dalam materi organik, karya tersebut terdiri dari serangkaian proyeksi cahaya digital yang tampak seperti tanaman yang tumbuh lebih tinggi dan melengkung tertiup angin. Sering kali terletak di dalam ruang interior perkotaan, keberadaan bentuk botani digital yang menyemai sendiri dan bergelombang menciptakan kontras yang tajam dengan lingkungan dan arsitektur ruang modern yang terkendali: sebuah pengingat akan hilangnya kehadiran alam dalam kehidupan sehari-hari.

Rumput Liar Simon Heijdens
Proyeksi tanaman digital yang tumbuh dan bergerak di dalam ruangan sebagai reaksi terhadap kondisi lingkungan di luar.

Tanaman digital tumbuh dari kode yang mencerminkan pola pertumbuhan dan struktur kehidupan organik. "Benih" yang dihasilkan oleh kode menentukan struktur genetik dan perilaku setiap tanaman. Semua tanaman yang dihasilkan dari famili yang sama dihasilkan oleh benih yang sama dan karenanya tumbuh dan bertindak sesuai dengan kode "genetik" yang sama. Setiap tanaman digital tumbuh, hidup, dan mati sesuai dengan benih digitalnya. Tanaman tersebut bahkan menyerbuki tanaman baru, yang menghasilkan pertumbuhan lebih lanjut. Informasi tentang kondisi cuaca luar (kelembapan, angin, dan suhu) serta pergerakan orang di depan proyeksi diterjemahkan menjadi data yang menentukan bagaimana tanaman digital bergerak dan berkembang. Batas antara digital dan organik menjadi kabur saat proyeksi digital bergelombang dan menyebar sebagai respons terhadap kondisi lingkungan nyata.



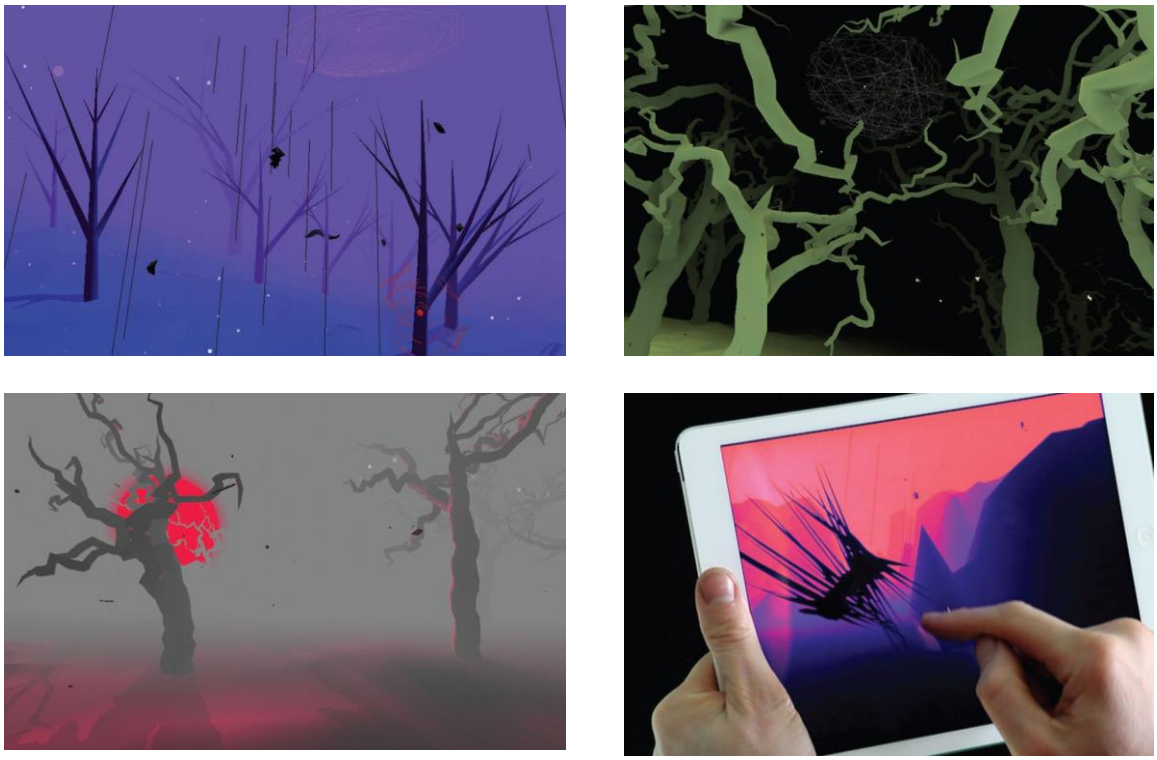
Logograf Interaktif Golan Levin
Contoh percobaan awal dalam menciptakan logograf interaktif. Logo BP dikonsep ulang sebagai bidang dinamis dari bentuk-bentuk individual yang bergerak secara kolektif sebagai respons terhadap interaksi pengguna, yang secara bertahap berubah untuk menciptakan logo. Perilaku kelompok dari bentuk-bentuk tersebut mencerminkan gerakan makhluk hidup yang organik dan alami saat mereka berkumpul dan berkerumun bersama.

Perilaku yang Mirip dengan Kehidupan Nyata

Kode dapat digunakan untuk membangun dan menerapkan perilaku alami pada objek grafis individual atau kelompok objek tersebut, mengubah bentuk, huruf, atau garis digital menjadi objek yang interaktif dan reaktif. Perilaku memberikan "kehidupan" pada objek di layar, mengubahnya menjadi organisme digital, yang memungkinkan objek tersebut bergerak, tumbuh, berubah, berinteraksi dengan objek lain dan bahkan memunculkan dan menghasilkan kembali bentuk dan rupa baru. Grafik dengan kualitas perilaku seperti ini menjadi versi digital dari bentuk kehidupan sederhana: tanaman, benih, atau organisme sel hidup.

Sama seperti keluarga atau spesies makhluk hidup yang memiliki atribut dan perilaku yang sama, kelompok objek digital juga dapat mewarisi dan berbagi pola perilaku.

Pemrograman berorientasi objek (OOP) memungkinkan programmer untuk membuat kelompok objek digital yang memiliki fungsi inti yang sama dan menentukan serangkaian perilaku generik yang umum (misalnya, kemampuan untuk tumbuh) sekaligus memungkinkan setiap objek memiliki kapasitas untuk mempertahankan atribut dan aspek individualnya sendiri dari "kepribadian"nya sendiri (misalnya, tingkat pertumbuhan tertentu). Oleh karena itu, objek digital dapat diciptakan yang, meskipun memiliki atribut individualnya sendiri, dapat bertindak dan bergerak dengan tujuan yang sama, semacam "mentalitas kelompok" bersama, yang menciptakan pola gerak yang meniru fenomena alam seperti cara burung berkelompok dan ikan berenang.



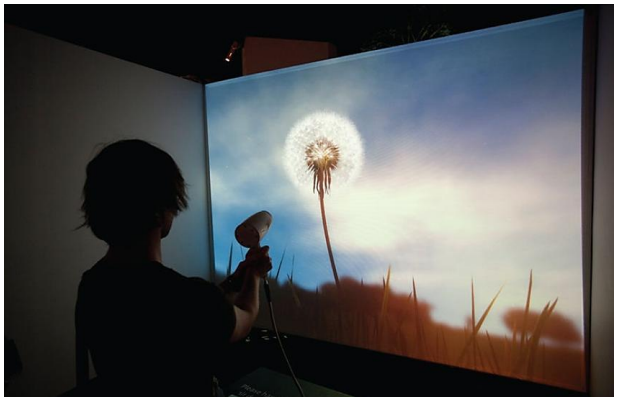
PolyFauna Radiohead, Nigel Godrich, Stanley Donwood, dan Universal Everything
Lingkungan audiovisual eksperimental dan imersif untuk iOS dan Android yang dihuni oleh bentuk kehidupan yang dihasilkan secara komputasional.

Radiohead, Nigel Godrich, Stanley Donwood, dan Universal Everything Polyfauna Digambarkan sebagai "lingkungan layar sentuh yang hidup, bernapas, dan berkembang" dan terinspirasi oleh lukisan lanskap atmosferik dari J. W. Turner hingga Peter Doig, serta bentuk kehidupan komputasional Carl Sims, PolyFauna merupakan hasil kolaborasi kreatif antara Radiohead, Nigel Godrich, Stanley Donwood, dan Universal Everything.

Dibuat menggunakan mesin Unity dan diprogram dalam C#, aplikasi ini menyajikan lingkungan digital audiovisual yang imersif tentang kehidupan primitif, cuaca, matahari terbenam, gunung, dan hutan. Lingkungan abstrak yang ada diisi dengan bentuk-bentuk kehidupan yang dihasilkan secara komputasional. Variasi musik, cuaca, palet warna, fase bulan, dan spesies makhluk menawarkan kepada pengguna serangkaian pengalaman unik setiap kali digunakan.

Lingkungan digital dieksplorasi melalui gerakan dan isyarat, kamera internal di dalam ruang terus bergerak maju, dan gerakan ke kiri dan kanan dicapai melalui rotasi fisik tubuh. Proyek ini mengeksplorasi penggunaan kehidupan digital yang dihasilkan secara komputasional dan pertumbuhan organik untuk menciptakan pengalaman pengguna yang imersif.

Gambar Sennep dan Yoke: Dandelion Interaktif



Pergerakan benih dandelion saat tertiup angin direplikasi secara digital dalam instalasi interaktif yang indah dan menyenangkan ini. Pengguna berinteraksi dengan proyeksi dandelion dengan menggunakan pengering rambut untuk meniup benih darinya. Benih-benih tersebut hanyut oleh angin virtual hingga tidak ada yang tersisa; kemudian mereka terbentuk kembali dan memulai lagi. Kode digunakan untuk meniru gerakan benih yang melayang tertiup angin dan menciptakan pengalaman yang mengingatkan pada masa kanak-kanak. Kode digital menerjemahkan interaksi dengan alam menjadi pengalaman digital yang puitis dan mendalam.

Dandelion Sennep and Yoke
Dibuat menggunakan Processing, program baru ini mengeksplorasi cara fisik berinteraksi dengan lingkungan. Benih dandelion virtual ditiupkan ke layar dengan pengering rambut.

BOIDS

Pada akhir tahun 1980an, ilmuwan komputer Craig Reynolds mengembangkan perilaku kemudi algoritmik untuk karakter animasi. Perilaku ini memungkinkan elemen individu menavigasi lingkungan digital mereka dengan cara yang "mirip dengan kehidupan nyata" dengan strategi untuk melarikan diri, mengembara, tiba, mengejar, menghindar, dan sebagainya.

Dengan membangun sistem yang terdiri dari beberapa karakter yang mengarahkan diri mereka sendiri menurut serangkaian aturan sederhana, muncul tingkat kompleksitas yang mengejutkan. Contoh yang paling terkenal adalah model "boids" Reynolds untuk perilaku "berkelompok/berkerumun". Model perilaku ini telah diterapkan kembali oleh para desainer dan seniman untuk mengembangkan kawanan karakter yang menampilkan jenis perilaku kelompok yang ditampilkan dalam aktivitas berkelompok dan mengarahkan sekelompok ikan atau burung.

3.6 GAYA LINGKUNGAN GRAVITASI, ELASTISITAS

Selain menciptakan pola perilaku individu dan kelompok, gerakan, dan interaksi objek individual, kode juga dapat menentukan karakteristik lingkungan tempat objek tersebut bergerak. Gaya virtual dan digital yang meniru sifat fisik lingkungan alami, seperti angin, gesekan, atau gravitasi, dapat meningkatkan lingkungan digital dan memengaruhi gerakan dan gerak semua objek di dalamnya.

Simulasi gaya yang meniru "angin" dalam dunia layar, misalnya, dapat berarti bahwa grafik dan potongan tipografi tertiup dengan anggun di layar seperti daun yang tertiup angin. Menambahkan atau membuat perubahan pada faktor lingkungan (misalnya, nilai yang mengubah atribut berat atau ringan) mengubah sifat pemandangan virtual dan dapat menentukan apakah objek di layar tampak seolah-olah berada di udara, di air, atau di luar

angkasa. Mengubah jumlah "gravitasi" komputasional dalam suatu pemandangan, misalnya, akan memengaruhi kecepatan dan gerakan objek menuju "tanah".

Dengan menambahkan kualitas perilaku dan lingkungan ke layar, kode dapat mengubah objek dalam lingkungan terprogram sehingga objek tersebut bertindak seperti makhluk digital yang hidup, yang dapat berubah seiring waktu dan mampu hidup, tumbuh, bergerak, dan bahkan mereplikasi diri. Objek yang dibuat secara digital dapat menciptakan berbagai lingkungan yang menarik secara visual yang dapat menghasilkan dirinya sendiri, seperti ekosistem taman atau kolam, atau dihasilkan dan diinformasikan oleh interaksi pengguna.

CONTOH

Desain Bibliotheque: Ollo

Identitas visual yang dibuat untuk Ollo, sebuah perusahaan telekomunikasi digital, mengeksplorasi potensi kode interaktif untuk membentuk bagian penting dari solusi desain. Sebagai bagian dari pengembangan merek, logo dibayangkan bukan sebagai visual statis yang tetap, tetapi sebagai bentuk organik yang hidup dengan sifat-sifat dinamis visualnya sendiri. Agar sesuai dengan konsep merek "satu jalur komunikasi", logo Ollo dibuat sebagai satu garis yang fleksibel dan dinamis yang dapat dimanipulasi melalui interaksi berbasis sentuhan, yang mendorong pengguna untuk mendorong dan menarik bentuk tersebut di layar.

Manipulasi logo secara interaktif menjadi alat kreatif dalam membangun identitas visual, yang memungkinkan desainer untuk membuat aset digital unik dalam jumlah tak terbatas yang dapat diintegrasikan ke dalam merek. Kekuatan fisik yang dibuat secara komputasional, seperti elastisitas dan kelenturan, memberikan bentuk tersebut karakteristik ekspresif yang menyenangkan, yang mencerminkan kepribadian dan informalitas merek.

Desain Perpustakaan Ollo
Logo yang lembut dan responsif dibuat untuk Ollo dan dibuat agar sesuai dengan kepribadian merek tersebut. Fleksibilitas digital dari logo tersebut menciptakan citra merek yang mendorong interaksi yang menyenangkan dengan pelanggan dan terbuka terhadap variasi visual yang hampir tak terbatas.

Di Mana Saja: Oasis

Oasis adalah lingkungan interaktif yang menyenangkan, ekosistem seperti kolam yang berdiri sendiri yang dihuni oleh bentuk-bentuk kehidupan yang dihasilkan secara digital. Aplikasi interaktif ini mengisi lingkungan tersebut dengan berbagai organisme digital, masing-masing dengan karakteristik, perilaku, dan gerakannya sendiri. Dengan menggunakan kode untuk meningkatkan dan menciptakan perilaku setiap makhluk, pengguna dapat bermain dengan lingkungan tersebut dengan membentuk kolam-kolam bentuk kehidupan dari pasir di permukaan. Gerakan dan interaksi makhluk-makhluk digital tersebut menarik bagi pemirsa yang dapat menjelajahi dan menonton; hal itu menciptakan kembali minat seperti anak-anak untuk mengintip ke dalam kolam batu virtual.

Ekosistem yang diciptakan secara digital memiliki bentuk yang ringan, indah, dan unik yang tidak selalu tepat secara matematis atau ilmiah, tetapi merupakan interpretasi artistik dari bentuk-bentuk organik. Bentuk, warna, bentuk, dan gerakan tersebut menggemakan bentuk yang ringan dan indah dari dunia nyata. Hidup dalam ruang layar atau proyeksi memadukan lingkungan nyata dan virtual.

Oasis Segala Peranti
Bentuk kehidupan komputasional virtual muncul dalam "kolam batu" digital.

SOROTAN PADA

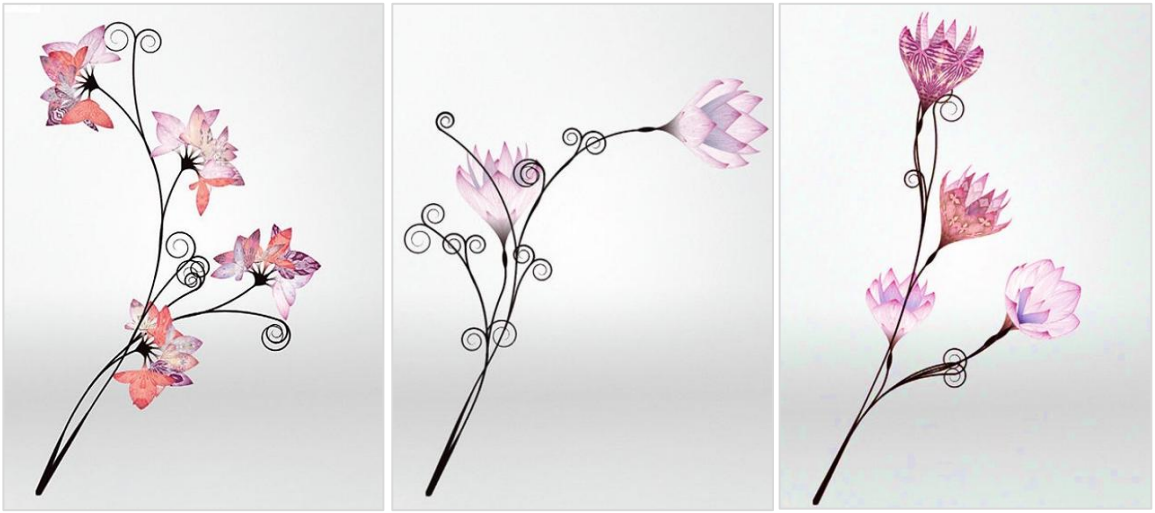
Daniel Brown

Daniel Brown adalah seorang desainer, programmer, dan seniman, serta salah satu dari generasi pelopor yang mengeksplorasi penggunaan dan aplikasi kreatif dari teknologi interaktif. Ia telah menjadi yang terdepan dalam desain digital dan interaktif sejak meluncurkan eksperimen web yang menyenangkan dan inovatif, Noodlebox, pada tahun 1997.

Tentang Pertumbuhan dan Bentuk Daniel Brown
Ditugaskan oleh Universitas Dundee, ini adalah karya seni generatif yang dibuat sebagai penghormatan kepada buku perintis D'Arcy Thompson berjudul On Growth and Form (1917) untuk Museum Zoologi D'Arcy Thompson. Terinspirasi oleh bentuk dan tekstur dalam pameran museum, karya ini menghasilkan tanaman dan bunga digital yang terus tumbuh dan beregenerasi. Karya ini merupakan bagian dari rangkaian karya seni generatif Daniel yang sedang berlangsung, di mana bentuk dan rupa botani dibuat dari algoritme kompleks dan proses komputasi.

Karya Daniel menggabungkan teknologi dan pemrograman mutakhir dengan rasa estetika yang terinspirasi oleh cahaya dekoratif dan kesegaran bentuk dan rupa alami. Ia telah menghasilkan karya untuk berbagai merek mode dan mewah selain menciptakan instalasi pribadi dan publik.

Keindahan botani alami terutama bentuk, rupa, dan warna kupu-kupu dan bunga—memenuhi karya Daniel. Banyak karyanya menerjemahkan bentuk botani ke dalam lingkungan digital bunga dan tanaman yang tumbuh dan berkembang biak. Proses teknologi yang rumit dan algoritma generatif disembunyikan di balik keanggunan karya akhir sehingga pemirsa tidak hanya menghargai penerapan kode yang cerdas, tetapi juga keindahan estetika karya itu sendiri.



Bunga Cinta Mulberry Daniel Brown
Proyek Love Blossoms yang diprakarsai oleh Mulberry menawarkan kepada pengguna kesempatan untuk mengirimkan bunga unik yang dibuat secara digital sebagai hadiah Valentine. Dengan memilih pola Mulberry musiman, pengguna mengirimkan benih yang tumbuh menjadi bunga digital yang unik. Proses komputasi digunakan untuk menciptakan bentuk dan rupa setiap bunga dan kelopak, yang berarti tidak ada dua bunga yang sama, dan pengalaman bagi setiap penerima adalah unik. Kombinasi media interaktif dengan bentuk naturalistik yang elegan menciptakan pengalaman digital yang indah bagi pemirsa dan penerima.

3.7 LINGKUNGAN DIGITAL

Banyak gaya dari dunia nyata dapat disimulasikan secara komputasional dalam lingkungan digital dengan menerapkan kalkulasi matematika sederhana untuk menciptakan bentuk dan gerakan. Bagian berikut menguraikan beberapa contoh yang bermanfaat.

Gelombang Sinus

Gelombang sinus adalah kurva yang dijelaskan secara matematis, sering digunakan dalam trigonometri, yang dapat diterapkan untuk menggambar garis lengkung atau menciptakan gerakan memantul yang halus. Fungsi sinus menggunakan sudut untuk membuat deret angka: Saat sudut meningkat, deret angka "memantul" antara nilai minimum (-1) dan maksimum (+1).

```
sine(angle); // syntax example
sin (radians (90)); // outputs 1
sin (radians (270)); // outputs -1
```

Mengalikan hasil dengan angka yang lebih besar, nilai "besarnya", "memperbesar" angka-angka ke dalam rentang yang lebih dapat digunakan.

```
float magnitude = 20; // create a "magnify" value
sin (radians (90)) * magnitude; // outputs 20
sin (radians (270)) * magnitude; // outputs -20
```

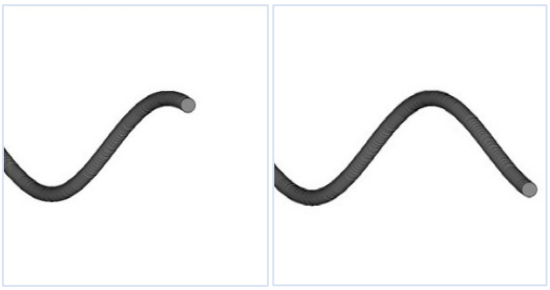
Saat nilai sudut terus meningkat, angka yang dibuat oleh fungsi sinus "memantul" antara nilai minimum dan maksimum. Kode berikut menghitung ulang lokasi lingkaran pada sumbu y dengan menggunakan fungsi sinus untuk menggerakkan bentuk ke atas dan ke bawah.

```
float angle = 0; // starting angle
void draw () {
  // a "bouncing" value between -20 and 20
  float ypos = sin(radians(angle)) * 20;
  ellipse (10, ypos, 10, 10); // draw circle
  angle++; // increase angle by 1
}
```

Dengan menambahkan variabel untuk menggerakkan objek melintasi panggung pada saat yang sama, gerakan gelombang (pantulan) sederhana tercipta.

```
float angle = 0;
float xpos = 10;
void draw () {
  float ypos = sin(radians(angle)) * 20;
  // draw ellipse with variables for x and y
  ellipse (xpos, ypos + 50, 10, 10);
```

```
    angle++; // increase angle by 1
    xpos += 2; // increase xpos value
  }
```



Gelombang sinus dapat digunakan untuk menggerakkan objek dalam gerakan seperti gelombang di layar. Melakukan penyesuaian pada perhitungan, seperti laju peningkatan nilai sudut atau angka magnitudo, mengubah jenis gelombang dan gerakan yang dihasilkan.

Secara bertahap meningkatkan ukuran objek saat bergerak menghasilkan gerakan gelombang yang tampak lebih organik.

Gerakan dan perilaku objek interaktif dapat dibuat lebih realistis dengan menambahkan simulasi komputasional gaya dunia nyata seperti gesekan, gravitasi, angin, dan elastisitas.

Gesekan dan Peredaman

Gerakan objek di dunia nyata dipengaruhi oleh gesekan dan gaya hambatan lainnya, yang membuat objek melambat secara bertahap sebelum berhenti. Gulingkan bola di lantai dan bola tersebut tidak langsung berhenti; sebaliknya, bola tersebut melambat secara bertahap. Efek perlambatan ini dicapai secara matematis dengan mengalikan nilai "kecepatan" dengan pecahan. Mengalikan angka secara berulang dengan nilai pecahan (misalnya, 0,5 atau 0,9) secara perlahan akan menurunkan ("meredam") nilainya dan merupakan cara yang baik untuk mensimulasikan efek perlambatan. Misalnya, jika variabel "kecepatan" dimulai dengan nilai 100, mengalikan dengan 0,5 akan mengurangi kecepatan hingga 50% dari nilainya setiap kali.

```
float speed = 100;
speed = speed * 0.5; // speed becomes 50
speed = speed * 0.5; // speed becomes 25
speed = speed * 0.5; // speed becomes 12.5
speed = speed * 0.5; // speed becomes 6.25
speed = speed * 0.5; // speed becomes 3.125
```

Membagi dua nilai setiap kali berarti bahwa laju perubahan antara angka-angka tersebut secara bertahap berkurang. Memetakan perubahan ini secara visual menciptakan efek perlambatan.

Mengubah nilai pecahan akan mengubah laju gerakan. Dalam contoh berikut, nilai kecepatan, yang digunakan untuk mengubah lokasi x bentuk, terus dikalikan dengan 0,9, yang memiliki efek "meredam" kecepatan dan memperlambat gerakan bentuk.

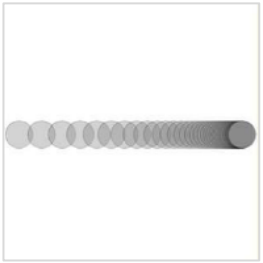
```
// create starting variables for speed and x location
float speed = 5;
float xpos = 10;
void draw () {
  speed = speed * 0.9; // "dampen" the speed
  xpos = xpos + speed; // update the shape location
  ellipse (xpos, 50, 10, 10); // draw shape
}
```

Menggunakan nilai bilangan pecahan untuk "meredam" kecepatan horizontal suatu objek dapat digunakan untuk menciptakan efek di mana objek melambat secara bertahap saat mendekati target. Dalam contoh berikut, sebuah lingkaran digerakkan menuju titik tetap, dan melambat secara bertahap saat bergerak.

Selama bagian "gambar" berulang dari kode, jarak antara lokasi x objek (xpos) dan titik targetnya (targetX) dihitung ulang. Nilai ini menunjukkan jarak total antara bola dan tujuan akhirnya, dan disimpan sebagai variabel ("distanceToMove"). Alih-alih menggerakkan lingkaran ke targetnya dalam satu lompatan, nilai "peredam" digunakan untuk menggerakkan lingkaran sebagian dari total jarak yang dibutuhkan.

Saat proses ini diulang, jarak antara bola dan targetnya menjadi lebih kecil dan pecahannya menjadi lebih kecil, yang memiliki efek memperlambat objek secara bertahap hingga berhenti.

```
float damping = 0.1;
float targetX = 90;  // target x location to aim for
float xpos = 10;     // starting x location
void draw() {
  background(0);
  // find how far to move
  float distanceToMove = targetX - xpos;
  // move a fraction of distance
  xpos = xpos + (distanceToMove * damping);
  ellipse(xpos, 50, 10, 10);
}
```



Suatu benda secara bertahap melambat hingga berhenti.

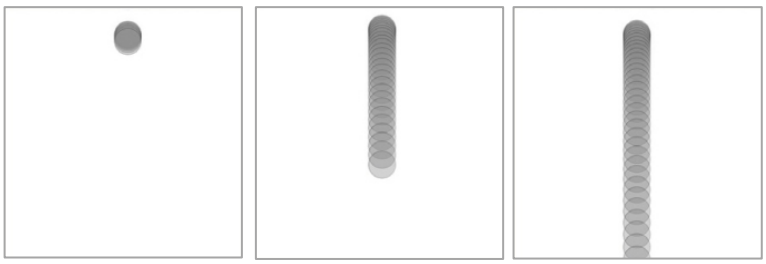
Gravitasi

"Gravitasi" komputasional direpresentasikan dalam kode sebagai angka, yang terus-menerus ditambahkan ke nilai kecepatan dan secara bertahap meningkatkan pergerakan suatu objek. Kecepatan dimulai dari 0, tetapi saat nilai gravitasi ditambahkan berulang kali, kecepatan akan bertambah cepat.

```
float gravity = 0.3; // a constant gravity value
// use gravity to increase speed
speed = speed + gravity;
// change y location using speed
ypos = ypos + speed;
ellipse (10, ypos, 5, 5); // draw object
```

Dengan mengulang proses ini, seorang programmer menciptakan efek sebuah objek jatuh karena gaya gravitasi:

```
float gravity = 0.1; // a set value
float speed = 0; // a changing value that increases
// the starting y location of the ball
float ypos = 0;

void draw () {
  // gradually increase the speed
  speed = speed + gravity;
  // update the y location of the object
  ypos = ypos + speed;
  ellipse (10, ypos, 5, 5
}
```



Bentuk jatuh yang dipengaruhi oleh gravitasi komputasional akan meningkatkan kecepatan saat mendekati tanah.

Pantulan

Gravitasi membuat bentuk jatuh dan jatuh dari tepi layar. “Pantulan” dapat ditambahkan dengan membalikkan kecepatan objek saat menyentuh “tanah.” Membalikkan kecepatan akan membalikkan arah gerakan. Untuk membalikkan nilai kecepatan, kalikan dengan angka negatif. Mengalikan apa pun dengan -1 akan mengubahnya dari positif menjadi negatif, atau dari negatif menjadi positif: 10 * -1 = -10;

```
−10 * −1 = 10;
```

Oleh karena itu, mengalikan kecepatan dengan angka negatif akan membalikkan arah objek, sehingga menciptakan efek “pantulan”. Menggunakan -0,95 daripada -1 “meredam” kecepatan, yang menyebabkan objek kehilangan sedikit momentum pada setiap “pantulan.”

```
// reverse direction and lose energy
speed = speed * -0.95;
```

Efek "pantulan" perlu dipicu pada titik tertentu (misalnya, saat bola jatuh terlalu jauh atau "menghantam tanah"). Dalam istilah kode, ini berarti memicu pembalikan kecepatan saat lokasi y objek telah mengenai, atau jatuh di bawah, "permukaan tanah" di layar. Pernyataan "if" digunakan dalam bagian draw untuk terus memeriksa lokasi y objek saat jatuh. Jika bola mengenai, atau jatuh melewati tepi bawah layar, maka kecepatan dibalik dan bola memantul.

```
if (ypos > height) {
  speed = speed * -0.95;
}
```

Angin

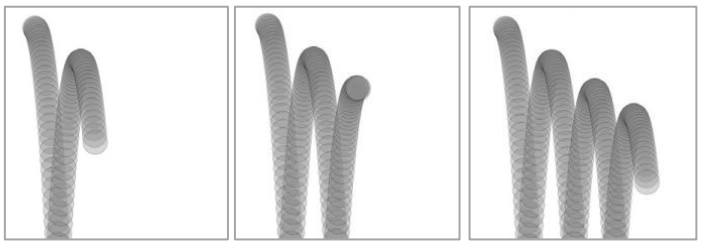
Sama seperti gravitasi adalah angka yang digunakan untuk memengaruhi lokasi y suatu objek, maka "angin" komputasional dapat dihasilkan dengan menggunakan angka untuk memengaruhi lokasi x suatu objek dan mendorongnya ke samping. Menambahkan gaya untuk mengubah gerakan ke samping dapat digunakan untuk mensimulasikan "angin sepoi-sepoi" yang bertiup melintasi layar.

```
float wind = 0.4;
xpos = xpos + wind;
ellipse (xpos, 100, 50, 50);
```

Efek gabungan dari gaya gravitasi dan angin memberikan properti lingkungan digital yang mulai meniru perilaku dalam ekosistem fisik alami. Objek digital menjadi hidup, bergerak dengan cara yang sederhana namun realistis.

```
float speed = 0;
float gravity = 0.3;
float wind = 0.4;
float ypos = 0;
float xpos = 10;

void draw () {
  speed = speed + gravity; // increase speed
  ypos = ypos + speed; // move ball's y position

  xpos = xpos + wind; // move ball's x position
  ellipse (xpos, ypos, 10, 10); // draw ball
```

```
  // reverse speed when the ball reaches bottom of screen
  if (ypos > height) {
    speed = speed * -0.95;
  }
}
```



Contoh bentuk yang digerakkan oleh gaya komputasional "gravitasi" dan "angin." Gravitasi meningkatkan kecepatan bentuk saat bergerak ke tanah. Saat mencapai dasar layar, kecepatannya dibalik untuk "memantulkan" bentuk tersebut. Gaya "angin" tambahan menggerakkan objek ke

Pegas dan Elastis

Gaya elastisitas menciptakan gerakan yang memberi objek perilaku seperti pegas yang lincah. Gerakan pegas direplikasi dalam kode menggunakan kalkulasi standar yang digunakan untuk memodelkan gaya elastis semacam ini.

```
spring force = stiffness * distance stretched;
```

Kekakuan menunjukkan seberapa "memantulnya" pegas; jarak yang diregangkan bisa jadi adalah jarak antara grafik dan lokasi istirahat yang dituju. Setelah dihitung, gaya pegas ditambahkan untuk mengubah gerakan suatu objek. Peredaman diterapkan untuk memastikan pegas melambat secara bertahap hingga berhenti. Gaya pegas dapat diterapkan pada gerakan suatu objek dengan cara berikut:

```
// create starting variables
float stiffness = 0.1; // a constant value
// a constant value to dampen the force
float damping = 0.9;

float targetX = 80; // location to "spring" to
float xpos;
float speed;
void draw ()
 {
// calculate spring force
  float spring_force = stiffness * (targetX - xpos);
  // add force to speed
  speed = speed + spring_force;
  // apply damping
  speed = speed * damping;
  xpos = xpos + speed;
  ellipse (xpos, 50, 20, 20);
}
```

Jenis gaya pegas yang memantul ini dapat sangat efektif saat memindahkan bentuk ke suatu tujuan, memberikan gerakan tingkat "vitalitas" ekstra. Ini juga dapat memberikan grafis elemen fisik yang lebih tinggi saat digunakan untuk mengubah ukuran bentuk: semacam efek goyangan dan peregangan.

Objek dan Grup

Contoh di atas adalah contoh di mana satu bentuk (bola) digerakkan oleh nilai yang mensimulasikan gaya gravitasi dan angin. Ini berfungsi dengan baik untuk satu bentuk, tetapi bagaimana jika Anda ingin membuat lebih banyak objek yang bergerak serupa misalnya, sebagai partikel atau daun yang tertiup angin?

Mengulangi proses yang sama untuk membuat ratusan bentuk bergerak serupa akan memerlukan serangkaian variabel dan kalkulasi baru untuk setiap bentuk baru, dan akan dengan cepat menjadi sangat rumit. Cara yang lebih berguna dan efisien untuk melakukan ini adalah dengan menggunakan proses berorientasi objek dan membuat perilaku generik sebagai

"kelas."

"Kelas" Pemrograman

Dalam pemrograman, "kelas" adalah blok kode yang bertindak sebagai templat (cetak biru) tempat banyak objek individual dapat dibuat. Kelas mendefinisikan properti dan perilaku generik setiap objek. Ketika objek individual dibuat, objek tersebut mewarisi properti dan perilaku generik yang sama, tetapi setiap objek mungkin memiliki nilai yang berbeda untuk properti tertentu (misalnya, memiliki ukuran atau nilai warna yang berbeda). Membuat kelas memungkinkan programmer untuk membuat ulang dan mereproduksi banyak objek serupa yang memiliki perilaku dan tampilan generik yang sama.

Contoh Kelas "Mobil"

Kelas adalah templat yang digunakan untuk membuat banyak objek serupa. Objek tersebut dapat dibuat sesederhana atau serumit yang dibutuhkan.

Kelas ditulis sebagai kumpulan variabel dan fungsi yang menentukan properti (karakteristik) dan metode (perilaku) suatu objek. Setiap contoh kelas mewarisi serangkaian properti dan perilaku yang sama. Misalnya, bayangkan sebuah mobil. Setiap mobil memiliki properti umum yang sama (masing-masing memiliki pintu, mesin, gigi, dll.) dan perilaku (dapat berakselerasi, berbelok, mengerem).

Saat dibuat, setiap "objek" mobil memiliki perbedaan tersendiri (warna, merek, model); bentuk dan ukurannya mungkin berbeda, tetapi masing-masing berasal dari templat generik yang sama. Hal ini sama untuk objek yang dibuat oleh suatu kelas: Objek tersebut memiliki templat yang sama tetapi masing-masing memiliki kualitas tersendiri.

Dalam contoh kita, kelas "mobil" akan ditulis untuk menentukan properti dan fungsi generik mobil sebagai berikut:

```
class Car {
// define properties of the car
float engineSize;
int numberOfDoors;
float maxSpeed;
```

```
// define the "constructor" to create a car
Car (float ms) {
  maxSpeed = ms;
}

// define some behaviors: functions of the car
void changeSpeed () {
  // add functions here
}

void changeDirection () {
  // add functions here
}
}
```

Setelah kelas ditulis, instance (objek) individual dapat dibuat darinya. Mobil baru dapat dibuat dari kelas mobil dengan membuat instance “baru” yang memanggil fungsi konstruktor yang ditulis ke dalam kelas. Dalam contoh ini, mobil baru dibuat dan menetapkan nilai maxSpeed dari setiap instance mobil baru:

```
// a new “car” instance
Car car1 = new Car (120);
// another new "car" instance
Car car2 = new Car (200);
```

Setelah instance kelas dibuat, properti dan fungsi yang ditulis dalam kelas "template" dapat diakses menggunakan notasi "titik". Misalnya, setiap mobil yang telah dibuat (car1, car2) dapat mengatur atau mengubah properti engineSize-nya sebagai berikut:

```
car1.engineSize = 200;
car2.engineSize = car1.engineSize + 20;
```

Demikian pula, mereka dapat memanggil fungsi changeSpeed() atau changeDirection().

```
car1.changeSpeed; car2.changeDirection;
```

Kelas untuk Membuat Beberapa Bentuk

Dalam contoh sebelumnya, satu bentuk digerakkan oleh nilai gravitasi. Pergerakan bentuk tersebut dihasilkan menggunakan tiga variabel:

```
float gravity = 0.1;
float speed = 0;
float ypos = 0;
```

Program utama mengulang tiga instruksi utama: meningkatkan kecepatan, memperbarui lokasi, dan menggambar bentuk.

```
// increase the speed
speed = speed + gravity;
// update the location of the object
ypos = ypos + speed;
// draw the shape
ellipse (10, ypos, 5, 5);
```

Variabel-variabel dan instruksi ini dapat digunakan sebagai dasar untuk "kelas" lingkaran generik yang dapat digunakan sebagai templat untuk membuat banyak bentuk lingkaran jatuh yang serupa. Kelas lingkaran contoh dapat mencakup variabel untuk mengendalikan gravitasi, kecepatan, dan posisi y objek, serta fungsi untuk meningkatkan kecepatannya, memperbarui lokasinya, dan menggambar bentuknya. Kode berikut adalah contoh titik awal untuk membuat jenis kelas lingkaran yang dapat digunakan kembali ini.

```
class Circle {
  // name the properties
  float speed;
  float ypos;
  float gravity;

  // set the “constructor” to set the gravity and y position each
instance
  Circle (float g, float y) {
    gravity = g;
    ypos = y;
  }

  // create simple functions: re-locate the circle
  void updateSpeedAndLocation () {
    speed = speed + gravity;
    ypos = ypos + speed;
  }
}
// draw the circle
void drawShape () {
  ellipse (10, ypos, 5, 5);
}
```

Setelah dibuat, kelas lingkaran dapat digunakan untuk membuat banyak contoh individual baru, masing-masing dengan nilai gravitasi dan ypos awalnya sendiri (sebagaimana didefinisikan dalam fungsi “konstruktor”).

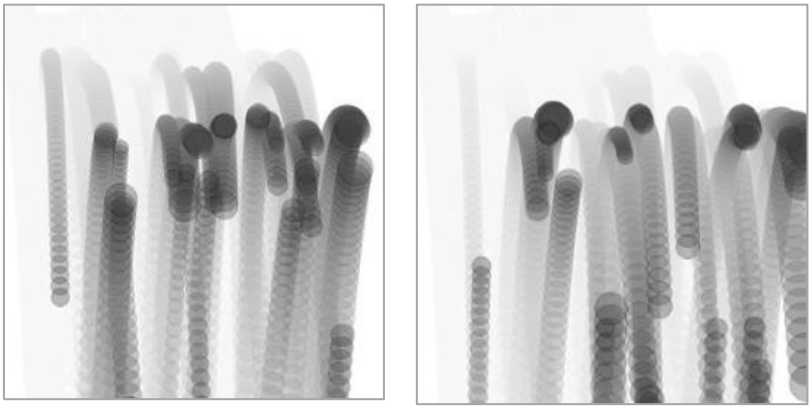
```
// create a new circle with a gravity of 0.1 and y position of 10
Circle c1 = new Circle (0.1, 10);
// create a new circle with a gravity of 0.2 and y position of 20
Circle c2 = new Circle (0.2, 20);
```

Menentukan nilai gravitasi dan ypos yang berbeda membuat setiap lingkaran dimulai pada ketinggian yang berbeda dan jatuh pada kecepatan yang berbeda. Kedua contoh lingkaran baru (“c1” dan “c2”) sekarang dapat memanggil fungsi yang ditulis dalam kelas yang digunakan untuk memindahkan dan menggambar setiap bentuk di layar:

```
c1.updateSpeedAndLocation();
c1.drawShape();
c2.updateSpeedAndLocation();
c2.drawShape();
```

Contoh sederhana ini menunjukkan bagaimana struktur kelas digunakan untuk membuat objek dan perilaku yang dapat digunakan kembali. Menetapkan nilai individual ke properti setiap instans akan memberikan identitas unik tersendiri bagi setiap objek. Setelah struktur dasar dibuat, kompleksitas yang lebih tinggi dapat ditambahkan dengan, misalnya, menambahkan lebih banyak variabel (misalnya, untuk menentukan ukuran atau lokasi x setiap lingkaran), atau dengan menambahkan fungsi tambahan, seperti "bounce".

Pemrograman berorientasi objek memungkinkan sekelompok objek dibuat dengan perilaku generik yang sama.



Mengisi array dapat mengisi layar dengan objek dengan perilaku serupa. Instansi individual dari setiap objek memiliki sedikit perbedaan yang mengubah atribut seperti ukuran, lokasi awal, atau kecepatan gerakan.

Setelah kelas dibuat, array dapat dibuat untuk menampung dan menyimpan banyak contoh sekaligus. Ini menyediakan cara cepat untuk membuat dan mengakses puluhan atau ratusan contoh objek sekaligus. Contoh berikut membuat daftar ("circleList") yang dapat menampung hingga 10 contoh lingkaran.

```
Circle [ ] circleList = new Circle [10];
```

Perulangan "for" digunakan untuk mengisi daftar dengan objek Lingkaran, yang memberikan setiap contoh nilai gravitasi dan ypos yang berbeda:

```
for (int i=0; i<circleList.length; i++) {
circleList[i] = new Circle (random (0.7), i*10);
}
```

Perulangan “for” lainnya kemudian dapat digunakan untuk menelusuri daftar dan memanggil fungsi updateSpeedAndLocation() dan drawShape ke setiap instance lingkaran secara bergantian:

```
for (int i=0; i<circleList.length; i++) {
circleList[i]. updateSpeedAndLocation();
 circleList[i]. drawShape();
}
```

BAB 4
TIPOGRAFI DINAMIS

"Tipografi harus dapat didengar. Tipografi harus dapat dirasakan.
Tipografi harus dapat dialami."
Helmut Schmid

4.1 BENTUK DAN ISI

Tipografi telah memainkan peran penting dalam komunikasi visual dan menjadi bagian penting dari praktik desain grafis. Sejak awal abad ke-20 hingga saat ini, desainer grafis telah bereksperimen dengan kemungkinan kreatif dari jenis huruf dan bentuk huruf sebagai mode komunikasi visual.

Lanskap teknologi yang berubah telah membuka sejumlah peluang kreatif bagi desainer untuk bereksperimen dan mengeksplorasi cara bermain dengan bentuk dan tampilan huruf dan fon. Lapisan piksel dalam dokumen bitmap, titik dan jalur dalam berkas vektor, objek dalam lanskap 3D, dan serangkaian efek filter: semuanya memungkinkan teks digital untuk diubah dan dikonfigurasi ulang dalam berbagai cara yang berbeda. Konsep "teks sebagai gambar" adalah umum; perangkat lunak memungkinkan dan mendorong keceriaan visual dengan kata-kata dan huruf saat kata-kata tersebut diregangkan, digabungkan, dan disusun berlapis-lapis.

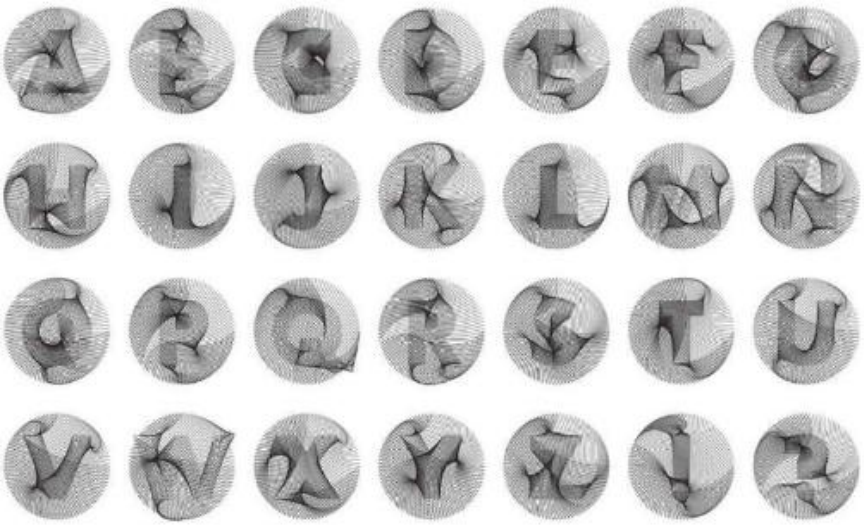
Selain mengubah estetika tipografi, teknologi digital juga telah mengubah cara membaca jenis huruf dan kata-kata. Media digital telah mengubah konsep teks dari sekumpulan kata cetak yang tetap dan permanen menjadi sumber data digital yang konten dan bentuknya dapat dibaca, dibaca ulang, dan dimanipulasi. Dalam dunia digital, kata-kata tidak terpaku pada halaman; kata-kata tersebut ada dalam lingkungan yang bentuk dan isinya dapat berubah.

Desainer dapat menggunakan kode untuk secara kreatif menata ulang bentuk dan isi teks. Dengan menggunakan kode pemrograman, teks digital dapat dimanipulasi baik sebagai data bentuk titik, garis, dan kurva yang menciptakan visual animasi yang dinamis sebagai bentuk huruf atau sebagai konten misalnya, berkas teks informasi digital, novel, buku, dan puisi yang kontennya digunakan untuk membuat dan menghasilkan grafik. Bab ini akan memberikan gambaran umum tentang cara-cara penggunaan data digital untuk membuat dan memanipulasi teks dan tipografi baik sebagai bentuk maupun sebagai konten.

Tipe Sebagai Bentuk

Aturan dan struktur pemrograman dapat diterapkan secara kreatif untuk mengubah dan menggabungkan potongan teks secara dinamis atau untuk memanipulasi bentuknya. Aturan dan struktur tersebut dapat diterapkan untuk memanipulasi fon dan bentuk huruf yang ada atau untuk membuat bentuk baru yang dihasilkan secara komputasional. Tidak seperti prosedur yang digerakkan oleh perangkat lunak "tertutup", yang terbatas pada serangkaian teknik manipulasi gambar yang telah ditetapkan sebelumnya (filter, lengkungan, dll.), metode pemrograman membuka kemungkinan kreatif untuk bekerja dengan detail tipe, yang

memungkinkan desainer untuk bermain dan bereksperimen dengan atribut visual mendasar dari bentuk tipografi dan untuk menghasilkan variasi bentuk yang unik dan asli.



Kode-Tipe Kyuha Shim: *Contoh bagian dari proyek "Kode dan Tipe" Kyuha Shim (halaman sebelumnya), yang menggunakan proses komputasi untuk menghasilkan bentuk huruf digital. Aturan kode memungkinkan bentuk diulang dan diubah, dan pengulangan garis yang berputar menciptakan serangkaian huruf berbasis kode yang unik.*

Aturan dan Transformasi

Menerapkan fungsi pemrograman pada bentuk huruf atau kata mendorong desainer untuk secara mendasar memikirkan kembali dan membayangkan kembali bentuk, struktur, dan komposisi tipografinya untuk membangun kembali huruf digital. Aturan pemrograman dapat digunakan untuk menerapkan transformasi matematika yang menghitung ulang bentuk secara visual dengan cara yang dapat diprediksi dan dikendalikan secara numerik.

Kode yang secara sistematis atau acak mengulang, mengubah skala, mengubah, memindahkan, atau memanipulasi bagian atau seluruh huruf dapat menciptakan komposisi dan huruf yang unik. Bentuk huruf individual dapat diubah dan digabungkan menjadi pola ritmis dengan menghitung ulang atau mendekonstruksi posisi atau bentuk huruf secara berulang.

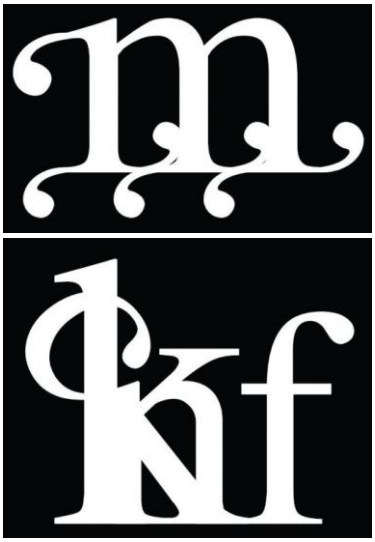
CONTOH

Ricard Marxer: Caligraft

Caligraft adalah proyek eksperimental yang terinspirasi dari kaligrafi tradisional (seni menulis dekoratif) dan menggunakan pemrograman untuk menghasilkan bentuk huruf "kaligrafi" digital. Kode digunakan untuk menafsirkan ulang huruf secara dinamis dalam berbagai cara, mengabstraksi bentuk asli menjadi serangkaian bentuk animasi, yang masing-masing memiliki karakter dan gaya khasnya sendiri. Goresan dan garis digital yang menghidupkan dan menghasilkan huruf mencerminkan estetika kaligrafi tradisional yang digambar dengan tangan dengan cara digital yang unik. Proyek ini berfungsi sebagai serangkaian eksperimen daring yang menyenangkan yang mengeksplorasi cara-cara menciptakan karya "buatan tangan" digital menggunakan kode.

Kaligraft Ricard Marxer

Proyek Caligraft Ricard Marxer menggunakan sistem generatif buatan sendiri untuk bermain-main dengan pembuatan ulang huruf. "Sketsa" menghidupkan dan menggambar karakter dengan cara yang lancar. Serangkaian eksperimen mengeksplorasi berbagai aturan untuk memodifikasi bentuk huruf. Setiap percobaan merupakan representasi visual dari proses desain yang lebih luas di mana serangkaian aturan dan fungsi telah diterapkan secara sistematis untuk menghasilkan karya



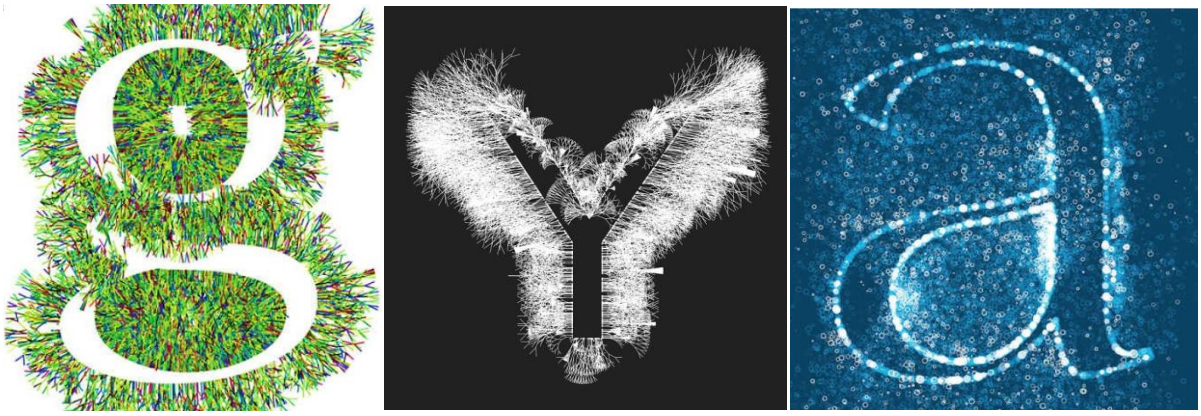
Kaligraft Ricard Marxer

Eksperimen lain dari proyek Caligraft menggabungkan bentuk huruf sesuai dengan aturan generatif yang ditetapkan dengan baik.

Jalur dan Titik Bentuk

Kode mengubah kata dan huruf menjadi bentuk yang lentur secara digital. Titik dan jalur koordinat individual yang menggambarkan dan mendefinisikan titik dan kurva bentuk huruf dapat diubah menggunakan kode untuk mengubah tampilan visualnya secara dinamis. Desainer dapat menggunakan fungsi pemrograman untuk memisahkan bentuk huruf secara visual. Ketika tunduk pada aturan dan gaya komputasional (seperti rekursi, pengulangan, gerakan fisik, keacakan), goresan dan garis, titik dan kurva bentuk huruf dapat dibuat ulang menjadi bentuk tipografi dan menjadi grafik baik untuk layar maupun cetak.

Metode pemrograman yang digunakan untuk mendekonstruksi jenis huruf dan bentuk huruf mendorong semacam inventivitas visual yang dapat mengembangkan versi baru dari bentuk huruf. Jalur garis besar dan kurva huruf dapat digunakan sebagai panduan gerakan yang dianimasikan oleh objek, yang secara visual menciptakan kembali bentuk huruf sebagai serangkaian garis bergerak yang dinamis. Titik-titik individual dari bentuk huruf tunggal dapat digunakan sebagai koordinat untuk titik-titik “penyemaian” tempat garis-garis digital tumbuh dan berkembang biak secara organik atau bahkan sebagai titik-titik “magnetik” yang mengarah ke dan di sekitar tempat grafik, garis, bentuk, dan warna lain tertarik dan berinteraksi.



Yeohyun Ahn

Contoh karya Yeohyun Ahn, seorang tipografer dan desainer visual interaktif, yang mengeksplorasi hubungan antara seni, desain, dan tipografi. Karya tipografi eksperimental ini menggunakan Processing dan pustaka Geomerative milik Ricard Marxer untuk menata ulang bentuk huruf sebagai lingkungan yang dihasilkan oleh kode.

Bentuk Huruf Dari Tanggal

Berbagai sumber data eksternal juga dapat mengubah titik koordinat dan nilai huruf. Misalnya, gerakan tetikus, audio, atau serangkaian penekanan tombol dapat ditangkap dan diterapkan sebagai informasi untuk mengubah detail visual bentuk huruf. Ukuran, bentuk, atau warna tipografi dapat dihasilkan sebagai respons visual terhadap file audio eksternal dari, misalnya, ucapan atau kicauan burung.

Menghubungkan bentuk tipografi dengan sumber data membuka cara baru untuk mengekspresikan dan menciptakan pengalaman tipografi. Kata-kata menjadi ekspresi visual dari pengalaman atau ide tertentu. Jenis huruf komputasional yang unik dapat dibuat yang secara visual mewakili kumpulan data tertentu atau serangkaian tindakan yang ditangkap dan "dibekukan" ke dalam format visual.

CONTOH

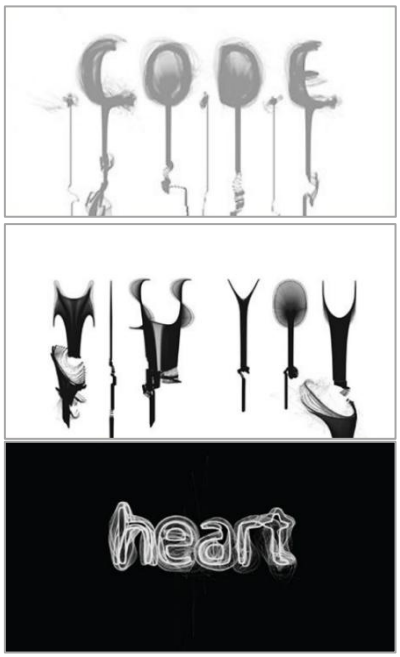
Reza Ali mis.shap.en.ness

Reza Ali adalah seorang programmer dan desainer yang penelitiannya telah membawanya untuk mengeksplorasi kemungkinan visual dalam menerapkan kode untuk menghasilkan bentuk dan rupa secara dinamis. Dengan menghubungkan sumber data dan gaya fisik yang dihasilkan secara komputasional ke dalam kerangka kerja untuk menciptakan bentuk dan gambar, ia telah mampu menghasilkan sekumpulan bentuk eksperimental yang menarik.

Melihat tipografi sebagai bentuk komputasional yang lentur yang bentuknya dapat dibentuk dan diubah sebagai respons terhadap gaya dan data eksternal, Reza telah bereksperimen dengan menciptakan tipografi "generatif": bentuk huruf yang responsif terhadap sistem partikel bawaan serta file audio eksternal.

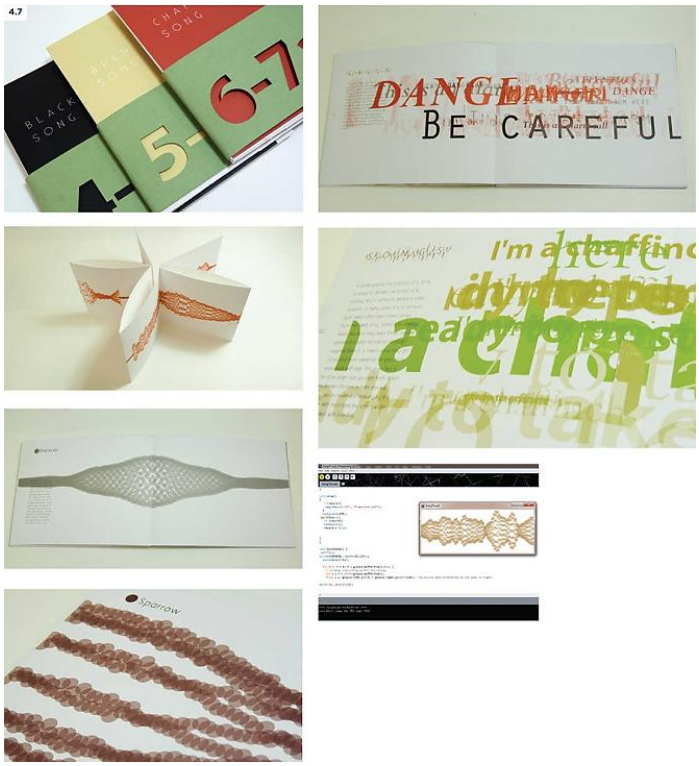
Hasilnya, yang didokumentasikan di blognya, adalah serangkaian bentuk tipografi yang sangat berwarna, cair, dengan kualitas yang dapat diubah secara unik; bentuk-bentuk tersebut berfungsi baik sebagai grafik statis maupun animasi. Eksperimen tersebut menyoroti

bagaimana, setelah mengabstraksi bentuk huruf menjadi serangkaian warna dan titik individual dan menciptakan koneksi dengan data (misalnya, file audio), Reza terus bereksperimen dengan mengubah parameter untuk mendapatkan berbagai hasil yang berbeda. Reza menerapkan dan menggunakan kembali gaya terprogram sederhana (misalnya, pegas dan partikel) untuk memengaruhi warna dan bentuk huruf. Perubahan sederhana dan halus pada parameter dan nilai ini menghasilkan serangkaian visual berbeda yang dihasilkan dari satu konsep.



bentuk tubuh yang tidak terbentuk Reza Ali (mis.shap.en.ness Reza Ali)

Garis besar huruf diubah menjadi bentuk yang fleksibel dengan menerapkan partikel yang dihasilkan kode yang bergerak dan jatuh, mendistorsi kata-kata saat mereka "menetes" ke bawah layar.



Melihat Suara Alam Elena Kalaydzhieva

Dalam contoh karya desain eksperimental ini, tipografi dibuat sebagai respons terhadap berkas audio kicauan burung yang direkam selama paduan suara fajar. Pemrosesan digunakan untuk memvisualisasikan setiap kicauan burung melalui gelombang suara grafis dan memanipulasi tipografi. Hasil cetak akhir mencakup gambar yang dibuat dari aplikasi Pemrosesan.

4.2 GERAKAN DAN JENIS INTERAKTIF

Kode tidak hanya menghasilkan bentuk dan rupa huruf baru, tetapi juga dapat digunakan untuk menciptakan pengalaman tipografi yang beranimasi dan "reaktif": huruf yang bergerak sebagai respons terhadap interaksi pengguna. Metode pemrograman dapat membagi dan membagi lagi struktur formal dari satu baris atau paragraf teks menjadi serangkaian kata, huruf, atau titik individual. Baris teks yang kaku dapat diubah menjadi objek yang fleksibel dan mengambang bebas, yang tunduk pada aturan dan gaya interaktif yang dihasilkan secara internal atau eksternal. Membebaskan huruf dari format aslinya memungkinkan tipografi dianimasikan secara menyenangkan dan interaktif. Huruf individual menjadi objek grafis kinetik yang posisi, titik, dan konturnya dapat dimanipulasi.

Aplikasi kreatif pemrograman memungkinkan kata-kata untuk melompat dari halaman ke lingkungan interaktif: Gerakan tetikus, isyarat pengguna, input suara, dan sebagainya dapat digunakan untuk memutar dan memanipulasi tipografi secara dinamis menjadi bentuk dan gerakan yang fleksibel secara interaktif, membuatnya berputar, berputar, atau menari di layar, tertiup angin virtual dari interaksi yang dihasilkan pengguna. Aturan komputasi dapat membuat tipografi digital menjadi interaktif, yang memungkinkan pemirsa untuk terlibat dengan konten dengan cara baru.

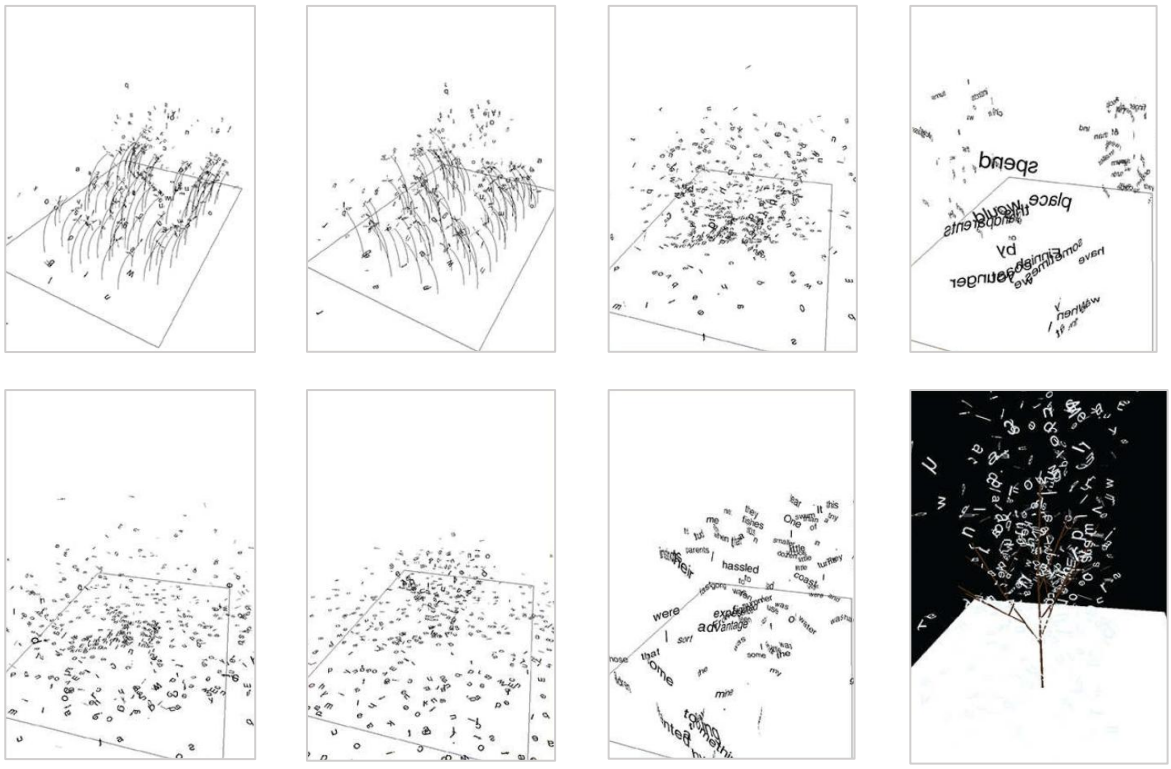
CONTOH

Nanika (Andreas Müller): Aplikasi for All Seasons

Aplikasi For All Seasons adalah versi dari karya tipografi interaktif tahun 2005 yang dibuat oleh Andreas Müller. Karya tersebut merupakan eksplorasi tipografi interaktif yang eksperimental dan menyenangkan. Karya tersebut dibagi menjadi empat bagian teks, yang masing-masing menggambarkan memori yang terkait dengan musim tertentu dalam setahun. Dalam setiap kasus, huruf dan kata dari bagian tersebut berubah menjadi lingkungan interaktif yang sesuai.

Kata-kata tersebut berubah menjadi biji dandelion yang tertiup angin (musim semi), ikan yang berenang (musim panas), tumpukan daun yang gugur (musim gugur), dan kepingan salju yang jatuh (musim dingin). Pengguna dapat menjelajahi dan berinteraksi dengan setiap lingkungan, yang menyebabkan huruf-huruf tersebut tertiup dengan anggun di layar, berenang, atau jatuh dengan lembut di pohon (seperti salju). Gerakan huruf yang organik dan naturalistis mencerminkan lingkungan digital tempat mereka berada, menciptakan asosiasi yang menarik, menyenangkan, dan puitis dengan setiap bagian teks.

Setiap layar adalah lingkungan baru. Huruf-huruf tersebut menjadi objek di lingkungan (benih, ikan, daun, salju) yang digerakkan oleh kekuatan alami dari dunia virtual. Proyek ini merupakan contoh yang baik tentang kedekatan antara lingkungan organik dan digital, karena pergerakan objek yang diprogram mengambil inspirasi dari dan mencerminkan dengan indah apa yang ditemukan di dunia alami.



Untuk Semua Musim Nanika
Penggunaan tipografi yang menyenangkan, di mana emosi dieksplorasi melalui kenangan interaktif dari setiap musim. Setiap lingkungan alam menampilkan huruf dan kata yang tampak bergerak secara organik saat aturan komputasi memandu mereka.

Majalah Born

Sebuah eksperimen awal dalam penggunaan media digital untuk menciptakan pengalaman teks interaktif yang menarik, born (bornmagazine.org) adalah/dulu merupakan proyek kolaboratif eksperimental yang mempertemukan desainer, seniman, dan penulis untuk menghasilkan puisi daring yang interaktif dan mengeksplorasi kualitas puitis dari tipografi yang interaktif dan bergerak.

Born menyediakan tempat bagi desainer, seniman, dan penulis digital untuk berkolaborasi guna menciptakan puisi dan cerita interaktif yang kaya secara visual. Kombinasi pemrograman grafis dan penulisan membantu menciptakan area terbuka untuk mengeksplorasi dan mendorong batasan cara interaktif digital dalam mengalami dan membaca teks.

Zoologi Sasha West (penulis) dan Ernesto Lavandera (artis)
Cuplikan layar dari proyek yang dipublikasikan di majalah daring Born.

Tipografi yang bergerak dan interaktif memberikan kehidupan dan karakter baru pada huruf digital. Kode membuat bentuk huruf yang dianimasikan menjadi interaktif dan dapat diubah; programmer dapat mengubah konten bentuk huruf sesuai dengan masukan dan interaksi pengguna dan memungkinkan pengguna untuk membaca dan merasakan kata-kata dengan cara yang unik, interaktif, dan digital.

Menciptakan lingkungan teks yang interaktif dapat menghidupkan keterlibatan pengguna dengan konten dan mengubah cara kata-kata dipahami. Membaca menjadi pengalaman yang menyenangkan dan puitis, di mana pengguna dapat membangun makna dari eksplorasi kata-kata mereka sendiri. Kehadiran layar genggam dan tablet membawa pengalaman membaca teks digital lebih dekat dengan membaca buku fisik dan menghadirkan lebih banyak peluang untuk membuat konten yang kaya dan interaktif.

4.10

Berjalan Bersama Apa yang Tersisa Chris Green (penulis) dan Erik Natzke (artis)
Cuplikan layar dari proyek yang dipublikasikan di majalah daring Born.

4.11

Bentang Kata Peter Cho
Karya tipografi interaktif Peter Cho merupakan eksplorasi kemungkinan kreatif dari teks interaktif komputasional. Wordscapes merupakan kumpulan dua puluh enam lanskap tipografi interaktif yang menyenangkan.

4.3 TEKS SEBAGAI SUMBER DATA

Kode memungkinkan desainer untuk mengakses dan menggunakan teks sebagai sumber data penting yang dapat diinterpretasikan secara visual. Menggunakan karakteristik sumber teks sebagai informasi untuk mendorong dan mengarahkan gaya dan format visual membangun hubungan antara konten dan penanganan grafisnya. Kode dapat "menambang" detail teks, mengungkap detail tersembunyi (jumlah kata, frekuensi kata, jumlah huruf, panjang baris, panjang kata, hubungan antarkata, dll.), menerjemahkannya ke dalam nilai angka dan menerapkannya untuk menentukan elemen visual (warna, bentuk, geometri, atau rotasi).

Menggunakan kode dengan cara ini membuka peluang untuk menyelidiki struktur dan konten kata, "menggali di bawah" makna permukaan, mengungkap dan memvisualisasikan pola dan hubungan "tersembunyi" dalam konten. Seniman dan desainer dapat menghasilkan asosiasi visual baru mereka sendiri antara teks dan gambar, mengubah dan memvisualisasikan atribut teks yang dipilih dan menciptakan ekspresi visual baru dari kata-kata.

Mengabstraksikan Teks

Ide mengubah teks menjadi sekumpulan atribut visual mungkin tampak sebagai konsep yang abstrak atau sulit. Namun, seperti banyak informasi yang digunakan dalam kode pemrograman, nilai dan atribut dari sepotong teks diproses dan dipahami oleh komputer sebagai angka. Nilai angka membentuk bagian yang sangat penting dari setiap kode pemrograman; nilai tersebut mengatur dan mendefinisikan detail visual setiap objek.

Angka dalam kode adalah nilai transformatif: potongan data fleksibel yang dapat digunakan untuk mendefinisikan dan mengubah banyak elemen dan atribut visual. Angka dapat mendefinisikan warna, bentuk, tekstur, atau gerakan. Bagi desainer yang menggunakan kode, angka menyediakan cara yang berguna dan langsung untuk mengubah dan memanipulasi grafik. Oleh karena itu, gagasan bahwa sebuah tulisan dapat direpresentasikan dalam format angka merupakan kunci untuk memahami bagaimana teks dapat divisualisasikan dalam format grafik. Hal ini membuka pandangan tentang karya desain kreatif.

Data teks dapat diperoleh langsung dari masukan pengguna atau dari data yang disimpan dan dimasukkan dari berkas eksternal. Sumber informasi huruf dan kata dapat \dianalisis secara komputasional, diabstraksikan, dan diubah untuk menghasilkan rangkaian grafik yang unik. Contoh sederhana terkait dengan bagaimana grafik dapat dihasilkan dari masukan keyboard.

CONTOH

Boris Müller: Puisi di Jalan

Puisi di Jalan adalah proyek desain visual eksperimental yang sedang berlangsung berdasarkan festival sastra tahunan yang diadakan di Bremen, Jerman. Setiap tahun sejak 2002, desainer Boris Müller telah ditugaskan untuk membuat tema visual yang mencerminkan karya para penulis yang ditampilkan dalam festival tersebut.

Tantangan grafis, untuk merepresentasikan beragam tulisan dengan cara yang koheren secara visual, merupakan tantangan yang menarik, yang dipecahkan Boris dengan menggunakan kode sebagai cara memvisualisasikan karya sastra. Sistem pemrograman khusus

dibuat untuk mengubah teks menjadi gambar; setiap grafik secara langsung mewakili bagian tulisan tertentu.

Puisi di Jalan Boris Müller
Contoh sampul dari tiga edisi Puisi di Jalan, yang dirancang oleh Boris Müller. Detailnya menunjukkan komposisi yang dibuat sebagai visualisasi dari beberapa bagian puisi.

Meskipun konsep dasarnya tetap sama, setiap sistem pemrograman baru dirancang untuk menggambar dan memvisualisasikan kata-kata dengan cara yang berbeda, menciptakan hubungan baru dan menghasilkan berbagai karya grafis dan interaktif yang menarik. Setiap versi proyek menciptakan interpretasi visual baru dari puisi karena metode yang berbeda untuk mengubah kata-kata menjadi gambar dieksplorasi. Hubungan antara isi puisi dan grafik dibuat dalam berbagai cara yang menarik, yang masing-masing menghasilkan hasil visual yang unik.

Atribut kata dan huruf dalam setiap puisi didekonstruksi dan digunakan untuk membuat bentuk abstrak yang atribut visualnya (warna, bentuk, posisi, arah, dll.) ditentukan oleh nilai angka dan karakteristik yang ditemukan dalam kata itu sendiri (misalnya, jumlah huruf, frekuensi kata, dan ukuran). Dengan metode ini, huruf-huruf individual diabstraksikan ke dalam kotak dan garis berwarna; kata-kata umum ditemukan, diperbesar, dan dihubungkan bersama; dan seluruh puisi digambar sebagai bidang kata "partikel" dengan rangkaian gayanya sendiri.

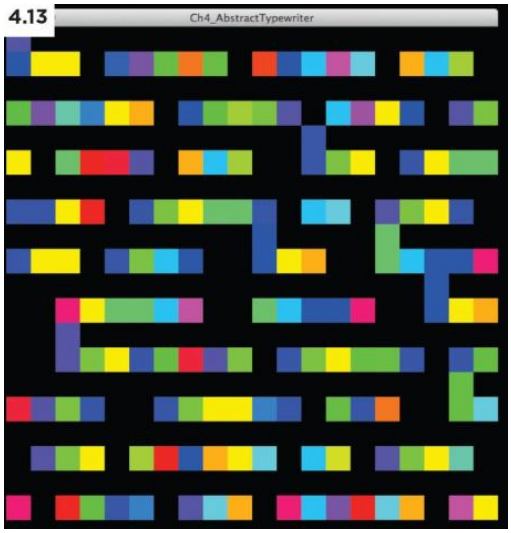
Eksperimen lain mendorong lebih jauh pada hubungan visual antara kata dan gambar: Puisi diubah menjadi kolase gambar daring yang diberi tag, bentuk 3D, dan bahkan serangkaian kode batang interaktif. Hasil eksperimen Poetry on the Road dapat dilihat secara daring, bersama dengan versi interaktif dari program yang digunakan untuk menghasilkan gambar. Karya akhir menyajikan serangkaian grafik mencolok yang sangat beragam yang memberikan wawasan tentang berbagai cara penggunaan kode untuk memvisualisasikan kata-kata sebagai grafik.

4.4 INPUT PENGGUNA KEYBOARD

Setiap huruf dan karakter pada keyboard komputer diberi nilai numerik, yang berasal dari nilai ASCII yang sesuai. Nilai-nilai ini digunakan untuk memungkinkan program pengolah kata membedakan antara penekanan tombol tertentu. Bahasa pemrograman dapat

menangkap penekanan tombol pengguna secara individual saat diketik, menggunakannya untuk membuat dan menentukan variabel yang menentukan bentuk dan atribut grafis tertentu (misalnya, warna, tinggi, bentuk, atau ukuran).

Oleh karena itu, baris huruf dan kata yang diketik dapat diabstraksikan secara visual menjadi serangkaian warna, bentuk, wujud, dan grafik yang sesuai dengan kata-kata yang diketik ke keyboard. Pola visual yang dinamis dapat dihasilkan sebagai respons visual langsung terhadap input keyboard pengguna. Semacam "mesin tik visual" dapat dibuat di mana kombinasi huruf, kata, dan kalimat yang unik divisualisasikan sebagai serangkaian pola bentuk dan warna yang unik. Irama dan nada kata, baris, dan kalimat diubah menjadi pola visual.



Bentuk yang Dihasilkan Oleh Papan Ketik
Terinspirasi oleh "Mesin Ketik Berwarna" milik John Maeda, komposisi abstrak dari bentuk dan warna ini dihasilkan oleh masukan papan ketik. Huruf-huruf yang diketik diabstraksikan menjadi bentuk-bentuk berwarna yang menciptakan komposisi kata dan frasa yang berirama.

Berkas Teks Berukuran Besar

Sama seperti huruf tunggal dan frasa pendek yang dapat digambar dan divisualisasikan ulang, teks yang bersumber dari berkas data eksternal berukuran besar juga dapat diubah menjadi informasi visual. Fungsi pemrograman memberi desainer alat untuk membaca dan menganalisis konten digital dari ribuan baris teks dalam hitungan beberapa detik. Nilai data yang ditambang dari sepotong teks berukuran besar dapat diabstraksikan menjadi visualisasi grafis atau digunakan untuk menggambar ulang atau mengubah gaya tipografi kata-kata itu sendiri.

Berkas digital dari seluruh buku dapat dibaca dengan cepat dan efisien ke dalam program menggambar, dianalisis secara komputasi, dibedah, dan dipecah menjadi kelompok paragraf, kalimat, atau kata yang berguna. Kekuatan pemrosesan komputer yang berulang memungkinkan satu fungsi, yang digunakan untuk menemukan dan memvisualisasikan satu kata, untuk diulang dan diterapkan ribuan kali dan pada ribuan kata.

Rincian dan kerumitan struktural dari seluruh bagian atau halaman buku dapat dengan cepat ditemukan dan diformat ulang ke dalam format visual baru (misalnya, sebagai gambar yang diinformasikan data atau potongan tipografi). Pola, koneksi, dan karakteristik tersembunyi dalam teks dapat ditemukan, diungkapkan, dan divisualisasikan.

Misalnya, menghitung berapa kali kata-kata "kunci" diulang dapat mengungkapkan ide dan tema umum dalam teks. Frekuensi kata-kata tertentu dapat ditemukan dan disimpan sebagai angka yang digunakan untuk memengaruhi gaya visual teks misalnya, untuk mengatur ukuran font kata-kata yang muncul kembali. Teknik ini umumnya digunakan dalam "Wordles," awan teks yang menyorot frasa dan jumlah kata penting, tetapi format visual lain juga dapat menggunakannya.

Data teks digital tersedia dari sejumlah sumber yang berbeda; misalnya, versi digital dari banyak karya sastra klasik tersedia secara gratis daring untuk diunduh dari situs-situs seperti Project Gutenberg *(www.gutenberg.org)*. Sebuah novel dapat menjadi sumber data teks yang menarik; Kemampuan untuk menerapkan fungsi pemrograman pada keseluruhan buku menciptakan peluang luar biasa bagi para desainer untuk memanipulasi konten, menghadirkan pendekatan baru pada teks yang sudah dikenal, dan menciptakan karya desain yang tidak mungkin dicapai dengan cara lain. Seluruh bagian dan buku teks dapat ditampilkan ulang dalam format visual yang mencerminkan tema kontennya. Misalnya, kemunculan kata atau frasa tertentu, atau ritme dan pola dalam kelompok kata, dapat ditampilkan secara visual dan diterapkan pada gaya tipografi, yang memperlihatkan pola dan karakteristik teks.

Awan Kata (Wordle Cloud)
"Wordle" adalah contoh dari "word cloud," yang memvisualisasikan frekuensi relatif kata-kata dalam sepotong teks.

CONTOH

Stephan Thiel: Memahami Shakespeare

Proyek Memahami Shakespeare adalah contoh bagus tentang data yang digunakan untuk menginformasikan konten teks secara visual, mengubah cara teks tersebut dibaca dan dipahami. Proyek yang diselesaikan sebagai proyek pascasarjana oleh mahasiswa desain antarmuka Stephan Thiel, mengeksplorasi dan meneliti karya-karya Shakespeare. Thiel menggunakan pemrograman untuk mengekstrak dan secara grafis merepresentasikan potongan informasi tertentu yang ditemukan dalam teks.

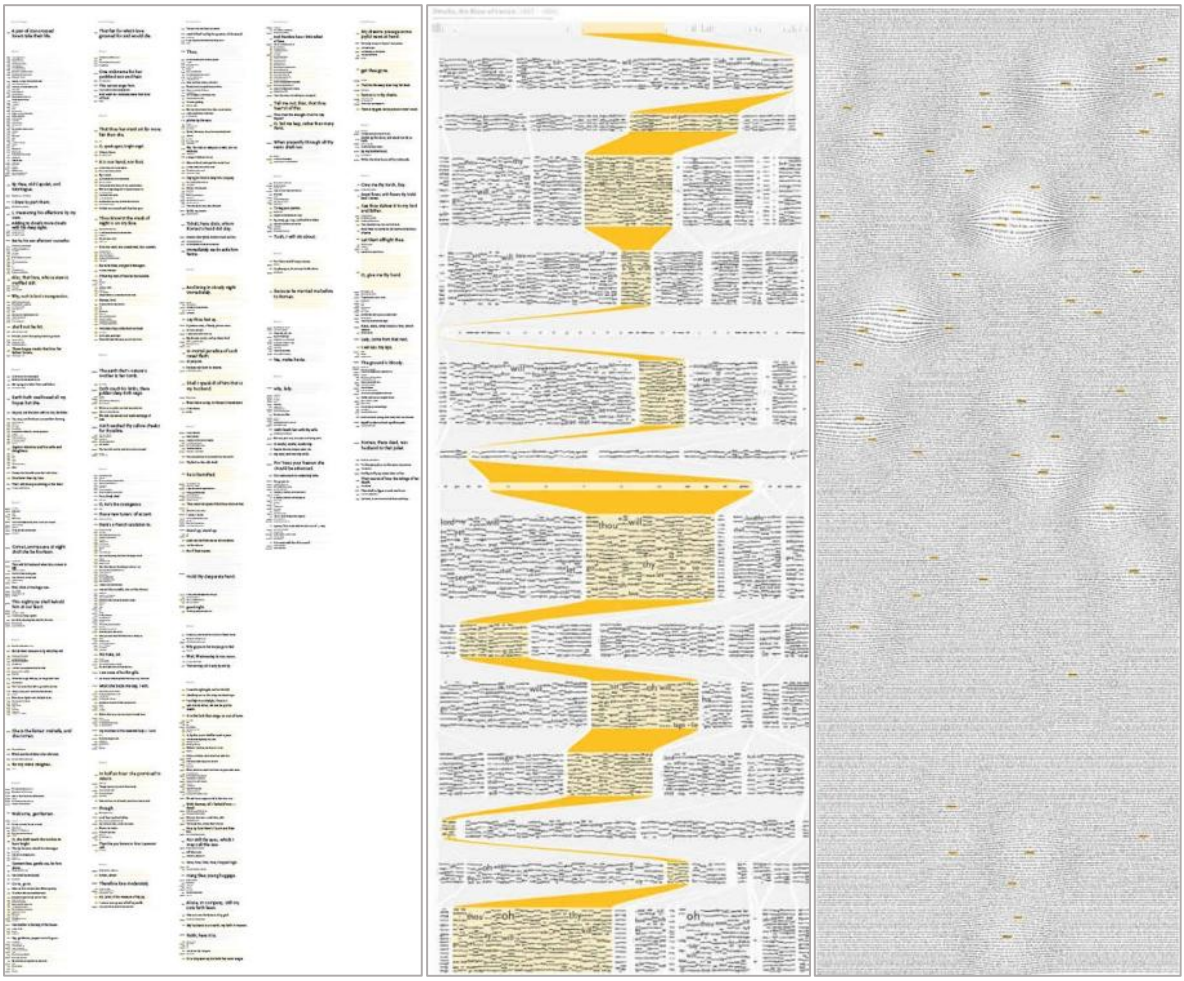
Dengan menggunakan kode untuk memvisualisasikan kembali bagian-bagian drama Shakespeare, proyek tersebut mengungkap elemen-elemen tersembunyi dari narasi yang mendasarinya dalam teks dan membuka cara-cara baru untuk membaca karya tersebut. Proyek tersebut menggunakan alat pemrograman untuk menganalisis drama dalam lima cara berbeda, menghasilkan untuk masing-masing drama serangkaian gambar yang secara grafis merangkum elemen-elemen utama yang ditemukan dalam setiap narasi, sehingga merepresentasikan kumpulan karya yang mengesankan ini dalam format yang ringkas dan menarik secara visual.

Fungsi pemrograman digunakan untuk memilih dan menunjukkan kata-kata yang sering digunakan untuk karakter dalam setiap drama; kalimat yang merangkum pidato; contoh pernyataan yang dimulai dengan kata ganti orang, seperti saya, saya, kami, dan seterusnya; petunjuk panggung untuk menunjukkan pintu masuk dan keluar yang dramatis; dan elemen populer dari setiap lakon, menurut penelusuran Google. Hasil dalam setiap kasus menyajikan ikhtisar lakon secara grafis. Untuk setiap penelusuran data, serangkaian cetakan skala besar diproduksi, yang memungkinkan pemirsa memperoleh ikhtisar visual yang luas tentang pola dan ritme dalam struktur dramatis setiap lakon. Semua informasi diringkas dan disajikan dalam

format buku.

Understanding Shakespeare adalah contoh hebat dari penceritaan visual di mana metode pemrograman telah diterapkan secara kreatif ke bidang komunikasi grafis. Perancang telah menggunakan dan menerapkan data dengan cara yang bijaksana dan bermakna, menerapkan kemampuan kode untuk mengekstrak kata dan frasa tertentu untuk menciptakan hasil visual yang dipimpin oleh desain. Hasil akhir adalah serangkaian gambar yang kuat yang memberikan pendekatan visual baru pada teks lama dan familiar.

Memahami Shakespeare Stephan Thiel



Contoh gambar dari proyek, yang menghasilkan visualisasi dinamis dari beberapa karya Shakespeare yang paling terkenal, yang memungkinkan pemirsa untuk "membaca" dan memahami teks yang sudah dikenal dengan cara baru.

4.5 DATA EKSTERNAL

Selain menggunakan dan mengungkap informasi dan karakteristik dari dalam sebuah tulisan, gaya visual dari sebuah teks digital juga dapat dipengaruhi oleh sumber data di luar teks. Informasi eksternal yang memiliki relevansi langsung atau hubungan dengan konten teks dapat digunakan untuk mengubah detail bentuk dan rupa teks, menciptakan komposisi visual

yang merangsang secara visual dan intelektual, menyajikan kata-kata dalam format yang mengungkap aspek dan asosiasi baru dalam teks yang jika tidak demikian akan tersembunyi. Hubungan visual yang menarik dapat dibuat yang mendorong pemirsa untuk membaca teks dengan cara baru, membangun hubungan baru yang menarik perhatian pada hubungan antara teks dan konteks atau lingkungannya yang lebih luas.

Misalnya, aktivitas Twitter, data GPS, atau aktivitas daring lainnya, dapat ditautkan ke bagian teks untuk mengungkap aktivitas daring di sekitar teks tertentu atau untuk menghubungkan kata-kata ke lokasi atau lanskap tertentu. Teks bahkan dapat ditautkan ke berkas suara; kode menafsirkan elemen trek audio untuk menghasilkan ritme visual dalam tipografi. Menciptakan hubungan antara data dan tipografi menciptakan hubungan langsung antara makna dan atribut visual kata-kata, yang didefinisikan oleh data itu sendiri. Menggunakan dan menerapkan data pada teks dan tipografi dengan cara ini memberi desainer sejumlah alat kreatif baru untuk mengomunikasikan aspek konten tertulis yang bermakna secara visual.

CONTOH

Vladimir V. Kuchinov: Generative Gatsby

Generative Gatsby adalah proyek yang dibuat oleh Vladimir V. Kuchinov yang mengubah tipografi prosa asli dengan mengubah data dari ritme sembilan standar jazz "big band" dari tahun 1920an. Partitur musik, yang dapat dimainkan di pesta-pesta Gatsby, digunakan sebagai templat untuk tipografi berbasis kode, yang menggunakan nada, kecepatan, posisi, dan durasi setiap not untuk menentukan posisi dan gaya huruf dan kata di halaman.

Lebih jauh, setiap instrumen dalam aransemen musik (piano, gitar, bas, brass, dan perkusi) diberi jenis huruf tertentu yang dianggap menggugah atribut visual atau emosional instrumen tersebut. Misalnya, senar gitar jazz diwakili oleh jenis huruf tipis modern (Brandon Grotesque Thin) dan drum serta perkusi diwakili oleh jenis huruf mesin ketik klasik (Remington Typewriter) yang menggugah suara ritmis dan perkusi mesin ketik pada tahun 1920an. Tata letak akhir yang dihasilkan oleh proses desain generatif ini menonjolkan ritme era jazz,

Generatif: Gatsby Vladimir V. Kuchinov
Pengerjaan ulang visual novel The Great Gatsby (1925) yang menggunakan ritme dan not dari partitur musik jazz untuk menafsirkan ulang tipografi teks aslinya.

mengubah novel tersebut menjadi ekspresi tipografi visual dari gaya musik Roaring Twenties.

CONTOH

Ariel Malka

Ariel Malka adalah seorang desainer dan programmer yang telah menciptakan berbagai proyek interaktif. Sebagian besar penelitian kreatifnya difokuskan pada eksperimen dengan berbagai bentuk tipografi interaktif. Karyanya dicirikan oleh penciptaan lingkungan yang mengubah kata-kata menjadi untaian dan pita teks yang bergerak dan mengalir, yang memungkinkan pengguna untuk membaca dan merasakan konten dengan cara yang menyenangkan dan menarik. Dua hasil yang biasanya menarik untuk proses penelitian kreatifnya yang sedang berlangsung adalah "Javascriptorium" dan transformasi interaktif yang lebih baru dari Ulysses karya James Joyce (1922): "He Liked Thick Word Soup."

Javascriptorium: **Ariel Malka**
Ariel Malka menggunakan pemrograman untuk menghasilkan versi teks spiritual keagamaan yang dianimasikan secara dinamis dan interaktif; ini adalah karya tipografi interaktif yang sangat puitis yang mengeksplorasi hubungan antara teks dan konten.

Dia Suka Sup Kata Tebal: Ariel Malka
Aplikasi Ariel Malka mengubah bagian-bagian teks dari Ulysses (1922) karya James Joyce menjadi pita-pita teks interaktif. Tindakan membaca menjadi pengalaman yang rumit, menantang, tetapi menyenangkan yang selaras dengan karakteristik karya aslinya

.

4.6 TIPOGRAFI KOMPUTASI

Anatomi Teks sebagai Data

Fungsi pemrograman dapat digunakan untuk menangkap dan menampilkan teks di layar. Teks dalam program disimpan sebagai data String. String adalah serangkaian karakter (biasanya huruf) yang "dirangkai bersama" di antara serangkaian tanda baca:

```
//example String
String s = "a String of characters";
```

Huruf atau karakter individual disimpan sebagai tipe data char.

```
char c = 'a'; // a single character
```

String pada dasarnya adalah daftar karakter individual dan dapat ditulis sebagai berikut:

```
char[] letters = {'S', 't', 'r', 'i', 'n', 'g'};
```

Data String dapat ditampilkan dalam berbagai cara dan dapat diakses dari berbagai sumber. Setelah disimpan atau dibuat dalam program, data String dapat dimanipulasi dan diformat ulang dalam berbagai cara visual yang berbeda. Menampilkan dan Memformat Teks

Sebagian besar bahasa pemrograman memiliki fungsi untuk menampilkan teks di layar. Dalam Pemrosesan, fungsi untuk menampilkan kata adalah fungsi text(). Fungsi text() menggunakan tiga parameter (argumen): String teks yang akan ditampilkan, dan sepasang angka untuk menentukan lokasi x dan y kata di layar.

```
text (String, x, y);
// put message on screen at x=10, y=50
text ("hello world", 10, 50);
```

Dalam contoh ini, font, ukuran, dan warna default digunakan. Fungsi tambahan dapat mengubah ukuran dan warna font. Ukuran font dapat diatur oleh fungsi textSize(). Warna font diatur menggunakan fungsi fill():

```
textSize(48);
fill (255,0,0);
text ("hello", 10, 50);
```

Contoh Pemrosesan Jenis yang Digambar Ke Layar.

Memilih dan Mengontrol Font (PFont)

Untuk menambahkan dan menggunakan lebih banyak font dalam Processing, kelas PFont digunakan. Kelas PFont dapat memuat font ke dalam sketsa, siap untuk ditampilkan di layar. Sebelum font baru dapat digunakan, berkas fontnya harus dibuat dan disimpan di dalam folder "data" di dalam sketsa; ini dilakukan melalui menu Tools > Create Font, yang membuat

berkas font (.vlw) yang diperlukan.

Setelah berkas font ditambahkan ke dalam folder data, berkas tersebut siap untuk dimuat ke dalam sketsa melalui kelas PFont. Fungsi loadFont() memuat berkas font yang siap digunakan. Fungsi textFont() memilih font yang akan digunakan di layar.

```
PFont myFont; // create a new PFont object to use
// load the font,
the file has to be in the data folder
myFont = loadFont("LetterGothicStd-38.vlw");
// select the loaded font and set the size
textFont (myFont, 38);
text ("word", 10, 50);
```

Kelas Processing PFont memberikan opsi untuk memilih dan menggunakan berbagai font di layar.

Menggunakan Variabel dengan Teks

Menggunakan variabel alih-alih nilai tetap memungkinkan teks mengubah posisi atau kontennya. Elemen data yang digunakan dalam fungsi text untuk mengatur konten dan lokasi kata dapat diganti dengan nilai variabel yang lebih berguna. Variabel String dapat menggantikan parameter pertama dan variabel angka dapat menggantikan nilai yang mengatur lokasi layar teks x dan y:

```
String message = "hello world";
float xpos = 100;
float ypos = 50;
text (message, xpos, ypos);
```

Perubahan pada konten salah satu variabel akan mengubah cara teks ditampilkan. Penggunaan variabel dapat mengubah posisi dan konten teks di layar secara dinamis. Dalam contoh berikut, variabel digunakan untuk mengganti angka tetap yang menetapkan lokasi x. Mengubah variable misalnya, dengan menambah 10 akan memperbarui lokasi x teks, membuatnya bergulir secara horizontal di sepanjang layar.

```
float xpos = 10;
void draw() {
text ("hello", xpos, 100);
xpos += 10;
}
```

Variabel sistem mouseX, mouseY juga dapat digunakan untuk memindahkan teks secara dinamis dengan lokasi kursor mouse.

```
void draw () {
```

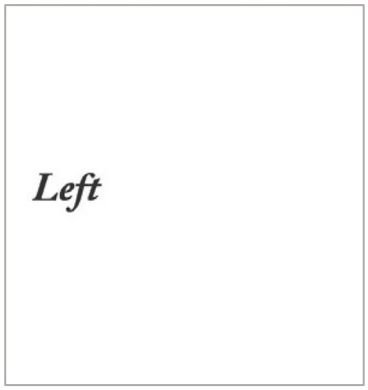
```
  text ("hello", mouseX, mouseY);
}
```



Menggunakan variabel mouseX dan mouseY untuk mengatur lokasi x dan y teks akan membuat teks mengikuti pergerakan kursor.

Selain mengubah lokasi, variabel juga dapat mengubah konten teks secara dinamis. Dalam contoh berikut, variabel String (“pesan”) digunakan untuk mengatur kata yang akan ditampilkan. Konten variabel berubah seiring gerakan tetikus secara horizontal. Saat tetikus berada di sisi kiri layar (<50), nilai variabel, “pesan,” berubah menjadi “kiri,” dan saat tetikus berada di kanan, variabel mengubah teks menjadi “kanan.”

```
String message =“”;
void draw() {
  background(0);
  text (message, mouseX, 50);
  if (mouseX<50) message = “left”;
  if (mouseX>50) message = “right”;
  }
```



Menggunakan variabel sebagai konten teks dapat menciptakan jenis interaksi sederhana: Kata-kata dapat berubah saat mouse digerakkan ke sisi kiri dan kanan layar.

Memanipulasi String

Pemrosesan mencakup berbagai fungsi yang berguna untuk menganalisis karakteristik String. Fungsi dapat digunakan untuk membandingkan String, menemukan karakter tertentu, atau beralih antara huruf kecil dan huruf besar. Berikut adalah contoh beberapa fungsi utama: charAt (n): menemukan huruf ke-n ("karakter") dalam String.

`indexOf(string)`: menemukan kemunculan pertama String yang ditentukan. Dapat digunakan untuk menemukan pertama kali kata atau bagian dari kata digunakan.

`length()`: mendapatkan jumlah karakter dalam bagian String.

`equals()`: membandingkan satu String dengan yang lain.

`subString()`: menemukan sub-bagian dari String.

`split()`: digunakan untuk membagi String menjadi beberapa bagian terpisah.

Memprogram komputer untuk memilah teks dengan cepat dengan cara ini dapat menjadi sumber interpretasi kreatif dan visual yang bermanfaat.

Memisahkan String

Memindahkan dan memanipulasi seluruh kata sebagai satu blok memiliki penggunaan visual yang terbatas. Memanipulasi karakter individual dari sebuah String dapat menghasilkan hasil yang lebih menarik secara visual.

Sebuah String adalah daftar, sebuah array, dari karakter individual. Seperti array lainnya, sebuah String memiliki fungsi “length()” yang mengembalikan jumlah karakter yang dikandungnya:

```
String message = “hello world”;
int numOfChars = message.length();
println(numOfChars); // outputs 11
```

Karakter tertentu dalam String dapat ditemukan menggunakan fungsi charAt(). Fungsi charAt() mengembalikan karakter yang ditemukan pada titik tertentu (indeks) dalam String. Karakter pertama berada pada titik indeks 0 sehingga ditemukan oleh charAt(0);

```
String message = “hello world”;
// finds the first character ('h')
message.charAt(0);
```

Setiap karakter dalam String dapat ditemukan dan ditampilkan sebagai berikut:

```
String message = "hello world";
println (message.charAt(0)); // prints 'h'
println (message.charAt(1)); // prints 'e'
println (message.charAt(6)); // prints 'w'
println (message.charAt(10)); // prints 'd'
```

Cara yang lebih efisien untuk menemukan setiap karakter dari sebuah String secara bergantian adalah dengan menggunakan loop “for”, yang menyediakan cara cepat untuk melakukan loop melalui daftar karakter:

```
String message = “hello world”;
for (int i=0; i<message.length(); i++)
{
  char letter = message.charAt(i);
  println(letter);
}
```

Dengan menambahkan fungsi text() dalam loop “for”, struktur dasar ini dapat digunakan untuk menempatkan dan menggambar setiap huruf di layar satu per satu.

```
float xloc = 10; // starting loc for first letter
String message = "hello world";
for (int i=0; i<message.length(); i++) {
  char letter = message.charAt(i); // get a letter
  text (letter, xloc, 50); // draw a letter at an xloc
  xloc += 10; // increase xloc
}
```

Memisahkan String untuk menggambar huruf satu per satu berarti bahwa setiap karakter dapat diubah secara individual dan digambar dengan cara yang unik. Dalam contoh di bawah ini, fungsi ditambahkan ke loop "for" untuk menetapkan nilai ukuran dan warna acak ke setiap karakter dalam String.

```
float xloc = 10;
String message = "hello world";
for (int i=0; i<message.length(); i++) {
  char letter = message.charAt(i); // get a letter
  fill (random (255)); // select a random fill color
  // select a random text size
  textSize (random (4, 50));
  text (letter, xloc, 50);
  xloc += textWidth(letter);
}
```

Memisahkan serangkaian teks menjadi karakter-karakter individual memungkinkan huruf-huruf digambar secara individual, masing-masing dengan ketebalan dan tinggi yang berbeda.

Menambahkan Animasi: Bergoyang, Memutar

Kemampuan untuk menemukan dan menggambar setiap karakter dalam String membuka peluang untuk menganimasikan kata-kata dengan menggerakkan setiap huruf secara independen. Dalam contoh berikut, sebuah variabel digunakan untuk menggeser lokasi “x” setiap huruf secara acak dan memvariasikan tampilan keseluruhan kata. Menempatkan kode untuk melakukan ini dalam fungsi “draw()” yang berulang terus-menerus akan menggambar ulang teks dan menciptakan efek “goyang” keseluruhan dari huruf-huruf yang

dianimasikan:

```
String message = “hello world”;
void draw () {
  background (125);
  float xloc = 10;
  for (int i=0; i<message.length(); i++) {
    char letter = message.charAt(i);
    float offset_x = random (-3, 3);
    text (letter, xloc + offset_x, 50);
    xloc += textWidth(letter);
  }
}
```

Setiap huruf digambar ulang secara acak dari lokasi aslinya, sehingga menciptakan efek “animasi” sederhana untuk setiap huruf.

Huruf sebagai Objek Dinamis

Kemampuan untuk mengubah dan menganimasikan huruf-huruf individual dapat ditingkatkan dengan menggunakan teknik pemrograman berorientasi objek (OOP). Proses pemrograman berorientasi objek dapat meningkatkan kualitas perilaku dan fleksibilitas huruf-huruf individual, mengubah masing-masing menjadi objek “reaktif” yang beranimasi dengan serangkaian fungsi, perilaku, dan atributnya sendiri.

Fungsi OOP dapat digunakan untuk mengambil huruf-huruf individual dan menerapkan perilaku animasi dan interaktif pada masing-masing huruf. Berikut ini adalah contoh kelas Huruf “generik” yang berisi variabel untuk menyimpan huruf, lokasinya, dan kecepatannya. Kelas ini juga berisi fungsi untuk memindahkan dan menggambar huruf:

```
class Letter {
float ypos;
float xpos;
char letter;
float speed;
Letter (char let, float x, float y) {
  letter = let;
  xpos = x;
  ypos = y;
  speed = random (1, 5);
}
void moveLetter () {
  ypos += speed;
```

```
}
void drawLetter () {
  text (letter, xpos, ypos);
}
```

Contoh kelas ini dibuat menggunakan perintah “new”, yang memanggil fungsi “constructor” di kelas:

```
Letter letter1 = new Letter (‘a’, 100, 200);
```

Setelah dibuat, fungsi di dalam kelas untuk memindahkan dan menggambar contoh huruf dipanggil sebagai berikut:

```
letter1.moveLetter();
letter1.drawLetter();
```

Susunan contoh Letter dapat dibuat dari setiap karakter individual dari sebuah String. Contoh berikut mengambil sebuah String dan mengubah setiap karakter secara bergantian menjadi contoh Letter baru:

```
String message = “drop”;
Letter [] letters;
float xpos = 10;
void setup () {
  // create an array for the letters
  letters = new Letter [message.length()];
  for (int i=0; i<letters.length; i++) {
    // addan letter instance
    letters[i] = new Letter (message.charAt(i), xpos, 20);
    xpos += 10;
  }
}
```

Setiap contoh objek huruf dibuat dengan karakter, lokasi, dan kecepatan acak tertentu. Fungsi moveLetter() dan drawLetter() dapat dipanggil untuk setiap contoh dalam array untuk menganimasikannya.

```
void draw () {
  background (125);
  for (int i=0; i<letters.length; i++) {
    letters[i.moveLetter();
    letters[i.drawLetter();
  }
}
```

Pemrosesan: Penggunaan pendekatan berorientasi objek untuk menggambar huruf memungkinkan setiap huruf memiliki serangkaian perilakunya sendiri.

Penggunaan OOP dengan cara ini memungkinkan huruf menjadi objek yang dianimasikan dan reaktif secara individual dengan kemampuan untuk bergerak secara independen satu sama lain. Lebih banyak fungsi dapat ditulis ke dalam kelas Letter untuk memperluas jangkauan gerakan dan perilaku untuk setiap huruf.

Muat String

Selain menggunakan variabel String (pendek) yang ditulis di dalam program, data String dapat dimuat dari sumber eksternal. Fungsi loadStrings mengimpor konten file teks eksternal dan membuat larik dari baris-baris individualnya.

```
// create an array of the lines of text from an external file
string [] lines = loadStrings("wutheringHeights.txt");
// display the number of lines in the text
println ("number of lines = " + lines.length);
```

Setelah diimpor, setiap baris teks dalam array dapat dibaca dan ditampilkan di layar:

```
float ypos = 20;
String [] lines = loadStrings("wutheringHeights.txt");
for (int i = 0; i < lines.length; i++) {
  text(lines[i], 10, ypos); // draw a line of text
  // change the y position for the next line of text
  ypos += 10;
}
```

Dengan menggunakan fungsi split() atau splitTokens(), baris teks dapat dibagi lagi menjadi larik kata-kata individual. Baik fungsi split maupun splitTokens() "membagi" String (misalnya, baris teks) menjadi larik bagian-bagian individual (misalnya, kata-kata). Karakter tertentu digunakan untuk menentukan di mana pemisahan ini terjadi (biasanya spasi di antara setiap kata). Tidak seperti split(), splitTokens() dapat menggunakan lebih dari satu karakter untuk membagi String, dan juga dapat menghapus tanda baca tambahan serta spasi.

```
// example using split () and splitTokens ()
String message = “Hi. A greeting!”; // String to be divided up
```

```
    // splits String at the spaces: creates an array of 3 words
    String words1[] = split (message, “”);

    // splits String at the spaces and removes the punctuation (., and!)
    String words2[] = splitTokens (message, “!”);
```

Mendapatkan kata-kata individual dari berkas teks yang besar merupakan titik awal yang berguna untuk menggali lebih dalam teks: misalnya, mencari pola atau kemunculan kata. Kata-kata dalam setiap baris dapat dihitung (untuk mendapatkan jumlah kata secara keseluruhan) atau diperiksa untuk jumlah kemunculan kata tertentu. Dalam contoh berikut, setiap baris teks dibagi menjadi kata-kata. Setiap kata kemudian dibandingkan untuk melihat apakah kata tersebut cocok dengan nama karakter dari novel. Setiap kali kecocokan ditemukan, variabel “wordSearch” akan bertambah satu.

```
String [] lines = loadStrings("wutheringHeights.txt");
int wordSearch = 0;
int wordCount = 0;

for (int i = 0; i < lines.length; i++) {
  String words [] = splitTokens(lines[i]," ,;:.!?”);

  wordCount += words.length;
  for (int n = 0; n < words.length; n++) {
    // look for a specific word
    if (words[n]equals("Heathcliff")) {
      wordSearch++;
    }
  }
}
// print the total number of words
Println ("total words = " + wordCount);

// print the number of words matching the search term
Println ("word search count = " + wordSearch);
```

Dengan cara ini, sejumlah besar data teks dapat dengan mudah disortir dan disaring, mengungkap pola tersembunyi penggunaan kata yang dapat divisualisasikan dalam sejumlah cara berbeda.

BAB 5
MELIHAT DUNIA

"Visi adalah seni melihat apa yang tidak terlihat oleh orang lain."
Jonathan Swift

5.1 RUANG DIGITAL

Lingkungan modern kita telah menjadi ruang yang semakin dipenuhi dengan objek dan pengalaman digital di mana garis antara fisik dan virtual telah kabur. Media digital lebih fleksibel dari sebelumnya; ia ditemui dalam berbagai cara dan tempat dan semakin banyak mengelilingi kita dalam kehidupan sehari-hari. Teknologi telah memindahkan gambar digital dari batasan layar komputer desktop ke lingkungan dan tempat baru, yang berarti bahwa gambar tersebut dapat ditemui sebagai bagian dari pengalaman kita sehari-hari.

Gambar interaktif yang diproyeksikan digunakan untuk mengubah ruang atau objek sehari-hari yang biasa saja (misalnya, gedung, kamar, meja) menjadi lingkungan pengguna yang imersif dan hidup. Proyeksi layar besar, perangkat genggam layar kecil, dan permukaan interaktif menggunakan teknologi digital untuk melengkapi ruang ritel, galeri, museum, dan area lainnya.

Ruang Ritel

Lingkungan ritel difokuskan pada tampilan dan selalu ingin menggunakan cara yang menarik perhatian untuk menarik perhatian pada merek dan produk mereka. Dalam menghadapi persaingan ketat dari belanja daring, toko fisik semakin mengeksplorasi interaktivitas untuk memberi informasi, melibatkan, dan menghibur, menjadikan berbelanja sebagai pengalaman yang lebih menarik. Tampilan digital yang cerah kini menjadi fitur umum dari belanja modern; layar sentuh menawarkan informasi; monitor berfungsi sebagai papan reklame bergerak. Gambar yang diproyeksikan ke dinding atau lantai dapat dibuat interaktif dengan menggunakan kamera dan sensor yang mendeteksi gerakan manusia dan memungkinkan pengunjung untuk "menendang" dedaunan virtual atau bermain dan berinteraksi dengan hewan virtual. Tampilan digital di dalam toko dapat mencakup cermin virtual interaktif yang memungkinkan pembeli untuk melihat diri mereka sendiri mengenakan pakaian yang berbeda dari toko.

Etalase toko dan etalase yang besar dapat diubah dengan kamera dan sensor menjadi pengalaman interaktif yang menghasilkan grafik atau pesan interaktif dengan merasakan dan menanggapi gerakan manusia di depan toko.

Ruang Galeri/Museum

Dengan latar belakang teknologi yang semakin berkembang dan persaingan untuk mendapatkan perhatian khalayak yang semakin paham teknologi, museum dan galeri juga semakin bersemangat untuk memanfaatkan teknologi interaktif guna meningkatkan pengalaman pengunjung dan meningkatkan keterlibatan serta pemahaman terhadap berbagai benda dalam koleksi museum.

Perangkat interaktif telah merevolusi cara museum menyajikan pameran, memperluas

pengalaman pengunjung melampaui batasan vitrin kaca konvensional dan pendekatan "lihat tanpa menyentuh". Teknologi ini menghadirkan lingkungan imersif melalui layar besar maupun kecil yang memungkinkan eksplorasi koleksi dan navigasi pameran secara lebih mendalam dan personal. Inovasi ini mencerminkan perkembangan metode baru dalam penyampaian konten museum.

Penggunaan layar sentuh dan citra digital yang aman menciptakan antarmuka grafis interaktif, seperti model tiga dimensi, peta interaktif, serta visualisasi data lainnya yang efektif dalam menyampaikan informasi kompleks yang sulit dijelaskan secara verbal.

Selain itu, teknologi ponsel pintar memfasilitasi akses pengunjung terhadap informasi tambahan mengenai elemen-elemen tertentu dalam pameran, serta membantu mereka menavigasi ruang pamer secara lebih efisien. Teknologi realitas tertambah *Augmented Reality* (AR) juga berperan dalam memperkaya konteks fisik pameran dengan menghadirkan representasi tempat, tokoh, dan peristiwa yang relevan, sehingga narasi pameran dapat tersampaikan secara lebih hidup.

Dengan pendekatan ini, pengunjung dapat menggali latar belakang objek yang dipamerkan dan mengikuti alur cerita sejarah yang disusun, menciptakan pengalaman mendalam yang menyerupai perjalanan melintasi ruang dan waktu ke era atau lokasi tertentu yang menjadi fokus pameran.

Bermain Game

Terlibat dengan konten tidak harus berarti mengisi kepala dengan fakta dan angka; bermain game dan keceriaan secara umum juga dapat menjadi cara yang baik untuk melibatkan pemirsa dan pengunjung. Informasi yang kurang nyata dapat ditawarkan kepada pengunjung dengan cara yang eksploratif. Membungkus ide dan konsep ke dalam lingkungan game merupakan cara yang baik untuk memungkinkan orang menemukan dan menjelajahi subjek yang sangat berguna untuk mengeksplorasi ide yang lebih abstrak, seperti kerja sama atau kreativitas.

Museum Dunia Bede

Penanda Skala Besar

Lingkungan interaktif yang merasakan atau mendeteksi gerakan manusia dapat membenamkan pemirsa dengan cara yang kuat dan menarik, mendobrak batasan layar, mouse, dan keyboard, yang mengarah ke pengalaman interaktif yang memuaskan. Karya yang melibatkan pemirsa secara menyenangkan

dalam pengalaman "menyeluruh", di mana gerakan tubuh digunakan sebagai pemicu interaksi, dapat menjadi cara yang ampuh untuk melibatkan audiens untuk tujuan pendidikan dan komersial.

Augmented Reality (AR) digunakan sebagai cara untuk menghidupkan objek-objek di museum. Saat dilihat melalui ponsel pintar atau perangkat serupa, objek-objek menjadi "ditambah" dengan gambar-gambar tambahan yang dibuat secara digital atau model-model 3D. Aplikasi Museum Dunia Bede menggunakan AR untuk mempertemukan pengunjung secara langsung dengan hantu Yang Mulia Bede.

Uji coba untuk konsep augmented reality ini menggunakan penanda skala besar untuk menghasilkan pengalaman AR yang sangat besar dalam hal ini, mamut virtual.

5.2 PROYEKSI LAYAR BESAR - GERAKAN TUBUH

Proyeksi layar besar memungkinkan lingkungan merespons kehadiran figur dalam suatu ruang; proyeksi ini dapat digunakan dalam berbagai konteks. Baik di etalase pusat perbelanjaan besar atau di ruang interior terkendali museum atau galeri, proyeksi digital besar dapat menciptakan lingkungan imersif yang menyenangkan yang dapat dirasakan sepenuhnya oleh pengguna. Jenis lingkungan ini sering kali melacak dan menggunakan gerakan dan tindakan pengguna yang besar yang melibatkan seluruh figur. Lingkungan ini dapat menggunakan teknologi layar hijau yang memungkinkan orang melihat diri mereka sendiri ditumpangkan pada latar belakang baru dan dengan demikian "dipindahkan" ke dunia baru. Objek grafis dan animasi dapat berinteraksi dengan gerakan pengguna, mungkin tertarik atau ditolak oleh figur pengguna saat mereka bergerak, menendang, atau melambaikan tangan.

Jenis lingkungan interaktif ini menghasilkan berbagai respons interaktif yang berbeda. Selain diproyeksikan ke dinding atau gedung, gambar interaktif berskala besar dapat diproyeksikan dari atas ke lantai, melacak posisi orang saat mereka bergerak di suatu ruang. Grafik interaktif di permukaan lantai bereaksi terhadap gambar saat mereka bergerak di dalam dan melintasi lantai (misalnya, menyebarkan daun, dll.).

Vitrine Baggar dan Yoram Mesuere
Instalasi jendela interaktif eksperimental yang dibuat oleh agensi desain Baggar yang bekerja sama dengan Yoram Mesuere.

CONTOH

Yoram Mesuere dan Baggar: Vitrine

Vitrine merupakan instalasi jendela interaktif eksperimental yang dikembangkan oleh studio desain interaktif Baggar bekerja sama dengan seniman Yoram

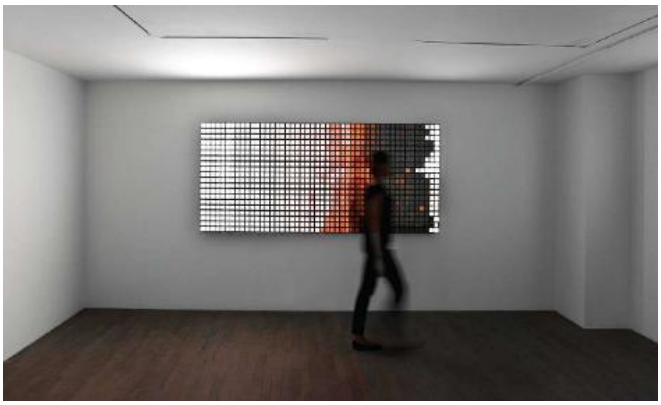
Mesuere. Karya ini menampilkan elemen-elemen grafis yang secara dinamis mengelilingi logo di pusat layar, bergerak dalam pola yang selaras. Interaksi tercipta melalui penggunaan kamera yang merekam gerakan orang-orang yang melintas di depan instalasi. Data gerakan tersebut kemudian diproses untuk mengaktifkan respons visual, di mana bentuk-bentuk grafis tampak bergerak mengikuti keberadaan dan arah pergerakan pengunjung saat mereka melewati jendela instalasi.

Pengaturan komputer digunakan untuk melacak posisi umum orang-orang di depan layar. Hal ini memungkinkan grafis bergerak ke lokasi umum pengguna, daripada harus melacak garis luar atau gerakan secara tepat. Interaksi tersebut menciptakan hubungan yang sederhana namun kuat antara penonton dan gambar serta memberikan umpan balik visual langsung, yang memungkinkan penonton melihat efek gerakan mereka pada animasi.

Pergerakan bentuk-bentuk tersebut memiliki nuansa organik; bentuk-bentuk tersebut secara bertahap berkumpul dan berkerumun bersama seperti burung atau daun yang tertiup angin. Kualitas gerakan yang naturalistis menambah rasa senang yang sederhana pada pengalaman interaktif yang menutupi proses digital dan komputasional.

5.3 TAMPILAN LAYAR KECIL DALAM CERMIN DIGITAL

Layar berukuran kecil sering kali dilengkapi dengan kamera yang berfungsi sebagai alat pengenal gambar, umumnya digunakan untuk menciptakan interaksi berbasis pengguna tunggal. Dalam konteks ini, layar bertindak layaknya cermin digital yang merespons kehadiran pemirsa secara langsung. Melalui umpan video waktu nyata, citra pemirsa dapat dimodifikasi atau ditingkatkan secara digital, menghasilkan perubahan visual yang segera terlihat. Grafik tambahan dan elemen visual lainnya dapat disisipkan, misalnya dengan menyorot bagian tubuh atau wajah pengguna.



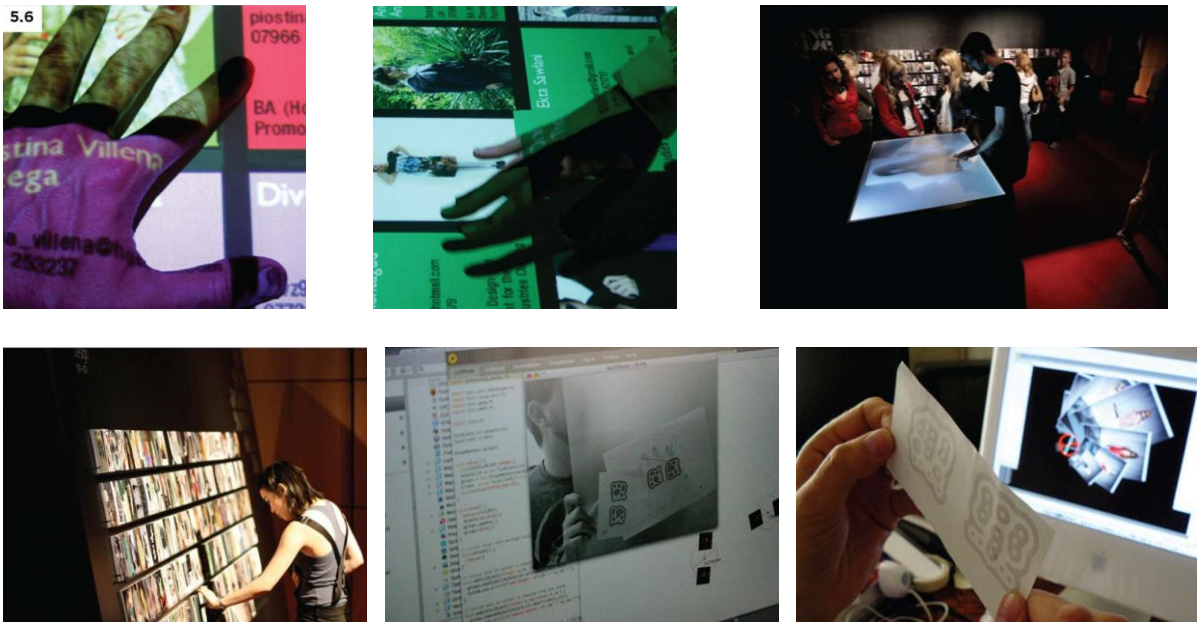
Kau Memudar Menjadi Cahaya Acak Internasional
Instalasi berskala besar yang dibuat menggunakan kisi-kisi LED dan sistem pelacakan gerakan berbasis kamera. Berfungsi sebagai semacam kisi-kisi cermin digital, pemirsa didorong untuk terlibat secara fisik guna menciptakan citra diri mereka sendiri.

Pengalaman interaktif ini menghadirkan sensasi "cermin ajaib" yang mampu mengubah penampilan pemirsa saat mereka melihat ke dalam layar. Efek visual yang menyerupai suasana dalam kisah *Alice in Wonderland* tercipta ketika seseorang melihat refleksi dirinya dalam bentuk yang telah dimodifikasi oleh teknologi. Pengguna dapat menyaksikan dirinya tampil dengan kostum digital, mengenakan topeng, atau bahkan aksesori kepala virtual. Lebih jauh lagi, visualisasi di layar dapat menyulap representasi tubuh menjadi lebih fantastis, seperti ditumbuhi sayap atau anggota tubuh tambahan. Bahkan transformasi sederhana sekalipun dapat memberikan pengalaman visual yang memikat dan menggugah rasa ingin tahu.

CONTOH

Merek yang bergerak: Peragaan busana LC

Ditugaskan oleh London College of Fashion, Moving Brands menciptakan permukaan meja reaktif sebagai cara untuk memamerkan secara interaktif ke-500 karya dalam Pameran Portofolio Pascasarjana.



Meja Portofolio Interaktif untuk London College of Fashion, Moving Brands
Layar dapat diubah menjadi permukaan meja responsif yang bereaksi terhadap objek yang diletakkan di atasnya, seperti yang ditunjukkan dalam solusi desain imajinatif yang dibuat oleh Moving Brands, untuk memamerkan karya lebih dari 500 portofolio virtual mahasiswa tingkat akhir.
Penanda "fiducial" khusus digunakan pada kartu sebagai pemicu untuk memproyeksikan gambar untuk setiap mahasiswa. Kombinasi fisik (kartu cetak) dengan digital menciptakan pengalaman yang menarik dan intuitif.

Setiap mahasiswa dalam pameran diwakili oleh kartu pos yang dirancang khusus, yang memiliki contoh visual dari karya cetak mereka di bagian depan, dan label pengenal unik di bagian belakang. Ketika diletakkan di permukaan meja, label pada setiap kartu pos memicu serangkaian gambar digital dari portofolio mahasiswa yang bersangkutan. Memindahkan kartu menciptakan interaksi; memutar dan merotasi kartu memungkinkan pemirsa untuk mengacak-acak rangkaian gambar.

Proyek ini menggunakan kombinasi teknologi Pemrosesan dan Reaktivasi untuk membuat label cetak sebagai pemicu interaktif. Kamera web dan lampu inframerah di bawah permukaan meja digunakan untuk melacak lokasi label di atas meja dan mengirimkan informasi ke komputer, yang memproyeksikan gambar digital dari bawah. Pengalaman keseluruhannya dengan indah menggabungkan kualitas fisik kartu pos dengan proyeksi digital di atas meja, membuat interaksi menjadi intuitif dan pengalamannya menarik: kartu pos cetak

biasa diubah menjadi sesuatu yang interaktif, mengejutkan, bahkan ajaib.

5.4 CARA MELIHAT

Ada banyak cara berbeda yang dapat dilakukan komputer untuk "melihat" guna mendeteksi keberadaan dan gerakan manusia, objek, lampu, dan sebagainya. Sebagai perangkat input, komputer biasanya digunakan sebagai pengamat "buta" di dunia, terbatas pada penginderaan gerakan hanya melalui input tetikus dan papan ketik.

Namun, sebagai perangkat penginderaan, komputer jauh lebih canggih dari itu, dan mampu merespons berbagai jenis input. Banyak seniman, desainer, dan teknolog kreatif telah menyadari bahwa membatasi cakupan input dan interaksi komputer hanya pada input papan ketik dan tetikus mengabaikan kemampuan komputer yang sangat besar sebagai perangkat input dan pemrosesan data yang canggih. Oleh karena itu, dalam beberapa tahun terakhir, seniman, desainer, dan programmer telah bekerja keras untuk mengeksplorasi kemampuan komputer sebagai perangkat multisensori dengan penuh semangat, menciptakan lingkungan digital yang dapat merasakan dunia di sekitar mereka.

Aspek utama aktivitas di bidang ini berpusat pada komputer sebagai perangkat penglihatan ("penglihatan komputer"). Memungkinkan komputer untuk mengakses dan memproses informasi visual telah terbukti menjadi minat khusus bagi seniman dan desainer kreatif. Kemajuan terkini dalam pemrosesan komputer telah menjadikan visi komputer sebagai bidang yang telah diterapkan pada berbagai proyek kreatif.

"Visi" Komputer

Meskipun kemajuan teknologi terus berkembang pesat, prinsip dasar dalam pengolahan visual oleh komputer tetap tidak banyak berubah. Dengan dukungan perangkat dan sumber daya yang memadai yang kini banyak tersedia secara daring seorang programmer dapat mengembangkan sistem yang memungkinkan komputer mengenali objek maupun manusia secara efisien, cepat, dan berbiaya rendah.

Salah satu metode paling umum untuk menghubungkan komputer dengan lingkungan sekitarnya adalah melalui kamera, seperti webcam, yang berfungsi sebagai "indera penglihatan" perangkat. Setelah terhubung, komputer menjalankan peran sebagai "otak", yaitu memproses dan menafsirkan data visual guna memperoleh pemahaman terhadap dunia di sekitarnya. Terdapat beragam pendekatan dan teknik yang dapat diterapkan untuk mengenali jenis-jenis objek serta mendeteksi gerakan dalam citra yang ditangkap.

Melihat Warna

Salah satu cara utama bagi komputer untuk memahami lingkungannya adalah melalui analisis informasi warna yang ditangkap oleh lensa kamera. Kamera merekam gambar bergerak dan mengirimkan rangkaian bingkai ke komputer dengan kecepatan sekitar 28 frame per detik. Setiap bingkai tersebut terdiri dari ratusan ribu elemen kecil yang disebut piksel, yang masing-masing merepresentasikan nilai warna tertentu. Misalnya, video dengan resolusi 320 x 240 piksel menghasilkan setiap bingkai dengan total 76.800 piksel warna, yang kemudian dianalisis oleh komputer sebagai daftar data warna.

Melalui pemrograman, komputer dapat memilah dan menyaring ribuan nilai warna ini

untuk mencari corak atau rona tertentu. Informasi tersebut digunakan untuk mendeteksi pola visual, mengenali bentuk, dan menafsirkan elemen-elemen dalam lingkungan sekitarnya. Oleh karena itu, kemampuan komputer dalam memproses, mengelompokkan, dan memanipulasi data warna dari citra piksel merupakan komponen mendasar dalam sistem penglihatan komputer *(computer vision)*.

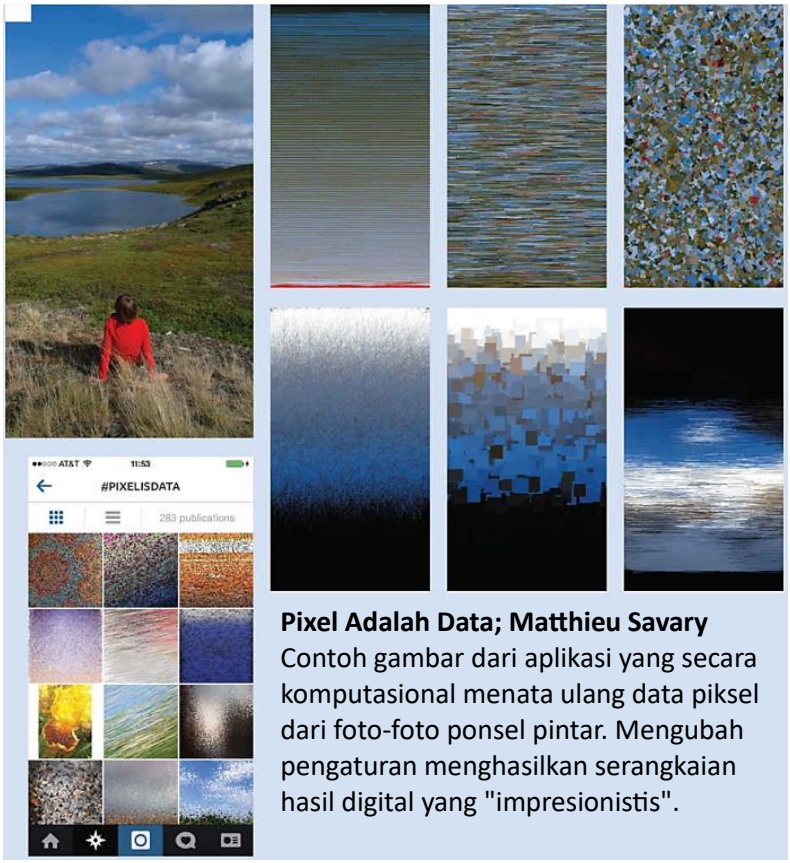
CONTOH

Matthieu Savary: Pixel Data

Pixel Data merupakan aplikasi gratis yang dikembangkan oleh Matthieu Savary sebagai eksplorasi ulang terhadap potensi visual fotografi ponsel pintar. Aplikasi ini memperlakukan foto sebagai himpunan data warna, kemudian memanfaatkan informasi tersebut untuk menyusun ulang piksel gambar berdasarkan parameter tertentu, seperti rona *(hue)* dan nilai warna.

Dalam prosesnya, gambar diurai menjadi elemen-elemen warna individu, lalu direkonstruksi ulang menggunakan parameter berbeda, misalnya rona atau komponen RGB. Pengguna dapat menyesuaikan parameter penyusunan ulang ini melalui serangkaian penggeser *(sliders)* yang tersedia pada antarmuka aplikasi.

Dengan mengurutkan data warna menurut nilai logis, aplikasi ini menantang asumsi bahwa piksel dalam foto digital harus dilihat dalam urutan yang "benar", dan memungkinkan pengguna untuk menafsirkan ulang fotografi mereka untuk menciptakan gambar abstrak dan impresionis. Gambar seperti konfeti yang dihasilkan menyajikan cara berpikir baru tentang cara informasi warna gambar ponsel pintar diproses dan ditampilkan.



Pixel Adalah Data; Matthieu Savary
Contoh gambar dari aplikasi yang secara komputasional menata ulang data piksel dari foto-foto ponsel pintar. Mengubah pengaturan menghasilkan serangkaian hasil digital yang "impresionistis".

Gambar Statis

Setiap citra bitmap digital, seperti format GIF atau JPEG, terdiri atas ratusan hingga ribuan nilai warna individual yang dikenal sebagai piksel. Bahkan citra digital yang tampak paling kompleks pada dasarnya merupakan susunan mosaik dari kotak-kotak warna yang tersusun secara sistematis. Ketika seorang desainer menggunakan perangkat lunak grafis untuk menerapkan filter misalnya efek buram (*blur*) atau peningkatan kontras pada dasarnya perangkat lunak tersebut sedang menjalankan proses komputasi untuk memodifikasi

informasi piksel dari gambar tersebut.

Berbeda dari penggunaan antarmuka grafis, pemrograman melalui kode memungkinkan desainer maupun pengembang mengakses langsung daftar nilai warna setiap piksel dalam gambar. Akses langsung ini memberi keleluasaan lebih besar dalam memanipulasi data warna secara presisi dan sesuai kebutuhan. Kemampuan ini memungkinkan pemrogram untuk mengekstrak serta menganalisis informasi warna dari citra bitmap secara waktu nyata (*real-time*), dan sangat bermanfaat dalam konteks interaksi berbasis video atau rangkaian gambar bergerak.

Gambar Bergerak

Daftar data warna piksel dari sumber video langsung dapat diambil, dicari, dan dianalisis. Mencari dan mengekstrak nilai warna individual dari umpan kamera langsung memberi desainer cara yang berharga untuk mengamati lingkungan sekitar. Kode dapat digunakan untuk mencari dan menemukan bagian umpan video yang berisi warna tertentu (misalnya, hijau) atau menemukan area dengan kontras, kecerahan, atau kegelapan yang ekstrem.

Memiliki kemampuan untuk menemukan warna yang telah ditentukan sebelumnya atau area dengan kecerahan atau kegelapan tertentu memungkinkan desainer memulai untuk membuat lingkungan yang mengidentifikasi gambar bergerak sebagai warna, memisahkan elemen latar depan dan latar belakang, atau melacak warna atau cahaya. Data warna dari piksel dalam gambar video dapat digunakan untuk membuat “cermin digital” yang di dalamnya gambar diubah menjadi mosaik bentuk grafis yang bergerak.

Contoh “Gambar Mozaik”

Teknik analisis warna dapat diterapkan dalam pengembangan berbagai jenis lingkungan interaktif. Salah satu contohnya adalah teknik *green screen* (layar hijau), yang memungkinkan pemisahan antara latar depan dan latar belakang dalam suatu citra. Teknologi ini dapat digunakan untuk mengganti latar belakang asli dengan gambar yang sepenuhnya baru, atau untuk memodifikasi efek visual pada objek atau individu yang berada di depan kamera.

Dengan pendekatan serupa, identifikasi terhadap piksel-piksel yang sangat terang juga dapat dimanfaatkan sebagai dasar untuk menciptakan bentuk interaksi fisik. Misalnya, pergerakan sumber cahaya seperti lampu LED atau senter yang diarahkan ke layar dapat menghasilkan respons visual, membentuk interaksi antara gerakan pengguna dan tampilan digital. Mekanisme ini merupakan salah satu prinsip yang mendasari teknologi interaktif seperti yang digunakan dalam konsol permainan Nintendo Wii.

- Efek "layar hijau" dapat dibuat dengan menemukan area piksel yang luas dengan rona tunggal tertentu yang membentuk dinding latar belakang; area ini dapat diabaikan atau diganti, sehingga gambar tetap berada di latar depan.

- Area dengan nilai kecerahan rendah dapat memperlihatkan keberadaan dan pergerakan orang; bayangan atau siluet gambar dapat ditemukan saat mereka lewat di depan atau di belakang layar.
- Menemukan dan mengikuti piksel paling terang di depan kamera adalah cara mudah untuk melacak pergerakan lampu.
- Alternatif untuk layar hijau adalah "teknik perbedaan", yang merupakan variasi dari konsep layar hijau. Nilai warna lingkungan kosong terus-menerus dibandingkan dengan nilai warna pemandangan saat ini. Perbedaan antara keduanya menunjukkan keberadaan orang yang memasuki ruang tersebut.

CONTOH

Sennep: Dinding Pesta

Dinding Pesta adalah instalasi proyeksi interaktif yang dibuat khusus untuk perayaan pasca-pernikahan. Pergerakan tamu di lantai dansa pesta dilacak, dan jumlah energi digunakan untuk memicu proyeksi dinding. Sedikit atau tidak ada gerakan menghasilkan layar kosong; namun, segera setelah tingkat energi fisik tertentu terdeteksi oleh sistem, gambar besar para penari diproyeksikan ke dinding dalam bentuk titik-titik piksel. Sistem, yang merespons gerakan tarian yang energik, karenanya mendorong partisipasi yang lebih besar dari para tamu.

Dinding Pesta Sennep
Gambar proyeksi interaktif yang dibuat sebagai bagian dari pesta pernikahan. Gerakan lantai dansa ditangkap dan digunakan untuk memproyeksikan gambar ke dinding.

5.5 MELIHAT ORANG

Mencari informasi warna adalah cara paling sederhana dan tercepat untuk menemukan gerakan dan keberadaan figur dalam suatu lingkungan. Teknik yang digunakan berguna untuk menemukan sesuatu yang baru atau berbeda dalam suatu ruang. Namun, informasi warna tidak dapat memberikan informasi terperinci tentang figur tertentu dalam ruang tersebut misalnya, jumlah orang atau lokasi wajah, gerakan, atau anggota tubuh mereka.

Informasi terperinci semacam ini memerlukan metode pemrosesan gambar dan visi komputer yang lebih canggih. Metode ini melibatkan banyak proses matematika yang lebih rumit. Untungnya bagi desainer kreatif, ada beberapa pustaka dan sumber daya yang tersedia yang dapat membantu dan membuat prosesnya tidak terlalu sulit. Berikut adalah ringkasan dari konsep-konsep utama.

Gumpalan

Kemampuan untuk mengenali bentuk dalam sebuah citra merupakan pendekatan canggih dan efektif dalam mendeteksi keberadaan serta pergerakan individu dalam suatu adegan. Teknik ini menawarkan informasi yang lebih rinci dibandingkan sekadar analisis berdasarkan nilai warna. Proses ini dikenal dalam bidang visi komputer sebagai deteksi gumpalan (*blob detection*), yang secara khusus digunakan untuk mengidentifikasi kontur atau siluet manusia dalam gambar.

Deteksi gumpalan merupakan metode matematis yang kompleks untuk mengidentifikasi area-area dalam citra yang memiliki warna dan tingkat kecerahan yang kontras dibandingkan dengan sekitarnya. Dengan menemukan wilayah yang memiliki kesamaan warna dan corak, sistem dapat memperoleh representasi yang lebih akurat tentang bentuk dan pergerakan manusia dalam suatu lingkungan visual. Karena gumpalan mengelompokkan area dengan karakteristik warna yang serupa misalnya warna kulit dan pakaian metode ini mampu meningkatkan akurasi dalam mendeteksi batas tubuh manusia.

Lebih dari sekadar mengenali bentuk, deteksi gumpalan juga memungkinkan sistem untuk membedakan dan melacak individu secara spesifik saat mereka bergerak melintasi pemandangan. Kemampuan untuk mendeteksi dan mengikuti figur secara individual ini sangat berperan dalam menciptakan bentuk interaksi pengguna yang responsif dan dinamis.

Menemukan Tepi

Deteksi gumpalan orang dalam suatu adegan dapat digunakan untuk menemukan tepi figur dalam suatu lingkungan, yang dapat digunakan sebagai dasar interaksi imajinatif yang bekerja dengan dan di sekitar bentuk tubuh peserta. Garis atau grafik animasi dapat tertarik ke garis luar tubuh seseorang; "hujan" digital dapat jatuh ke tangan atau lengan; grafik dapat tumbuh dan muncul dari sekitar garis luar tubuh. Menemukan bentuk dan garis luar orang dengan cara ini membantu menempatkan mereka secara efektif di tengah layar atau permainan interaktif, memberikan ilusi interaksi yang lebih lengkap dan menambah tingkat keterlibatan, kesenangan, dan "keajaiban".

CONTOH

Chris Milk: The Treachery of Sanctuary

The Treachery of Sanctuary merupakan instalasi interaktif berskala besar yang terdiri dari tiga panel proyeksi, di mana masing-masing panel menampilkan siluet tubuh partisipan yang dimodifikasi secara digital menggunakan citra burung untuk menyampaikan narasi tentang kelahiran, kematian, dan transfigurasi.

Pengkhianatan Sanctuary Chris Milk
Gambar dari triptych interaktif di mana para peserta menggunakan tubuh mereka untuk menciptakan hibrida makhluk mirip manusia dan burung. Fisik langsung dari interaksi tersebut menciptakan transformasi luar biasa dari bentuk manusia.

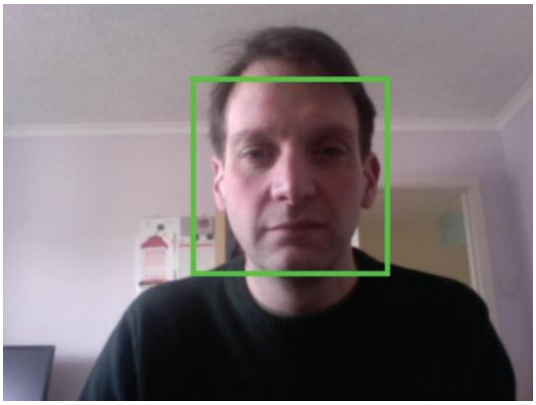
Saat pengguna bergerak di depan panel, siluet tubuh mereka diolah menjadi berbagai representasi visual: sekumpulan burung terbang yang melambangkan kelahiran, sosok yang diserang burung sebagai metafora kematian, serta figur yang perlahan menumbuhkan sayap sebagai simbol transfigurasi. Proses transformasi ini membangun hubungan emosional yang kuat antara partisipan dan tema-tema yang diangkat dalam karya.

Interaksi yang intuitif, berpadu dengan palet warna monokrom, menambah kesan visual yang elegan sekaligus mendalam. Gerakan dan bayangan tubuh pengguna direkam melalui kamera Kinect, lalu dianalisis menggunakan kode berbasis *openFrameworks*. Informasi ini kemudian diproses lebih lanjut dalam perangkat lunak grafis berbasis Unity untuk menghasilkan animasi burung interaktif secara real-time.

Kemampuan sistem *blob detection* dalam melacak wajah dan individu dalam suatu adegan membuka peluang lebih luas bagi desainer kreatif untuk merancang interaksi berbasis kelompok, yang melibatkan partisipasi banyak orang secara simultan. Animasi, grafik, permainan, dan bentuk interaksi lainnya dapat diarahkan secara spesifik kepada satu individu dalam kerumunan, sementara kehadiran pengguna baru dapat dikenali secara otomatis, yang kemudian memicu munculnya rangkaian animasi dan visual grafis baru yang disesuaikan secara personal. Kostum atau atribut virtual dapat mengikuti pergerakan masing-masing pengguna, dan gaya visual yang unik dapat ditampilkan untuk setiap partisipan baru yang terlibat dalam adegan.

Deteksi Wajah

Selain mendeteksi kontur dan melacak individu dalam sebuah adegan, algoritma *blob detection* juga mampu mengidentifikasi dan mengenali wajah. Kemampuan ini sangat bermanfaat dalam aplikasi grafis dan animasi yang berfokus pada bagian kepala atau wajah suatu figur. Atribut virtual seperti topi, telinga, atau topeng dapat ditampilkan secara dinamis pada area wajah setiap pengguna, dan elemen-elemen ini dapat dianimasikan serta diubah secara individual, mengikuti karakteristik masing-masing partisipan.



Kemampuan untuk melacak wajah merupakan titik awal yang berguna untuk menciptakan interaksi berdasarkan pengguna individu.

Contoh: Hellicar dan Lewis (*Topeng Interaktif)*

Hellicar dan Lewis adalah duo kreatif yang menggabungkan teknologi, seni, dan desain untuk menciptakan pengalaman interaktif lintas konteks, mulai dari dunia mode, pertunjukan, hingga pendidikan dan seni visual. Karya-karya mereka menekankan eksplorasi hubungan antara ruang, teknologi, dan audiens, serta mengintegrasikan pendekatan dari berbagai disiplin seni dan ilmu pengetahuan.

Hellicar dan Lewis: Topeng Diaghilev

Proyek topeng interaktif ini merupakan hasil kerja sama antara Museum Victoria and Albert di London dan departemen pendidikannya, yang dikembangkan untuk mendampingi pameran mereka tentang Diaghilev. Mengusung tema topeng dalam teater, instalasi ini memanfaatkan teknologi pengenalan wajah untuk menampilkan topeng digital secara real-time kepada partisipan yang berada di depan kamera komputer. Kode pemrograman yang dirancang khusus memungkinkan penciptaan berbagai desain topeng serta pelacakan gerakan pengguna selama interaksi berlangsung.

Topeng Diaghilev: Hellicar dan Lewis
Perangkat lunak pelacak wajah digunakan untuk menemukan pengguna dalam adegan dan memposisikan topeng hewan grafis ke wajah peserta.

Hellicar dan Lewis juga mengembangkan berbagai instalasi interaktif lainnya. Salah satunya adalah Uniqlo Stripes, sebuah instalasi berbasis dinding bertema garis-garis yang dirancang untuk mendukung promosi acara label mode Uniqlo. Proyek lainnya, Norwegian Wood, merupakan proyeksi visual yang merespons suara, dibuat untuk pemutaran perdana film adaptasi novel Norwegian Wood karya Haruki Murakami di Galeri Haunch of Venison, London. Instalasi ini menggabungkan elemen-elemen visual dari film dengan citra khas musim-musim

dalam kebudayaan Jepang.

Proyek topeng interaktif ini dikembangkan atas inisiatif Museum Victoria and Albert di London, bekerja sama dengan departemen pendidikannya, sebagai bagian dari penyelenggaraan pameran bertema Diaghilev. Mengangkat konsep topeng dalam tradisi teater, instalasi ini menggunakan teknologi pengenalan wajah untuk menerapkan topeng digital secara real-time kepada pengunjung yang berada di depan kamera komputer. Sistem ini dijalankan melalui pemrograman khusus yang memungkinkan penciptaan berbagai desain topeng sekaligus memantau gerakan pengguna selama berlangsungnya interaksi.

Selain proyek tersebut, Hellicar dan Lewis juga menghasilkan sejumlah instalasi interaktif lainnya. Salah satunya adalah Uniqlo Stripes, sebuah karya berbasis dinding bertema garis-garis yang dikembangkan sebagai bagian dari kampanye promosi merek fesyen Uniqlo. Karya lain berjudul Norwegian Wood merupakan proyeksi visual yang merespons elemen audio, dirancang untuk mengiringi pemutaran perdana film adaptasi novel Norwegian Wood karya Haruki Murakami. Instalasi ini ditampilkan di Galeri Haunch of Venison, London, dan memadukan elemen visual dari film dengan simbol-simbol musim dalam tradisi budaya Jepang.

5.6 *ZOOM IN* DAN *ZOOM OUT*

Kamera digital standar, seperti webcam, dapat dimanfaatkan secara efektif dalam proyek komputer vision untuk mendeteksi warna maupun bentuk. Namun, perangkat ini memiliki keterbatasan dalam hal kemampuan mendeteksi kedalaman dan gerakan pengguna secara spesifik. Sebaliknya, Kinect merupakan perangkat kamera yang lebih canggih karena mampu mengenali keduanya kedalaman dan gerakan.

Dalam beberapa tahun terakhir, industri gim telah menjadi pelopor dalam pengembangan teknologi komputer vision, dengan tujuan menciptakan pengalaman bermain yang semakin imersif. Sebagai contoh, pengenalan sensor gerak dan kontrol gestur inovatif oleh Nintendo melalui konsol Wii telah mendefinisikan ulang cara pandang terhadap pengontrol gim konvensional.

Setiap kemajuan dalam pengembangan perangkat teknologi komersial kerap dimanfaatkan dengan cepat oleh para kreator dan pengembang yang ingin mengeksplorasi potensi aplikatif yang lebih luas. Salah satu contoh signifikan adalah Kinect, perangkat yang awalnya dikembangkan untuk konsol Xbox. Kinect mampu mengenali berbagai gerakan tubuh manusia melalui kombinasi kamera RGB standar, kamera inframerah, dan pemancar cahaya inframerah, yang bekerja bersama untuk mendeteksi kedalaman serta gestur secara real-time.

Meskipun dirancang sebagai aksesori game, Kinect juga dapat digunakan secara independen dengan komputer, menjadikannya alat sensor yang sangat bermanfaat untuk mengembangkan berbagai aplikasi interaktif. Sejumlah pustaka kode telah tersedia untuk memberi akses langsung ke data sensor Kinect, memungkinkan pemrogram untuk mengolah tampilan sebagai citra RGB konvensional, mengukur jarak, serta mendeteksi pergerakan tubuh secara keseluruhan.

Kamera Panorama

Selain tampilan kamera biasa dari suatu pemandangan, perangkat kamera Kinect juga mampu membuat "gambar kedalaman" dari suatu lingkungan: gambar skala abu-abu di mana nilai warna setiap piksel secara akurat berhubungan dengan jaraknya dari kamera. Objek yang paling dekat dengan kamera tampak putih, dan yang terjauh tampak hitam. Hasilnya adalah "gambar kedalaman." Diferensiasi antara kedekatan objek dari kamera berarti bahwa objek di latar depan dapat lebih mudah dilacak sementara elemen latar belakang dapat diabaikan atau dihapus.

Kamera Kinect memiliki kamera RGB internal, cahaya inframerah, dan sensor kedalaman, yang dapat digunakan untuk memvisualisasikan gerakan tubuh, gerakan, dan kedekatan dengan kamera.

Contoh gambar biasa (RGB) (kanan) dan gambar "kedalaman" (kiri) yang diambil oleh kamera Kinect yang terhubung ke perangkat komputer. Pustaka Pemrosesan Dan Shiffman, "Open Kinect for Processing," digunakan untuk mengekstrak gambar kedalaman.

CONTOH

Robert Hodgin: Dismorfia Tubuh

Karya Body Dysmorphia menggunakan sensor Kinect untuk menciptakan visualisasi interaktif secara real-time yang merepresentasikan gangguan dismorfia tubuh sebuah kondisi psikologis di mana individu mengalami kekhawatiran berlebihan terhadap apa yang mereka anggap sebagai cacat fisik. Data kedalaman yang dikumpulkan oleh kamera Kinect dimanfaatkan untuk memperoleh representasi visual penonton, yang kemudian dimanipulasi secara komputasional dan digambar ulang menjadi sebuah refleksi yang terdistorsi. Informasi mengenai permukaan tubuh yang diperoleh dari kamera tersebut dimodifikasi dengan mendorong bagian-bagian tertentu ke dalam atau ke luar, menghasilkan efek visual yang memperlihatkan tubuh penonton sebagai lebih gemuk atau sangat kurus secara ekstrem, semuanya terjadi secara waktu nyata.

Dismorfia Tubuh Robert Hodgin
Sensor Kinect menciptakan "peta" 3D dari suatu figur yang dapat disesuaikan secara komputasional untuk menciptakan distorsi waktu nyata pada citra penonton.

Inframerah dan Distorsi Visual

Teknologi cahaya dan kamera inframerah sangat berguna dalam mendeteksi bentuk dan figur dalam suatu adegan. Sumber cahaya inframerah, yang lazim digunakan dalam sistem kamera pengawas, memancarkan cahaya intens yang tidak tampak oleh mata manusia namun dapat ditangkap oleh kamera inframerah. Lampu jenis ini tersedia secara komersial melalui pemasok perangkat keamanan. Sementara itu, kamera webcam konvensional umumnya tidak mampu menangkap cahaya inframerah karena dilengkapi dengan filter penyaring khusus. Namun, filter tersebut dapat dilepas melalui proses modifikasi sederhana sering disebut sebagai “peretasan” yang memungkinkan kamera tersebut digunakan sebagai sensor inframerah dalam proyek eksperimental atau artistik.

Salah satu masalah dengan penggunaan kamera biasa untuk mendeteksi figur adalah sumber cahaya lain, terutama proyektor atau layar tampilan, menambahkan cahaya ekstra yang mengganggu banyak "gangguan visual" yang membuat tugas mendeteksi figur menjadi jauh lebih sulit bagi kamera. Menyorotkan lampu sorot inframerah ke suatu pemandangan dan menggunakan kamera yang hanya melihat area yang diterangi inframerah adalah cara untuk menghilangkan semua sumber cahaya tampak yang menyebabkan gangguan, sehingga kamera dapat mendeteksi orang-orang di ruangan tersebut.

Deteksi Gerakan

Selain mampu merasakan jarak dan kedalaman, kamera Kinect dapat mendeteksi figur individu, menemukan dan melacak posisi tangan, dan bahkan mengenali gerakan tangan dan gerakan seluruh tubuh. Kemampuan untuk melacak gerakan tangan dan tubuh secara akurat menciptakan fondasi untuk pengalaman interaktif yang sepenuhnya imersif di mana tubuh pengguna menjadi pengontrol permainan, yang secara intuitif terlibat dalam lingkungan. Ini adalah fitur hebat yang telah dieksplorasi dan dimanfaatkan oleh beberapa permainan dan lingkungan interaktif.

Pelacakan tangan memungkinkan pengguna untuk terlibat dengan layar berbasis gerakan atau permukaan meja interaktif secara main-main. Deteksi figur seluruh tubuh dapat digunakan untuk menciptakan pengalaman layar lebar yang imersif, yang melibatkan satu atau beberapa pengguna secara main-main.

Proyek Lokal: Museum Seni Cleveland

Proyek Lokal adalah sebuah studio desain media yang berfokus pada penciptaan pengalaman interaktif di ruang publik dan institusi budaya seperti museum. Dengan menekankan pemanfaatan teknologi sebagai sarana berbagi narasi, perusahaan ini mendorong terciptanya cara-cara baru dalam berinteraksi baik dengan seni, lingkungan perkotaan, maupun antarindividu. Dalam sebuah proyek berskala besar yang baru-baru ini mereka kerjakan, Proyek Lokal mengembangkan serangkaian instalasi interaktif untuk Museum Seni Cleveland, khususnya di Galeri Satu (*Gallery One*), dengan tujuan menginspirasi pengunjung untuk menjelajahi dan menafsirkan ulang koleksi museum melalui pendekatan yang segar dan inovatif.

Salah satu hasil utama dari proyek ini adalah pengembangan berbagai "lensa interaktif" yang dirancang secara khusus dan ditempatkan di seluruh ruang pamer. Perangkat ini

memungkinkan pengunjung memperkaya pengalaman mereka terhadap koleksi museum melalui keterlibatan aktif. Dua instalasi utama, strike a pose (juga dikenal sebagai "Lensa Patung") dan Making Faces, menggunakan teknologi pengenalan wajah serta pelacakan gerakan untuk menciptakan interaksi yang personal antara pengunjung dan karya seni. Kedua instalasi ini mendorong pengunjung untuk melihat refleksi diri mereka dalam representasi artistik tubuh manusia, sekaligus mengajak mereka merenungkan bagaimana bentuk tubuh menjadi inspirasi utama dalam sejarah seni rupa.

Dinding Koleksi Museum Seni Cleveland
Dinding interaktif yang menampilkan lebih dari 4.100 karya seni dari koleksi permanen yang disimpan di Museum Seni Cleveland. Teknologi layar sentuh memungkinkan pengunjung menjelajahi koleksi dan terhubung dengan objek, menjadikan kunjungan mereka sebagai pengalaman yang lebih personal.

Strike a Pose merupakan sebuah permainan interaktif yang mengajak pengunjung museum untuk meniru pose khas dari patung-patung dalam koleksi museum. Interaksi ini memanfaatkan sensor Kinect untuk melacak posisi tubuh pengguna, lalu membandingkan titik-titik kunci tubuh mereka dengan pose patung yang ditampilkan. Sistem kemudian menganalisis dan memberi skor berdasarkan tingkat kesesuaian gerakan pengguna dengan objek seni. Dengan mengundang pengunjung untuk meniru bentuk dan postur patung, instalasi ini mendorong keterlibatan fisik yang lebih mendalam dan memungkinkan pemirsa mengamati karya secara lebih cermat.

Sementara itu, instalasi Make a Face menggunakan teknologi pengenalan wajah untuk mencocokkan ekspresi pengunjung dengan ekspresi yang muncul dalam karya seni dari koleksi museum. Ekspresi wajah pengguna direkam melalui webcam, kemudian fitur-fitur utama seperti jarak antar mata atau bentuk mulut dianalisis dan dibandingkan dengan basis data yang terdiri atas 180 citra karya seni. Sistem ini akan memilih pasangan ekspresi yang paling sesuai, lalu menyusunnya dalam bentuk foto-strip yang dapat dibagikan melalui email, ditampilkan di layar-layar lain dalam museum, atau disimpan sebagai kenang-kenangan digital.

Membuat Wajah Museum Seni Cleveland oleh Proyek Lokal
Perangkat lunak pengenalan wajah digunakan untuk mencocokkan ekspresi wajah pengguna dengan salah satu karya seni di museum, sehingga menciptakan cara unik untuk menjelajahi koleksi.

Garis dan Bentuk Museum Seni Cleveland oleh Proyek Lokal
Dalam interaktif ini, pengunjung menggambar garis di layar, yang kemudian dicocokkan dengan salah satu objek dalam koleksi museum yang memiliki garis serupa, sehingga pengunjung dapat membuat hubungan visual baru antara objek.

CONTOH

Theo Watson dan Emily Gobeille

Theo Watson dan Emily Gobeille telah mengembangkan berbagai lingkungan interaktif yang mendorong pengguna untuk secara jenaka menjelajahi dan berinteraksi dengan dunia animasi. Mereka memadukan teknologi layar hijau secara langsung dengan sistem pelacakan gerakan dan gestur, guna menciptakan dunia imajinatif di mana partisipan dapat memicu badai, menyulut petir, atau menurunkan hujan dedaunan hanya melalui gerakan tubuh. Skala instalasi yang luas, dipadukan dengan kehalusan visual dan integrasi teknologi kreatif yang cermat, menjadikan karya-karya mereka sebagai pengalaman visual yang imersif dan memikat.

Knee Deep

Knee Deep adalah instalasi imersif yang dirancang untuk mengajak anak-anak masuk

ke dalam dunia fantasi melalui perpaduan teknologi komputer dan efek layar hijau secara langsung. Umpan video dari kamera yang merekam peserta di depan layar hijau diproses menggunakan platform openFrameworks, lalu dikombinasikan dengan latar belakang digital untuk menciptakan citra anak-anak yang muncul seolah-olah berada dalam lanskap khayalan. Sistem *computer vision* melacak posisi dan pergerakan mereka secara real-time, menghasilkan animasi responsif yang berinteraksi dengan setiap langkah dan lompatan. Hasilnya adalah pengalaman eksploratif yang menyeluruh, di mana anak-anak digambarkan sebagai sosok raksasa yang dapat mengendalikan makhluk dan lingkungan yang secara dinamis menanggapi aksi mereka.

<table>
<tr>
<td>  </td>
<td>  </td>
<td rowspan="4">Knee Deep Theo Watson dan Emily Gobeille
Contoh diambil dari instalasi interaktif yang imersif. Pengguna diarahkan ke salah satu dari beberapa dunia imajiner dengan layar hijau dan menjadi peserta aktif, berinteraksi dengan karakter dan makhluk dengan menggerakkan dan menghentakkan kaki mereka.</td>
</tr>
<tr>
<td>  </td>
<td>  </td>
</tr>
<tr>
<td>  </td>
<td>  </td>
</tr>
<tr>
<td>  </td>
<td>  </td>
</tr>
</table>

5.7 MELIHAT DUNIA

Warna Sebagai Tipe Data

Membuat program komputer yang dapat "melihat" dimulai dengan memperoleh dan memproses informasi warna dari gambar digital; ini dapat berasal dari satu bitmap atau bahkan dari umpan video langsung.

Warna, seperti banyak hal lain dalam lingkungan komputasi, didefinisikan dan dipahami sebagai tipe data. Dalam Pemrosesan, tipe data "warna" digunakan untuk menyimpan dan menyimpan nilai warna yang dapat dirujuk kembali oleh kode nanti.
Contoh berikut membuat dua tipe data warna merah dan biru.

```
color c1 = color (255, 0, 0);
color c2 = color (0, 0, 255);
fill(c1);
stroke(c2);
```

Gambar bitmap pada dasarnya adalah daftar besar nilai warna individual (RGB), yang digambar di panggung sebagai piksel individual. Menemukan dan menggunakan nilai-nilai ini adalah titik awal untuk dapat membuat program yang dapat "melihat" dan merespons dunia luar.

Informasi warna dapat diambil dari gambar statis atau bergerak. Sebaiknya mulai dengan melihat gambar statis tunggal. Dalam Pemrosesan, gambar dimuat ke dalam program dengan menggunakan kelas PImage() dan fungsi loadImage() sebagai berikut:

```
PImage img;
img = loadImage("face.jpg");
```

Fungsi image digunakan untuk menempatkan gambar yang dimuat ke layar:

```
// draw the image on the screen
Image (img, 100, 100);
```

Ekstraksi data warna dari SATU piksel dalam sebuah gambar dapat dilakukan dengan menggunakan fungsi get, yang mendapatkan informasi warna dari koordinat x, y tertentu:

```
get (x, y);
// gets the color of a pixel x100, y10 from the image "img"
img.get (100, 10);
```

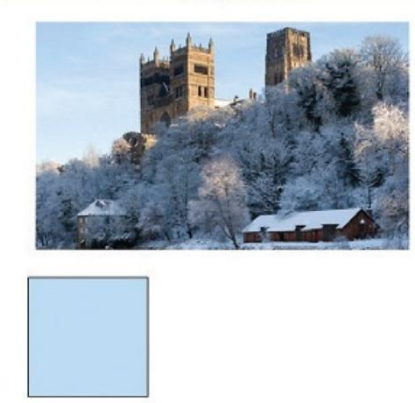
Sebagai alternatif, data warna dapat diekstraksi dengan mereferensikan piksel tertentu dari daftar array. (Ini merupakan cara komputasi yang lebih cepat.)

```
pixels[pixel_in_list];
// get the pixel at position 144 in the array
```

```
img.pixels[144];
```

Fungsi ini dapat diterapkan untuk mendapatkan warna piksel tunggal dari gambar yang dimuat:

```
size (800, 800);
PImage img = loadImage("face.jpg"); // load image
Image (img, 0, 0); // put image on stage

// save pixel color from image
color c1 = img.get (100, 100);
color c2 = img.pixels[400];

fill(c1); // use pixel color as fill color
rect (600, 0, 100, 100);
fill(c2);
rect (600, 150, 100, 100);
```



Contoh mendapatkan dan menampilkan warna dari piksel dalam gambar. Satu warna piksel dari foto dapat ditemukan dan digunakan sebagai warna pengisi untuk grafik lain di layar.

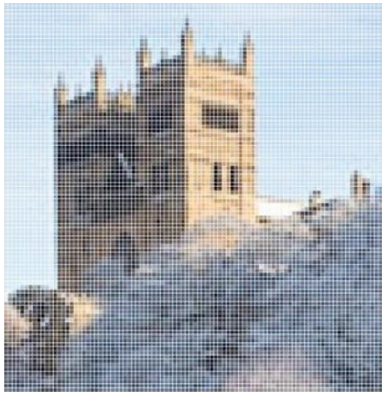
Dapatkan Semua Nilai Warna dari Gambar Statis

Selain mendapatkan satu nilai dari gambar, kode dapat mengambil setiap piksel warna dari gambar dan menggambarnya ulang di layar.

Perulangan “for” bersarang, satu perulangan “for” di dalam yang lain, adalah cara umum untuk menelusuri setiap baris dan kolom gambar secara berurutan. Perulangan luar berjalan melalui “lokasi y” setiap piksel; perulangan dalam mendapatkan “lokasi x” setiap piksel. Dengan menggunakan struktur ini dan fungsi get, warna setiap piksel dapat ditemukan dan digambar ulang secara berurutan.

```
PImage img = loadImage("me.jpg"); // load image
noStroke ();
for (int y = 0; y < img.height; y++) {
  for (int x = 0; x < img.width; x++) {
    // find the color of the current pixel
    color c = img.get (x, y);
```

```
      fill(c); // use found color as the fill color
      ellipse (x, y, 1, 1); // redraw as a circle
    }
  }
```



Efek gambar "mosaik" berpiksel dicapai dengan menggambar ulang semua piksel. Informasi warna dari setiap piksel ditemukan dan digunakan untuk menggambar setiap kotak.

Ini adalah jenis “manipulasi gambar” sederhana di mana warna dari piksel dalam gambar dapat digambar ulang sebagai bentuk, sehingga menciptakan efek mosaik.

Ekstrak Informasi Warna Lainnya

Fungsi tambahan dapat digunakan untuk mengekstrak informasi warna yang lebih terperinci dari piksel misalnya, menemukan nilai kecerahannya atau jumlah merah, hijau, atau biru.

Fungsi kecerahan menemukan tingkat kecerahan dalam suatu warna, disimpan sebagai nilai antara 0 dan 255, di mana 0 adalah yang paling gelap dan 255 adalah yang paling terang.

```
color c1 = (64, 0, 0);
// finds the brightness level from a color
float br = brightness(c1);
```

Fungsi red(), green(), dan blue() mengekstrak nilai yang memberikan jumlah masing-masing dalam suatu warna:

```
// extracts the amount of red from a color
float r = red(c1);
float g = green(c1); // extracts the amount of green
float b = blue(c1); // extracts the amount of blue
```

Informasi warna jenis ini dapat digunakan untuk memengaruhi cara gambar digambar ulang. Misalnya, mendapatkan nilai kecerahan setiap piksel dalam kode contoh di atas dapat menentukan apakah bentuk "piksel" individual digambar di panggung atau tidak, sehingga hanya piksel paling gelap yang digambar.

```
// Use the following code within the example for loop
color c = img.get (x, y);
```

```
fill(c);
// get the brightness of the color
float br = brightness(c);
// only draw shape if brightness is less than 125
if (br < 125) {
  ellipse (x, y, 1, 1);
}
```

Demikian pula, nilai kecerahan warna dapat digunakan untuk menentukan ukuran setiap bentuk lingkaran, yang berarti piksel gelap digambar ulang pada ukuran terbesar. Bagian contoh skrip ditunjukkan di bawah ini:

```
color c = img.get (x, y);
fill(c);
// get the brightness
float br = brightness(c);
// map the brightness number to a useful range
br = map (br, 255, 0, 0, 20);
// use the brightness value to define the
width and height of the circle
ellipse (x, y, br, br);
```

Contoh manipulasi gambar sederhana di mana kecerahan setiap warna piksel digunakan untuk menentukan ukuran setiap kotak saat digambar ulang. Area yang lebih gelap digambar dalam ukuran yang lebih besar, sehingga menciptakan gaya gambar yang "impresionistis".

Impor Sumber Video Langsung

Contoh-contoh sejauh ini membahas tentang menemukan dan menggunakan informasi warna yang bersumber dari berkas gambar statis; namun, teknik-teknik ini dapat diterapkan secara setara pada gambar bergerak yang bersumber dari umpan video langsung (dari webcam, misalnya). Mengalihkan sumber input dari satu berkas gambar ke umpan kamera langsung berarti bahwa efek yang sama dari mengekstraksi dan memanipulasi informasi piksel warna dapat digunakan untuk menciptakan efek yang lebih dinamis secara visual.

Untuk memulai, pustaka Video diimpor ke sketsa Pemrosesan:

```
import processing.video.*;
```

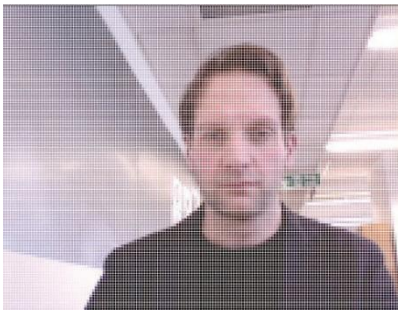
Setelah diimpor, pustaka tersebut kemudian dapat digunakan untuk mendapatkan dan menampilkan umpan video langsung dari webcam atau kamera internal yang terhubung. Objek Tangkap baru dibuat yang membuat koneksi ke kamera, memulai umpan, dan menggambar gambar di layer:

```
// CAPTURE AND DISPLAY A VIDEO FEED //
import processing.video.*;
Capture camera;
void setup () {
  size (800, 600);
  camera = new Capture (this, 320, 240);
  camera.start();
}

void draw () {
  if (camera.available()) {
    camera.read();
  }
}
  }
  Image (camera, 0, 0);
}
```

Setelah umpan langsung dibuat, informasi warna dari setiap bingkai video dapat diekstraksi, menggunakan struktur loop "for" bersarang dasar yang sama yang sebelumnya telah diterapkan pada gambar statis tunggal. Hasilnya adalah semacam mosaik digital, sejenis cermin berpiksel.

```
// PIXEL MIRROR FROM CAMERA FEED /////
import processing.video.*;
Capture camera;
float spacing = 5;
float dotSize = 4;

void setup () {
  size (800, 600);
  camera = new Capture (this, 160, 120);
  camera.start();
  noStroke ();
}

//////////

void draw () {
  background (255);
  if (camera.available()) {
    camera.read();
  }

  // go through all the pixels in the video image
  // get the color of each pixel, and draw a circle
```

```
  for (int y=0; y<camera.height; y++) {
    for (int x=0; x<camera.width; x++) {

      color c = camera.get(x, y);
      fill(c);
      ellipse(x * spacing, y * spacing, dotSize, dotSize);
    }
  }
}
```



Contoh "cermin" berpiksel yang dibuat dengan menggambar ulang piksel dari sumber video langsung sebagai kisi-kisi persegi.

Sesuaikan Kecerahan

Fungsi warna yang sama yang sebelumnya diterapkan pada gambar statis red(), green(), blue(), kecerahan juga dapat diterapkan untuk mengekstrak informasi warna dari piksel gambar bergerak dan digunakan untuk memanipulasi gambar video. Misalnya, kode berikut dapat ditambahkan dalam contoh umpan kamera (di atas) untuk mendapatkan kecerahan setiap warna piksel; kode tersebut menggunakan nilai untuk mengatur ukuran setiap bentuk. Hasilnya adalah gambar bergerak terdistorsi yang berubah seiring dengan tingkat terang dan gelap.

```
// get the color from the pixel
color c = camera.get (x, y);
fill(c); // select color for fill
float br = brightness(c); // get the brightness value
// map brightness to usable range (0-8)
br = map (br, 255, 0, 0, 8);
// use brightness to set size of circle
ellipse (x*5, y*5, br, br);
```

Lingkaran digambar untuk setiap piksel dalam video, yang menciptakan efek seperti kisi-kisi. Ukuran setiap lingkaran disesuaikan menurut nilai kecerahan warna. Area terang menciptakan lingkaran yang lebih kecil dan area gelap menciptakan lingkaran yang lebih besar.

Ikuti Piksel Paling Cerah

Mengekstrak data warna dari piksel dalam gambar langsung memberikan kemampuan untuk melakukan hal-hal yang berguna seperti menemukan dan melacak area warna atau kecerahan tertentu. Dengan membandingkan nilai setiap piksel dalam gambar, piksel paling terang, paling merah, paling biru, atau paling hijau secara keseluruhan dapat ditemukan. Ini membuka kemungkinan kreatif untuk pembuatan karya interaktif. Misalnya, mendapatkan piksel paling terang dalam suatu pemandangan dapat menjadi langkah pertama dalam

membuat grafik interaktif yang mengikuti gerakan senter.

Untuk mendapatkan piksel paling terang dari suatu pemandangan, diperlukan variabel untuk menyimpan nilai piksel paling terang secara keseluruhan. Dimulai dengan nilai rendah, dan digunakan untuk menyimpan nilai kecerahan tertinggi yang ditemukannya:

```
float overallBrightest = 0;
```

Dalam loop "for" bersarang, nilai kecerahan setiap warna piksel ditemukan secara bergantian dan dibandingkan dengan nilai overallBrightest. Jika piksel ditemukan dengan tingkat kecerahan yang lebih tinggi dari level tertinggi saat ini, maka piksel tersebut disimpan sebagai nilai overallBrightest yang baru, dan lokasinya (x, y) disimpan (sebagai brightestX, brightestY). Hasil dari proses ini adalah untuk setiap bingkai dalam umpan video, piksel yang paling terang secara keseluruhan dan lokasi x, y-nya ditemukan. Sebuah bentuk kemudian dapat digambar pada lokasi x, y yang paling terang; berikut ini adalah ekstrak dari kode yang akan digunakan untuk menemukan piksel yang paling terang dalam sebuah gambar.

```
float overallBrightest = 0;

for (int y=0; y<camera.height; y++) {
  for (int x=0; x<camera.width; x++) {
    // get the COLOR of the current pixel
    color c = camera.get (x, y);
    // get the BRIGHTNESS of current pixel
    float currentBrightness = brightness(c);
    // compare this pixel with the BRIGHTEST found so far
    // if the current pixel is brighter than the overall brightest,
    // then save it and its location
    if (currentBrightness > overallBrightest) {
      overallBrightest = currentBrightness;
      brightestX = x;
      brightestY = y;
    }
  }
}
ellipse (brightestX, brightestY, 50, 50);
```

Dalam contoh ini, bentuk lingkaran akan terus melacak pergerakan lampu atau objek terang (misalnya, dapat melacak pergerakan pengguna yang melambaikan senter atau lampu LED). Dengan mengekstraksi sejumlah warna merah, hijau, atau biru, pendekatan yang sama dapat digunakan untuk melacak titik-titik dengan warna tertentu (misalnya, untuk melacak dan mengikuti pergerakan lampu yang dipilih).

Contoh pelacakan kecerahan. Lingkaran merah digambar di lokasi piksel paling terang dalam gambar. Senter dapat digunakan untuk mengontrol pergerakan bentuk di layar.

Green Screening

Pengekstrakan informasi warna juga dapat digunakan sebagai dasar teknik green screening, di mana warna yang dekat dengan warna tertentu (“kunci”) (misalnya, hijau terang) diabaikan (tidak digambar dalam pemandangan), yang memungkinkan gambar latar belakang atau grafik lain untuk ditampilkan.

Untuk mengetahui seberapa dekat piksel dengan warna kunci tertentu, perlu untuk mendapatkan nilai merah, hijau, dan biru untuk setiap warna dan menghitung keseluruhan “jarak” di antara mereka: Pertama, nilai jumlah merah, hijau, dan biru dari warna “kunci” ditemukan dan disimpan dalam hal ini, hijau paling terang:

```
color chromaKey = color (0, 255, 0);
float chromaR = red(chromaKey);
float chromaG = green(chromaKey);
float chromaB = blue(chromaKey);
```

Berikutnya, warna setiap piksel dalam umpan video ditemukan dalam struktur loop “for” dan nilai RGB diekstraksi secara bergantian:

```
color c = camera.get (x, y);
fill(c);
// extract RGB values from the current pixel
float r = red(c);
float g = green(c);
float b = blue(c);
```

Kemudian "jarak" antara nilai warna ditemukan. Fungsi dist mengembalikan satu angka, yang merupakan selisih keseluruhan antara dua set angka (dalam kasus ini, nilai RGB); dengan demikian, fungsi ini menemukan "jarak" antara dua warna (yaitu, seberapa dekat kecocokannya).

```
float colorDiff = dist (r, g, b, chromaR, chromaG, chromaB);
```

Warna yang memiliki "jarak" kecil di antara keduanya menunjukkan bahwa warna "kunci" dan warna piksel sangat cocok. Warna yang paling cocok dengan warna hijau "kunci" diabaikan. Hanya warna dengan jarak lebar di antara keduanya yang digambar:

```
if (colurDiff > 180)
{
rect (x, y, 1, 1);
}
```

Efek keseluruhan dalam contoh ini adalah piksel paling hijau diabaikan; piksel tersebut tidak digambar. Oleh karena itu, gambar atau warna latar belakang apa pun akan dapat ditampilkan melalui celah-celah ini. Dengan cara ini, efek "layar hijau" dapat dicapai, dan suatu gambar dapat dibuat muncul dalam adegan baru.

Contoh sederhana dari efek "layar hijau". Piksel hijau terang dari gambar diabaikan, sehingga gambar latar belakang dapat ditampilkan. Selembar kertas hijau berubah di depan kamera menjadi langit dan awan.

BAB 6
DATA EKSTERNAL YANG BESAR DAN LANGSUNG

"Visualisasi data adalah tentang menggabungkan visual dan konseptual."

6.1 KEHIDUPAN YANG DIJALANI DENGAN DATA

Kita hidup di dunia yang terhubung secara digital dan digerakkan oleh data. Teknologi seluler yang canggih memungkinkan orang untuk terhubung satu sama lain dan dengan data yang belum pernah ada sebelumnya. Sistem navigasi satelit (GPS) menghubungkan kita dengan peta dan informasi lalu lintas langsung; jam tangan lari merekam dan berbagi pola latihan individu, termasuk gerakan, kecepatan, dan rute yang diambil; teks dan foto dapat diunggah dan dibagikan dari lokasi terpencil di seluruh dunia. Dari statistik pemerintah melalui lembaga global hingga pesan dan gambar media sosial individu, data dibagikan setiap menit, menambah kumpulan informasi yang terus bertambah. "Masyarakat informasi" kita berenang dalam data; data ada di mana-mana dan di mana-mana.

Istilah "data" dapat terdengar dingin dan kering, memunculkan ide tentang daftar angka dan fakta yang tidak dapat dipahami; namun, meskipun sering kali bersumber secara digital, data berpusat pada manusia. Setiap set data mewakili catatan pola dan cerita aktivitas manusia. Di balik setiap teks atau daftar angka terdapat cerita, pengalaman, perasaan, atau interaksi. Data digital dapat memberikan "gambaran" suasana hati atau aktivitas sekelompok orang selama waktu tertentu atau di lokasi tertentu. Data digital merupakan catatan berharga tentang narasi, emosi, ide, dan pengalaman.

Mengambil dan memvisualisasikan data dalam format grafis merupakan cara penting dan ampuh untuk menghubungkan ide, pengalaman, dan cerita tentang interaksi manusia yang mungkin tidak terungkap atau tidak terlihat. Informasi yang kompleks atau sulit dicerna dapat diberi kehidupan dan makna baru melalui gambar visual yang mengomunikasikan fakta dan informasi secara langsung dan mudah diakses.

Praktik penyajian data dalam format visual, yang terkadang disebut sebagai "grafik informasi", telah lama menjadi bagian penting dari desain grafis dan masih tetap menjadi cara yang ampuh dan penting untuk menyajikan gambar, fakta, cerita, dan ide secara visual. Bagan, diagram, dan grafik telah digunakan sepanjang sejarah desain untuk meringkas dan mengomunikasikan informasi dengan cara yang penting dan bermakna.

Sejak tahun 1858, Florence Nightingale menggunakan diagram pai yang sekarang dikenal sebagai cara visual untuk meringkas angka kematian tentara dan data yang menyoroti bagaimana kebersihan yang buruk di rumah sakit menyebabkan penyakit yang dapat dicegah. Representasi grafisnya terhadap halaman-halaman data rumah sakit membawa tujuannya ke permukaan dengan meringkas fakta-fakta menjadi satu informasi visual langsung, yang membantu mengubah kondisi perawatan para prajurit yang terluka.

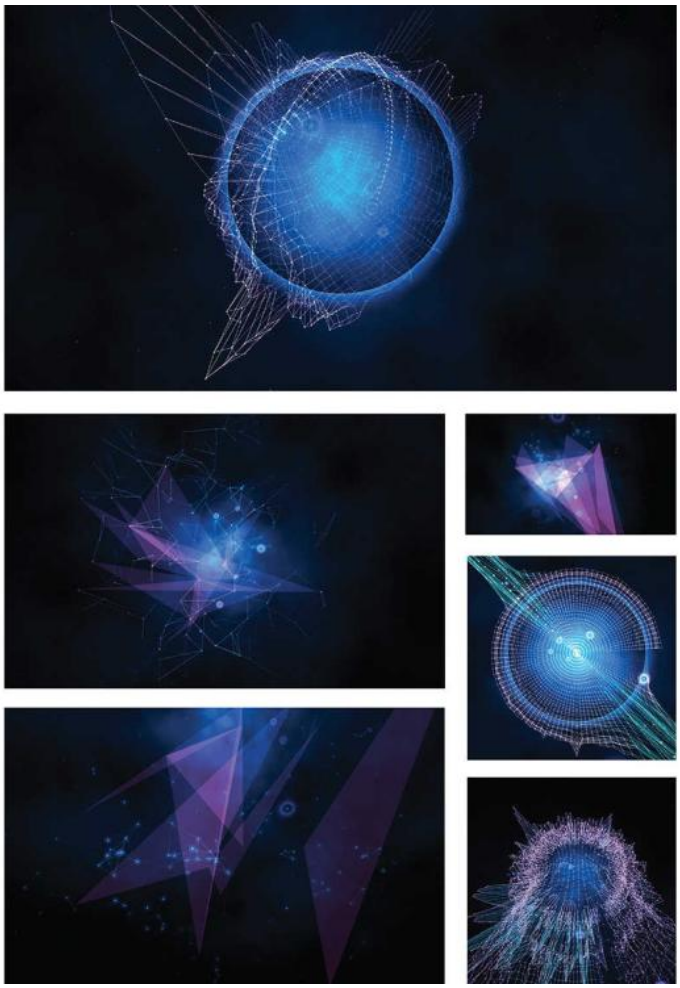
Banyak bidang, mulai dari olahraga hingga pemerintahan, menggunakan grafik informasi untuk menyajikan dan mengomunikasikan informasi untuk berbagai alasan dan dalam konteks yang beragam. Grafik informasi bahkan dapat digunakan untuk membuat data

tampak indah.

Menyajikan data dalam format visual dapat membuat konten menarik dan memikat bagi pemirsa. "Mempercantik" data merupakan cara penting untuk membuat data diperhatikan atau dilihat dengan cara baru. Desain visual merupakan alat yang ampuh untuk mengungkap pola tersembunyi dalam data dan mengomunikasikannya dengan cara yang bermakna dan mencolok. Oleh karena itu, keterampilan kreatif desainer grafis berperan penting dalam mengomunikasikan "cerita" data.

6.2 KODE SEBAGAI ALAT VISUALISASI DATA

Meskipun visualisasi data dan grafik informasi telah lama digunakan sebagai bagian dari praktik desain grafis, peningkatan yang belum pernah terjadi sebelumnya, dan akses ke data digital selama beberapa tahun terakhir telah membuat bidang desain ini dengan cepat menjadi terkenal karena data menjadi tersedia dalam jumlah yang semakin banyak. Di era data modern, alat yang paling ampuh untuk mengakses dan memvisualisasikan kumpulan data besar bukanlah perangkat lunak grafik dan gambar tradisional, melainkan pemrograman dan kode.



Grid Futura Episode 1
Kode dapat digunakan untuk memvisualisasikan data dari sumber audio. Ini adalah visualisasi suara multisentuh interaktif yang dibuat untuk acara langsung yang menerjemahkan trek audio ke dalam ruang grafis 3D. Aplikasi ini terdiri dari versi desktop (dibuat menggunakan Processing) untuk visualisasi grafis musik secara real-time dan juga versi iOS untuk berinteraksi dengan aplikasi Processing (dibuat menggunakan openFrameworks).

Kode dapat digunakan untuk memvisualisasikan data dari sumber audio. Ini adalah

visualisasi suara multisentuh interaktif yang dibuat untuk acara langsung yang menerjemahkan trek audio ke dalam ruang grafis 3D. Aplikasi ini terdiri dari versi desktop (dibuat menggunakan Processing) untuk visualisasi grafis musik secara real-time dan juga versi iOS untuk berinteraksi dengan aplikasi Processing (dibuat menggunakan openFrameworks).

Kode adalah lingkungan visualisasi data yang kuat, yang menciptakan hubungan langsung antara sumber data dan keluaran grafis. Kode memberi desainer dan programmer cara unik untuk mengekstrak sejumlah besar informasi digital dan langsung menerapkannya ke keluaran visual. Nilai data langsung dapat secara langsung mengubah variabel yang tertanam dalam sepotong kode yang menentukan visual grafik atau gambar digital (misalnya, ukuran, warna, bentuk, posisi, dll.).

Desainer dan programmer menggunakan kode untuk menghubungkan sumber data eksternal ke grafik dan visual dengan cara yang dinamis dan responsif. Skala data yang dapat diproses dengan cara ini memberi desainer kesempatan untuk menggunakan dan merepresentasikan kumpulan data yang besar dengan cara yang semakin kompleks. Kekuatan pemrosesan dari proses pemrograman digital memberi desainer kesempatan untuk membuat visual data dalam skala yang sebelumnya tidak terbayangkan. Kode dapat menyaring dan mengatur sejumlah besar data dengan cara yang tidak dapat dilakukan oleh proses manual; kode dapat membuat asosiasi visual di antara kualitas yang ditemukan dalam nilai data, yang mengungkap pola menarik dari aktivitas manusia dalam kelompok.

6.3 SUMBER DATA

Menemukan dan memilih sumber data yang paling tepat dan paling menarik merupakan bagian penting dari setiap proses visualisasi data. Dunia daring dibanjiri dengan data saham dan obligasi, suhu, berita utama, gambar, opini, lokasi geografis, dan sebagainya yang masing-masing dapat menjadi titik awal yang berpotensi menarik untuk karya kreatif. Data dapat berupa nilai tunggal, langsung, dan bervariasi (misalnya, informasi cuaca atau nilai saham dan obligasi) di situs web atau sebagai lembar kerja besar yang dapat diunduh (misalnya, dokumen Excel atau dokumen Google) berisi informasi statistik yang dikumpulkan oleh pemerintah atau lembaga. Dalam lingkungan yang memiliki begitu banyak informasi digital, masalahnya sering kali bukanlah kurangnya data, tetapi jumlahnya.

Apa pun sumber datanya, sebagian besar pekerjaan pemrograman awal dengan nilai data melibatkan penyaringan dan pencarian melalui sumber untuk menemukan elemen informasi yang paling berguna atau menarik dan mengekstraknya dalam format yang "bersih" dan dapat digunakan. Ini dapat berarti menghapus tag dan teks yang salah dan tidak diinginkan dari berkas HTML atau menyortir baris dan kolom pada tabel informasi. Proses pencarian nilai data terkadang disebut sebagai "penambangan data". Berikut ini adalah ringkasan beberapa format utama untuk mengakses data daring.

HTML

Cara mudah untuk mendapatkan data adalah dengan mengakses informasi langsung dari sumber web. Dokumen HTML *(HyperText Markup Language)* menyediakan titik awal yang baik untuk melakukan ini. Banyak nilai data yang ada sebagai elemen yang diperbarui secara

dinamis dalam badan halaman web yang diperbarui secara dinamis (HTML). Informasi yang dapat diubah, seperti skor olahraga dan nilai saham dapat ditemukan dalam sumber HTML dokumen web, yang menyediakan sumber data numerik yang sederhana dan langsung. Nilai data individual tersebut dapat diekstrak dari halaman web dan digunakan untuk menggerakkan grafik dinamis.

Kode sumber dokumen HTML dapat diimpor langsung ke dalam program, yang kemudian menyortir dan menyaringnya, mengekstrak bit informasi yang berguna. Setelah diekstrak, nilai data dari situs web dapat digunakan sebagai bagian dari grafik yang sederhana namun dinamis. Sumber (teks berformat HTML) situs web prakiraan cuaca dapat, misalnya, berisi nilai-nilai yang berhubungan dengan suhu saat ini dan mendatang di kota tertentu di dalam badan halaman. Memuat HTML, lalu menyaring dan mengekstrak nilai-nilai ini, dapat menghasilkan grafik yang menyesuaikan dengan perubahan kondisi cuaca mungkin yang dialami di belahan dunia lain.

Meskipun merupakan sumber data yang dapat diakses, konten berformat HTML tidak menyajikan informasi dengan cara yang dapat langsung diakses; hal itu mengharuskan programmer melakukan banyak penyortiran dan pencarian (penambangan) sebelum sepotong data dapat diekstraksi dan digunakan.

XML

Informasi yang terkubur dalam di dalam badan dokumen web (berformat HTML) dapat dibaca oleh manusia, yang mampu, setelah beberapa kali mencari, untuk melihat dan memilih data yang diperlukan. Dengan melihat kode sumber secara saksama, kita mungkin dapat menemukan nilai yang sesuai dengan bagian informasi tertentu yang kita inginkan. Namun, kemampuan kita sendiri untuk menyaring tag HTML yang rumit untuk menemukan bagian data penting tertentu dari dokumen HTML tidak dapat dengan mudah diterjemahkan ke dalam program komputer.

Menulis kode untuk melakukan tugas yang setara dengan menggali item data dari dokumen HTML dapat menjadi rumit dan bermasalah. Di sisi lain, dokumen XML *(Extensible Markup Language)* menyediakan cara yang lebih berguna dan mudah diakses untuk mengatur data, yang membantu meringankan masalah ini.

XML adalah bahasa pemformatan yang banyak digunakan untuk mendeskripsikan data dalam format yang berguna dan mudah diakses. Meskipun HTML digunakan untuk mendeskripsikan konten halaman, XML adalah cara yang berguna untuk mendeskripsikan data di dalam halaman dan menyediakan cara yang efisien untuk mendapatkan inti data secara daring. XML tidak "melakukan" apa pun; XML merupakan alat untuk membawa informasi dan format ideal untuk mengakses dan menemukan informasi.

Daripada menggunakan tag yang telah ditetapkan sebelumnya (seperti dalam HTML), elemen data dalam file XML dibungkus dalam tag yang ditetapkan oleh programmer, sehingga setiap item data memiliki deskripsi yang akurat dan mudah dipahami. XML merupakan format yang sangat fleksibel untuk mendeskripsikan data. XML digunakan untuk berbagai tujuan (konten halaman, basis data, katalog, dll.). XML merupakan format yang bagus untuk mengambil dan menggunakan sumber data yang menarik.

Data dalam dokumen XML biasanya bertingkat (yaitu, disimpan di dalam satu sama lain), menciptakan struktur percabangan dari "item" dan "sub item" (setara dengan folder dan sub folder). Struktur bertingkat yang terorganisasi ini membantu mendefinisikan dan menyusun informasi dengan cara yang logis dan mudah diakses. Struktur ini memungkinkan para programmer untuk menggali dan menavigasi melalui apa yang mungkin merupakan kumpulan data yang kompleks. Komputer membaca dokumen XML jauh lebih mudah daripada dokumen HTML. Kode dapat ditulis untuk mencari detail data, menavigasi dengan mudah melalui gaya direktori dokumen XML yang terorganisasi.

Data XML ada di seluruh web. Data daring seperti umpan cuaca atau berita dalam format ini dapat diidentifikasi, dipilih, dan diimpor ke dalam aplikasi pemrograman yang bebas untuk mewakilinya dengan cara yang baru dan asli.

Data Tabular

Sejumlah besar data statistik nasional dan global dikumpulkan, disimpan, dan disediakan untuk penggunaan publik dari beberapa sumber daring. Beberapa data ini berisi fakta menarik dan angka latar belakang yang berkaitan dengan kesehatan, ekonomi, isu sosial, dan bidang lainnya, jenis data ini biasanya diformat sebagai satu lembar kerja besar atau tabel nilai dan properti. Informasi semacam ini biasanya diekspor dan diformat ulang sebagai satu daftar besar nilai yang dipisahkan oleh koma atau tab. Tidak mengherankan, ini disebut sebagai *Comma Separated Values* (CSV) atau *Tab Separated Values* (TSV).

Daftar besar data TSV atau CSV, meskipun terlalu sulit dibaca oleh manusia, berada dalam format yang berguna dan mudah dicerna untuk digunakan oleh bahasa dan program komputer. Format data yang sistematis (pemisahan koma), yang menghapus semua format teks yang tidak perlu, menyajikan data "mentah" dalam satu penyajian sederhana, sehingga pemrogram dapat dengan mudah menemukan dan mengekstrak informasi individual. Format CSV khususnya berguna untuk data angka dalam skala besar mulai dari suhu di seluruh dunia hingga tren kesehatan dunia hingga peta informasi dari rute dan koordinat GPS.

API (Antarmuka Pemrograman Aplikasi)

API atau "antarmuka pemrograman aplikasi" memungkinkan aplikasi untuk berkomunikasi satu sama lain. API merupakan cara yang baik untuk mengekstrak konten daring dalam format yang bermanfaat (misalnya, baik sebagai konten berformat XML atau JSON) dan khususnya berguna untuk mencari sumber informasi yang dibuat daring oleh aplikasi media sosial. Ada banyak API yang tersedia yang memungkinkan programmer dan pengembang untuk mengakses data dalam aplikasi mereka.

Facebook dan Twitter, misalnya, memiliki API mereka sendiri, yang memungkinkan pengembang dan pembuat kode untuk mengekstrak dan mengakses komentar, gambar, pembaruan status, dan elemen lain dalam format yang dapat digunakan untuk membuat aplikasi baru (dengan Pemrosesan, misalnya) atau visualisasi data. Setiap API memiliki panduan referensi dan dokumentasinya sendiri, yang biasanya tersedia di bagian “pengembang” penyedia konten di situs web mereka.

6.4 DATA PEMETAAN

Peta adalah representasi grafis dan interpretasi dari dunia fisik. Perancang peta menyajikan versi lanskap geografis tertentu yang telah diedit, dengan menambahkan atau menghilangkan landmark, tempat, dan titik menarik yang terlihat sesuai dengan fungsi dan tujuan peta. Selain fitur dunia fisik yang terlihat, ada lapisan data tambahan yang dapat dihubungkan ke lokasi dan tempat tertentu di seluruh dunia.

Semakin banyak data yang terhubung dengan Lokasi baik melalui perangkat GPS berbasis lokasi individual atau melalui data statistik nasional dan internasional sekarang tersedia untuk membuat peta berbasis data yang informatif secara visual. Peta tradisional menandai lokasi fisik dan karakteristik suatu lokasi; peta data menandai dan menemukan informasi tak terlihat yang melibatkan pergerakan, pengalaman, dan interaksi manusia.

Pemetaan Orang

Data lokasi tersebar luas dalam kehidupan digital kita. Perangkat yang memetakan posisi pengguna semakin umum digunakan. Teknologi GPS (satelit) yang tertanam di ponsel, komputer, dan perangkat yang dapat dikenakan menempatkan pengguna dengan tingkat akurasi yang luar biasa. Aplikasi media sosial dapat menggunakan dan mengirimkan informasi lokasi, yang memungkinkan orang untuk berbagi "apa" dan "di mana" dalam kehidupan digital mereka.

Selain menjadi alat navigasi, data lokasi menghasilkan jejak data digital kehidupan kita. Catatan data tentang lokasi dan pergerakan orang saat mereka berinteraksi menghasilkan lapisan informasi digital yang lancar yang dapat dikomunikasikan oleh programmer dalam peta data.

Mengumpulkan sejumlah besar data menciptakan awan data besar berisi pemikiran dan pengalaman peta digital yang lancar tentang pengalaman pengguna dan aktivitas manusia. Data lokasi media sosial yang dihubungkan dengan informasi lain informal atau formal mengungkapkan jejak data digital berisi kata-kata, tempat, gambar, dan pemikiran bersama yang dapat diambil oleh seniman, desainer, dan programmer untuk ditafsirkan ulang secara visual guna menciptakan pola kehidupan manusia digital. Peta-peta ini menciptakan pandangan global tentang aktivitas individu, yang secara kolektif menciptakan perspektif baru dan terbuka tentang aktivitas manusia.

CONTOH

Onformative: Pohon Facebook 4010

Deutsche Telekom menugaskan onformative untuk mendesain dan membuat Dinding Galeri toko utama mereka di Berlin. Toko tersebut berkomunikasi dengan pelanggan di Facebook, memasang foto, penawaran khusus, dan acara. Tim desain mencetak visualisasi data untuk menutupi dinding, yang dibuat dari pesan dan komunikasi Facebook milik toko itu sendiri.

Dengan menggunakan Facebook Graph-API, tim desain mengakses dan menganalisis data yang direkam selama periode empat tahun dengan perangkat lunak khusus yang ditulis dalam Processing. Tantangannya adalah mengambil sejumlah besar elemen data yang berbeda dari Facebook API (misalnya, kiriman, jumlah suka, topik, komentar, dan waktu) dan

mengilustrasikannya dalam satu grafik data yang menarik.

Struktur tanaman organik terbukti menjadi metafora visual yang berguna untuk menghubungkan dan mengilustrasikan berbagai untaian dan elemen jaringan sosial. Daun-daun dibuat berdasarkan "karakteristik" yang ditemukan pada setiap kiriman misalnya, tanggal dan waktu pembuatan, jumlah komentar dan like yang diterima setiap kiriman, serta topik kiriman.

Nilai dari setiap elemen ini dipetakan untuk membuat karakteristik, bentuk, dan warna masing-masing daun. Jumlah "warna merah" pada daun, misalnya, ditentukan oleh jumlah like yang diterima setiap kiriman, dan waktu kiriman dibuat menentukan seberapa banyak daun tersebut terbuka (kiriman yang dibuat pada pagi atau sore hari lebih sedikit terbuka daripada kiriman yang dibuat pada siang hari) tunas yang muncul di sekitar daun mencerminkan jumlah komentar di sekitar kiriman.

Setiap jenis kiriman baik pesan, tautan, atau foto menghasilkan gaya daunnya sendiri dengan karakteristik visual yang dapat diidentifikasi. Cabang digunakan untuk menghubungkan kiriman berdasarkan tanggal pembuatannya, yang memungkinkan pemirsa untuk membaca perkembangan komunikasi Facebook dari waktu ke waktu, dari akar pohon hingga puncaknya. Untuk menghasilkan tata letak akhir, tim menggunakan Pemrosesan untuk menghasilkan berbagai variasi secara otomatis, lalu mengeditnya secara manual untuk mendapatkan tata letak dan tampilan yang diinginkan. Hasil akhirnya sangat memukau secara visual dan menarik karena penanganan datanya yang terperinci. Melihat komunikasi kelompok sebagai bentuk organik menciptakan citra visual akhir yang kuat yang beresonansi dengan komunitas media sosial.

Desain informatif Pohon Facebook 4010

Visualisasi aktivitas Facebook pelanggan untuk Deutsche Telekom, Berlin. Setiap cabang dan pakis terkait dengan tema, topik, dan kiriman tertentu yang diambil dari halaman Facebook perusahaan itu sendiri. Struktur organik menyoroti untaian percakapan dan aktivitas daring sebagai bagian dekoratif dari wallpaper berbasis data.

Eric Fischer: Lihat Sesuatu Katakan Sesuatu; Turis dan Penduduk Lokal

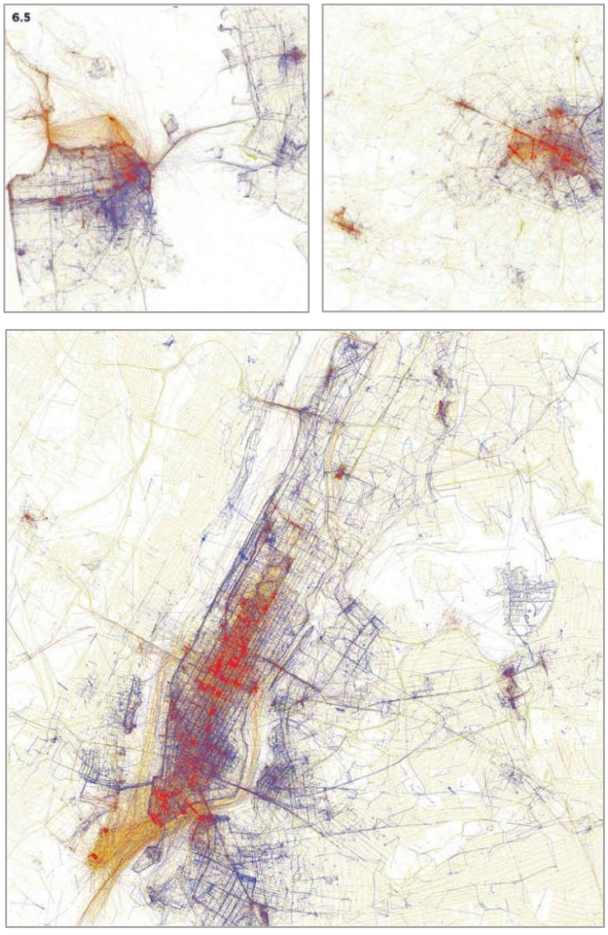
Eric Fischer adalah seniman/desainer visualisasi data yang menggunakan informasi media sosial untuk memetakan lokasi orang saat mereka bergerak melalui dan di sekitar kota.

Hasilnya adalah serangkaian peta data yang menakjubkan secara visual.

Lihat Sesuatu, Katakan Sesuatu
Eric Fischer
Visualisasi data New York, London, dan Tokyo, dibuat dari Tweet pengguna yang diberi tag geografis. Titik-titik berwarna menunjukkan tempat seseorang membagikan foto Flikr (oranye), atau mengeposkan Tweet (biru), atau keduanya (putih).

Fischer menggunakan kode pemrograman (biasanya C) untuk mengambil data geolokasi dari jaringan media social terutama Twitter API atau Flickr. Ia kemudian menggunakan kembali informasi tersebut untuk memetakan piksel yang mengungkapkan lokasi pengguna saat mereka bergerak melalui suatu ruang dan terhubung dengan jaringan sosial. Hasil proyek dibagikan secara daring dan membuat peta data yang sangat terperinci yang menyajikan visi baru dan menarik tentang dunia kita. Kota, kota besar, dan bahkan negara dipetakan dengan cara ini, yang menghidupkan pola aktivitas media sosial yang dipetakan ke lokasi geografis tertentu.

Turis dan Penduduk Lokal Eric Fischer



Beberapa proyek penting dari rangkaian eksperimen yang sedang berlangsung ini meliputi "Lihat Sesuatu, Katakan Sesuatu" dan "Turis dan Penduduk Lokal." Dalam setiap eksperimen ini, Fischer mengeksplorasi perbedaan dalam data untuk membedakan dan menemukan di antara jenis aktivitas pengguna.

Proyek "Lihat Sesuatu, Katakan Sesuatu" menggunakan warna untuk menunjukkan pengguna yang melihat sesuatu (berbagi foto), mereka yang mengatakan sesuatu (men-Tweet beberapa teks), dan mereka yang melakukan keduanya. Titik-titik berwarna berbeda dipetakan untuk menunjukkan lokasi masing-masing kelompok ini.

Dalam proyek "Turis dan Penduduk Lokal", warna digunakan untuk menunjukkan pengguna yang baru mengenal kota dan mereka yang sedang berkunjung. Perbedaan

ini diperoleh dengan melihat riwayat lokasi setiap pengguna untuk menentukan apakah mereka baru saja bepergian ke kota tersebut. Warna di seluruh peta menunjukkan keterbatasan penjelajahan wisatawan di sekitar kota, menyoroti tempat-tempat menarik dan batas-batas pengembaraan mereka.

Menandai dan menjelajahi perbedaan sederhana ini secara visual dalam data pengguna akan membantu kita memahaminya dan menambah kekuatan visualnya.

Visualisasi data peta New York, Paris, dan San Francisco, yang menunjukkan aktivitas penduduk lokal dan pengunjung kota. Titik biru menunjukkan gambar yang diambil oleh penduduk lokal; titik merah adalah gambar yang diambil oleh wisatawan; titik kuning adalah gambar pengguna yang mungkin merupakan salah satu dari keduanya.

Memetakan Perjalanan

Selain dapat menemukan dan memetakan lokasi pengguna perorangan, data dapat diakses dari perangkat GPS, seperti jam tangan lari, yang merekam dan menyimpan data rute atau perjalanan perorangan. Dengan menggunakan teknologi ini, setiap orang dapat memetakan dan membagikan rute lari atau jalan kaki pribadi mereka, menyimpan dan membagikan setiap langkah saat mereka melakukan perjalanan melintasi lanskap. Daftar nilai GPS (lintang dan bujur) yang disusun dari lari atau jalan kaki memberikan banyak informasi angka yang menarik dan dapat digunakan, yang dapat diubah menjadi grafik atau gambar digital.

Lari atau jalan kaki tunggal akan menghasilkan daftar besar nilai angka: rekaman setiap langkah, putaran, dan belokan perjalanan pengguna tunggal. (Daftar ini juga akan mencakup data tambahan, seperti jarak, kecepatan elevasi, dll.) Nilai-nilai ini, saat dimasukkan ke dalam program gambar, dapat diubah menjadi garis dan guratan, yang menciptakan representasi grafis dari setiap perjalanan perorangan.

jenis data yang berfokus pada manusia ini mengaitkan nilai numerik dengan pengalaman hidup yang nyata, memungkinkan desainer dan programmer untuk menciptakan visualisasi yang bermakna dan menarik, didasarkan pada pengalaman individu yang khas.

Sebagai kelanjutan dari proyek awal, sebuah instalasi ritel dikembangkan untuk memvisualisasikan data lari selama satu tahun yang dikumpulkan dari situs Nike+. Perangkat lunak khusus dirancang untuk memutar ulang puluhan ribu aktivitas lari dari tiga kota besar: New York, London, dan Tokyo. Animasi visual yang dihasilkan berdasarkan data dari ribuan pelari menampilkan perjalanan masing-masing individu sekaligus menyoroti semangat kolektif dari komunitas pelari perkotaan di setiap kota tersebut.

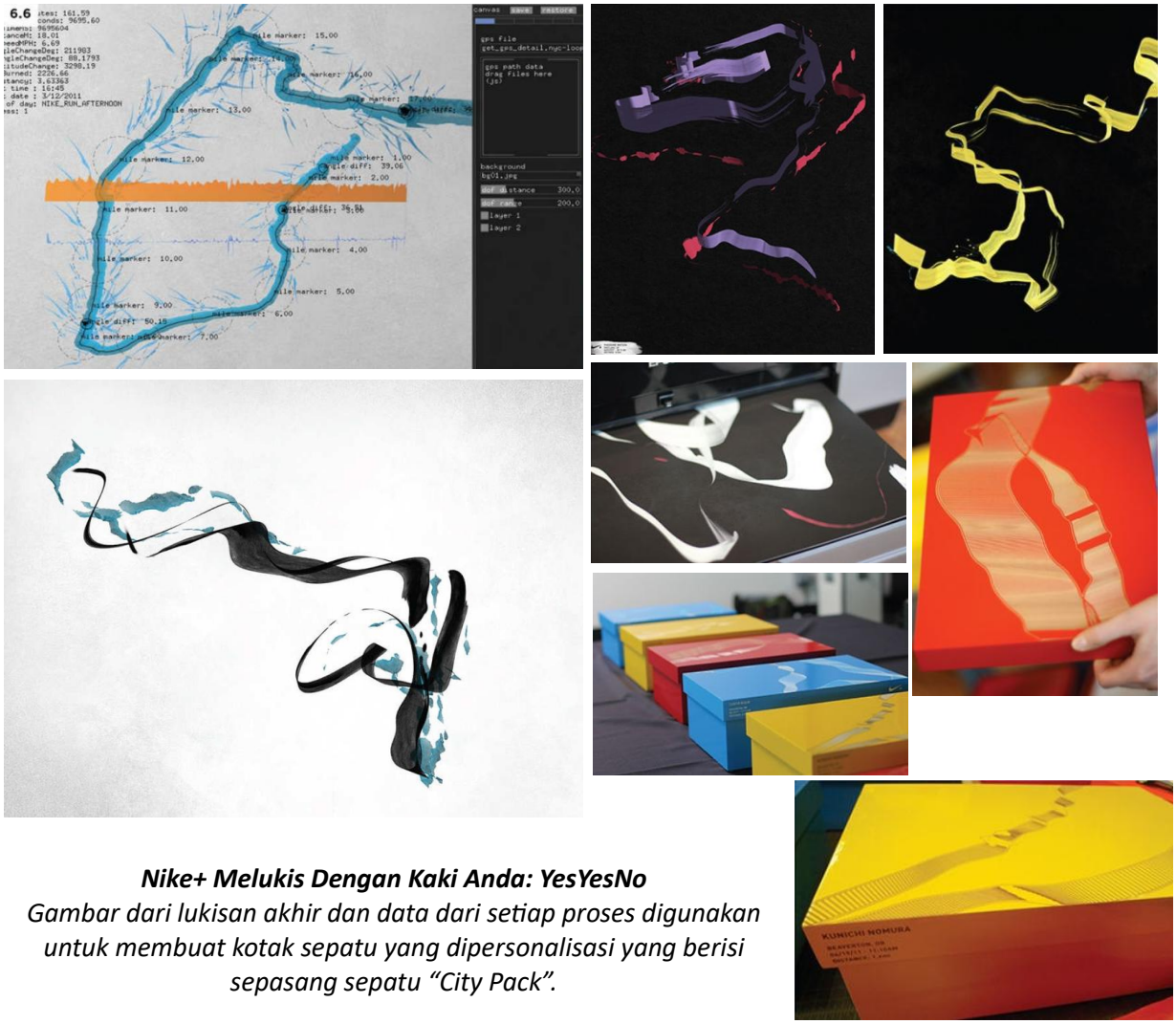
CONTOH

Nike +: Lukis dengan Kaki Anda

Lukis dengan Kaki Anda adalah proyek yang dibuat oleh agensi desain YesYesNo untuk bekerja sama dengan peluncuran seri Nike Free Run+ 2 City Pack. Dengan menggunakan data GPS yang diambil dari perangkat Nike+, para pelari diundang untuk membuat lukisan dinamis mereka sendiri dengan kaki mereka.

Data yang diambil dari para pelari selama periode dua hari dikumpulkan dan diimpor ke dalam perangkat lunak khusus, yang menggunakannya untuk membuat grafik berdasarkan

parameter tertentu: kecepatan, konsistensi, dan gaya lari unik setiap orang.



Nike+ Melukis Dengan Kaki Anda: YesYesNo
Gambar dari lukisan akhir dan data dari setiap proses digunakan untuk membuat kotak sepatu yang dipersonalisasi yang berisi sepasang sepatu "City Pack".

6.5 PEMETAAN MASYARAKAT

Peta data dapat dibuat dari sumber daring lain yang tidak terlalu personal. Arsip yang berisi sejumlah besar data statistik yang disusun oleh organisasi atau lembaga (sosial) besar sering kali menyusun dan berbagi temuan dan analisis isu sosial, seperti kesehatan, kekayaan, politik, atau kebahagiaan sekelompok besar orang.

Temuan dari sumber data semacam ini biasanya diperingkat menurut lokasi (misalnya, kota, kabupaten, negara bagian, provinsi, kode pos, atau negara). Kode yang memberikan nilai visual pada data angka mentah dan memetakan data menurut lokasi dapat digunakan untuk memetakan perubahan sosial dan sikap di seluruh wilayah dan negara. Menemukan, mengakses, dan memvisualisasikan informasi semacam ini memungkinkan desainer untuk merepresentasikan data dengan cara yang secara langsung mengomunikasikan isu-isu penting sosial dan ekonomi.

CONTOH

Jed Carter: Eyes on the Sky

Eyes on the Sky adalah investigasi kreatif terhadap desain generatif dan cuaca. Carter mengumpulkan data gambar dari enam puluh empat webcam akses publik dari seluruh Eropa (satu jam sekali selama dua puluh empat jam setiap hari) dan menggunakan informasi tersebut untuk menghasilkan serangkaian gambar digital abstrak. Warna langit diekstraksi dari setiap foto webcam sebagai nilai numerik merah, hijau, dan biru, dan Pemrosesan digunakan untuk memetakan warna kembali ke lokasi geografis yang benar. Rangkaian gambar terakhir, yang memetakan perubahan warna langit di seluruh Eropa selama satu minggu, disusun menjadi buku peta cuaca berbasis data.

Mata di Langit: Jed Carter
Menggunakan data yang diambil dari webcam publik, buku ini mendokumentasikan secara grafis perubahan cuaca dan warna langit di berbagai kota di Eropa selama satu minggu.

Memetakan Emosi

Selain dapat menemukan dan menggunakan informasi berbasis lokasi, kode juga dapat mengumpulkan dan memetakan pikiran, ide, dan perasaan yang kurang nyata. Dunia daring seperti tempat pertemuan virtual; forum publik tempat orang-orang dari berbagai latar belakang sosial, agama, etnis, dan politik saling mengomunikasikan pendapat, keyakinan, ide, harapan, dan ketakutan pribadi.

Komunitas digital dipenuhi dengan percakapan dan pertukaran, mulai dari yang sepele hingga yang mendalam, saat pengguna berdiskusi dan berdebat. Lingkungan media sosial merupakan sumber emosi manusia daring yang sangat besar, yang memetakan komunikasi pengguna di seluruh belahan dunia. Tweet, pembaruan status, gambar, video, tautan, komentar, kiriman ulang, dan kiriman blog semuanya menambah percakapan digital global yang luas dan terus berkembang.

Alat pemrograman dapat memanfaatkan kumpulan diskusi digital yang luas ini untuk mengumpulkan informasi dan memberikan wawasan tentang pikiran, opini, dan emosi komunitas daring. Kode dapat mencari media sosial (Twitter, Facebook, kiriman blog) untuk menemukan informasi tentang tema, waktu, atau tempat tertentu. API khusus untuk aplikasi media individual memberi programmer akses ke data di balik aktivitas media sosial, yang

memungkinkan mereka untuk mencari dan mengambil berbagai macam data, termasuk konten postingan, media yang disematkan, tautan, detail kapan dan di mana postingan ditulis, oleh siapa, serta tautan ke pengguna atau percakapan terkait.

Pencarian dalam kumpulan data yang sangat besar berarti bahwa programmer perlu menetapkan batasan untuk mencegah terciptanya serangkaian hasil yang begitu besar hingga tidak berarti. Penting untuk memiliki gagasan yang jelas tentang jenis data spesifik yang diinginkan; ini membantu memberi hasil lebih banyak bobot dan makna. Pencarian yang luas, menggunakan kata kunci yang luas seperti "cinta," akan menghasilkan terlalu banyak hasil, yang memerlukan penyaringan dan penyortiran lebih lanjut.

Membatasi pencarian ke frasa atau rangkaian kata kunci tertentu memberikan hasil yang lebih koheren dan bermakna. Menggunakan kombinasi istilah atau menempatkan parameter pada pencarian membantu. Menggabungkan frasa, seperti "cinta" dan "London," secara alami akan mempersempit hasil, menawarkan wawasan yang lebih besar ke dalam topik tertentu. Setelah hasil ditemukan, masih banyak pekerjaan yang harus dilakukan untuk menyempurnakan atau mengkategorikannya guna mengungkap pola atau tema tersembunyi. API aplikasi jejaring sosial dapat membantu proses penyempurnaan ini, yang memungkinkan programmer untuk menggali lebih dalam hasil tersebut. Misalnya, melihat waktu penulisan posting, lokasi, atau usia penulis memberikan wawasan lebih dalam dan menambahkan lebih banyak lapisan pada hasil data.

Serupa dengan itu, membatasi pencarian pada jangka waktu atau lokasi tertentu juga dapat membantu untuk memperjelas dan memfilter hasil. Misalnya, mencari kata "gol" selama final Piala Dunia akan menghasilkan serangkaian hasil yang berhubungan dengan pertandingan saat berlangsung, saat orang-orang berkomentar, mengantisipasi, atau bereaksi selama dan setelah gol tercipta. Mencari kata "gol" di waktu lain (tentu saja) akan menghasilkan serangkaian hasil yang jauh lebih luas dan kurang menarik.

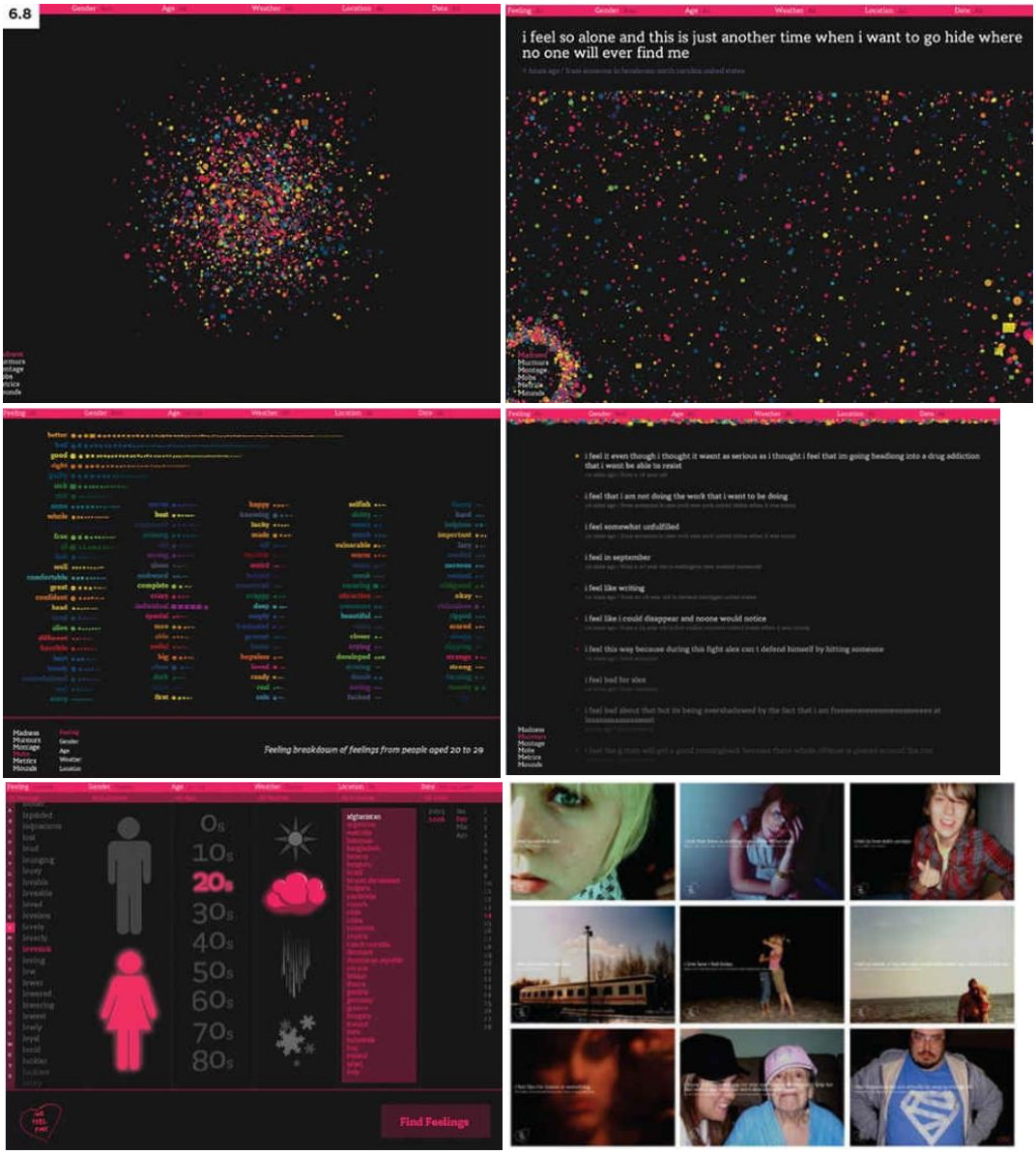
Hasil pencarian akan selalu berisi elemen yang tidak relevan atau tidak relevan dengan tema yang dimaksud; namun, serangkaian hasil yang cukup bermakna akan mengesampingkan sejumlah kecil jawaban yang salah.

CONTOH

Jonathan Harris dan Spe Kamvar: We Feel Fine

We Feel Fine adalah aplikasi daring inovatif yang dibuat oleh Jonathan Harris dan Spe Kamvar (awalnya pada tahun 2005). Konsep di balik karya ini sederhana, tetapi hasilnya adalah eksplorasi visual yang luar biasa tentang emosi manusia secara daring.

Karya ini terus-menerus menelusuri berbagai entri blog yang baru diposting, mencari frasa "Saya merasa" atau "Saya sedang merasa." Setiap kali kecocokan dengan salah satu frasa ini ditemukan, kalimat lengkapnya direkam dan disimpan dalam basis data emosi dan perasaan yang terus bertambah yang diambil dari seluruh dunia. Oleh karena itu, setiap entri ke dalam basis data tersebut merupakan catatan unik tentang suasana hati seseorang pada suatu titik waktu tertentu.



Dario Taraborelli, Giovanni Luca Ciampaglia, dan Moritz Stefaner: Notabilia
Notabilia adalah visualisasi berbasis penelitian dari diskusi yang terjadi seputar penghapusan artikel di Wikipedia. Komunitas Wikipedia meninjau setiap artikel yang diposting berdasarkan pedoman "keterkenalan" untuk menentukan apakah topik yang dimaksud cocok untuk dimasukkan atau tidak. Editor dapat menominasikan artikel untuk dihapus dan, jika nominasi tersebut sah, diskusi komunitas akan berlangsung, di mana para anggota mendiskusikan alasan untuk atau menentangnya.

Selain merekam suasana hati seseorang yang tercatat di blog mereka ("Saya merasa senang, sedih, dll.''), aplikasi ini juga menangkap data lain untuk memberikan pemahaman yang lebih menyeluruh tentang, dan latar belakang, perasaan tersebut. Data yang diekstrak dari blog, seperti usia, jenis kelamin, dan lokasi penulis, disimpan, bersama dengan tanggal dan kondisi cuaca pada saat postingan ditulis. Gambar-gambar dari posting blog juga disusun bersama kalimat emosional. Setiap hari antara lima belas hingga dua puluh ribu perasaan baru ditambahkan, menghasilkan basis data besar perasaan manusia.

Data itu sendiri ditampilkan sebagai awan partikel berwarna-warni yang terorganisasi sendiri. Properti visual setiap partikel warna, ukuran, bentuk, dan opasitasnya didefinisikan menurut perasaan yang diwakilinya. Setiap partikel dapat diklik untuk menampilkan datanya: frasa lengkap, gambar, dan informasi latar belakang.

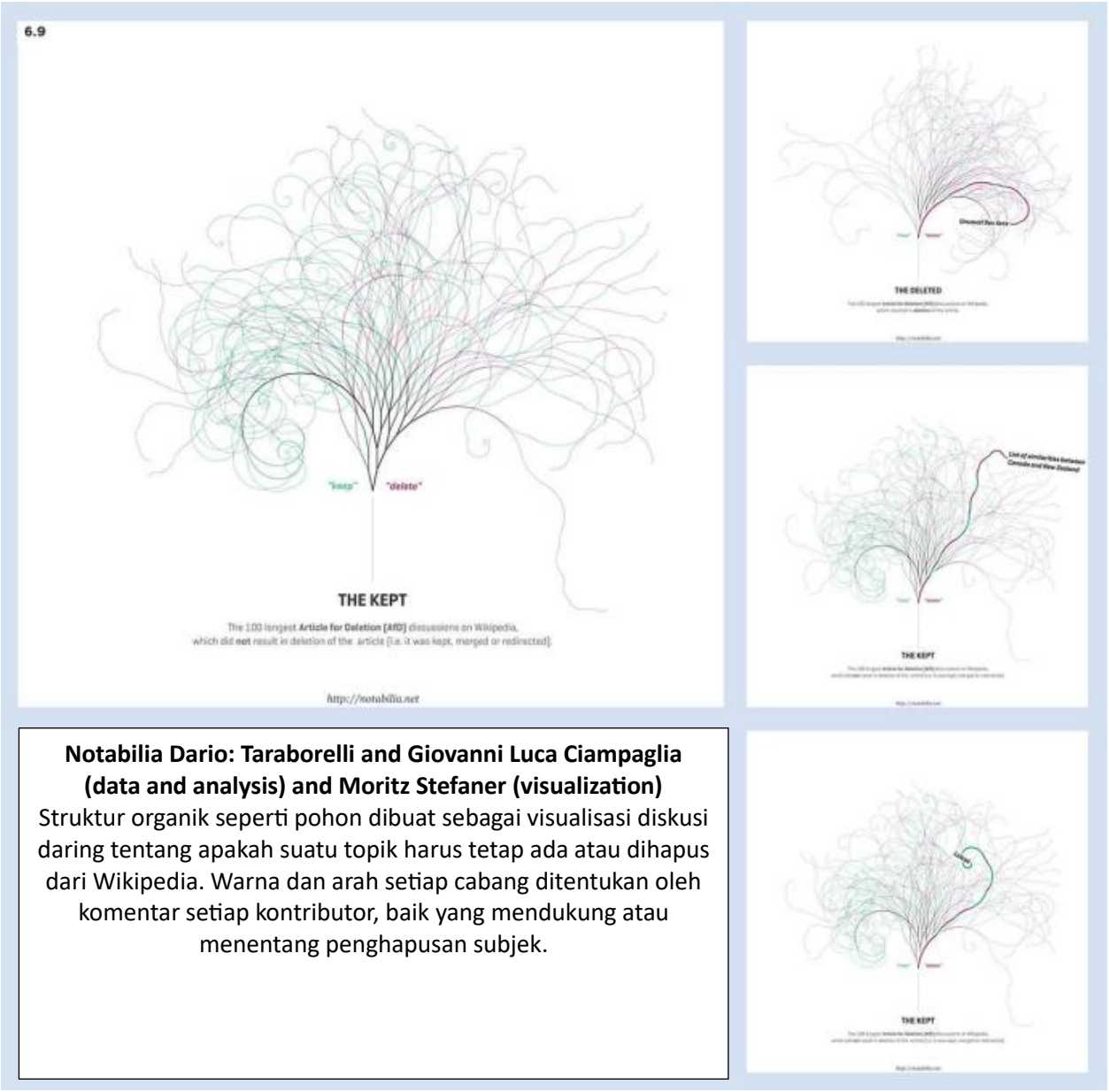
Melihat awan partikel berwarna-warni memberikan gambaran menyeluruh tentang jenis emosi yang dialami. Visualisasi partikel dapat disortir secara visual dan dicari menurut berbagai parameter: suasana hati, jenis kelamin, waktu, atau lokasi penulis. Kelompok partikel berwarna bergeser atau menggumpal, menurut parameter pencarian yang digunakan. Ini berarti bahwa pengguna dapat menemukan visualisasi grafis mereka sendiri tentang emosi manusia.

Menggali lebih dalam data individual menggabungkan teks dan gambar dari posting blog individual, yang menciptakan cuplikan pengalaman individual yang menyentuh dan bermakna. Proyek ini berfungsi sebagai visualisasi data yang kuat dan berkembang yang menggunakan grafik untuk mencerminkan suasana hati komunitas daring dengan cara yang menarik. Pekerjaan ini telah disampaikan sebagai proyek daring dan juga telah diproduksi dalam format buku, yang menambahkan lapisan lebih lanjut pada desain proyek.

Dengan mencari frasa "Saya merasa" atau "Saya sedang merasa" dalam entri blog baru, visualisasi data emosi manusia tercipta. Hasilnya dapat dilihat sebagai awan partikel titik-titik yang dapat difilter dan diurutkan berdasarkan kriteria yang berbeda. Montase foto secara otomatis menggabungkan frasa dari blog dengan gambar untuk menciptakan potret emosi.

Proyek Notabilia mengambil diskusi Artikel untuk Dihapus dan memvisualisasikan data sebagai serangkaian cabang dalam struktur seperti pohon. Arah dan warna setiap segmen cabang ditentukan oleh pandangan masing-masing pengguna terhadap artikel tersebut saat diskusi berkembang. Segmen hijau yang condong ke kiri dibuat saat pengguna merekomendasikan untuk menyimpan, menggabungkan, atau mengalihkan artikel. Segmen merah yang condong ke kanan dibuat setiap kali pengguna merekomendasikan untuk menghapus artikel.

Oleh karena itu, warna dan bentuk keseluruhan setiap baris merupakan representasi visual dari keseluruhan suasana diskusi saat berkembang. Seratus diskusi terpanjang telah disusun menjadi struktur cabang seperti pohon yang menyoroti alur umum diskusi seputar setiap topik, yang pada akhirnya disimpan atau dihapus.



Notabilia Dario: Taraborelli and Giovanni Luca Ciampaglia (data and analysis) and Moritz Stefaner (visualization)
Struktur organik seperti pohon dibuat sebagai visualisasi diskusi daring tentang apakah suatu topik harus tetap ada atau dihapus dari Wikipedia. Warna dan arah setiap cabang ditentukan oleh komentar setiap kontributor, baik yang mendukung atau menentang penghapusan subjek.

Jer Thorp

Jer Thorp adalah seniman perangkat lunak generatif dan pendidik yang praktiknya mengeksplorasi batas-batas antara sains, data, seni, dan budaya. Ia menggunakan kode Pemrosesan untuk menerjemahkan sejumlah besar data menjadi hasil visual, yang menambahkan makna dan narasi, membantu pemirsa memahami dan mengendalikan informasi yang ada di sekitar mereka.

Ia pernah bekerja sebagai Data Artist in Residence di New York Times dan merupakan Dosen tamu dalam program ITP Universitas New York serta salah satu pendiri Office for Creative Research, sebuah kelompok penelitian multidisiplin yang mengeksplorasi cara-cara baru untuk terlibat dengan data.

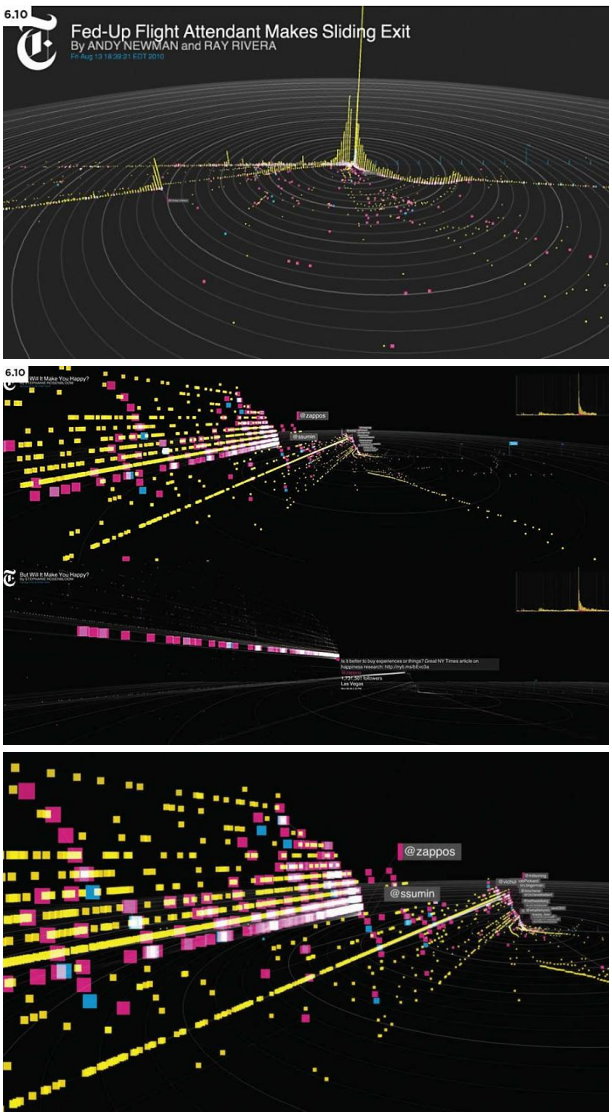
Eksplorasi visualnya terhadap data telah menghasilkan beberapa karya desain cetak dan layar yang mencolok. Contoh-contoh penting termasuk proyek 138 Tahun Sains Populer dan Cascade.

Cascade

Cascade adalah alat interaktif yang dibuat untuk departemen R&D New York Times yang memvisualisasikan bagaimana berita dan informasi dari organisasi dibagikan melalui jejaring sosial. Alat yang dibuat dalam Processing ini memberi pengguna kesempatan untuk menjelajahi bagaimana lebih dari enam ribu konten yang dibuat setiap bulan dibagikan dan "diturunkan" melalui Internet.

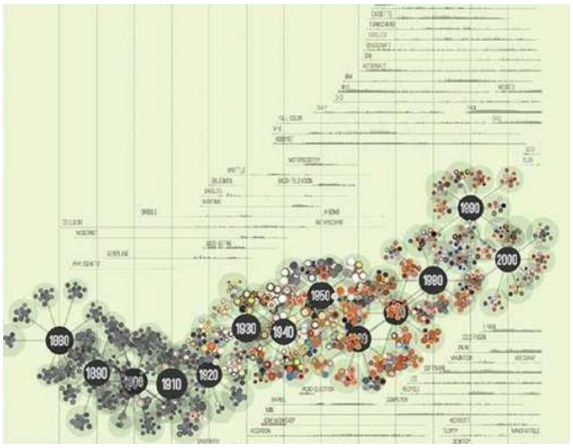
Aplikasi ini adalah alat eksplorasi yang sangat visual yang memungkinkan pemeriksaan yang cermat terhadap pergerakan konten dan dapat dilihat baik dalam mode Story maupun mode Cascade. Mode Story menyoroti serangkaian cerita, yang dapat diminta baik melalui pencarian kata kunci, pencarian bagian, atau menggunakan serangkaian parameter "ketertarikan". Mode Cascade menunjukkan bagaimana suatu peristiwa atau cerita mengalir melalui dunia daring saat dibagikan dan bergerak melalui jejaring sosial selama beberapa menit, jam, atau hari.

Elemen aplikasi yang berbasis hampir waktu nyata memberikan wawasan unik tentang cara setiap cerita "terungkap" dan berkembang melalui Internet. Berbagai perspektif 2D dan 3D dari mode Cascade memungkinkan pemirsa menganalisis pergerakan percakapan dan informasi dalam berbagai cara yang informatif. Melihat data ini dengan cara baru berbasis waktu ini memberikan wawasan berharga tentang cara pesan dibagikan, menangkap pergerakannya saat menyebar melalui media sosial. Selain beroperasi sebagai aplikasi layar kecil, proyek Cascade juga diimplementasikan dalam mode "pameran", dipentaskan di dinding video lima layar.



Cascade Jer Thorp

Cascade adalah alat interaktif dan eksploratif yang menyampaikan bagaimana konten New York Times dibagikan (cascade) di seluruh jaringan sosial, yang menunjukkan cara cerita dan ide menyebar melalui dunia daring.



138 Tahun Sains Populer

Gambar visualisasi yang menjelajahi arsip majalah Popular Science untuk menunjukkan bagaimana istilah teknis dan budaya yang berbeda telah digunakan dan tidak digunakan lagi di majalah tersebut sejak awal penerbitannya. Setiap tahun bertindak sebagai jangkar yang menghubungkan berbagai kata. Thorp menggunakan Processing sebagai alat untuk menemukan frekuensi setiap kata dan mengidentifikasi kata yang paling menarik untuk digunakan dalam grafik akhir.

6.6 MENDAPATKAN DAN MENGGUNAKAN DATA EKSTERNAL

Memuat Data dari Berkas Teks

Ada banyak cara dan banyak tempat asal data, termasuk GPS, informasi suhu, dan statistik lainnya. Apa pun sumbernya, data biasanya diekspor ke dalam dokumen teks atau nilai angka yang dapat dibaca ke dalam program, disortir, dan dimanipulasi.

Cara paling sederhana untuk memuat data eksternal ke dalam program adalah sebagai berkas teks dasar.

Data dalam berkas teks dapat berupa kata-kata (String) atau angka, dan dapat dibaca ke dalam berkas menggunakan fungsi loadStrings sederhana dan referensi ke nama dokumen yang akan dimuat.

```
loadStrings (“nama_dokumen”);
```

(Ingat bahwa “String” adalah istilah pemrograman untuk teks atau informasi yang diatur dalam tanda kutip terbalik sebuah “string” karakter.) Perintah loadStrings() membaca data teks eksternal ke dalam program dan memformatnya sebagai array (daftar) dari baris-baris individual. Sebuah file teks contoh (misalnya “test.txt”) dapat diimpor dengan loadStrings().

```
test.txt:
I wandered lonely as a cloud
That floats on high over vales and hills,
When all at once I saw a crowd,
A host, of golden daffodils;

String lines [] = loadStrings("test.txt");
```

Data dari berkas teks akan diformat secara otomatis dan dibagi menjadi daftar baris-baris individual. Hasilnya adalah array yang disebut “baris”:

```
lines [0] = “I wandered lonely as a cloud”
lines [1] = “That floats on high over vales and hills,”
```

Mendapatkan teks sebagai daftar baris, bukan sebagai satu blok teks besar, berarti bahwa data sekarang berada dalam format baris-baris individual yang berguna dan dapat digunakan, yang dapat lebih mudah disortir dan dikeluarkan dalam format visual.

Data yang Dipisahkan Koma

Selain digunakan untuk mengimpor kata, berkas teks eksternal juga dapat menyimpan dan digunakan untuk mengimpor data numerik yang berguna. Nilai angka yang diekspor ke dalam format berkas teks (misalnya, daftar data GPS) dapat diformat sebagai satu daftar panjang nilai yang dipisahkan oleh koma dan tanpa spasi. Misalnya:

```
Weather_data.txt: 78,82,93,85,45,44,56,78,98
```

Nilai angka dari dokumen yang dipisahkan koma dapat dimuat ke dalam program, lalu dibagi dan diubah menjadi daftar angka (yang menghasilkan sekumpulan nilai yang dapat digunakan). Fungsi loadStrings() mengimpor angka-angka tersebut sebagai satu string tunggal yang panjang.

```
String [] loadedData = loadStrings (“Weather_data.txt”);
```

Ini menempatkan satu baris angka ke dalam elemen pertama dalam array.

```
loadedData [0] = “78,82,93,85,45,44,56,78,98”
```

Item daftar tunggal, yang berisi semua data sebagai satu string tunggal, dapat dibagi menjadi array nilai angka individual. Fungsi split adalah fungsi yang digunakan untuk membagi teks. Dalam hal ini, fungsi ini digunakan untuk membagi string angka menjadi daftar baru nilai angka individual, dengan menggunakan setiap koma sebagai titik potong untuk setiap angka (“pemisah”). Berikut ini membagi array menjadi array item individual, menggunakan koma sebagai titik untuk membuat pembagian:

```
// split the data into individual values
String dataAsStrings = split (loadedData [0], ‘,’);
```

Hasil dari proses ini adalah array angka individual, yang ditulis sebagai String. Ada perbedaan, dalam lingkungan pemrograman, antara String (teks) dan integer (angka) meskipun bagi kita, keduanya mungkin terlihat sama. Misalnya, String (“42”) tidak sama dengan integer (42). Oleh karena itu, array data harus diubah dari String (“78,” “82”) menjadi integer (78, 82) sebelum dapat dilihat dan digunakan sebagai angka.

Fungsi int digunakan untuk mengubah data menjadi nilai angka (“int”). Seluruh daftar data dalam “dataAsStrings” diubah menjadi array angka baru:

```
// convert the entire array from Strings to number
int [] dataAsNumbers = int (dataAsStrings);
```

Hasilnya adalah serangkaian nilai angka individual yang dapat digunakan untuk membuat serangkaian bentuk atau grafik.

```
dataAsNumbers [0] = 78;
dataAsNumbers [1] = 82;
dataAsNumbers [2] = 93;
```

Perulangan “for” biasanya digunakan untuk mengurutkan dan menelusuri setiap nilai dalam array.

Memuat Data Tabel

Selain dapat memperoleh data dari berkas teks sederhana sebagai rangkaian nilai, data juga dapat diekstrak ke dalam program dari tabel data. Data yang diformat ke dalam tabel kolom dan baris merupakan cara umum untuk memformat informasi. Spreadsheet dan basis data, daring atau luring, mengatur informasi ke dalam kelompok kolom dan baris yang terorganisasi. Contoh sederhana:

```
data.tsv:
Name        Age        Gender
Andrew      42         M
Steve       36         M
Charlie     24         F
```

Data yang diformat dengan cara ini merupakan cara yang sangat berguna dan umum untuk menyortir dan menyampaikan konten. Informasi yang disusun dengan cara ini dapat berasal dari berbagai sumber daring atau luring (misalnya, lembar kerja Excel atau Google, atau nilai yang dihasilkan dari GPS atau jenis perangkat pemetaan lainnya). Kolom mengelompokkan informasi di bawah tajuk topik tertentu (Nama, Usia, Jenis Kelamin). Baris data ditambahkan untuk contoh individual (Andrew, Steve, Charlie) untuk setiap kelompok.

Informasi yang disusun dalam tabel dapat diekspor ke dalam format CSV *(Comma*

Separated Values) atau TSV *(Tab Separated Values)*, yang masing-masing menyediakan data mentah dalam format yang paling mudah diambil dan disortir oleh program ke dalam baris dan kolom. Membaca dan mengimpor nilai dari tabel memungkinkan data diekstraksi dan kemudian diformat ulang.

Tabel dari sumber CSV atau TSV dapat dengan mudah diimpor dengan menggunakan fungsi loadTable, yang membuat kisi data yang dapat digunakan dari dokumen yang masuk.

```
Table table = loadTable("data.tsv");
```

Fungsi ini bekerja dengan cara yang sama dengan perintah loadStrings(), tetapi alih-alih membuat daftar informasi String (teks), fungsi ini membuat objek Tabel khusus yang mengurutkan data ke dalam baris dan kolom yang mudah dibaca. Ini menyediakan cara yang jauh lebih mudah untuk bekerja dengan dan mengurutkan data daripada jika diimpor sebagai String, yang kemudian harus dibagi dan diurutkan.

Setelah tabel dimuat, data dapat diekstraksi dengan melihat melalui baris informasi individual. Mengakses data berarti mencari informasi tertentu dalam baris atau kolom. Data dalam objek tabel dibagi menjadi "kotak" baris dan kolom. Setiap baris dan kolom diberi nomor, sehingga menciptakan kotak sortir x, y yang sederhana.

TIPS: Tabel contoh di atas, data.tsv, memiliki 4 baris dan 3 kolom. Mengingat komputer mulai menghitung dari nol, bukan satu, item di baris 1, kolom 0, misalnya, adalah “Andrew.” Baris 1 adalah baris kedua dalam daftar. Kolom 0 adalah kolom pertama di sebelah kiri.

Data dari salah satu "sel" individual dalam tabel dapat diakses dengan mudah oleh fungsi sederhana, yang "mendapatkan" tipe data tertentu, menggunakan nomor baris dan kolom sebagai referensi. Data yang akan diperoleh bisa berupa teks, getString, atau angka, getInt atau getFloat. Ingat, "int" adalah bilangan bulat (integer); "float" mencakup angka desimal (floating point). Misalnya, kode berikut mendapatkan data String (nama) dari item di baris 2, kolom 0:

```
String nama = tabel.getString (2, 0);
println(name); // outputs "Steve"
```

Demikian pula, nilai usia (int) dapat ditemukan dengan melihat baris 2, kolom 1:

```
// get data from row 2 column 1
int age = table.getInt(2, 1);
println(age); // outputs "36"
```

Cara lain untuk mendapatkan informasi dari tabel adalah dengan menemukan semua data dari baris atau kolom tertentu menggunakan getRow atau getColumn.

Misalnya, perintah berikut mengekstrak seluruh baris data dari baris 1 file contoh:

```
// gets data first row 1
TableRow row1 = table.getRow(1);
```

Setelah ini ditemukan, item dalam baris (nama, usia, jenis kelamin) dapat diakses secara berurutan:

```
// gets the 1st item in row1 (a String)
String name = row1.getString(0);
// gets the 2nd item in the row (a number)
int age = row1.getInt(1);
// gets the 3rd item in the row (a String)
String gender = row1.getString(2);
```

Fungsi Pemrosesan sederhana ini merupakan alat yang berguna dan efisien untuk mengimpor, menemukan, mengekstrak, dan memvisualisasikan data dari basis data atau lembar kerja yang besar. Fungsi ini dapat diterapkan pada berbagai macam kumpulan data dan digunakan untuk membuat berbagai tujuan dan hasil visual.

Peta Data Sederhana

Dengan menggunakan fungsi ini, kumpulan data yang besar dapat diakses dan didaur ulang dengan cepat. Misalnya, kumpulan data yang lebih besar yang menunjukkan lokasi setiap negara bagian di Amerika disimpan sebagai file “locs.tsv” (tersedia melalui contoh daring). Dalam file tersebut, singkatan setiap negara bagian disimpan di kolom pertama, dan koordinat grid yang sesuai dari negara bagian yang berlokasi di dua kolom berikutnya.

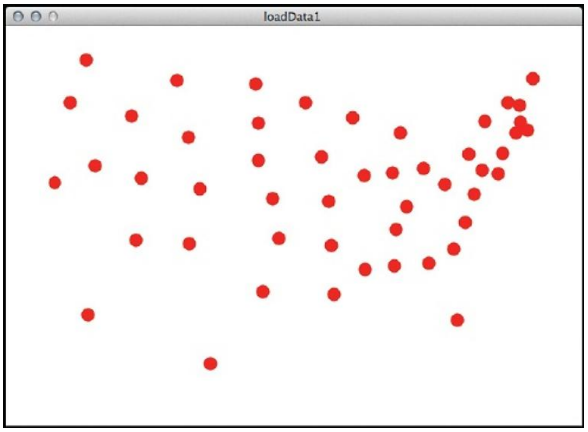
```
locs.tsv:
AL    439    270
AK     94    325
. . .
WI    392    103
WY    207    125
```

Nilai angka dapat digunakan sebagai koordinat x, y yang berguna untuk menggambar bentuk di layar. Perulangan "for" dapat melakukan pekerjaan berat dalam menelusuri daftar data dan digunakan untuk menelusuri semua baris, satu per satu, dan untuk setiap baris, mendapatkan baris data, mengekstrak dua koordinat angka dari kolom, dan menggunakannya untuk memplot bentuk di layar:

```
for (int i=0;
i<table.getRowCount(); i++) {
// get a row of data
TableRow row = table.getRow(i);
// get the first number value (column 1)
int x = row.getInt(1);
// get the second number value (column 2)
int y = row.getInt(2);
ellipse (x, y, 10, 10); // draw a shape
```

```
}
```

Hasilnya adalah “peta” berbasis data yang sederhana namun efektif.



Contoh sederhana peta lokasi negara bagian yang diplot dari data dari berkas data eksternal.

Data XML

XML adalah format yang berguna untuk membaca dan mendapatkan data dari web. XML lebih berguna dan lebih mudah dibaca untuk program komputer daripada HTML, yang melibatkan lebih banyak “penambangan” untuk mengekstrak data.

Struktur XML

Data XML disusun menjadi struktur elemen dan tag yang terdefinisi dengan baik yang dapat diletakkan (disarangkan) di dalam satu sama lain untuk membuat pohon (struktur seperti cabang).

```
clients.xml
<?xml version="1.0"?>
<client_list>
  <person>
    <name>andrew</name>
    <age>42</age>
    <address>
      <street>main street</street>
      <town>durham</town>
    </address>
  </person>
  <person>
    <name>bill</name>
    <age>34</age>
<address>
  <street>high road</street>
  <town>manchester</town>
</address>
</person>
```

```
</client_list>
```

Tingkat teratas dokumen disebut "root." Elemen di dalam root disebut elemen "anak". Setiap elemen dapat memiliki elemen sub ("anak") sendiri yaitu, elemen lain di dalamnya (seperti direktori file) atau bahkan elemen "induk", yang berada satu tingkat di atasnya.

Dalam contoh di atas, **<client_list>** adalah elemen root dokumen, tempat semua hal berada. Ia memiliki dua elemen "anak" **<person>** yang mewakili orang yang berbeda dalam tag **<person>.** Setiap **<person>** memiliki kumpulan elemen anaknya sendiri, yang digunakan untuk menyimpan informasi tentang nama, usia, dan alamat. Elemen induk dari **<age>** adalah elemen **<person>** tempat elemen tersebut berada. Elemen-elemen **(<street>** dan **<town>)** adalah elemen anak dari **<address>.**

Cara terorganisasi (root parent, child) dalam mendefinisikan dan mengatur data ini menjadikannya cara sederhana untuk membuat dokumen berlapis-lapis yang dapat dipelajari elemen pemrograman (milik saya) untuk mengekstrak data.

Memuat dan Mengakses XML

Muat data XML menggunakan fungsi loadXML. Ini dapat digunakan untuk memuat data dari dokumen XML yang diunduh (offline), atau langsung ke sumber XML langsung. Misalnya:

```
XML xml = loadXML (“clients.xml”);
```

Setelah dokumen XML dimuat, ada banyak fungsi khusus XML yang dapat mencari dan mengambil data dari dalam file XML. Fungsi getChild adalah cara yang berguna untuk mendapatkan data dan menggali (dan menavigasi) ke dalam struktur dokumen.

Fungsi getChild mengambil item dalam dokumen dengan tag yang sesuai dengan nama yang digunakan dalam fungsi tersebut. Misalnya, perintah berikut akan mengambil item “orang” pertama dalam daftar:

```
XML firstPerson = xml.getChild (“person”);
```

Alternatif yang berguna adalah mengambil semua item “orang” sebagai satu daftar, yang kemudian dapat diulang dan diekstrak untuk mendapatkan informasi lebih lanjut. Perintah berikut akan mengembalikan daftar semua item “orang” dalam dokumen XML.

```
XML [] listOfPeople = xml.getChildren("person");
```

Setelah mengambil daftar lengkap konten utama, setiap elemen (orang) dalam daftar dapat ditemukan dan dilihat untuk mendapatkan detail nama, usia, dan alamat setiap orang. Ini dilakukan dengan mengambil item dalam daftar dan menggunakan fungsi getChild untuk mengambil detail ini. Contoh berikut menemukan elemen XML “nama” dan “usia” orang pertama dalam daftar.

```
// gets XML element called name and age from the first person in
the list // gets <name> Andrew </name>
XML name = listOfPeople [0]. getChild("name");
XML age = listOfPeople [0]. getChild("age");
```

Proses ini mendapatkan item XML, termasuk tag (misalnya, `<name> Andrew </name>`). Fungsi getContent() akan mengekstrak konten data dalam tag, format yang jauh lebih berguna:

```
String name_data = name.getContent();
String age_data = age.getContent();
println(name_data);
println(age_data);
```

Meskipun ini merupakan contoh yang sangat sederhana, kemampuan untuk mengekstrak dan menggunakan data XML dari sumber offline maupun online merupakan cara yang berharga untuk mendapatkan data langsung yang dapat digunakan untuk menginformasikan grafik dan visual.

KESIMPULAN

Masing-masing bab dalam buku ini telah menunjukkan berbagai cara di mana kode pemrograman dapat membuka pendekatan kreatif untuk membantu desainer berpikir dengan cara baru dan menciptakan jenis karya digital baru. Diharapkan buku ini telah memberikan apresiasi yang lebih besar terhadap peran yang dapat dimainkan oleh kode sebagai bagian dari praktik desain kreatif dan mendorong para pembaca untuk melihat bahwa desain berbasis komputer mencakup berbagai pendekatan dan kemungkinan kreatif.

Melihat kode sebagai bagian utama dari praktik desain kreatif adalah penting. Ini adalah cara untuk mengembangkan pendekatan baru dalam menciptakan karya untuk lingkungan berbasis data, untuk memikirkan kembali ketergantungan default pada perangkat lunak siap pakai, dan untuk menginspirasi perspektif yang berbeda tentang proses digital kreatif. Memiliki pemahaman dan apresiasi yang lebih besar terhadap kemungkinan untuk menciptakan desain berbasis komputasi memberikan batu loncatan penting untuk menghasilkan konsep kreatif asli yang memanfaatkan kemampuan input/output data dari lingkungan digital.

Ini juga menumbuhkan pendekatan terhadap desain digital yang akan terus bernilai bahkan saat teknologi berubah. Meskipun bukan media yang nyata dan fisik seperti cat atau tinta, mengotori tangan Anda dengan kode adalah proses eksperimental dan menyenangkan yang sangat penting bagi imajinasi dan kreativitas. Jenis pendekatan ini memerlukan perspektif yang lebih luas terhadap kode yang tidak dibatasi dalam batasannya atau ambisi kreatifnya.

Visi kode yang sempit dapat membatasi, membatasi pemrograman menjadi alat yang murni fungsional, seperti mesin, yang hanya berguna untuk tugas komputasi yang berulang dan melelahkan. Namun, ketika diserahkan ke tangan seniman, programmer, dan desainer

yang kreatif, kode membuka ranah baru kemungkinan imajinatif ranah yang dapat menciptakan hasil visual baru yang unik dari grafik, gambar, dan interaksi.

Proses eksperimental dan menyenangkan dalam mengeksplorasi kode sebagai bagian dari praktik desain digital dapat disamakan dengan jenis proses, lingkungan, dan media kreatif lainnya. Sama seperti seorang seniman mengeksplorasi proses dan bahan tradisional cat, tinta, atau kapur untuk menemukan dan mengembangkan gaya visualnya sendiri, demikian pula seniman dan desainer yang karyanya menggunakan "bahan" pemrograman terlibat dalam proses eksplorasi serupa untuk menemukan pendekatan mereka sendiri terhadap, dan cara bekerja dengan, kode.

Seperti menemukan gaya menggambar atau melukis pribadi, berhasil menulis kode untuk membuat karya yang unik adalah masalah mengembangkan pendekatan individual terhadap jenis dan penggunaan fungsi dan instruksi pemrograman. Meskipun kode dapat dengan mudah disalin dan ditempel untuk membuat karya yang tampak generik, "mata" kreatif dari desainer atau programmer individu perlu dimanfaatkan untuk menciptakan karya yang menonjol di atas yang lain.

Terlibat dengan proses kreatif untuk mengeksplorasi elemen tertentu dari data komputasional mengangkat karya melampaui generik menjadi sesuatu yang individual dan unik. Setiap karya yang ditampilkan dalam buku ini merupakan ekspresi dari visi kreatif seniman atau desainer individu di balik karya tersebut; ini adalah produk dari eksperimen dan investigasi kreatif mereka sendiri, yang digunakan untuk mengeksplorasi aspek tertentu dari pengodean dan untuk membuat karya komputasi yang memiliki gaya tersendiri.

Baik mengeksplorasi kumpulan data besar, membuat gambar komputasi eksperimental, atau menghasilkan tipografi yang dihasilkan kode, masing-masing desainer dan seniman yang ditampilkan di sini telah mengembangkan, melalui praktik pribadi mereka, pendekatan dan cara mereka sendiri dalam menggunakan kode untuk mengekspresikan visi dan identitas kreatif mereka yang unik.

Ketika mengeksplorasi cara untuk mengintegrasikan proses kode sebagai bagian dari praktik kreatif, oleh karena itu penting untuk mengembangkan pendekatan pribadi untuk mengeksplorasi ide dan visi Anda sendiri. Jangkauan dan jenis kode dan data sangat luas dan bervariasi sehingga, seperti setiap seniman atau desainer lainnya, bidang minat atau spesialisasi tertentu perlu dikembangkan.

Apakah minat utamanya adalah pada grafik, gambar, atau tipografi, setiap desainer perlu mengembangkan titik fokus, area minat utama, di mana ide dapat tumbuh dan berkembang. Anda mungkin ingin mengeksplorasi kapasitas kode untuk menghasilkan gambar generatif skala besar, bentuk huruf komputasi, atau visualisasi media sosial. Titik fokus, apa pun itu, membentuk hub penting untuk eksplorasi dan permainan kreatif.

Bermain dengan kode memungkinkan ide muncul. Karya komputasi kreatif tumbuh dari titik awal yang sangat sederhana.

Bahkan bagian pemrograman yang paling rumit pun dimulai dengan beberapa baris yang menetapkan parameter umum, konsep keseluruhan, dari karya tersebut. Dengan menambahkan dan mengubah elemen satu per satu secara bertahap, ide berkembang dan

tumbuh ke berbagai arah. Penjelasan umum atau tutorial sederhana dapat menjadi titik awal, "benih," untuk berbagai macam ide dan eksperimen baru. Mungkin beberapa, atau mungkin hanya satu, dari ide atau konsep yang ditampilkan dalam buku "benih" ini akan menjadi titik awal untuk seluruh taman yang penuh dengan eksplorasi kreatif unik Anda sendiri ke dalam kode.

DAFTAR PUSTAKA

Alvarado, M. (2019). The artistry of coding: A new paradigm in graphic design. Design Philosophy Papers, 1(1), 15–28.

Ambrose, G., & Harris, P. (2010). *Design Thinking*. AVA Publishing.

Arnheim, R. (1974). *Art and Visual Perception: A Psychology of the Creative Eye*. University of California Press.

Blais, J. (2021). Visualizing ideas: Computational approaches in graphic design. Boston, MA: MIT Press.

Bowers, A. (2019). Creative coding: A practical guide to visual programming. New York, NY: Routledge.

Buffington, C., & White, L. (2020). Programming languages for visual creatives: An introduction. Computers & Graphics, 90, 56–67.

Buxton, B. (2007). *Sketching User Experiences: Getting the Design Right and the Right Design*. Morgan Kaufmann.

Cairo, A. (2013). *The Functional Art: An Introduction to Information Graphics and Visualization*. New Riders.

Chamberlain, E. (2021). Imagining visuals: Creative coding and augmented reality in design. Journal of Design Innovation, 8(3), 178–191.

Cooper, A., Reimann, R., & Cronin, D. (2007). *About Face 3: The Essentials of Interaction Design*. Wiley.

Cross, N. (2008). *Engineering Design Methods: Strategies for Product Design* (4th ed.). Wiley.

Curcio, A., & Kim, Y. (2022). Creative coding as a pedagogical tool in graphic design curriculum. Journal of Design and Technology Education, 27(1), 56–70.

Dabner, D., Stewart, S., & Vickress, A. (2017). *Graphic Design School*. Wiley.

DeLuca, M., & Frank, J. (2019). Visual thinking through code: Enhancing graphic design workflows. Graphic Design Journal, 22(2), 121–135.

Dumas, J., & Redmond, J. (2020). The aesthetics of code: Visual programming for creative expression. Art Journal, 79(2), 24–39.

Eick, S. (2019). Integrating creative coding into the digital design process. Design and Culture, 11(3), 341–357.

Elam, K. (2001). *Geometry of Design: Studies in Proportion and Composition*. Princeton Architectural Press.

Engel, S. (2020). Open source creative coding platforms for graphic design education. Education and Information Technologies, 25, 529–544.

Few, S. (2012). *Show Me the Numbers*. Analytics Press.

Fisher, R., & Park, G. (2021). The influence of creative coding on contemporary visual branding. Journal of Marketing Communications, 27(5), 411–427.

Frisch, M. (2020). The intersection of art and technology: Creative coding in graphic design. Journal of Visual Communication, 19(3), 245–260.

Fuchs, L., & Schmitt, R. (2021). Creative coding workflows in graphic design projects. International Journal of Art & Design Education, 40(3), 610–625.

Gao, M. (2020). Interactive visual design using creative coding. Journal of Multimedia Tools and Applications, 79(33), 24411–24426.

Golan, O., & Shalev, A. (2021). Exploring the role of creative coding in contemporary graphic design education. Design Studies, 72, 1–20.

Green, D. (2019). Reimagining graphic design with interactive coding. New Media & Society, 21(11–12), 2590–2605.

Harvey, L. (2019). Programming as a medium of creative expression in graphic design. Design Issues, 35(4), 92–104.

Heer, J., & Bostock, M. (2010). *Declarative Language Design for Interactive Visualization*. IEEE Trans. Visualization & Comp. Graphics.

Heller, S. (2020). Graphic design: A new history. New York, NY: Thames & Hudson.

Hidayatullah, P. (2015). *Visual Basic.net Membuat Aplikasi Database dan Program Kreatif*. INFORMATIKA.

Huang, Y., & Tsai, M. (2020). Creative coding for interactive visual narratives in graphic design. Interactive Technology and Smart Education, 17(3), 211–224.

Ikeda, R. (2022). New frontiers in creative coding: Generative design methods. Information Design Journal, 28(1), 4–18.

Jensen, L., & Kaur, S. (2019). Visual arts and programming: Teaching creative coding with p5.js. Computers in Art and Design, 5(4), 299–312.

Johnson, M. (2014). *Now Try Something Weirder*. Laurence King Publishing.

Johnson, P., & Liu, S. (2020). From concept to code: Creative coding in visual story-telling. Computers in Human Behavior, 112, 106480.

Kahn, J. (2019). Coding as a creative medium: The impact of programming on visual design. International Journal of Design, 13(2), 45–58.

Keller, A. (2019). Creative coding for visual communication: Exploring the boundaries. Visual Communication, 18(2), 133–152.

Klein, R. (2020). Algorithmic aesthetics and the role of code in graphic design. Visual Studies, 35(3), 291–304.

Kress, G., & van Leeuwen, T. (2006). *Reading Images: The Grammar of Visual Design*. Routledge.

Lawson, B. (2006). *How Designers Think: The Design Process Demystified* (4th ed.). Architectural Press.

Lee, Y. (2021). The role of creative coding in evolving graphic design practices. Design and Technology Education, 26(4), 35–49.

Lopez, A. (2021). Creative coding for immersive graphic experiences. Journal of Digital Media & Policy, 12(2), 154–167.

Matthews, J. (2020). The elegance of code: Visual programming languages in design. Journal of Visual Literacy, 39(2), 75–89.

McCormick, J. (2014). *Generative Design: Visualize, Program, and Create with Processing*. Princeton Architectural Press.

McCullough, M. (2020). Digital ground: Architecture, pervasive computing, and environmental knowing. Cambridge, MA: MIT Press.

Meggs, P. B., & Purvis, A. W. (2016). *Meggs' History of Graphic Design* (6th ed.). Wiley.

Morris, B., & Chen, X. (2019). Designing with code: Bridging creativity and technical skill in graphic design. International Journal of Technology and Design Education, 29(1), 89–103.

Munroe, R. (2014). *Thing Explainer: Complicated Stuff in Simple Words*. Houghton Mifflin Harcourt.

Munzner, T. (2014). *Visualization Analysis and Design*. CRC Press.

Nascimento, D., & Raby, F. (2019). Creative coding techniques for experimental typography. Typography Papers, 9, 50–67.

Nielsen, J. (1994). *Usability Engineering*. Morgan Kaufmann.

O'Connor, H. (2022). Coding the visual: Creative coding in motion graphics design. Animation Studies, 17, 88–102.

Papert, S. (1980). *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books.

Park, J., & Kim, H. (2021). Generative art and creative coding: Impact on graphic design aesthetics. Computer Graphics Forum, 40(2), 123–135.

Primasanti, H. (2020). *Belajar Desain Grafis Untuk Pemula*. Gramedia Pustaka Utama.

Pritchard, J. (2021). The future of graphic design: Integrating creative coding into design practice. Design Issues, 37(1), 34–45.

Quinn, S. (2020). Teaching creative coding to graphic design students: Challenges and opportunities. International Journal of Art & Design Education, 39(3), 567–580.

Resnick, M., & Rosenbaum, E. (2020). Designing for a creative coding experience: The role of visual programming in education. Computers & Education, 148, 103798.

Rodriguez, P. (2019). Creative coding and data visualization: A new approach to graphic design. Information Design Journal, 25(3), 237–251.

Shneiderman, B., Plaisant, C., Cohen, M., Jacobs, S., & Elmqvist, N. (2016). *Designing the User Interface: Strategies for Effective Human-Computer Interaction* (6th ed.). Pearson.

Smith, D. (2020). Processing: Creative coding and generative design. Berkeley, CA: Apress.

Sribu Blog. (2021). *8 Daftar Buku Belajar Desain Grafis Untuk Pemula*.

Sweeney, J. (2019). Creative coding for visual artists: A guide to programming with p5.js. San Francisco, CA: No Starch Press.

Tang, L., & Zhao, Y. (2021). Creative coding frameworks for visual graphic design: A comparative study. Multimedia Tools and Applications, 80, 12345–12361.

Tufte, E. R. (2001). *The Visual Display of Quantitative Information*. Graphics Press.

Ullman, E. (2019). Programming as a creative design tool in visual arts. The Journal of Arts Management, 27(2), 147–160.

Vazquez, J. (2022). Impact of creative coding on branding and visual identity design. International Journal of Graphic Design, 6(1), 16–29.

Wang, S., & Chen, T. (2020). Machine learning and creative coding in generative graphic design. Journal of Creative Technologies, 12(4), 301–315.

White, A. W. (2011). *The Elements of Graphic Design*. Allworth Press.

Wong, B. (2019). Creative coding and its influence on new media art: An overview. Leonardo, 52(4), 399–406.

Xu, F. (2019). Digital art and creative coding: New frontiers in graphic design. Art & Design Review, 7(2), 22–37.

Young, A., & Lambert, M. (2021). Creative coding in the workflow of contemporary graphic designers. Design Management Journal, 16(1), 45–60.

Zahn, K. (2020). Algorithmic design and coding for graphic designers. Visual Communication Quarterly, 27(2), 89–97.

Ziegler, K. (2020). Coding the unseen: Visualizing algorithms for creative design. Visual Communication Quarterly, 27(4), 289–304.

Coding Kreatif dalam Desain Grafis Visual

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

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 di Fakultas Seni Rupa dan Desain, Institut Teknologi Bandung (ITB), dan kemudian melanjutkan studi pada program studi Teknologi Multimedia di 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 Akademik Lektor Kepala 700. 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 978-634-7227-10-2 (PDF)
9 786347 227102