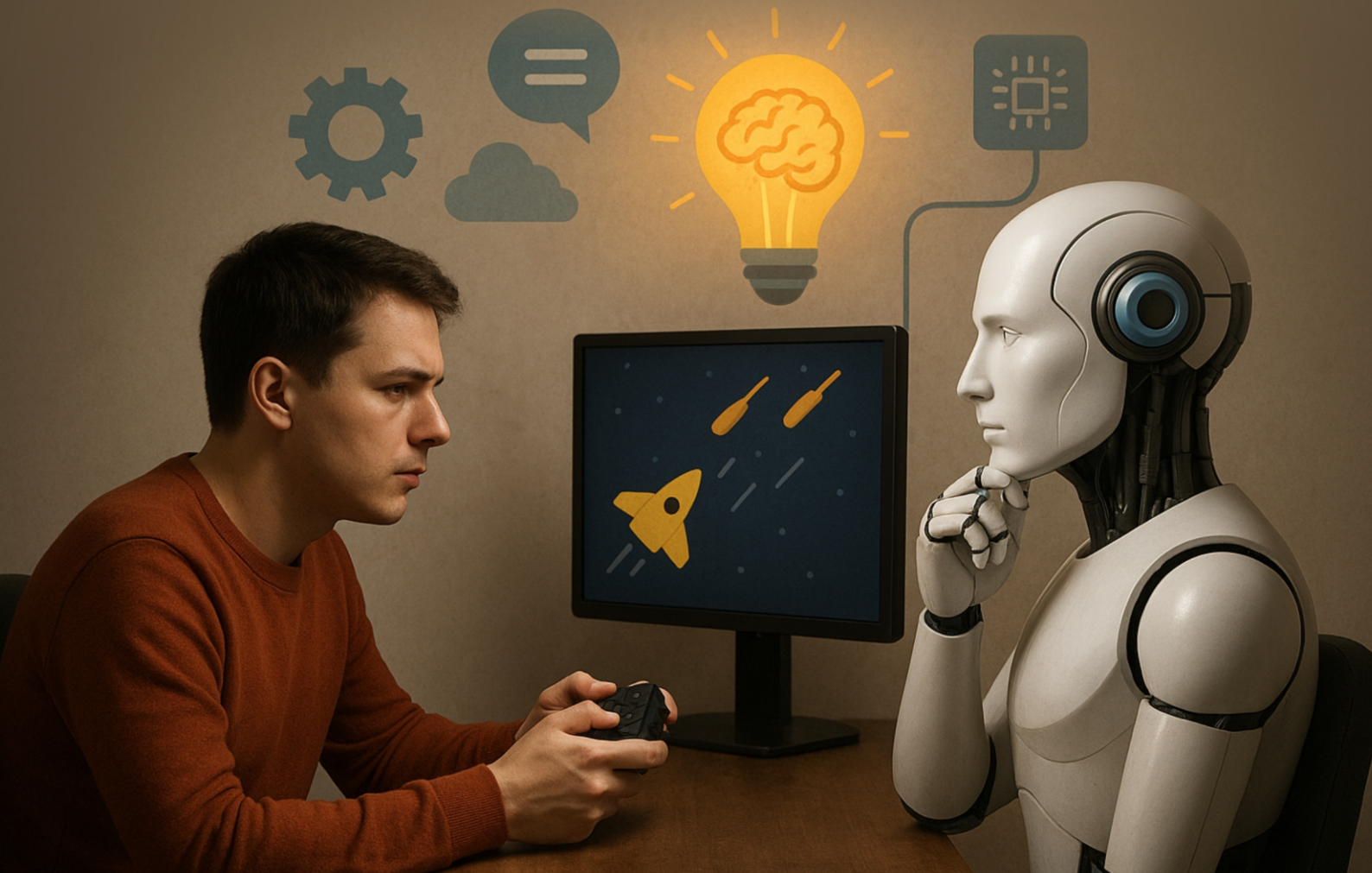
YAYASAN PRIMA AGUS TEKNIK

Pengaruh AI dalam Permainan Game dan berfikir cerdas

Dr. Ir. Agus Wibowo, M.Kom, M.Si, MM



Dr. Ir. Agus Wibowo, M.Kom, M.Si, MM

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

Pengaruh AI dalam Permainan Game dan berfikir cerdas

Penulis :
Dr. Ir. Agus Wibowo, M.Kom, M.Si, MM

ISBN : 978-634-7227-38-6

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

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

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. 08122925000
Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

Distributor Tunggal :
Universitas STEKOM
Jl. Majapahit no 605 Semarang
Telp. 08122925000
Fax. 024-6710144
Email : info@stekom.ac.id

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Tuhan Yang Maha Esa karena berkat rahmat dan karunia-Nya, buku berjudul ***Pengaruh AI dalam Permainan Game dan Berfikir Cerdas*** ini dapat terselesaikan dengan baik. Buku ini hadir sebagai upaya untuk mengulas dan memperkenalkan peran kecerdasan buatan (AI) yang semakin pesat berkembang dalam dunia permainan video sekaligus menyoroti bagaimana AI memengaruhi kemampuan berpikir cerdas baik bagi mesin maupun pemainnya.

Pada bab pertama, pembaca akan diajak menelusuri sejarah awal mula kecerdasan buatan dalam permainan, dimulai dari komputer yang berhasil bermain catur hingga perkembangan AI yang lebih maju dalam permainan Go. Bab ini membuka wawasan tentang fondasi teknologi AI di balik ingatan dan strategi permainan klasik tersebut. Bab kedua membahas bahwa kecerdasan bukanlah satu-satunya syarat untuk bermain game, melainkan proses pembelajaran yang terus berlangsung selama bermain. AI juga diterapkan pada karakter non-pemain (NPC) dalam game, menciptakan interaksi yang semakin realistis dan dinamis.

Selanjutnya, bab ketiga mengulas pengertian kecerdasan buatan, dengan membandingkan kecerdasan manusia sebagai makhluk hidup dengan kemampuan AI yang mampu meniru dan bahkan melampaui beberapa aspek perilaku manusia. Bab ini juga menyoroti bagaimana AI telah meningkatkan industri game secara signifikan. Bab keempat memperdalam pembahasan tentang keberadaan AI dalam gim video itu sendiri, menggambarkan bagaimana AI beroperasi dalam hitungan detik untuk mengatur gerakan dan keputusan lawan dalam permainan, sehingga memberikan pengalaman bermain yang menantang dan autentik.

Pada bab kelima, fokus berpindah pada proses tumbuhnya pemikiran dan pembelajaran dalam game, dimulai dari gagasan sederhana, simulasi otak kecil, hingga penerapan mekanisme uji coba dan kesalahan yang mempercepat adaptasi dan kelangsungan permainan. Bab keenam menyoroti bagaimana game belajar dari pemainnya. Sistem game dapat mengidentifikasi pola dan karakter pemain, sehingga mampu menyesuaikan tingkat kesulitan dan interaksi demi menciptakan pengalaman yang lebih personal dan menarik.

Bab ketujuh mengupas konsep otomatisasi kreativitas melalui AI, yang berfungsi seperti “dewa angka acak” dalam menghasilkan variasi dan keunikan dalam game. AI juga memungkinkan kolaborasi kreatif antara manusia dan mesin, membuka peluang baru dalam desain dan pengembangan game. Bab kedelapan membahas tentang desain game yang berorientasi pada AI, mengulas sejarah dan pola desain yang memanfaatkan teknologi kecerdasan buatan sebagai inti penggerak inovasi dalam gameplay dan interaktivitas.

Di bab terakhir, pembahasan berfokus pada kecerdasan umum dan permainan secara luas, termasuk bagaimana AI diterapkan pada komputer sebagai pemain dan bagaimana AI membentuk pengalaman bermain game secara umum di berbagai platform dan genre.

Penulis berharap buku ini dapat menjadi referensi bermakna bagi para akademisi, pengembang game, penggiat teknologi, dan pembaca umum yang tertarik memahami

bagaimana AI tidak hanya mengubah permainan, tetapi juga membuka wawasan baru tentang kecerdasan dan pembelajaran dalam konteks digital. Kritik dan saran yang membangun sangat penulis harapkan demi penyempurnaan buku ini di masa mendatang. Akhir kata, selamat membaca dan semoga buku ini memberikan manfaat serta inspirasi dalam melihat lebih jauh hubungan antara teknologi AI dan dunia permainan.

Semangat Dan Selamat Membaca Dan Belajar...!!!

Semarang, September 2025
Penulis

Dr. Ir. Agus Wibowo, M.Kom, M.Si, MM

Teknologi tanpa batas, kecerdasan tanpa henti, menciptakan dunia permainan yang menginspirasi dan membentuk pikiran cerdas masa depan. Buku ajar ini dipersembahkan untuk para mahasiswa, pengembang, dan pecinta teknologi yang selalu haus akan pengetahuan dan inovasi. Semoga karya ini mampu membuka wawasan dan menginspirasi perjalanan kalian dalam memahami dan mengembangkan kecerdasan buatan dalam dunia permainan, serta menumbuhkan cara berpikir yang kritis dan kreatif di era digital. Menggabungkan kekuatan kecerdasan buatan dan kreativitas manusia untuk membuka cakrawala baru dalam permainan dan cara berpikir.

Dr. Ir. Agus Wibowo, M.Kom, M.Si, MM

DAFTAR ISI

BAB 1
PADA AWAL MULA AI, TERDAPAT PERMAINAN

Komputer digital pertama yang berfungsi dikembangkan pada akhir 1940an atau awal 1950an, tergantung definisi komputer Anda, dan langsung digunakan untuk bermain permainan. Bahkan, setidaknya dalam satu contoh, sebuah program untuk bermain permainan ditulis dan dijalankan secara manual, menggunakan pena dan kertas karena komputer yang cukup canggih untuk menjalankan program tersebut belum dibangun. Penemu (dan pemain) yang bersemangat itu tak lain adalah Alan Turing, salah satu pendiri ilmu komputer dan kecerdasan buatan. Saat itu tahun 1948. Permainannya adalah Catur (gambar 1.1). Turing bertindak sebagai komputer (menghitung semua langkah secara manual) ketika menggunakan algoritma ini untuk bermain melawan seorang teman baik.

Gambar 1.1

Catur telah ada selama ribuan tahun sebelum menjadi pusat penelitian kecerdasan buatan.

Mengapa Catur? Nah, ini adalah permainan yang sudah ada sejak lama, aturannya mudah ditulis dalam bahasa Inggris dan kode komputer, dan banyak orang memainkannya. Entah karena alasan apa atau kombinasi berbagai alasan. Catur secara tradisional dianggap sangat serius. Mungkin karena jarang, bahkan mungkin tidak pernah, dimainkan demi uang, yang mungkin juga karena tidak ada unsur keberuntungan dan tidak ada informasi tersembunyi (tidak ada dadu atau kartu dan Anda dapat melihat seluruh papan). Mungkin karena Catur memiliki banyak kedalaman: ada banyak hal yang dapat dipelajari tentang permainan ini, sehingga Anda dapat terus menjadi lebih baik seumur hidup. Permainan ini memungkinkan beragam strategi, dan pemain tingkat master biasanya memiliki gaya bermain yang mudah dikenali.

Jadi, bukan hal yang mengada-ada ketika, di awal penelitian kecerdasan buatan, Catur diusulkan sebagai masalah penting untuk dikerjakan. Mustahil bagi siapa pun untuk dapat memainkannya di tingkat tinggi tanpa menjadi benar-benar cerdas, karena bagaimana mungkin seseorang dapat memainkan permainan ini tanpa perencanaan yang matang, menilai

nilai sebenarnya dari posisi papan, dan memahami pemikiran lawan serta memprediksi langkahnya?

Permainan ini tampaknya mendekati pemikiran murni. Atau, dapatkah Anda memikirkan aktivitas lain yang lebih jelas membutuhkan kecerdasan daripada bermain Catur? Wajar jika kita berasumsi bahwa jika kita membangun program yang merupakan pemain Catur ahli, kita akan memecahkan masalah kecerdasan buatan. Maka, orang-orang mulai mengerjakan masalah yang terdefinisi dengan baik ini.

Meskipun Turing sendiri mungkin adalah orang pertama yang menjalankan program bermain Catur, banyak peneliti lain menganggap ini sebagai topik penting. Permainan catur berkembang menjadi subbidang penelitian kecerdasan buatan yang dinamis, dengan konferensi, jurnal, dan kompetisi yang dikhususkan untuk mempelajari dan mengembangkan perangkat lunak yang dapat memainkan catur dan permainan papan serupa. Beberapa perkembangan penting dalam kecerdasan buatan terjadi dalam konteks permainan papan, seperti ketika ilmuwan komputer IBM, Arthur Samuel, pada tahun 1958 menemukan versi pertama dari apa yang sekarang disebut pembelajaran penguatan untuk membuat program permainan catur belajar dari pengalaman.

Ketika program permainan catur pertama dikembangkan, banyak yang berpikir bahwa program komputer tidak akan pernah bisa menyaingi pemain manusia tingkat master karena program ini hanyalah kode dan manusia cerdas. Dan catur, perlu diingat, adalah permainan canggih yang membutuhkan kecerdasan untuk dimainkan. Namun selama beberapa dekade penelitian yang berdedikasi, perangkat lunak permainan catur menjadi semakin kuat. Meskipun program-program ini awalnya hampir tidak dapat mengalahkan seorang pemula, secara bertahap mereka melampaui kinerja menengah, dan mendekati permainan tingkat master. Hal ini banyak berkaitan dengan tersedianya prosesor yang lebih cepat dan ukuran memori yang lebih besar, tetapi juga banyak berkaitan dengan perangkat lunak yang menjadi lebih baik pada dasarnya penyempurnaan dan penambahan pada algoritma dasar yang sama yang telah digunakan semua program ini sejak awal.

Pada tahun 1997, perkembangan ini akhirnya sejalan dengan perkembangan teknologi manusia, yang telah berkembang perlahan, jika pun ada. Dalam pertandingan yang dipublikasikan secara luas, IBM menurunkan komputer Catur Deep Blue khusus miliknya melawan juara dunia saat itu, Garry Kasparov. Komputer tersebut menang. Peristiwa ini menjadi titik awal perdebatan sengit tentang makna kecerdasan dan kecerdasan buatan setelah mesin menaklukkan Catur. Sebagian besar pengamat menyimpulkan bahwa Deep Blue sama sekali tidak cerdas, karena tampilan dan fungsinya sama sekali tidak seperti otak manusia. Inti dari Deep Blue adalah sebuah algoritma sederhana, meskipun dilengkapi dengan segudang fitur tambahan. Faktanya, algoritma ini adalah algoritma yang sama persis yang diciptakan (kembali) oleh Turing pada tahun 1940an. Jadi, bagaimana cara kerjanya?

1.1 KOMPUTER BERMAIN CATUR

Pendekatan yang digunakan hampir semua program permainan Catur adalah menggunakan beberapa varian dari algoritma minimax. Ini sebenarnya adalah algoritma yang

sangat sederhana. Algoritma ini bekerja dengan konsep status dan langkah papan. Keadaan papan adalah posisi semua buah di papan, dan langkah adalah transisi dari satu keadaan ke keadaan lain (misalnya, menggerakkan pion Anda dua langkah ke depan akan mengubah papan Anda ke keadaan yang mirip tetapi berbeda dari keadaan papan sebelum langkah).

Dari keadaan papan mana pun, cukup mudah untuk membuat daftar semua langkah yang dapat diambil pemain; rata-rata Anda dapat mengambil tiga puluh lima atau lebih langkah yang berbeda, dan pada giliran pertama, Anda memiliki dua puluh langkah untuk dipilih. Terkadang Anda hanya dapat mengambil satu atau dua langkah yang berbeda ketika ini terjadi, Anda biasanya memiliki masalah. Minimax berasumsi bahwa Anda dapat menyimpan beberapa keadaan papan dalam memori atau, dengan kata lain, Anda dapat menyimpan banyak salinan permainan. (Ini bukan masalah pada komputer modern mana pun, karena keadaan papan Catur dapat direpresentasikan hanya dalam beberapa byte.) Minimax juga berasumsi bahwa Anda memiliki cara untuk mengevaluasi nilai (atau utilitas, atau kebaikan) suatu kondisi papan. Untuk saat ini, Anda dapat menganggap nilai suatu kondisi papan sebagai jumlah bidak yang tersisa di papan, dikurangi jumlah bidak yang dimiliki lawan Anda. Jika hasilnya positif, Anda mungkin unggul dalam permainan.

Beginilah cara kerja Minimax. Bayangkan Anda bermain dari perspektif pemain putih, dan ingin mengetahui langkah terbaik yang harus diambil dari kondisi papan tertentu. Anda mulai dengan membuat daftar semua kemungkinan langkah yang dapat Anda lakukan dari kondisi tersebut. Kemudian, Anda mensimulasikan pengambilan setiap langkah dan menyimpan semua kondisi papan yang dihasilkan. Jika Anda kurang berwawasan, Anda bisa berhenti di sini; cukup perkirakan nilai setiap kondisi papan yang dihasilkan (misalnya, dengan menghitung bidak) dan pilih langkah yang mengarah ke kondisi papan dengan nilai tertinggi. Itu akan menjadi bagian maksimal dari algoritma Minimax. Namun, itu memang picik, karena langkah yang memberikan keuntungan langsung (misalnya, dengan menangkap salah satu bidak lawan) mungkin memberi lawan peluang untuk membalas dengan satu atau beberapa tangkapannya sendiri, sehingga menjadi bencana dalam jangka waktu yang tidak terlalu pendek. Setiap orang yang telah memainkan lebih dari satu permainan Catur mengetahui hal ini. Oleh karena itu, untuk setiap keadaan papan yang dihasilkan dari kemungkinan langkah pertama Anda, Anda membuat daftar semua kemungkinan langkah lawan dan mengevaluasi keadaan papan yang dihasilkan.

Perbedaan krusial dengan apa yang Anda lakukan pada langkah pertama adalah bahwa sementara pada langkah pertama Anda ingin menemukan langkah yang terbaik untuk Anda, pada langkah kedua, Anda berasumsi bahwa lawan akan mengambil langkah yang paling menguntungkan baginya, yaitu, langkah terburuk bagi Anda. Jika lawan Anda dapat menangkap bidak Anda saat gilirannya, dia akan melakukannya. Itulah bagian minimum dari algoritma minimum.

Dengan menggabungkan semuanya, langkah terbaik untuk Anda adalah langkah yang meminimalkan skor maksimum yang dapat dicapai lawan Anda pada gilirannya. Secara praktis, untuk masing-masing kemungkinan gerakan di giliran Anda, Anda menetapkan nilai papan terendah yang dapat dicapai lawan Anda melalui salah satu gerakannya di giliran kedua.

Anda sekarang melihat dua giliran ke depan. Namun, bagaimana dengan strategi cerdik di mana Anda membiarkan musuh melakukan tangkapan pada gilirannya hanya agar Anda sendiri dapat melakukan tangkapan yang lebih besar pada giliran berikutnya? Nah, Anda bisa saja membiarkan algoritma minimax berjalan selangkah lebih maju dan berasumsi bahwa lawan akan berasumsi bahwa Anda melakukan langkah yang paling menguntungkan Anda. Ini bisa berlangsung selamanya, atau sampai Anda mencapai akhir permainan dalam simulasi Anda. Bahkan, jika Anda melakukan simulasi hingga akhir permainan, mengeksplorasi semua langkah dan semua respons terhadap langkah-langkah tersebut, Anda akan menemukan strategi yang terbukti optimal, yang tidak dapat ditingkatkan lagi. Dengan kata lain, Anda akan bermain Catur dengan sempurna.

Tetapi Anda tidak akan melakukan simulasi hingga akhir permainan karena Anda tidak punya waktu untuk melakukannya. Jika ada tiga puluh kemungkinan langkah dari satu keadaan papan (angka umum untuk Catur di tengah permainan), maka setiap giliran yang Anda simulasikan akan mengalikan jumlah keadaan papan yang perlu Anda selidiki sebanyak tiga puluh kali. Dengan kata lain, Anda perlu mengevaluasi 30^t keadaan, di mana t adalah jumlah putaran yang Anda lihat ke depan.

Untuk melihat lima langkah ke depan, itu sekitar dua puluh empat juta. Jumlah keadaan yang perlu Anda selidiki dengan cepat mendekati jumlah atom di bumi dan angka-angka konyol lainnya. Oleh karena itu, mencari hingga akhir permainan Catur mustahil dilakukan oleh komputer mana pun yang dapat kita bangun di masa mendatang. Itulah sebabnya Anda memerlukan cara yang baik untuk memperkirakan nilai keadaan papan, karena Anda harus puas hanya dengan melihat beberapa putaran ke depan.

Dalam praktiknya, yang dilakukan agen Catur adalah mencari hingga kedalaman tertentu dan kemudian mengevaluasi status papan yang dicapainya, meskipun status tersebut biasanya bukan status menang atau kalah. Untungnya, dalam Catur, estimasi status papan sederhana seperti selisih buah catur antara hitam dan putih umumnya bekerja dengan cukup baik, meskipun banyak metode yang lebih kompleks telah dikembangkan.

Minimax disebut algoritma pencarian pohon, bukan karena membantu Anda mencari buah ceri terlezat di pohon ceri, tetapi karena apa yang dilakukannya dapat dipahami sebagai menumbuhkan pohon yang bercabang saat mencari langkah terbaik, pohon yang tumbuh terbalik. Bayangkan seperti ini: akar pohon adalah status papan asli, yang ingin Anda temukan langkah terbaiknya. Semua kemungkinan langkah dari status tersebut menjadi cabang yang tumbuh dari akar.

Di ujung setiap cabang terdapat status papan yang dituju oleh langkah tersebut. Tentu saja, dari setiap keadaan ini, sejumlah langkah dimungkinkan, dan langkah-langkah ini pada gilirannya dapat divisualisasikan sebagai cabang dari ujung cabang sebelumnya … dan begitu seterusnya hingga Anda mencapai keadaan papan di mana Anda tidak mencoba langkah apa pun lagi melainkan memperkirakan nilai keadaan tersebut. Keadaan tersebut disebut "daun", dalam analogi yang agak tidak sempurna ini. (Ilmuwan komputer tidak terkenal dengan analogi mereka.) Jumlah langkah yang mungkin pada setiap keadaan disebut "faktor percabangan".

Tentu saja, telah ada sejumlah penyempurnaan pada metode ini sejak Alan Turing sendiri pertama kali menyarankannya pada tahun 1940an. Ada cara untuk "memangkas" pencarian sehingga lebih sedikit keadaan papan yang diselidiki, ada cara untuk berkonsentrasi pada urutan langkah yang paling menjanjikan, dan ada cara yang jauh lebih baik untuk memperkirakan nilai papan. Namun prinsip minimax tetap ada. Prinsip ini merupakan inti dari hampir semua program permainan Catur yang sukses.

1.2 LANGSUNG MAJU KE GO

Go adalah permainan yang menempati tempat yang sama dalam budaya Asia Timur seperti halnya Catur dalam budaya Eropa. Go juga memiliki kesamaan lain dengan Catur, seperti memiliki dua pemain, informasi yang sempurna, tidak ada keacakan, dan satu pemain menggunakan buah catur putih dan yang lainnya hitam (gambar 1.2). Dalam hal lain, Go sebenarnya lebih sederhana. Go hanya memiliki dua atau tiga aturan, tergantung cara Anda menghitung, dan satu jenis buah catur dibandingkan dengan delapan buah catur dalam Catur. Mungkin agak mengejutkan, metode yang sama yang bekerja dengan sangat baik untuk bermain Catur gagal total dalam Go.

Algoritma berbasis Minimax umumnya memainkan permainan ini dengan buruk. Tampaknya ada dua alasan utama untuk ini: faktor percabangan (jumlah langkah) jauh lebih tinggi (sekitar 350 daripada 35 dalam Catur), dan sangat sulit untuk memperkirakan nilai keadaan papan secara akurat. Faktor percabangan yang tinggi berarti minimax hanya dapat melakukan pencarian yang sangat dangkal, dan kesulitan dalam memperkirakan nilai papan berarti "sinyal" yang digunakan algoritma minimax lebih buruk. Namun, untuk waktu yang lama, kita tidak mengetahui algoritma yang lebih baik untuk bermain Go. Oleh karena itu, program bermain Go terbaik terhenti di tingkat pemula, bahkan ketika program bermain Catur mencapai dan melampaui tingkat grandmaster.

Gambar 1.2

Go, sepupu catur dari Asia yang lebih sederhana namun lebih sulit (untuk komputer).

Maka wajar jika mata orang-orang beralih ke Go setelah Catur ditaklukkan. Go tampak jauh lebih sulit daripada Catur. Mungkinkah permainan ini tidak dapat ditaklukkan dengan teknik

sesederhana itu? Mungkinkah permainan ini benar-benar membutuhkan kecerdasan untuk memainkannya?

Kita akhirnya mulai melihat kemajuan nyata pada perangkat lunak permainan Go pada tahun 2007 ketika algoritma pencarian pohon Monte Carlo (MCTS) ditemukan. Seperti minimax, MCTS adalah algoritma pencarian pohon. Tidak seperti minimax, ia bersifat acak. (Itulah mengapa ia memiliki kata "Monte Carlo," seperti kasino Monaco yang terkenal, dalam namanya.)

Menerima kenyataan bahwa mustahil untuk mengeksplorasi semua langkah yang mungkin pada tingkat yang sama, MCTS memilih langkah mana yang akan dieksplorasi terlebih dahulu secara acak; kemudian melanjutkan untuk mengeksplorasi lebih lanjut langkah mana yang tampaknya paling menjanjikan pada awalnya. Alih-alih menghitung bidak untuk memperkirakan nilai papan (cara ini sangat buruk dalam Go), MCTS memainkan permainan secara acak hingga akhir berkali-kali, dan melihat persentase "permainan" yang dimenangkannya. Mungkin tampak gila dengan begitu banyak keacakan dalam algoritmanya, tetapi secara empiris cara ini bekerja dengan sangat baik.

Hampir dua puluh tahun setelah kemenangan Deep Blue atas Garry Kasparov, supremasi manusia dalam Go dijungkirbalikkan. Kali ini, perusahaan riset AI DeepMind, yang saat itu merupakan divisi dari Google, yang menyediakan perangkat lunaknya. Dalam serangkaian pertandingan pada tahun 2016, AlphaGo milik DeepMind menghadapi Lee Sedol, yang bisa dibilang pemain Go terbaik dunia, dan menang 4–1. AlphaGo dibangun di atas algoritma MCTS, dikombinasikan dengan jaringan saraf yang telah dilatih selama berbulan-bulan pada pertandingan sebelumnya dari banyak juara Go, dan dengan bermain melawan dirinya sendiri (saya akan membahas jaringan saraf nanti di buku ini).

Ini adalah permainan papan klasik penting terakhir yang menyerah pada mesin. Permainan ini juga yang paling sulit. Tidak ada lagi permainan papan klasik yang dimainkan manusia terbaik lebih baik daripada program komputer terbaik, setidaknya bukan permainan papan klasik yang dipedulikan orang-orang. Jadi, apakah AlphaGo cerdas? Kebanyakan orang akan menjawab tidak. Meskipun fungsinya berbeda dari Deep Blue dan mencakup unsur pembelajaran, ia tetap tidak seperti otak manusia. "Hanya sebuah algoritma," kata sebagian orang. Dan ia hanya bisa bermain Go. Ia bahkan tidak bisa bermain Catur (tanpa melatih ulang jaringannya), ia juga tidak bisa mengendarai mobil atau menulis puisi. Hal ini memunculkan beberapa pertanyaan penting: Apakah suatu benda perlu berfungsi seperti otak manusia agar cerdas? Dan apakah Anda perlu cerdas agar dapat bermain gim dengan baik? Mari kita coba jawab pertanyaan kedua terlebih dahulu.

BAB 2
KECERDASAN BUKAN SYARAT UTAMA UNTUK BERMAIN GAME

Maukah kita bermain game? Pilih: Catur, *Super Mario Bros*., atau *Angry Birds*. Saya memberi Anda pilihan karena saya tidak tahu apakah Anda familiar dengan ketiganya. Saya sudah membahas Catur di bab sebelumnya: permainan papan paling terkenal di dunia Barat, dimainkan dengan menggerakkan bidak-bidak seperti pion, raja, dan ratu secara fisik di atas papan dengan kotak hitam dan putih yang berselang-seling. Dengan menggerakkan bidak-bidak ini sehingga mengancam dan menangkap bidak lawan, Anda pada akhirnya dapat mengalahkan lawan dengan mengepung rajanya. Permainan ini hampir tidak berubah sejak diciptakan ribuan tahun yang lalu.

Gambar 2.1
Game platform penentu genre *Super Mario Bros*. (Nintendo, 1985).

Super Mario Bros. adalah permainan platform yang menyertai peluncuran konsol *Nintendo Entertainment System* (NES) 8-bit Nintendo di Eropa/Amerika pada tahun 1985 (gambar 2.1). Dengan menekan tombol pada kotak plastik kecil, Anda mengendalikan tukang ledeng periang, Mario, saat ia menghindari kura-kura jahat, menginjak manusia jamur yang mengancam, melompati celah, mengumpulkan koin, dan menyelamatkan sang putri yang telah diculik oleh kadal raksasa. Sekuel permainan terus dikembangkan untuk semua perangkat keras Nintendo dan selain ratusan juta salinan permainan Super Mario Bros. yang telah dijual secara sah, ada lusinan versi permainan yang tidak sah yang tersedia untuk setiap platform perangkat keras yang dapat dibayangkan.

Angry Birds adalah fenomena permainan seluler dari tahun 2009 oleh perusahaan Finlandia Rovio (gambar 2.2). Anda menunjuk dan menggesekkan jari Anda di atas layar sentuh ponsel Anda untuk melemparkan berbagai macam burung ke berbagai bangunan, dan tujuan Anda adalah membuat bangunan tersebut runtuh di atas babi hijau jahat yang telah mencuri

telur Anda. Game asli, serta segudang sekuelnya tersedia untuk perangkat iPhone, iPad, dan Android dan menduduki puncak daftar buku terlaris di semua platform tersebut.

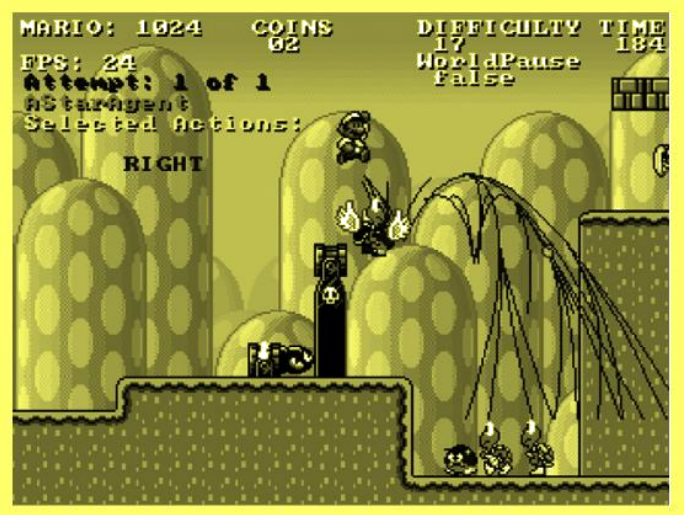
Gambar 2.2

Angry Birds (Rovio, 2009), game teka-teki fisika yang tampaknya ada di iPhone semua orang setelah debutnya.

Dugaan saya, Anda telah memainkan ketiga game ini, atau setidaknya melihat seseorang memainkannya. Jika tidak, Anda mungkin telah memainkan dua di antaranya, atau setidaknya satu. Seandainya Anda tidak mengenal Catur, *Super Mario Bros*., atau *Angry Birds*, saya agak bingung siapa Anda dan di dunia mana Anda tinggal. Apakah Anda membaca buku ini di masa depan yang jauh? Saya berasumsi Anda bermain game.

Setelah memastikan bahwa Anda bermain game, izinkan saya bertanya: Mengapa Anda bermain game? Untuk bersantai, bersenang-senang, atau sedikit melupakan diri sendiri? Mungkin sebagai cara bersosialisasi dengan teman? Hampir pasti bukan sebagai semacam latihan otak. Tapi mari kita lihat apa yang sebenarnya Anda lakukan:

Anda berencana. Dalam Catur, Anda merencanakan kemenangan Anda dengan membayangkan serangkaian beberapa gerakan yang akan Anda ambil untuk mencapai skakmat, atau setidaknya menangkap salah satu buah catur lawan. Jika Anda jago, Anda juga memperhitungkan gerakan lawan dan membuat rencana cadangan jika mereka tidak jatuh ke dalam perangkap yang Anda siapkan. Dalam *Super Mario Bros.*, Anda merencanakan apakah akan mengambil jalur yang lebih tinggi, yang memberikan lebih banyak hadiah tetapi lebih berisiko, atau jalur yang lebih rendah dan lebih aman (gambar 2.3). Anda juga berencana untuk menyusuri pipa yang mungkin membawa Anda ke ruang harta karun tersembunyi, atau terus melewatinya, tergantung pada sisa waktu dan seberapa besar keinginan Anda untuk menyelesaikan level.



Gambar 2.3

Algoritma perencanaan (versi algoritma A^*) memainkan klon *Super Mario Bros*. Garis hitam menunjukkan berbagai jalur masa depan yang sedang dipertimbangkan oleh algoritma tersebut.

Anda mungkin berencana untuk memakan power-up yang memungkinkan Anda menembus dinding itu sehingga Anda dapat menekan tombol yang melepaskan kacang yang darinya Anda dapat menumbuhkan pohon kacang yang memungkinkan Anda memanjat ke awan yang ingin Anda tuju. Dalam *Angry Birds*, Anda merencanakan ke mana harus melempar setiap burung agar mencapai kehancuran maksimum dengan jumlah burung paling sedikit. Jika Anda menghancurkan dinding es dengan burung biru, Anda kemudian dapat memukul rongga itu dengan burung bom hitam, meruntuhkan struktur utama, dan menghabisi babi pengecut yang bersembunyi itu dengan burung merah Anda.

Anda berpikir secara spasial. Catur berlangsung di petak dua dimensi, di mana sel-sel yang tidak ditempati oleh bidak putih atau hitam "kosong". Mereka yang telah memainkan permainan ini beberapa kali dan menghayati aturannya mulai melihat beberapa peluang dan ancaman secara langsung saat mereka melihat papan. Fakta bahwa ratu terancam tampak jelas seperti huruf X dalam deretan O, dan kemungkinan posisi kuda dapat langsung terlihat di papan. Di *Super Mario Bros.*, Anda perlu memperkirakan lintasan lompatan untuk melihat apakah Anda dapat melewati celah dan memantul dari musuh, yang berarti membayangkan lompatan tersebut sebelum Anda mengeksekusinya.

Anda juga perlu memperkirakan apakah Anda dapat melewati celah kecil itu dengan ukuran Anda saat ini (Mario dapat mengubah ukuran) dan apakah jalur di sana mengarah ke suatu tempat. Di *Angry Birds*, Anda juga perlu memperkirakan lintasan, terkadang sangat rumit yang melibatkan pantulan dan gravitasi yang aneh, dan Anda mungkin juga perlu menentukan apakah Anda dapat memasukkan burung itu ke dalam lorong sempit antara batu berpiksel dan tempat keras virtual itu.

Anda memprediksi permainan dan lawan Anda. Dalam Catur, memprediksi apa yang akan dilakukan lawan Anda sangat penting untuk permainan yang sukses. Jika Anda tahu bagaimana lawan Anda akan bereaksi terhadap gerakan Anda, Anda dapat merencanakan

strategi Anda dengan keyakinan penuh bahwa mereka akan berhasil. *Super Mario Bros.* dan *Angry Birds* biasanya bukan permainan yang bersifat adversarial (Anda tidak bermain melawan lawan manusia), tetapi tantangannya adalah memprediksi tindakan dan reaksi lingkungan. Kapan meriam akan ditembakkan? Ke arah mana kura-kura itu akan menghadap jika saya mendarat di sebelah kirinya? Akankah kadal monster itu maju sepenuhnya jika saya tidak melompat ke platform? Dan bagaimana tepatnya bangunan kompleks itu akan runtuh jika saya merobohkan penyangga bawahnya, di mana semua bidak akan mendarat, dan akankah salah satunya meledakkan kotak TNT itu untuk menciptakan reaksi berantai yang bagus? Meskipun keacakan mungkin berperan dalam *Angry Birds* (*Super Mario Bros.* sepenuhnya deterministik), kesulitannya terutama berasal dari interaksi yang sangat kompleks di antara berbagai objek dalam permainan.

Anda menilai diri sendiri. "Kenali diri Anda," kata Socrates. Dia mungkin tidak sedang membicarakan Catur, apalagi Angry Birds, tetapi sungguh, mengenal diri sendiri adalah aset yang tak ternilai saat bermain gim. Melebih-lebihkan kemampuan akan membuat Anda bermain gegabah dan kemungkinan besar kalah; meremehkan kemampuan berarti Anda tidak akan mencoba strategi berisiko yang sebenarnya bisa memenangkan permainan. Anda juga perlu memperhitungkan pengaruh diri dan memperbaikinya. Apakah Anda saat ini sedang tidak seimbang karena rencana Anda tidak berhasil, terlalu bersemangat dengan kesuksesan Anda baru-baru ini, atau mungkin didorong oleh nafsu balas dendam atas langkah buruk lawan yang baru saja merebut ratu Anda? Nah, Anda perlu mempertimbangkan hal itu.

Jangan mencoba strategi sepuluh langkah itu jika Anda tahu itu hanya berdasarkan angan-angan, bukan penilaian situasi yang cermat. Hal yang sama berlaku untuk *Super Mario Bros.* dan *Angry Birds*: jika Anda tidak mengetahui tingkat kemampuan Anda sendiri, Anda tidak akan bisa maju dalam permainan karena Anda akan mencoba strategi yang terlalu sulit bagi Anda. Anda mungkin juga lebih baik dalam menjalankan beberapa taktik, seperti lompatan jauh atau memasang jebakan dengan kuda Anda, dibandingkan taktik lain, seperti tembakan presisi atau bergerak cepat untuk mengepung raja.

Anda bergerak. Memang benar bahwa Catur tidak banyak melibatkan keterampilan motorik, setidaknya kecuali jika permainannya berubah menjadi perkelahian, tetapi dua permainan lainnya jelas melakukannya. *Super Mario Bros.* mengharuskan Anda menekan dua tombol dan D-pad, yang merupakan delapan tombol arah, sangat sering (seringkali beberapa penekanan per detik). *Angry Birds* menuntut kontrol yang sangat baik atas gerakan jari Anda di layar untuk menembak burung ke arah yang tepat dengan kekuatan yang tepat dan mengaktifkan kemampuan khususnya pada waktu yang tepat. Dalam kedua permainan, gerakan-gerakan ini harus dikoordinasikan dengan apa yang terjadi di layar dan diatur waktunya dengan sempurna. Aspek sensorimotor dari permainan inilah yang cenderung cepat dipahami oleh anak-anak berusia lima tahun tetapi tidak selalu oleh orang tua mereka yang frustrasi.

Tentu saja, permainan lain menawarkan tantangan lain. Penembak orang pertama seperti *Halo* atau *Call of Duty* menantang navigasi spasial Anda dalam tiga dimensi, dan dalam mode multipemain, mereka langsung membawa Anda ke dalam kompleksitas strategi tim.

Permainan peran seperti *Skyrim* dan *Mass Effect* mengharuskan Anda memahami motif di balik tindakan karakter yang kompleks, menyelesaikan dilema etika, dan menavigasi politik yang berbahaya (setidaknya jika Anda memainkannya sebagaimana mestinya, meskipun Anda bisa mencapai kemajuan yang cukup jauh di beberapa permainan hanya dengan menembaki semua yang bergerak). Permainan simulasi ekonomi seperti *SimCity* dan *Transport Tycoon* mengharuskan Anda memahami dan memengaruhi sistem ekonomi yang kompleks.

Salah satu cara untuk mencoba menguraikan jenis tantangan kognitif yang ditawarkan permainan adalah dengan melihat psikologi atau, lebih tepatnya, psikometrika, untuk melihat apakah ada daftar kemampuan kognitif yang praktis. Kita kemudian dapat mencoba mencari tahu bagaimana masing-masing kemampuan ini diperlukan (atau tidak) untuk memainkan berbagai jenis permainan. Ternyata memang ada daftar seperti itu. Secara khusus, teori *Cattell-Horn-Carroll* (CHC) membagi kecerdasan umum menjadi sebelas "kemampuan kognitif luas" yang berbeda, yang selanjutnya dibagi lagi menjadi banyak kemampuan kognitif yang lebih terspesialisasi. Taksonomi ini didasarkan pada analisis statistik dari ratusan tes kognitif yang berbeda dan diterima secara luas dalam komunitas psikometri (meskipun seiring dengan munculnya bukti empiris baru, kategori dimodifikasi dan ditambahkan).

Tabel 2.1 mencantumkan sebelas kemampuan kognitif luas dari teori CHC dan memberikan beberapa contoh situasi dalam permainan di mana kemampuan tersebut digunakan. Perlu dicatat bahwa ini masih jauh dari daftar lengkap; saya kurang lebih telah mencantumkan beberapa contoh pertama yang terlintas dalam pikiran, mencoba mendapatkan keragaman dalam hal genre permainan. Dugaan saya adalah hampir semua permainan akan menggunakan setidaknya lima kemampuan kognitif yang berbeda (*Super Mario Bros.*, *Angry Birds*, dan *Chess* tentu saja menggunakannya), tetapi ini hanya tebakan dan saya tidak mengetahui ada yang pernah melakukan penelitian tentang hal ini. Seseorang seharusnya melakukan penelitian itu.

Tabel 2.1 Berbagai kemampuan kognitif menurut teori Cattell-Horn-Carroll dan beberapa contoh penggunaannya dalam permainan

Kemampuan kognitif yang luas	Contoh penggunaan dalam permainan
Pemahaman-pengetahuan	Berkomunikasi dengan pemain lain di berbagai game multipemain, mulai dari *Bridge* hingga *Gears of War* dan *World of Warcraft*
Penalaran yang lancar	Menggabungkan bukti untuk mengisolasi tersangka di *Phoenix Wright*; memecahkan teka-teki di *Drop7*
Pengetahuan kuantitatif	Mengontrol sistem kompleks yang melibatkan banyak data kuantitatif, seperti di *SimCity*, atau manajemen karakter di *Dungeons and Dragons*
Kemampuan membaca dan menulis	Membaca instruksi dalam permainan, mengikuti percakapan, dan memilih opsi dialog dalam permainan peran seperti *Mass Effect*; menulis perintah dalam petualangan teks seperti *Zork*

Memori jangka pendek	Di mana-mana! Misalnya, mengingat kartu yang baru saja dimainkan di Texas hold'em poker atau *Hearthstone*
Penyimpanan dan pengambilan jangka panjang	Mengingat permainan Catur atau *StarCraft* sebelumnya yang menyerupai permainan saat ini untuk mendapatkan wawasan tentang strategi
Pemrosesan visual	Menemukan kemungkinan kecocokan ubin di *Candy Crush Saga* atau penembak jitu musuh di *Call of Duty*
Pemrosesan auditori	Menyadari kedatangan zombi (dan dari arah mana) di *Left 4 Dead*; menguping negosiasi rahasia di *Diplomacy*
Kecepatan pemrosesan	Memutar bidak dengan benar di *Tetris*; mengatur pertarungan secara detail di *StarCraft*; bermain Catur Cepat
Waktu/kecepatan pengambilan keputusan atau reaksi	Di mana saja! Misalnya, melawan gerakan di *Street Fighter* atau memutuskan buah mana yang akan dipotong di *Fruit Ninja*

Singkatnya, kita menggunakan berbagai bentuk kecerdasan saat bermain game, hampir sepanjang waktu. Kedengarannya seperti kerja keras. Memang luar biasa bermain game bisa membuat kita rileks, tapi memang begitulah kenyataannya.

2.1 BELAJAR KETIKA BERMAIN GIM

Sejauh ini, kita hanya membahas keterampilan individual yang Anda latih saat bermain gim. Namun, Anda tidak melatihnya dengan cara yang sama sepanjang waktu; Anda membangun keterampilan Anda saat bermain. Tentu saja, Anda tidak merasa seperti sedang mengikuti kelas saat bermain gim (jika ya, gim itu tidak terlalu bagus). Namun, Anda belajar. Ini buktinya: Anda jauh lebih mahir dalam gim setelah memainkannya selama beberapa waktu dibandingkan saat Anda memulainya. Cobalah mainkan kembali salah satu level awal di Super Mario Bros. atau Angry Birds. Atau coba mainkan kembali komputer Catur pada tingkat kesulitan pemula, yang mengalahkan Anda secara telak saat pertama kali mencoba. Mudah saja.

Raph Koster, seorang desainer gim ternama, berpendapat bahwa belajar adalah alasan utama mengapa gim itu menyenangkan. Gim yang bagus dirancang untuk mengajari Anda cara memainkannya; semakin baik gim tersebut mengajarkan Anda, semakin baik pula desainnya. Anda bersenang-senang karena Anda belajar bermain gim, dan ketika Anda berhenti belajar, Anda berhenti bersenang-senang. Jika tidak ada lagi yang bisa dipelajari, Anda akan bosan dengan permainan tersebut.

Oleh karena itu, permainan yang mudah dan bisa Anda taklukkan pada percobaan pertama tidaklah menarik, begitu pula permainan yang hampir mustahil dan tidak memungkinkan Anda untuk maju. Sebaliknya, permainan yang dirancang dengan baik menawarkan progresi tingkat kesulitan yang panjang dan lancar, tempat Anda dapat terus

belajar sambil bermain. Kita bisa mengatakan bahwa permainan ini mudah diakses dan mendalam.

Misalnya, ketika Anda mulai bermain *Super Mario Bros.*, pertama-tama Anda harus mempelajari fungsi tombol-tombolnya, tombol A membuat Mario melompat dan menekan D-pad ke arah yang berbeda membuatnya berjalan ke kiri atau kanan, lalu Anda harus mempelajari cara mengatasi berbagai tantangan yang dihadirkan permainan. "Jadi, ada jamur berjalan mendekat. Apa yang bisa kulakukan? Aha! Aku bisa melompatinya!" Seiring Anda maju melalui level-level *Super Mario Bros.*, Anda akan menyadari bahwa tantangan yang dihadirkan menjadi semakin sulit, tetapi juga bahwa Anda semakin siap untuk mengatasinya.

Desain level *Super Mario Bros.* yang sering ditiru biasanya memperkenalkan versi dasar dari beberapa tantangan (misalnya, melompati celah atau musuh yang terjebak di lembah di antara dua pipa) dan kemudian menyajikan versi yang lebih maju dari tantangan yang sama (celah yang lebih panjang, berbagai jenis musuh di lembah) atau kombinasi dari beberapa tantangan sebelumnya (lompatan jauh melewati celah, setelah itu Anda langsung mendarat di lembah yang penuh musuh). Setiap kali, penyelesaian beberapa tantangan sebelumnya telah mempersiapkan Anda untuk menangani tantangan baru yang lebih maju. Dan setelah beberapa saat, ketika Anda berpikir bahwa tidak ada cara tersisa untuk menghasilkan tantangan baru yang menarik dengan memvariasikan tantangan yang ada, permainan melemparkan beberapa bahan baru yang menawarkan variasi lebih lanjut dan tantangan yang lebih dalam. Salah satu bahan baru tersebut, diperkenalkan agak terlambat dalam permainan, adalah musuh berduri, yang tidak dapat dikalahkan dengan melompat di atasnya. Menambahkan musuh berduri ke tantangan yang ada memaksa Anda untuk mengembangkan strategi baru untuk mengatasi tantangan yang tampak familier tetapi segar.

Akhirnya, bahkan ketika Anda telah berhasil menyelesaikan seluruh permainan (mengalahkan bos di level terakhir dan menyelamatkan sang putri), masih banyak yang tersisa untuk ditemukan, termasuk area dan harta karun tersembunyi, dan cara untuk mengalahkan seluruh permainan dalam waktu kurang dari sepuluh menit (jika Anda dari persuasi itu). *Super Mario Bros.* secara luas dianggap sebagai mahakarya desain gim, sebagian karena menjadi mahakarya pedagogi: kursus yang mendalam dan bermanfaat di mana peningkatan berikutnya selalu dalam jangkauan.

Ceritanya hampir sama dengan *Angry Birds*. Pertama, Anda mempelajari keterampilan motorik dasar menggesekkan jari untuk melempar burung, sebelum melanjutkan untuk memahami bagaimana berbagai burung berinteraksi dengan material yang digunakan untuk membangun menara dan bagian menara mana yang paling penting untuk dipukul agar dapat menghancurkan seluruh menara. Sesekali, gim ini menambahkan jenis material baru, burung baru, dan perangkat lain untuk memperluas jangkauan tantangan. Bahkan dalam Catur, perkembangannya serupa, dengan pengecualian yang jelas bahwa sangat sedikit keterampilan motorik yang diperlukan dan pembelajaran terjadi di banyak permainan Catur, alih-alih di banyak level yang sama.

Pertama, Anda mempelajari aturan dasar Catur, termasuk cara bidak bergerak dan memukul. Kemudian, Anda mempelajari aturan yang lebih rumit, yang mensyaratkan

penguasaan aturan yang lebih sederhana, termasuk rokade dan saat permainan berakhir seri. Anda kemudian dapat melanjutkan mempelajari heuristik, yang pertama sederhana, lalu yang lebih rumit; lalu Anda mempelajari buku pembukaan (daftar langkah pembukaan yang baik), kebiasaan dan gaya bermain pemain tertentu, dan seterusnya.

Gagasan bahwa bermain (game atau lainnya) berjalan seiring dengan pembelajaran bukanlah hal yang unik dalam desain game. Psikolog perkembangan Lev Vygotsky berbicara tentang "zona perkembangan proksimal" dalam permainan anak-anak, di mana anak-anak biasanya memilih untuk bermain dengan objek dan tugas yang berada di luar kapasitas mereka karena ini adalah yang paling bermanfaat.

Terkait dengan itu, konsep aliran dari ahli teori kreativitas Mihaly Czikszentmihalyi menyatakan bahwa aliran dapat dialami saat melakukan tugas yang sangat sulit untuk menantang Anda tetapi tidak cukup mudah untuk membuat Anda bosan, dan di mana kesulitan tugas meningkat seiring dengan peningkatan kinerja Anda. Czikszentmihalyi mengembangkan konsep ini dengan mengacu pada kreativitas artistik dan ilmiah, tetapi berlaku juga untuk bermain game.

Dari perspektif yang tampaknya sangat berbeda, peneliti pembelajaran mesin Jürgen Schmidhuber memperkenalkan formalisasi matematis dari rasa ingin tahu. Dalam modelnya, agen yang ingin tahu (manusia atau buatan) mencari tugas yang memungkinkannya meningkatkan model tugasnya, dan dengan demikian meningkatkan kapasitasnya untuk melakukan tugas tersebut. Dengan kata lain, menurut teori Schmidhuber, agen yang secara matematis optimal ingin tahu melakukan hal yang sama seperti anak kecil yang belajar tentang dunia dengan bermain dengannya, atau seperti pemain yang cerdas memilih untuk memainkan permainan yang disukainya atau memilih tantangan yang tampaknya menarik dalam permainan itu.

Singkatnya, tampaknya permainan menantang otak Anda dalam lebih dari satu cara jauh lebih dari satu cara dan, lebih jauh lagi, permainan yang bagus dirancang untuk membuat Anda tetap tertantang dengan meningkatkan tantangan (dan memberikan tantangan tambahan) secara pedagogis. Sekolah harus memperhatikan (beberapa melakukannya). Sangat mungkin bahwa permainan yang bagus, yang kita pilih untuk mainkan dan terus kita mainkan, sangat bagus setidaknya sebagian karena permainan tersebut berhasil terus-menerus menantang otak kita dalam berbagai cara.

Jadi, Anda pasti menggunakan kecerdasan Anda saat bermain permainan. Pada saat yang sama, kita telah melihat di bab sebelumnya bahwa adalah mungkin untuk membangun perangkat lunak yang dapat bermain Catur atau Go lebih baik daripada manusia mana pun, meskipun tampaknya tidak cerdas. Jadi, mengapa kecerdasan dibutuhkan manusia untuk bermain gim, tetapi tidak dibutuhkan mesin untuk memainkannya? Apa yang terjadi di sini? Saatnya untuk mencoba memahami apa yang kita maksud dengan kecerdasan buatan dan, dalam prosesnya, apa yang kita maksud dengan kecerdasan.

Kecerdasan Buatan (AI) telah menjadi elemen penting dalam industri game, mengubah cara game dibuat, dimainkan, dan dinikmati. Mulai dari peningkatan interaktivitas karakter non-pemain (NPC) hingga pengembangan alur cerita yang dapat beradaptasi, AI tidak hanya

menambah tingkat realisme dalam game, tetapi juga menciptakan pengalaman bermain yang lebih mendalam dan personal bagi para pemain. Artikel ini akan membahas berbagai fungsi AI dalam dunia game serta dampaknya terhadap masa depan industri gaming.

1. Peningkatan NPC dan Musuh
   AI secara signifikan memperbaiki kecerdasan dan realisme karakter non-pemain dalam game. Dengan AI, NPC dapat menanggapi aksi pemain secara dinamis, memiliki strategi kompleks, dan menyesuaikan diri dengan gaya bermain yang berbeda, sehingga memberikan tantangan yang lebih nyata dan menghindari kejenuhan. Musuh yang dikendalikan AI juga mampu mengubah taktik, menyajikan strategi tak terduga, dan meningkatkan kesulitan serta daya tarik sebuah game.
2. Pengembangan Alur Cerita yang Adaptif
   AI memungkinkan alur cerita dalam game berkembang sesuai dengan keputusan dan tindakan pemain, menciptakan narasi yang kaya dan personal. Sistem ini membuat cerita menjadi organik dan memberi pemain rasa pengaruh nyata terhadap dunia game.
3. Pengoptimalan Gameplay dan Kesulitan
   AI juga digunakan untuk menyesuaikan tingkat kesulitan secara real-time agar pengalaman bermain tetap menantang namun adil. Sistem AI menganalisis performa pemain dan otomatis mengubah parameter game agar pemain tidak merasa frustrasi atau bosan, sehingga game tetap menarik bagi berbagai tingkat keterampilan pemain.
4. Pemrosesan Bahasa Alami dan Interaksi Suara
   Teknologi AI dalam pemrosesan bahasa alami memungkinkan pemain berinteraksi dengan game menggunakan suara.
5. *Procedural Content Generation* (PCG)
   AI digunakan untuk menghasilkan konten secara procedural, seperti peta, level, atau bahkan alur cerita yang dibuat secara otomatis dan acak. Hal ini menjamin variasi dan kesegaran pengalaman bermain, memastikan tidak ada dua permainan yang sama persis.
6. Pembelajaran dan Adaptasi
   AI dalam game bukan hanya meningkatkan gameplay tetapi juga bisa belajar dari gaya bermain, kebiasaan, dan keputusan pemain. AI menggunakan data tersebut untuk menciptakan tantangan yang lebih tepat dan memuaskan, menjadikan pengalaman bermain lebih personal dan menarik.

2.2 PENGGUNAAN AI DALAM GAME PADA NPC

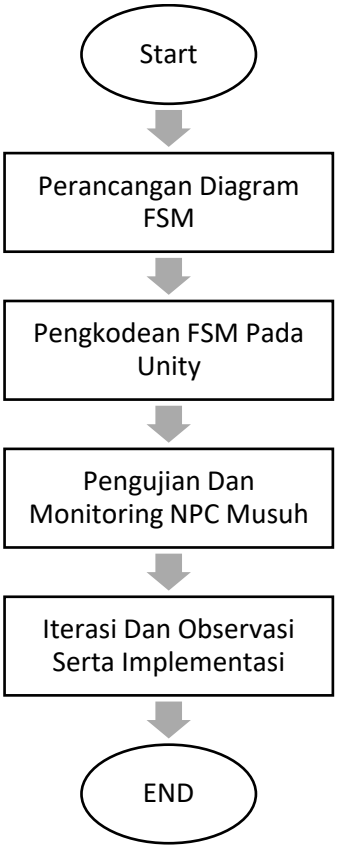
Permainan atau game adalah salah satu bentuk hiburan yang sangat digemari dan mengalami perkembangan pesat di era digital saat ini. Game biasanya dikelompokkan berdasarkan genre, yang meliputi berbagai macam jenis seperti aksi, petualangan, simulasi, teka-teki, dan lain sebagainya. Dalam sebuah game terdapat berbagai elemen, salah satunya adalah mekanik permainan, yang merupakan aturan dan sistem yang menentukan cara kerja dan interaksi game dengan pemain. Mekanik ini mencakup berbagai aspek, seperti kontrol,

grafis, suara, cerita, karakter, dan level. Salah satu elemen mekanik yang menarik dan sering menjadi fokus perhatian adalah karakter non-playable (NPC). NPC adalah karakter dalam game yang tidak dapat dimainkan langsung oleh pemain, melainkan dikendalikan oleh komputer. Terdapat berbagai jenis NPC dalam dunia game, antara lain NPC teman, NPC yang memberikan misi, NPC pendukung cerita, dan NPC sebagai musuh. NPC yang berperan sebagai musuh berfungsi untuk menghambat tujuan pemain, sehingga memberikan tantangan, variasi, dan dinamika dalam permainan. Oleh karena itu, kecerdasan dari NPC musuh sangat berpengaruh terhadap pengalaman bermain. Dengan demikian, menciptakan NPC musuh yang menantang dan beragam merupakan salah satu tantangan dalam pengembangan game. Salah satu pendekatan yang digunakan untuk mencapai hal ini adalah penerapan kecerdasan buatan atau *Artificial Intelligence* (AI). AI merupakan teknologi yang meniru cara berpikir manusia sehingga mesin dapat melaksanakan tugas-tugas seperti manusia. Dengan AI, NPC musuh dapat bersikap lebih dinamis dan adaptif, mampu menanggapi perubahan situasi, menyesuaikan strategi, bahkan belajar dari pemain

Finite State Machine (FSM) adalah sebuah mesin abstrak yang berfungsi untuk mendefinisikan sejumlah kondisi dan mengatur kapan suatu keadaan berubah. Menurut Rahadian, FSM adalah metode perancangan sistem kontrol yang menggambarkan perilaku atau prinsip kerja sistem menggunakan tiga elemen utama, yaitu keadaan (state), kejadian (event), dan aksi (action). Dengan pendekatan FSM, perilaku musuh dapat diorganisasikan ke dalam beberapa state yang saling terhubung, seperti menyerang, menghindar, berpatroli, dan lain-lain.

Dalam penerapan AI untuk game, FSM digunakan untuk mengatur mekanik NPC musuh. FSM sebagai metode perancangan sistem kontrol menjabarkan tingkah laku sistem dengan tiga komponen berikut :

1. Keadaan (State): menggambarkan kondisi spesifik NPC, misalnya menyerang, menghindar, atau berpatroli.
2. Kejadian (Event): kejadian yang memicu perubahan dari satu state ke state lain, seperti deteksi pemain atau terkena serangan.
3. Aksi (Action): tindakan yang dijalankan NPC saat berada di suatu state atau saat transisi antar state.



Gambar 2.1 Langkah-langkah implementasi FSM

Pada gambar di atas menggambarkan tahapan implementasi *Finite State Machine* (FSM) pada NPC musuh. Proses dimulai dengan perancangan diagram FSM yang menggambarkan state dan transisi yang diperlukan. Selanjutnya, state dan transisi tersebut dikodekan menggunakan bahasa pemrograman C# di Unity Engine. Setelah pengkodean selesai, dilakukan pengujian dan pengamatan terhadap perilaku NPC dalam permainan untuk memastikan FSM bekerja sesuai harapan. Terakhir, dilakukan iterasi dan penyesuaian sesuai kebutuhan berdasarkan hasil pengamatan tersebut. Alat dan teknologi yang digunakan dalam perancangan:

A. Unity Engine

   Unity Engine adalah aplikasi pengembangan game lintas platform yang dirancang agar mudah digunakan. Unity mendukung pembuatan grafis 2D dan 3D, dengan bahasa pemrograman utama yang digunakan adalah C-Sharp (C#). Mesin game ini dapat diakses secara gratis maupun dengan lisensi berbayar, serta dilengkapi dengan dokumentasi dan tutorial yang melimpah sehingga memudahkan proses pembelajaran.

B. Visual Studio Code

   Visual Studio Code merupakan editor kode yang kompatibel dengan sistem operasi Windows, macOS, dan Linux. Editor ini mendukung berbagai bahasa pemrograman seperti Javascript, PHP, C, dan Go. Selain itu, Visual Studio Code memiliki beragam

ekstensi yang membantu para pengembang dalam menyelesaikan pekerjaannya dengan lebih efisien.

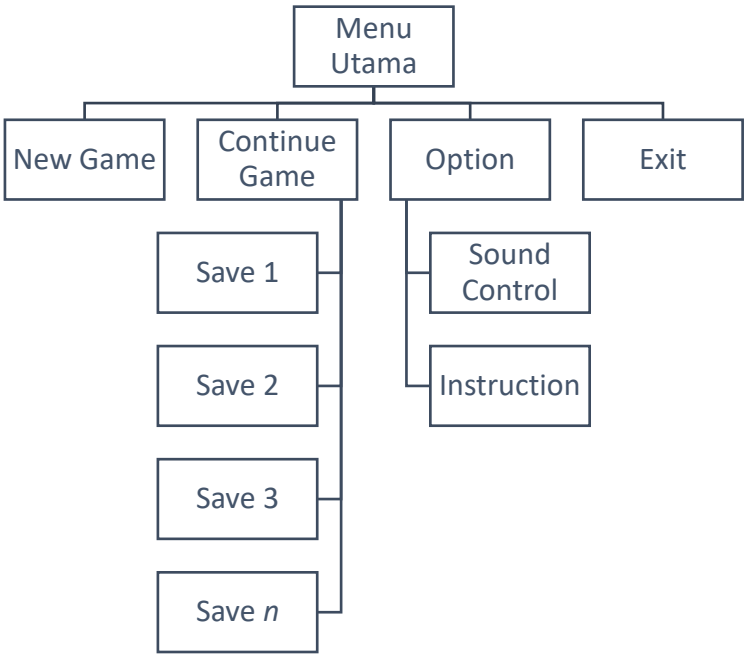
C. Aseprite

Aseprite adalah aplikasi pengedit sprite animasi dan seni piksel yang tersedia untuk Windows, Mac, dan Linux yang dikembangkan oleh Igara Studio. Aplikasi ini menyediakan berbagai alat gambar dan pengolahan piksel yang mempermudah pembuatan seni berbasis piksel.

D. C#

C# (dibaca "C Sharp") adalah bahasa pemrograman berorientasi objek yang dikembangkan oleh Microsoft. Bahasa ini dibangun dengan dasar C++ dan terpengaruh oleh bahasa pemrograman lain seperti Java, Delphi, dan Visual Basic, dengan sejumlah penyederhanaan. Dalam Unity, C# digunakan untuk scripting yang memungkinkan pengelolaan rotasi, skala, duplikasi, penghapusan, dan pengubahan properti objek dengan kode yang relatif singkat dan mudah

Berikut ini adalah diagram yang menggambarkan struktur menu utama dalam sebuah aplikasi atau game. Diagram ini menunjukkan pilihan utama seperti New Game untuk memulai permainan baru, Continue Game yang menyediakan beberapa slot penyimpanan (Save 1, Save 2, Save 3, dan Save n) untuk melanjutkan permainan dari data simpanan, Option yang berisi submenu pengaturan suara (Sound Control) dan instruksi (Instruction), serta Exit untuk keluar dari aplikasi atau game. Struktur ini membantu pengguna memahami dan mengakses berbagai fungsi utama secara mudah dan terorganisir.



Gambar 2.2 Struktur utama game

BAB 3
PENGERTIAN KECERDASAN BUATAN

Ini sudah bab ketiga buku ini, tetapi saya belum mendefinisikan apa yang sedang kita bicarakan. Mari saya coba. AI adalah singkatan dari "kecerdasan buatan", dan karena "buatan" adalah konsep yang cukup lugas, kita hanya perlu mendefinisikan kecerdasan. Pasti ada definisi yang tepat untuk kecerdasan, bukan?

Nah, kabar baiknya adalah banyak orang telah mendefinisikan kecerdasan. Kabar buruknya adalah definisi yang diajukan sangat berbeda satu sama lain dan sama sekali tidak mudah untuk diselaraskan. Bahkan, ada begitu banyak definisi sehingga sulit untuk mendapatkan gambaran umum dari semuanya. Ini memberi tahu kita dua hal: bahwa hakikat kecerdasan merupakan perhatian utama banyak pemikir dan bahwa masih banyak pekerjaan yang harus dilakukan. Dalam buku ini, saya menyajikan dan menggunakan beberapa definisi kecerdasan yang berbeda, khususnya kecerdasan buatan. Kita akan mulai dengan apa yang mungkin merupakan konsepsi kecerdasan buatan yang paling terkenal.

Bayangkan Anda sedang mengobrol daring dengan dua orang. Mungkin Anda menggunakan pesan Facebook, Twitter, Slack, SMS, atau yang lainnya. Jika Anda tidak suka mengobrol kata itu sendiri mungkin menyinggung Anda karena hanya dilakukan oleh generasi milenial bayangkan Anda sedang mengobrol dengan dua orang melalui pesan teks. Anda bahkan mungkin mengetik di selembar kertas dengan mesin tik dan mengirimkannya bolak-balik dalam amplop. Formatnya tidak penting. Yang penting adalah Anda berkomunikasi dengan cara lama yang hanya berupa teks dengan kedua orang tersebut.

Sekarang seseorang memberi tahu Anda bahwa salah satu dari orang-orang ini sebenarnya adalah mesin lebih tepatnya, perangkat lunak AI yang berjalan di komputer. Yang lainnya adalah manusia. Tugas Anda adalah mencari tahu yang mana yang mana atau, jika Anda mau, siapa yang mana. Anda dapat menanyakan apa pun kepada kedua rekan teks Anda, tetapi mereka tidak diharuskan menjawab dengan jujur, terutama jika Anda bertanya apakah mereka komputer.

Tes ini diusulkan pada tahun 1950 oleh Alan Turing, yang kita temui di bab pertama. (Ingatlah, ini terjadi sebelum komputer serba guna yang sebenarnya ada, apalagi Facebook dan pesan teks, jadi Turing berbicara tentang "teleprinter.") Turing menjawab pertanyaan, "Bisakah mesin berpikir?" dan mengusulkan bahwa salah satu cara untuk mengetahuinya adalah dengan melihat apakah komputer itu bisa menang dalam apa yang disebutnya "permainan imitasi" tetapi kemudian disebut "uji Turing".

Jika perangkat lunaknya begitu bagus sehingga Anda tidak dapat membedakan manusia dari komputer, apakah itu berarti komputer itu cerdas? Coba bayangkan situasinya. Jika Anda mau, Anda dapat membayangkan bahwa komputer itu "memenangkan" permainan tidak hanya sekali tetapi berkali-kali. Jika komputer itu bisa mengalahkan Anda, pastilah ia cerdas kecuali jika Anda memiliki pendapat yang sangat rendah tentang kecerdasan Anda sendiri.

Beberapa orang hanya menerima bahwa jika komputer dapat lulus uji Turing, ia akan menjadi cerdas. (Mungkin ia perlu lulus beberapa kali melawan beberapa juri manusia; mungkin para juri perlu dilatih secara khusus.) Yang lain, mungkin sebagian besar, tidak setuju. Menjadi menarik ketika Anda bertanya kepada orang-orang mengapa komputer itu tidak cerdas, meskipun ia lulus uji Turing.

Sayangnya, jawaban yang umum adalah, "Ia tidak mungkin cerdas karena ia komputer." Secara pribadi, saya merasa sulit untuk menjawab keberatan ini tanpa sarkasme. Namun, menyinggung perasaan orang lain bukanlah cara untuk melakukan diskusi yang konstruktif. Jawaban terbaik untuk keberatan "ia hanya komputer" adalah terus bertanya: "Mengapa komputer tidak bisa cerdas, sedangkan manusia bisa?" Beberapa orang mengatakan bahwa kata kecerdasan menurut definisi hanya berlaku untuk manusia. Baiklah. Mari kita cari kata lain yang berarti "kecerdasan", tetapi kata itu tidak terbatas hanya pada manusia. Yang lain menjawab bahwa komputer tidak mungkin cerdas karena terbuat dari komponen silikon seperti transistor, sementara manusia terbuat dari sel-sel biologis yang hidup. Jadi, mengapa memiliki sel-sel biologis diperlukan untuk kecerdasan?

Dan bagaimana Anda tahu? Ada juga yang mengklaim bahwa komputer tidak mungkin cerdas jika diprogram oleh manusia; ia pasti belajar sendiri, mungkin dengan tumbuh bersama manusia. Sekali lagi, bagaimana Anda tahu bahwa kecerdasan tidak dapat diprogram? Sudahkah Anda mencobanya? Dan bagaimana Anda tahu bahwa program komputer khusus ini, yang baru saja menipu Anda hingga mengira ia manusia, tidak tumbuh bersama manusia dan bersekolah dengan anak-anak lain? Yang Anda tahu hanyalah bahwa ia lebih pintar dari Anda.

Ada beberapa keberatan yang bagus juga. Salah satunya adalah bahwa berkomunikasi melalui teks tertulis agak terbatas, dan manusia nyata juga berkomunikasi melalui nada suara, ekspresi wajah, dan gerakan tubuh. Yang lainnya adalah bahwa situasi wawancara semacam ini memang sangat tidak alami, dan tidak benar-benar mewakili beragam aktivitas yang dilakukan manusia setiap hari. Beberapa orang menangani situasi wawancara tertulis dengan sangat buruk tetapi sebaliknya adalah manusia yang sangat kompeten. Sebaliknya, mampu menulis jawaban yang fasih atas pertanyaan tidak menjamin Anda dapat bangun dari tempat tidur, mengikat tali sepatu, memutuskan apa yang ingin Anda makan, menghibur orang yang Anda cintai, atau melukis. Atau bermain gim. Namun semua aktivitas ini tampaknya membutuhkan kecerdasan dalam bentuk tertentu.

3.1 MANUSIA TERMASUK MAKHLUK YANG CERDAS

Seperti yang bisa kita lihat, uji Turing bukannya tanpa masalah. Namun, ide dasar untuk mengambil sesuatu yang dapat dilakukan manusia dan menugaskan komputer untuk melakukan hal yang sama memang menarik. Masuk akal jika komputer benar-benar cerdas, ia seharusnya mampu melakukan semua hal yang dapat dilakukan manusia karena kecerdasannya. Namun, kriteria ini setidaknya mengasumsikan dua hal: bahwa manusia memang cerdas dan bahwa ini adalah satu-satunya (atau yang tertinggi) jenis kecerdasan. Manusia tampaknya secara implisit menjadi tolok ukur kecerdasan, sama seperti untuk hal-hal

lainnya. Jadi, mari kita balik pertanyaan ini dan tanyakan apakah, dari perspektif komputer, manusia cerdas.

Manusia, dibandingkan dengan komputer, akan tampak cukup bodoh dalam banyak hal. Mari kita mulai dengan yang paling jelas: manusia tidak bisa berhitung. Mintalah seorang manusia untuk menaikkan 3.425 pangkat 542 dan saksikan dia duduk di sana selama berjam-jam mencoba menghitungnya. Konyol. Hal yang sama berlaku untuk sejumlah tugas sepele lainnya, seperti menghitung usia rata-rata dalam populasi 300 juta. Seharusnya tidak lebih dari beberapa detik kecuali jika Anda manusia, yang dalam hal ini kemungkinan akan memakan waktu bertahun-tahun, dan bahkan dengan begitu pun Anda akan membuat sejumlah kesalahan.

Manusia juga hampir tidak memiliki ingatan. Mintalah manusia untuk memberi Anda nama dan alamat saat ini yang benar untuk nomor jaminan sosial yang dipilih secara acak (atau nomor registrasi pribadi, atau apa pun yang setara di negara Anda). Sekalipun ia memiliki semua informasi dalam format apa pun yang ia sukai (seperti katalog kertas besar), ia tetap membutuhkan setidaknya beberapa detik dan kebanyakan manusia bahkan tidak akan tahu di mana mendapatkan informasi tersebut.

Atau mintalah manusia untuk memberikan seratus alamat situs web yang membahas tentang kecerdasan buatan, atau bahkan daftar lengkap semua yang terjadi padanya kemarin. Manusia berbicara tentang "ingatan ikan mas", tetapi dari perspektif komputer, manusia dan ikan mas tidak terlalu jauh berbeda, dari segi kemampuan. Pada titik ini, banyak pembaca akan protes keras dan mengatakan bahwa saya sangat tidak adil kepada mereka. Saya hanya memilih tugas-tugas yang dikuasai komputer dan mengabaikan tugas-tugas yang diuntungkan manusia, seperti kontrol motorik dan pengenalan pola.

Benar. Komputer dapat mendaratkan pesawat jet dan menerbangkan helikopter. Bahkan, hampir semua komputer dapat melakukan hal-hal tersebut jika Anda memuat perangkat lunak yang tepat. Sangat sedikit manusia yang tahu cara mendaratkan pesawat jet, dan lebih sedikit lagi yang tahu cara menerbangkan helikopter. Banyak yang memiliki kapasitas untuk "memuat perangkat lunak" (belajar), tetapi ini adalah proses yang memakan waktu bertahun-tahun dan sangat mahal. Terkadang bahkan manusia yang terlatih pun gagal total dalam tugas-tugas ini. (Sulit untuk memahami mengapa ada orang yang ingin berada di pesawat yang diterbangkan oleh manusia sekarang karena ada alternatif.) Komputer dapat mengendarai mobil biasa di jalan raya dan di luar jalan raya, mematuhi semua peraturan lalu lintas. Ada banyak manusia yang bahkan tidak dapat melakukan itu.

Berbicara tentang pengenalan pola, memang benar bahwa manusia dapat mengenali wajah teman-teman mereka dengan akurasi yang cukup tinggi. Namun, manusia hanya memiliki beberapa ratus teman paling banyak. Perangkat lunak pengenal wajah yang digunakan Facebook dapat mengenali wajah ratusan ribu orang. Algoritma pengenalan pola lainnya dapat berhasil mencocokkan hasil pemindaian ibu jari manusia dengan sidik jari kanan dalam basis data yang berisi jutaan orang.

Sekarang mari kita ambil aktivitas lain yang seharusnya dikuasai manusia: bermain gim. Gim diciptakan oleh manusia untuk menghibur diri, dan karena manusia tampaknya merasa

terhibur untuk melatih kemampuan belajar, motorik, dan penalaran mereka, gim seharusnya disesuaikan secara sempurna dengan kecerdasan manusia. Manusia seharusnya unggul dalam bermain gim, bukan? Yah, tidak juga. Seperti yang telah kita lihat, komputer kini sepenuhnya mengungguli manusia dalam hampir semua gim papan klasik. Dan seperti yang akan kita lihat nanti, komputer juga berkinerja sangat baik dalam banyak gim video.

Masih ada gim di mana komputer berkinerja lebih baik, meskipun perkembangan perangkat keras dan perangkat lunak yang lebih baik berarti komputer terus-menerus mengejar ketertinggalan. Anda juga harus ingat bahwa semua gim yang kita bandingkan antara manusia dan komputer dirancang oleh manusia untuk manusia. Oleh karena itu, gim-gim tersebut sangat sesuai dengan kekuatan kognitif manusia. Akan sangat mudah untuk menciptakan gim yang begitu rumit sehingga hanya komputer yang dapat memainkannya. Komputer bahkan dapat menciptakan gim semacam itu sendiri.

Hal-hal lain yang telah disebut sebagai puncak pencapaian manusia adalah mengikat tali sepatu dan reproduksi diri. Namun, mengikat tali sepatu agak sia-sia; teknologi ini mulai usang, bahkan bagi manusia. Mengapa Anda membutuhkan tali sepatu jika Anda robot? Dan manusia tidak benar-benar tahu cara bereproduksi. Mereka tahu cara berhubungan seks, yang merupakan hal yang sangat berbeda dan cukup mudah. Reproduksi yang sebenarnya bergantung pada berbagai proses biokimia yang belum sepenuhnya dipahami manusia dan tidak tahu cara mereplikasinya.

Lalu bagaimana dengan uji Turing? Nah, komputer dapat mendefinisikan uji Turing mereka sendiri. Mereka mungkin akan mendefinisikan antarmuka sehingga alih-alih mengirimkan pesan yang diketik bolak-balik dengan kecepatan santai, proses tersebut akan berlangsung melalui kabel optik 100 megabit per detik. Saya rasa tidak ada manusia yang akan berhasil dalam uji ini. Jadi, dibandingkan dengan manusia, komputer tampaknya memang cukup baik setidaknya jika Anda bertanya kepada komputer. Semuanya tergantung pada apa yang Anda ukur.

Beberapa manusia akan keberatan bahwa perbandingan ini absurd karena manusialah yang membangun dan memprogram komputer. Oleh karena itu, kecerdasan apa pun yang dimiliki komputer seharusnya dikaitkan dengan penciptanya, manusia. Namun, argumen itu berbahaya bagi manusia, karena dalam hal ini, kecerdasan apa pun yang mungkin dimiliki manusia bukanlah benar-benar milik mereka sendiri, melainkan bagian dari proses evolusi melalui seleksi alam yang menciptakan mereka.

3.2 MELAKUKAN APA YANG MEREKA LAKUKAN DI DISCOVERY CHANNEL

Agaknya, beberapa halaman terakhir belum meyakinkan Anda bahwa Anda kurang cerdas dibandingkan komputer. Jelas ada sesuatu yang hilang dari diskusi ini. Pasti ada semacam asumsi tak terucapkan yang, ketika terungkap, meruntuhkan argumen tersebut. Saya setuju. Inilah masalahnya:

Semua contoh yang saya berikan menunjukkan komputer yang baik (dan manusia yang buruk) dalam melakukan tugas-tugas yang sangat spesifik dan memecahkan masalah yang sangat spesifik, padahal ciri kecerdasan sejati adalah mampu bekerja dengan baik dalam

berbagai situasi. Menjadi sangat baik dalam satu hal saja tidak pernah cukup untuk disebut kecerdasan.

Oleh karena itu, manusia memang lebih cerdas daripada komputer: program catur tidak dapat mendaratkan pesawat jet, dan program pengenalan wajah tidak dapat memainkan *Super Mario Bros.* atau melakukan eksponensial. Kecerdasan Anda bergantung pada kemampuan Anda untuk berkinerja baik dalam situasi apa pun yang Anda hadapi, dan manusia sangat pandai beradaptasi dengan beragam situasi dan masalah, sementara program komputer biasanya hanya cocok untuk jenis situasi atau masalah tertentu yang diprogram untuknya. Mari kita mundur selangkah dan pikirkan apa artinya ini dalam beberapa situasi konkret bagi hewan dan robot.

Etologi adalah cabang biologi yang mempelajari perilaku hewan dan mekanisme yang menghasilkan perilaku ini bisa Anda sebut "psikologi hewan". Konsep sentral dalam disiplin ini adalah perilaku adaptif, perilaku yang ditunjukkan hewan sebagai respons terhadap lingkungan tempat ia berevolusi dan yang berfungsi untuk meningkatkan peluangnya untuk bertahan hidup dan memiliki keturunan yang selamat. Mudah dipahami bagaimana rubah beradaptasi untuk bergerak sehingga meminimalkan peluangnya terdeteksi saat mendekati kelinci yang ingin dijadikan santapannya.

Demikian pula, mudah dipahami mengapa kelinci adaptif untuk mengubah arah pada interval yang tidak terduga ketika mencoba melarikan diri dari rubah yang lebih cepat tetapi lebih berat yang tidak ingin menjadi makan malamnya. Yang tidak mudah dipahami adalah rubah dan kelinci mana yang lebih cerdas. Memang, bagi seorang etolog, pertanyaan ini bahkan tidak masuk akal tanpa terlebih dahulu menentukan lingkungan apa dan masalah apa yang dihadapi hewan tersebut.

Sesekali Anda bertemu orang (atau surat kabar tabloid) yang mengklaim bahwa "lumba-lumba benar-benar secerdas manusia" atau "babi lebih cerdas daripada anjing" atau omong kosong serupa. Itu omong kosong bukan karena itu salah tetapi karena tidak masuk akal untuk membuat klaim seperti itu tanpa terlebih dahulu menetapkan lingkungan dan kondisi kehidupan di mana kecerdasan diukur. Letakkan lumba-lumba di kursi kantor, atau manusia di lautan, dan keduanya tidak akan melihat banyak keberhasilan.

Mengutip kata-kata ahli robotika ternama Rodney Brooks, "gajah tidak bermain Catur." Brooks memelopori robotika berbasis perilaku pada tahun 1980an, sebuah pendekatan robotika di mana robot yang sederhana secara komputasional dan mekanis dirancang untuk mengatasi lingkungan tertentu. Misalnya, Brooks mengembangkan serangga mekanis yang mampu mengikuti orang-orang dan menghindari rintangan di dalam ruangan hanya dengan menggunakan beberapa motor murah dan sensor cahaya.

Beberapa robotnya tidak memiliki komputer sama sekali, hanya kabel pintar antara input dan output. Sebaliknya, sebagian besar robot lain pada masa itu menggunakan komputer onboard canggih dan sensor canggih namun menjalankan tugasnya dengan buruk dan sangat sensitif terhadap modifikasi apa pun dari masalah yang ingin mereka selesaikan, seperti bayangan yang sedikit bergeser karena seseorang mengangkat bayangan. Robot yang sangat

canggih dan ambisius gagal dalam tugas-tugas yang sangat sederhana yang dapat diselesaikan dengan baik oleh robot yang lebih sederhana. Dan inilah inti dari apa yang Brooks sampaikan.

Gajah tidak bermain Catur karena mereka tidak perlu. Catur tidak adaptif bagi mereka. Mengapa mereka menyia-nyiakan kapasitas otak mereka yang berharga untuk hal ini, dan mengapa gen gajah menyia-nyiakan ruang untuk mengkode agar mereka dapat belajar bermain Catur? Dengan cara yang sama, Brooks menunjukkan bahwa robotnya dapat mengungguli banyak desain robot yang lebih canggih dengan membuang semua lapisan tambahan "kapasitas pemecahan masalah umum" dan langsung menyelesaikan masalah apa pun yang seharusnya dipecahkan oleh robot tersebut dengan menghubungkan masukan hampir secara langsung ke keluaran dan merancang beberapa aturan sederhana. Tampaknya jauh lebih mudah merancang robot yang benar-benar bekerja seperti itu. Jika Anda pernah bekerja di organisasi besar dengan berbagai lapisan manajemen dan birokrasi, dan mengamati betapa lebih mudahnya Anda menyelesaikan sesuatu jika Anda melewati semua manajemen dan birokrasi itu, Anda mungkin bisa memahaminya.

Di mana letak gagasan kecerdasan sebagai perilaku adaptif dalam kaitannya dengan pertanyaan tentang kecerdasan manusia dan kecerdasan mesin? Satu kemungkinan kesimpulan adalah bahwa sekarang tidak ada artinya membicarakan apakah komputer cerdas "secara umum", sama seperti tidak ada artinya membicarakan apakah hewan cerdas secara umum. Kita hanya bisa membicarakan seberapa cocok program komputer atau hewan dalam memecahkan masalah tertentu atau bertahan hidup di lingkungan tertentu. Namun, ini tentu saja jawaban yang agak membosankan. Jawaban ini juga tidak terlalu berguna, setidaknya tidak bagi para peneliti kecerdasan buatan yang masih ingin berpegang teguh pada gagasan bahwa ada yang namanya "kecerdasan" yang dapat dimiliki lebih banyak atau lebih sedikit oleh perangkat lunak (atau manusia, atau hewan). Bisakah kita berbuat lebih baik? Bisakah kita mempertahankan gagasan tentang perilaku adaptif dan menghasilkan definisi kecerdasan yang lebih baik, dan dengan demikian juga kecerdasan buatan?

3.3 MENJADI KURANG SPESIFIK

Mari kita lihat apakah kita dapat mempertahankan gagasan kecerdasan sambil mengakui bahwa kecerdasan selalu relatif terhadap suatu lingkungan atau tugas. Inilah yang Shane Legg dan Marcus Hutter, di Institut AI Swiss IDSIA tempat saya juga bekerja selama beberapa waktu, coba lakukan dalam sebuah makalah berpengaruh tahun 2007. Gagasan dasar Legg dan Hutter adalah bahwa kecerdasan universal suatu agen (manusia, program komputer, atau yang lainnya) sama dengan kemampuan Anda untuk melakukan tidak hanya satu tugas tetapi banyak tugas bahkan, semua tugas yang mungkin. Namun, tugas yang lebih sederhana lebih penting, dan semakin kompleks suatu tugas, semakin kecil bobotnya dalam penjumlahan akhir.

Hal ini mungkin perlu dijelaskan. Legg dan Hutter mengusulkan sebuah persamaan yang secara teori dapat digunakan untuk menetapkan nilai antara 0 dan 1 kepada agen mana pun (manusia, mesin, atau lainnya), di mana 0 berarti tidak mampu melakukan apa pun yang

berguna dan 1 berarti cerdas secara universal dan sempurna. Kecerdasan universal agen didefinisikan sebagai jumlah kinerjanya atas semua tugas yang mungkin.

Tugas pada dasarnya adalah apa pun yang dapat membuat agen gagal atau berhasil (memprediksi harga saham, mengikat tali sepatu, berteman di pesta). Kinerja agen pada setiap tugas diberi penghargaan berupa angka antara 0 dan 1, di mana 0 adalah kegagalan total dan 1 adalah keberhasilan total. Dengan membaginya dengan jumlah tugas, Anda mendapatkan kinerja rata-rata agen pada semua tugas. Untuk memberikan prioritas lebih pada tugas yang lebih mendasar, tugas-tugas tersebut diberi bobot lebih tinggi dalam perhitungan; pada dasarnya, pentingnya setiap tugas berbanding terbalik dengan deskripsi sesingkat mungkin dari tugas tersebut.

Apakah Anda masih bersama saya? Bagus. Deskripsi saya agak teknis, tetapi ide-ide dasarnya dapat diringkas: (1) kecerdasan dapat diukur sebagai kemampuan Anda untuk memecahkan masalah dan (2) Anda harus mengukur kecerdasan atas semua masalah yang mungkin, tetapi (3) masalah yang lebih sederhana (yang dapat dengan mudah dijelaskan) lebih mendasar dan kemampuan Anda untuk memecahkannya harus lebih diperhitungkan.

Saya pikir ini sangat masuk akal. Mungkin tidak secara akurat menangkap semua arti kata kecerdasan, tetapi saya pikir ini secara akurat menangkap satu arti kecerdasan yang sangat berguna untuk mengembangkan kecerdasan buatan. Anda dapat mendefinisikan pencarian kecerdasan buatan sebagai pencarian agen yang memiliki kecerdasan universal yang semakin tinggi. Namun, ini bukanlah ukuran praktis. Sebenarnya, itu agak meremehkan. Anda tidak dapat menguji kecerdasan universal agen mana pun menggunakan rumus yang diberikan oleh Legg dan Hutter, karena Anda perlu mengujinya pada semua tugas yang mungkin. Tetapi ada banyak tugas yang tak terhingga, dan Anda tidak punya banyak waktu. Selain itu, deskripsi terpendek dari suatu tugas (yang disebut kompleksitas Kolmogorov) tidak dapat dihitung. Anda tidak dapat, bahkan secara teori, yakin bahwa Anda telah menemukan deskripsi terpendek dari suatu tugas. Jadi, untuk benar-benar mengukur kecerdasan suatu program, kita harus mencari sesuatu yang lebih praktis.

3.4 MELAKUKAN LEBIH BAIK DARIPADA MANUSIA

Mendefinisikan kecerdasan dengan cara yang bermanfaat bagi kecerdasan buatan dan sekaligus sesuai dengan gagasan intuitif kita tentang kecerdasan tampaknya jauh dari mudah. Jadi, mungkin kita harus melihat definisi kecerdasan buatan, aktivitas dan teknologi yang biasanya kita rujuk ketika menggunakan istilah tersebut tanpa terlebih dahulu mencoba menetapkan definisi kecerdasan. Mari kita bersikap pragmatis. Berbagai versi definisi berikut telah diajukan oleh berbagai orang: "Kecerdasan buatan adalah upaya untuk membuat komputer mampu melakukan hal-hal yang saat ini dapat dilakukan manusia dengan lebih baik."

Ini adalah deskripsi yang menyegarkan dan tidak membatasi. Jika kita menciptakan perangkat lunak yang memahami ucapan manusia lebih baik daripada kebanyakan manusia, itu adalah kemajuan dalam kecerdasan buatan. Menciptakan perangkat lunak yang dapat melihat rontgen dada manusia, mendiagnosis penyakit, dan mengusulkan pengobatan juga

akan menjadi kemajuan dalam kecerdasan buatan. Mobil tanpa pengemudi yang mematuhi semua peraturan lalu lintas dan menghindari menabrak anak-anak yang tiba-tiba berlari ke jalan? Pasti AI. Dan menciptakan perangkat lunak yang mampu mengalahkan pemain manusia yang kuat dalam gim seperti *Star-Craft* atau *DOTA* tentu akan dianggap sebagai kemajuan dalam AI.

Namun, bukankah definisi ini terlalu luas? Bayangkan Anda menciptakan hati buatan. (Anda akan menjadi kaya!) Membersihkan darah adalah sesuatu yang saat ini tidak dapat kita lakukan dengan baik dengan sistem buatan; sebenarnya, hati adalah satu-satunya perangkat yang dapat melakukannya dengan baik. Itulah mengapa Anda membutuhkan transplantasi hati untuk bertahan hidup jika satu-satunya hati yang ada di tubuh Anda rusak. Namun, rasanya sangat aneh mengatakan bahwa menciptakan hati buatan akan mewakili kemajuan dalam kecerdasan buatan. Ini lebih seperti memecahkan masalah kimia, bukan?

Orang bisa berargumen bahwa untuk menjadi kecerdasan buatan, teknologinya harus mampu melakukan sesuatu yang dapat dilakukan manusia dengan lebih baik secara sadar. Saya tidak tahu tentang Anda, tetapi saya jelas tidak menyadari apa yang sedang dilakukan hati saya saat ini. Saya juga tidak sadar bagaimana saya memahami bahasa lisan, dan saya hanya sebagian sadar akan strategi yang saya gunakan saat bermain Catur atau gim aksi petualangan *Bloodborne.*

Masalah lain, atau mungkin fitur, dengan definisi ini adalah ia mencakup AI yang sempit. Sangat mungkin membayangkan sebuah sistem yang melaju dengan sempurna di lalu lintas kota atau sistem yang memberikan diagnosis penyakit dada yang lebih baik daripada dokter mana pun, tetapi tidak membuat kemajuan apa pun menuju AI yang lebih umum, tidak ada kemajuan menuju sesuatu yang, misalnya, akan lulus uji Turing.

Perbedaan antara AI sempit dan AI umum (atau kecerdasan umum buatan AGI, sebagaimana beberapa orang menyebutnya) penting karena alasan lain. Terkadang Anda mungkin mendengar orang berkata bahwa "AI telah gagal." Para peneliti telah meneliti AI sejak tahun 1950an, tetapi masih belum ada Robocop, HAL, atau Wall-E atau bahkan sesuatu yang dapat lulus uji Turing. Dari perspektif AI umum, memang benar bahwa kita belum menghasilkan AI. Namun, dibutuhkan waktu lebih dari lima puluh tahun sejak penemuan layang-layang kertas hingga Wright bersaudara membangun mesin terbang bertenaga sendiri pertama, termasuk ratusan tahun pengembangan teknis roda, mesin, teori, dan material. Dan bisa dibilang, telah terjadi banyak perkembangan teknis dalam AI sejak tahun 1950an.

Dari perspektif AI yang sempit, klaim bahwa AI telah gagal sepenuhnya salah. Sebagian besar teknologi yang Anda gunakan sehari-hari dan yang menjadi dasar masyarakat kita berawal dari penelitian AI. Perangkat lunak pengenal gambar di kamera ponsel Anda yang membantu Anda mengambil foto yang lebih baik, algoritma pemrosesan suara di perangkat lunak asisten pribadi Anda, navigator GPS Anda yang menemukan rute terpendek ke tempat konser, dan tentu saja saran-saran menyeramkan dari Facebook tentang siapa yang harus Anda tambahkan sebagai teman: semuanya adalah hasil penelitian AI.

Faktanya, gaya pemrograman berorientasi objek yang digunakan sebagian besar perangkat lunak Anda, dan model basis data relasional yang digunakan hampir setiap situs

web, juga berawal dari penelitian tentang cara membuat mesin benar-benar cerdas. Dapat dikatakan bahwa mereproduksi kecerdasan merupakan salah satu kekuatan pendorong bagi para penemu awal komputer. Namun, tampaknya segera setelah penelitian AI menghasilkan sesuatu yang benar-benar berfungsi dan bermanfaat, penelitian tersebut dipisahkan menjadi bidang penelitiannya sendiri dan tidak lagi disebut kecerdasan buatan. Dari perspektif ini, akan sedikit kurang ajar untuk mendefinisikan kecerdasan buatan sebagai teknologi komputer ambisius yang belum sepenuhnya berfungsi.

Anda mungkin dimaafkan jika kehabisan kesabaran saat ini. Saya telah menghabiskan seluruh bab ini berpindah-pindah dari satu definisi kecerdasan dan kecerdasan buatan ke definisi lainnya, tampaknya menemukan kekurangan di masing-masing definisi. Saya mulai dengan menggambarkan uji Turing sebagai sebuah uji dan secara implisit merupakan definisi kecerdasan buatan, tetapi menyimpulkan bahwa uji tersebut tidak menguji banyak hal yang dilakukan manusia normal dan yang tampaknya membutuhkan kecerdasan (memasak, mengikat tali sepatu, tersenyum penuh arti), dan dengan demikian, makhluk yang agak tidak cerdas pun mungkin lulus uji tersebut.

Selain itu, uji tersebut sangat bergantung pada interogator manusia tertentu; beberapa manusia mungkin gagal mengenali AI yang jelas, dan kita tidak ingin definisi apakah suatu mesin benar-benar cerdas bergantung pada penilaian manusia yang lemah. Selanjutnya, kami membahas gagasan kecerdasan sebagai perilaku adaptif, di mana kecerdasan akan menjadi sesuatu yang sama sekali berbeda tergantung pada lingkungan tempat agen (ahli bedah, ikan sturgeon, robot penyedot debu) tinggal.

Namun, hal ini seolah menghindari pertanyaan dan tidak memungkinkan kita untuk mengatakan bahwa satu agen lebih cerdas daripada yang lain. Jadi, kami kemudian mempertimbangkan gagasan bahwa kecerdasan universal adalah kinerja rata-rata agen pada semua kemungkinan masalah, yang dibobot oleh kesederhanaan masalah tersebut. Hal ini masuk akal secara teoretis tetapi mustahil diukur dalam praktik. Akhirnya, kami membahas gagasan bahwa AI hanyalah tentang mencoba menciptakan perangkat lunak (dan terkadang perangkat keras) yang mencoba melakukan hal-hal yang saat ini dapat dilakukan manusia lebih baik daripada komputer.

Kenyataannya, tidak ada definisi yang disepakati bersama untuk kedua konsep ini, dan bahkan para ahli sering membicarakan kecerdasan dan kecerdasan buatan dengan makna implisit yang berbeda, tergantung konteksnya. Kita harus menerimanya saja. Jadi, di sisa buku ini, saya akan menggunakan kecerdasan buatan untuk mengartikan salah satu hal berikut, tergantung apa yang saya bicarakan:

1. Upaya membangun mesin cerdas, untuk beberapa definisi kecerdasan.
2. Apa pun yang dilakukan orang yang menyebut diri mereka peneliti kecerdasan buatan.
3. Seperangkat algoritma dan ide yang dikembangkan oleh para peneliti kecerdasan buatan. Algoritma minimax dan MCTS dari bab 2 adalah contoh bagus dari algoritma AI, dan saya akan menyajikan lebih banyak algoritma semacam itu di bab-bab mendatang.

Terakhir, apa yang dipikirkan Alan Turing penemu tes Turing dan bisa dibilang orang pertama yang mengajukan beberapa masalah utama dalam AI? Nah, bertentangan dengan apa yang diyakini banyak orang, Turing tidak mengusulkan apa yang sekarang dikenal sebagai tes Turing sebagai definisi kecerdasan buatan; alih-alih, ia mengusulkannya untuk menunjukkan bahwa seluruh konsep kecerdasan kita cacat dan tidak ada gunanya berdebat tentang apakah suatu mesin itu cerdas. Turing berpikir bahwa kita pada akhirnya akan mengembangkan perangkat lunak yang akan lulus uji yang ia ciptakan, tetapi "pertanyaan awal, 'Bisakah mesin berpikir?', saya yakin terlalu tidak berarti untuk dibahas."

3.5 AI MENINGKATKAN INDUSTRI GAME

Industri game telah mengalami perkembangan yang sangat cepat selama beberapa dekade terakhir, dengan teknologi kecerdasan buatan (*Artificial Intelligence*/AI) sebagai salah satu faktor utama yang mendorong inovasi. AI dalam game memungkinkan terciptanya pengalaman bermain yang lebih mendalam dan dinamis, mulai dari pengembangan karakter *Non-Playable Characters* (NPC) hingga mekanisme gameplay yang adaptif. Teknologi ini tidak hanya memperkaya pengalaman pengguna, tetapi juga merubah cara perancangan dan cara bermain game, sehingga menghasilkan dunia virtual yang lebih realistis dan interaktif.

Artificial Intelligence (AI) telah membawa perubahan signifikan pada berbagai aspek dalam pengembangan game modern, mulai dari desain hingga mekanisme permainan. Salah satu penerapan utama AI dalam industri game adalah menciptakan karakter *Non-Playable Characters* (NPC) yang lebih cerdas dan hidup. Dengan AI, NPC mampu merespons tindakan pemain secara alami dan intuitif, memberikan kesan seolah-olah karakter-karakter ini memiliki kesadaran dan pemikiran sendiri. Selain itu, AI juga berperan penting dalam penciptaan dunia game dan level yang lebih imersif. Teknik seperti procedural generation memungkinkan AI membuat peta atau lingkungan game secara otomatis menggunakan algoritma tertentu.

AI juga mendukung gameplay adaptif, di mana tingkat kesulitan bisa disesuaikan secara dinamis berdasarkan kemampuan pemain. Ini membuat pengalaman bermain tetap menantang dan menarik tanpa membuat pemain merasa frustrasi. Peran AI dalam game tidak hanya meningkatkan realisme dan interaktivitas, tetapi juga secara keseluruhan mengubah cara game dirancang dan dijalankan, memberikan pengalaman yang lebih hidup dan personal bagi pemain. AI memungkinkan game menyesuaikan tingkat kesulitan sesuai dengan kemampuan pemain. Selain itu, AI juga dimanfaatkan dalam proses pengembangan game untuk melakukan pengujian secara otomatis, yang mencakup pemeriksaan bug, glitch, dan masalah performa dengan kecepatan dan efisiensi yang lebih tinggi dibandingkan pengujian manual oleh manusia.

Kecerdasan buatan diperkirakan akan terus memegang peranan penting dalam perkembangan industri game ke depannya, dengan sejumlah tren yang diprediksi semakin berkembang. Salah satunya adalah pembuatan konten game secara otomatis oleh AI, mulai dari misi hingga peta dan karakter, yang memungkinkan pengembang menghasilkan variasi konten tanpa batas sehingga meningkatkan nilai replayability game. Contohnya, AI dapat

digunakan untuk menciptakan dungeon yang berbeda setiap kali pemain memulai permainan baru dalam genre RPG.

Selain itu, kemajuan dalam machine learning dan reinforcement learning memungkinkan NPC (*Non-Playable Characters*) untuk terus belajar dan beradaptasi selama permainan berlangsung, membuat mereka semakin cerdas dalam memahami strategi pemain dan memberikan tantangan yang lebih dinamis dan bervariasi. Integrasi AI dengan teknologi Realitas Virtual (VR) dan *Augmented Reality* (AR) juga membuka peluang untuk menghadirkan pengalaman bermain yang lebih realistis dan imersif, dengan interaksi yang lebih natural antara pemain dan lingkungan virtual.

Dalam hal pengembangan cerita, AI dapat menciptakan alur narasi yang berkembang dan berubah berdasarkan keputusan pemain, sehingga setiap pengalaman bermain menjadi unik dan personal. AI juga akan berkembang sebagai rekan bermain dalam game multiplayer, bertugas sebagai teman atau lawan yang mampu berkolaborasi atau bersaing dengan pemain secara adaptif dan menantang.

Secara keseluruhan, perkembangan AI dalam industri game telah membawa dampak besar, dari pengembangan NPC yang lebih pintar hingga gameplay yang lebih personal dan adaptif. Ke depannya, AI akan terus mendorong batas kreativitas dan interaktivitas dalam dunia game, memungkinkan terciptanya pengalaman bermain yang lebih mendalam dan dinamis serta dunia virtual yang hidup dan penuh kemungkinan. Pengembang game akan semakin kreatif dalam memanfaatkan teknologi ini untuk menghadirkan inovasi tiada henti dalam industri game.

BAB 4
DALAM GIM VIDEO TERDAPAT KECERDASAN BUATAN

Mungkin sebagian besar, gim video menampilkan apa yang disebut karakter non-pemain (NPC). Ini bisa berupa musuh, sekutu, penonton, atau apa pun. Intinya adalah mereka dikendalikan bukan oleh pemain (Anda) tetapi oleh komputer. Biasanya orang menyebut cara NPC ini berperilaku sebagai "AI" dalam gim. Karena kita telah menetapkan bahwa ada banyak pandangan berbeda tentang apa itu kecerdasan buatan, mari kita terima saja julukan itu untuk apa pun yang mengendalikan NPC dalam gim video. Tetapi bagaimana tepatnya AI dalam gim video pada umumnya bekerja? Simak sedikit dramatisasinya.

4.1 TUJUH DETIK DALAM KEHIDUPAN MUSUH 362

Musuh 362 muncul 43 menit setelah sesi permainan dimulai. Gim ini telah memunculkan 361 musuh dalam sesi permainan ini; pemain telah membunuh 143 musuh, dan yang lainnya telah mati begitu saja ketika pemain meninggalkan zona permainan tempat mereka berada. Pemain telah berhasil mencapai level ketiga dari *first-person shooter* (FPS) generik ini (saya menganggapnya seperti game *Call of Duty*, tetapi bisa juga seperti *Gears of War*, atau *Half-Life*), dan karakternya kini sendirian menyerang tempat persembunyian seorang teroris internasional yang terkenal kejam (lihat gambar 4.1).

Gambar 4.1

Game tembak-menembak orang pertama disebut demikian karena Anda melihat dunia dari perspektif orang pertama dan, yah, menembaki berbagai hal. *Call of Duty*: *Modern Warfare 2* (Infinity Ward, 2009) adalah representasi yang baik dari genre ini.

Musuh 362, yang tampak seperti teroris kelas bawah pada umumnya dengan seragam tempur compang-camping, syal hitam menutupi separuh wajah bagian bawah, dan senapan serbu Kalashnikov, ditugaskan untuk gagal melindungi bos teroris di akhir level. Kecuali jika pemain benar-benar membuat kesalahan, tentu saja.

Seperti biasa di setiap sesi permainan musuh 362 muncul di tempat yang sama, di sebelah gubuk yang tampak terbengkalai, segera setelah pemain melewati pos pemeriksaan ketiga di level tersebut. Ketika musuh 362 muncul, pikirannya berada dalam keadaan 0. Beginilah rupa pikiran musuh 362:

- *Tahap 0: Berjaga*. Berjalanlah perlahan bolak-balik antara gubuk terbengkalai tempat ia muncul dan pohon palem, sambil melihat ke depan dan ke belakang. Jika karakter pemain muncul dalam jangkauan penglihatan, lanjutkan ke keadaan 1.
- *Tahap 1: Berlindung*. Lari secepat mungkin ke titik perlindungan terdekat. Tumpukan karung pasir ditempatkan dengan tepat di antara gubuk terbengkalai dan pohon palem. Setelah perlindungan tercapai, lanjutkan ke keadaan 2.
- *Tahap 2: Tetaplah berlindung*. Tetaplah berjongkok di balik perlindungan agar sekeras mungkin untuk dipukul. Atur timer untuk durasi acak antara 1 dan 3 detik. Setelah waktu tersebut berlalu, lanjutkan ke tahap 3. Jika pada suatu titik karakter pemain melampaui titik perlindungan, lanjutkan ke tahap 4.
- *Tahap 3: Tembak dari tempat perlindungan*. Berdiri di balik tempat perlindungan dan tembak karakter pemain, dengan deviasi acak 5 derajat agar tidak terlalu sering mengenai sasaran. Atur timer selama 1 atau 2 detik. Setelah waktu tersebut berlalu, lanjutkan ke tahap 3. Jika pada suatu titik karakter pemain melampaui titik perlindungan, lanjutkan ke tahap 4.
- *Tahap 4: Serang pemain*. Lari lurus ke arah pemain di sepanjang jalur terpendek, sambil terus menembak pemain.
- *Tahap 5: Mati*. Jika pada suatu titik kesehatan berkurang menjadi 0, jatuhlah ke tanah dan jangan melakukan apa pun lagi.

Arsitektur pikiran musuh 362 disebut mesin keadaan terbatas. Hal ini karena ia diorganisasikan sebagai sejumlah keadaan terbatas, di mana setiap keadaan berisi instruksi tentang bagaimana berperilaku dalam keadaan tersebut. Kebetulan, semua NPC dalam game ini memiliki arsitektur yang sama, tetapi jenis musuh yang berbeda memiliki status yang berbeda pula.

Pada status 1 dan 4, musuh 362 berlari menuju suatu posisi. Hal ini dicapai dengan menggunakan algoritma A^*, yang merupakan algoritma pencarian jalur. Dengan kata lain, ini adalah metode untuk menemukan jalur terpendek dari titik A (misalnya, tempat karakter berdiri) ke titik B (misalnya, di balik karung pasir). A^* bekerja sebagai berikut:

1. Mulai dari titik A, posisi awal, dan pilih ini sebagai posisi aktif.
2. Lihat semua posisi di sebelah posisi aktif dan cari tahu mana yang memungkinkan untuk dituju (misalnya, tidak berada di dalam dinding). Dalam contoh ini, algoritma mungkin melihat delapan titik dalam lingkaran berdiameter setengah meter di sekitar posisi aktif.
3. Posisi-posisi yang memungkinkan untuk dituju ditambahkan ke daftar posisi yang tersedia, yang diurutkan berdasarkan jaraknya ke tujuan (titik B) sepanjang garis lurus.
4. Pilih posisi yang paling dekat dengan tujuan dari daftar ini, dan hapus dari daftar. (Titik-titik lainnya tetap ada dalam daftar.) Tandai titik ini sebagai posisi aktif. Lanjutkan ke langkah 2.

Pada dasarnya, algoritma melacak sejumlah besar posisi, dan terus-menerus mengeksplorasi posisi yang paling menjanjikan. Menjalankan proses ini akan selalu menghasilkan pencarian jalur terpendek antara titik A dan titik B, dan biasanya akan menemukannya dengan cukup cepat, jauh lebih cepat daripada jika telah menyelidiki semua kemungkinan posisi di area tersebut. (Ada beberapa kerumitan lebih lanjut pada algoritme ini, tetapi hal-hal ini tidak perlu dibahas untuk memberi Anda gambaran umum.)

Arsitektur mesin berstatus terbatas dan algoritme A^* memainkan peran sentral dalam sebagian besar gim dan juga digunakan secara luas dalam robotika dan mobil self-driving. Banyak gim menggunakan algoritme tambahan selain ini untuk mengontrol perilaku NPC, dan beberapa tidak menggunakan teknik ini (dalam beberapa tahun terakhir, alternatif untuk mesin berstatus terbatas, yang disebut pohon perilaku, telah menjadi populer). Namun, dapat dikatakan bahwa mesin berstatus terbatas dan A^* adalah beberapa algoritme yang paling umum untuk mengimplementasikan perilaku NPC dalam gim video komersial.

Jadi, mari kita kembali ke musuh 362. Setelah tiba-tiba menemukan dirinya di dunia, ia dengan patuh mulai berjalan dari gubuk ke pohon palem. Namun, ia baru sampai setengah jalan sebelum melihat karakter pemain bergerak ke arahnya. Ia masuk ke keadaan 1 hanya sepersekian detik karena sudah berada di dekat karung pasir. Ia masuk ke keadaan 2 selama beberapa detik lalu ke keadaan 3, berdiri dan menembak langsung ke arah karakter pemain. Namun, karakter pemain telah bersembunyi di balik perlindungannya sendiri dan tidak terkena. Musuh 362 kembali ke keadaan 1 sementara karakter pemain melemparkan granat. Kekuatan ledakan langsung menghabiskan semua kesehatan, menyebabkan transisi cepat ke keadaan 5.

Musuh 362 tidak memiliki nama dan dengan cepat dilupakan oleh pemain saat ia maju lebih jauh. Biasanya tidak akan ada orang yang menulis biografi musuh 362, karena sebenarnya tidak banyak yang perlu diingat. Sisi sebaliknya adalah tak seorang pun akan merasa bersalah karena menyingkirkan musuh 362 begitu cepat. Lagipula, tidak ada pikiran yang benar-benar perlu diberantas.

Apakah Hanya Ini yang Ada?

Anda mungkin menyadari bahwa setelah Wizard of Oz dibuka, yang ada hanyalah asap dan cermin beberapa asap yang cukup mengesankan dan cermin yang dipoles dengan baik, tetapi tetap saja. Tentu saja, implementasi kontrol NPC yang sebenarnya dalam permainan apa pun jauh lebih kompleks daripada yang telah saya jelaskan, tetapi prinsipnya sangat mirip. Anda mungkin juga memperhatikan semua yang hilang. AI yang mengendalikan musuh 362 bukanlah pikiran yang lengkap. Ia tidak dapat melakukan apa pun selain apa yang tercatat dalam lima keadaan tersebut. Jika Anda bersembunyi di balik dinding selama satu jam, musuh 362 akan terus bertransisi antara keadaan 2 dan 3 sampai Anda keluar. Ia tidak akan memutuskan bahwa ia sudah cukup dan langsung menyerang Anda dari samping atau meminta bantuan teman-temannya.

Memang benar ada contoh NPC yang lebih menarik di beberapa gim yang sudah ada, bahkan di antara gim tembak-menembak orang pertama. Misalnya, gim tembak-menembak bertema horor $F.E.A.R.$ memperkenalkan penggunaan algoritma perencanaan dalam gim

aksi modern. Dengan perencanaan, musuh dapat mengoordinasikan serangan mereka dan melakukan hal-hal seperti mengapit pemain; pemain juga dapat menguping obrolan antar musuh untuk mencoba menebak rencana mereka.

Seri gim tembak-menembak orang pertama *Halo* juga telah menunjukkan bagaimana perilaku NPC yang lebih menarik dapat diimplementasikan; misalnya, musuh sering bergerak dalam regu, beberapa musuh mundur ketika yang lain terbunuh, dan beberapa musuh akan mencoba menebak di mana Anda akan muncul jika Anda mencoba bersembunyi dari mereka. Contoh yang lebih baru adalah *Shadow of Mordor*, sebuah gim di mana NPC mengingat pertemuan mereka dengan Anda dan merujuknya kembali dalam pertarungan selanjutnya.

Namun, contoh-contoh ini cukup mutakhir setidaknya untuk jenis gim ini dan setiap kemajuan dapat digambarkan sebagai trik yang sangat spesifik, alih-alih kemajuan dalam AI untuk keperluan umum. Layaknya musuh fiktif kita, 362, dari gim tembak-menembak orang pertama generik fiktif, NPC dalam gim yang paling canggih sekalipun terbatas dalam bentuk perilaku yang dapat mereka ekspresikan dan bentuk interaksi yang dapat mereka pahami. Berikut adalah daftar sebagian hal yang tidak dapat dilakukan musuh 362:

- Menyadari bahwa Anda bersembunyi di balik tembok selama satu jam, alih-alih menyerangnya, sehingga mempertimbangkan opsi alternatif, seperti mengapit Anda.
- Melempar kerikil ke arah Anda hingga Anda bergerak dari balik tembok itu.
- Memanggil bantuan.
- Merasa takut.
- Melakukan percakapan filosofis dengan Anda, diteriakkan dari balik tembok itu, tentang arti perang dan mengapa Anda dan perang saling berperang.
- Mengusulkan, dan bermain, permainan Catur yang menyenangkan dengan Anda.
- Mengikat tali sepatunya.
- Membuat secangkir kopi yang nikmat.

Tentu saja, sangat mungkin untuk menulis kode yang memungkinkan musuh 362 melakukan semua hal ini kecuali, mungkin, merasa takut. Memang, beberapa gim menyertakan NPC yang mengapit Anda, melempar kerikil ke arah Anda, melakukan percakapan filosofis (terprogram) dengan Anda, dan sebagainya.

Setiap kemampuan ini harus dibangun secara khusus oleh perancang manusia. Seseorang harus secara khusus menulis kode program yang memungkinkan musuh 362 melempar kerikil (mungkin menambahkan beberapa status ke mesin status terbatas dan algoritma untuk menentukan di mana harus melempar kerikil), atau menulis baris-baris dalam diskusi filosofis yang dapat Anda pilih untuk diikuti, lengkap dengan antarmuka tempat Anda dapat memilih respons.

Agar musuh 362 dapat menyimpan AK47 dan mengeluarkan papan Catur untuk bermain dengan Anda, pengembang game perlu menerapkan algoritma minimax untuk bertindak sebagai otak Catur musuh 362 dan, tentu saja, elemen grafis dan antarmuka untuk memungkinkan bermain Catur. Tak satu pun dari kemampuan ini akan muncul secara ajaib dari AI musuh 362 karena, seperti yang telah kita lihat, "otaknya" hanyalah mesin berstatus terbatas dan algoritma pencarian jalur.

Pada titik ini, saya ingin membandingkan apa yang baru saja saya ceritakan dengan imajinasi liar saya ketika berusia sebelas tahun dan bermain gim di Commodore 64 saya, seperti yang saya ceritakan di prolog. Saya terus berfantasi tentang apa yang akan terjadi jika saya memainkan gim dengan cara yang tidak saya sadari: berlayar melampaui batas peta di *Pirates!*, mengendalikan (atau sekadar berbicara dengan) orang-orang dalam gim strategi seperti *Civilization*, atau memasukkan karakter favorit saya dari gim lain ke dalam *Bubble Bobble*. Intinya, saya berfantasi bahwa gim itu tak terbatas dan memiliki ruang untuk kemungkinan yang tak terbatas.

Cara lain untuk melihat ini adalah saya membayangkan bahwa berinteraksi dengan gim dapat memiliki ruang kemungkinan yang luar biasa seperti berinteraksi dengan manusia, bahkan kucing, atau anjing. Anda sedang membaca buku ini sekarang dan memikirkan hal-hal yang belum pernah Anda pikirkan sebelumnya, mengikuti argumen saya atau mungkin merumuskan argumen balasan Anda sendiri. Reaksi Anda kemungkinan akan mengejutkan saya, atau setidaknya saya tidak akan dapat memprediksinya.

Mungkinkah gim tidak sama? Anda mungkin berharap bahwa saya seorang dewasa dan profesor yang telah menerbitkan ratusan artikel tentang kecerdasan buatan, khususnya tentang kecerdasan buatan dan gim telah mengatasi fantasi masa kecil ini dan mengadopsi pandangan yang lebih bijaksana. Ternyata tidak. Skenario fantasi yang terlalu mengada-ada diperlukan untuk kemajuan ilmiah. Jadi, izinkan saya memberikan satu visi tentang bagaimana jadinya jika ada gim yang dibangun lebih menyeluruh berdasarkan metode AI.

4.2 VIDEO GAME MEMILIKI AI YANG SESUNGGUHNYA

Mari kita melangkah ke masa depan dan berasumsi bahwa berbagai teknik AI yang sedang kita kembangkan saat ini telah mencapai kesempurnaan dan kita dapat membuat game yang menggunakannya. Dengan kata lain, mari kita bayangkan seperti apa game jika kita memiliki AI yang cukup baik untuk apa pun yang ingin kita lakukan dengan AI dalam game. Bayangkan Anda sedang memainkan game masa depan.

Anda sedang memainkan game dunia terbuka dengan kata lain, game di mana Anda menjelajahi ruang yang relatif terbuka dan mengejar tujuan game dalam urutan apa pun yang Anda pilih. (Contoh seri game dunia terbuka yang populer antara lain *Grand Theft Auto*, *The Elder Scrolls,* dan *The Legend of Zelda*). Dalam game dunia terbuka hipotetis masa depan ini, Anda memutuskan bahwa alih-alih langsung menuju tujuan misi berikutnya di kota tempat Anda berada, Anda ingin berkendara (atau berkuda) selama lima jam ke arah yang dipilih secara acak. Barat, mungkin. Game ini membentuk lanskap seiring Anda bermain, dan Anda berakhir di kota baru yang belum pernah dikunjungi pemain manusia sebelumnya.

Di kota ini, kamu bisa memasuki rumah mana pun (meskipun mungkin harus membuka beberapa kunci), mengobrol dengan semua orang yang kamu temui, terlibat dalam serangkaian intrik yang benar-benar baru, dan menjalankan misi-misi baru. Jika kamu pergi ke arah yang berbeda, kamu akan tiba di kota yang berbeda dengan arsitektur yang berbeda, orang-orang yang berbeda, dan misi yang berbeda atau hutan yang luas dengan hewan-hewan

dan pertapa yang realistis, atau laboratorium penelitian rahasia, atau apa pun yang dihasilkan oleh mesin gim ini.

Berbicara dengan orang-orang yang Anda temui di kota baru semudah berbicara ke layar. Karakter-karakter tersebut merespons Anda dalam bahasa alami yang memperhitungkan apa yang baru saja Anda katakan. Kalimat-kalimat ini tidak dibaca oleh aktor, melainkan dihasilkan secara langsung oleh permainan. Anda juga dapat berkomunikasi dengan permainan melalui lambaian tangan, tarian, ekspresi wajah, atau cara-cara eksotis lainnya. Tentu saja, dalam banyak (kebanyakan?) kasus, Anda masih menekan tombol pada kibor atau pengontrol karena itu seringkali merupakan cara paling efisien untuk memberi tahu permainan apa yang ingin Anda lakukan.

Mungkin tidak perlu dikatakan, tetapi semua NPC bernavigasi dan umumnya berperilaku dengan cara yang benar-benar meyakinkan. Misalnya, mereka tidak akan terjebak menabrak tembok atau mengulang kalimat yang sama berulang-ulang (yah, tidak lebih dari manusia biasa). Ini juga berarti Anda memiliki musuh dan kolaborator yang menarik untuk bermain game apa pun tanpa harus menunggu teman-teman Anda online atau harus berhadapan dengan anak-anak berusia tiga belas tahun yang menyebalkan.

Dalam permainan dunia terbuka, terdapat permainan lain yang dapat dimainkan, misalnya dengan mengakses konsol permainan virtual di dalam dunia permainan atau menawarkan untuk bermain dengan NPC. NPC ini mampu memainkan berbagai subpermainan pada tingkat kemahiran apa pun yang sesuai dengan fiksi permainan, dan mereka bermain dengan gaya bermain layaknya manusia. Permainan inti juga dapat dimainkan pada resolusi yang berbeda, misalnya, sebagai permainan manajemen atau sebagai permainan yang melibatkan kendali bagian tubuh individu, dengan memperbesar atau memperkecil tampilan. Aturan, mekanisme, dan konten apa pun yang diperlukan untuk memainkan subpermainan atau permainan turunan ini diciptakan oleh mesin permainan saat itu juga. Semua permainan ini dapat diangkat dari permainan utama dan dimainkan secara terpisah.

Permainan ini merasakan perasaan Anda saat bermain dan menentukan aspek mana yang Anda kuasai, serta bagian mana yang Anda sukai (dan, sebaliknya, bagian mana yang Anda kurang kuasai dan benci). Berdasarkan hal ini, gim ini terus beradaptasi agar lebih sesuai dengan keinginan Anda, misalnya, dengan memberi Anda lebih banyak cerita, tantangan, dan pengalaman yang akan Anda sukai di kota baru yang Anda capai dengan berkendara selama lima jam ke arah yang dipilih secara acak, mungkin dengan mengubah aturannya sendiri. Gim ini bukan hanya memberi Anda lebih banyak hal yang sudah Anda sukai dan kuasai. Lebih canggihnya lagi, gim ini memodelkan apa yang Anda sukai di masa lalu dan menciptakan konten baru yang merespons keterampilan dan preferensi Anda yang terus berkembang seiring permainan.

Meskipun gim yang Anda mainkan tidak ada habisnya, memiliki resolusi tak terbatas, dan terus beradaptasi dengan selera dan kemampuan Anda yang berubah, Anda mungkin masih ingin memainkan sesuatu yang lain di beberapa titik. Jadi mengapa tidak merancang dan membuat gim Anda sendiri? Mungkin karena sulit dan membutuhkan banyak pekerjaan? Tentu, memang benar bahwa pada tahun 2018, dibutuhkan ratusan orang yang bekerja selama

bertahun-tahun untuk membuat gim yang terkenal dan segelintir profesional yang sangat terampil untuk membuat gim yang terkenal sama sekali.

Tetapi sekarang karena ini adalah masa depan dan kita memiliki AI yang canggih, ini dapat digunakan tidak hanya di dalam gim tetapi juga dalam proses desain dan pengembangan gim, jadi Anda cukup mengganti mesin gim ke mode edit dan mulai membuat sketsa ide gim sedikit alur cerita di sini, karakter di sana, beberapa mekanik di sini, dan satu set piece di atasnya. Mesin gim segera mengisi bagian yang hilang dan memberi Anda gim yang lengkap dan dapat dimainkan. Beberapa di antaranya adalah saran. Jika Anda telah membuat sketsa ekonomi dalam gim, tetapi ekonomi tersebut tidak seimbang dan akan menyebabkan inflasi yang cepat, mesin gim akan menyarankan Anda untuk membuang uang.

Jika Anda telah merancang celah yang tidak dapat dilompati oleh karakter pemain, mesin gim akan menyarankan perubahan pada celah atau mekanisme lompatan. Anda dapat melanjutkan membuat sketsa, dan mesin gim akan mengubah sketsa Anda menjadi detail, atau langsung mulai memodifikasi detail gim.

Apa pun yang Anda lakukan, mesin gim akan bekerja sama dengan Anda untuk mengembangkan ide-ide Anda menjadi gim yang lengkap dengan seni, level, dan karakter. Kapan pun, Anda dapat langsung bermain. Anda juga dapat menyaksikan sejumlah pemain buatan memainkan berbagai bagian gim, termasuk pemain yang bermain seperti Anda memainkan gim tersebut atau seperti teman-teman Anda (dengan selera dan keterampilan yang berbeda).

Mengapa Masa Depan Belum Tiba?

Mengapa kita belum memiliki sesuatu seperti yang baru saja saya jelaskan? Karena kita belum memiliki teknologinya dan praktik desain dan pengembangan game belum begitu baik dalam mengintegrasikan teknologi AI yang kita miliki. Mari kita mulai dengan alasan kedua. Kecerdasan buatan sedang menjadi perbincangan hangat akhir-akhir ini, dan kemajuan dalam metode AI dipublikasikan hampir setiap hari. Namun, industri gim tampaknya kurang tertarik untuk menggabungkan sebagian besar teknik AI ke dalam gim mereka. Banyak peneliti AI akademis telah mengusulkan algoritma AI baru untuk gim dan dengan antusias mempresentasikannya kepada pengembang gim, hanya untuk melihat pengembang gim tersebut menjelaskan (dengan cara yang kurang lebih sopan) bagaimana algoritma baru tersebut tidak berguna bagi mereka. Terkadang hal ini dapat dikaitkan dengan peneliti AI yang tidak memahami gim atau pengembang gim yang tidak memahami AI, tetapi paling sering hal ini disebabkan oleh industri gim yang tidak bekerja seperti itu.

Pada dasarnya, industri gim dibatasi oleh realitas ekonomi sehingga sangat menghindari risiko dan cenderung picik. Gim video beranggaran besar biasanya membutuhkan waktu satu hingga tiga tahun untuk dikembangkan dan mungkin melibatkan ratusan profesional selama waktu tersebut; hal ini seringkali menghabiskan sebagian besar atau seluruh sumber daya dari satu studio. Pada saat yang sama, pasar gim digerakkan oleh hit-driven, dengan gim-gim yang biasa-biasa saja menghasilkan sangat sedikit uang.

Jadi, gamenya harus sukses atau studionya akan bangkrut. Tenggat waktu sangat ketat, jadi teknologinya harus dipastikan berfungsi. Karena banyak studio pengembang game tidak

tahu apakah mereka akan tetap ada setelah merilis game berikutnya, mereka biasanya hanya memiliki sedikit riset atau pengembangan jangka panjang.

Dalam kondisi seperti ini, beberapa teknologi AI baru yang mungkin bekerja dengan sangat baik, tetapi juga mungkin sangat sulit untuk digunakan, tidak mudah dijual kepada sebagian besar pengembang game. Sebaliknya, game dirancang berdasarkan teknologi yang sudah ada dan terbukti, seperti mesin berstatus terbatas dan pathfinding yang membentuk otak musuh 362. Game dirancang untuk tidak membutuhkan AI (yang tidak sepele). Kita akan kembali ke pertanyaan mengapa game dirancang berdasarkan ketiadaan AI dan apa yang dapat dilakukan untuk mengatasinya.

Sekarang mari kita beralih ke alasan pertama mengapa masa depan belum tiba. Tentu saja kita tidak memiliki game seperti yang saya bayangkan karena kita belum memiliki teknologinya. Saat ini, teknik AI kita yang matang sebagian besar memungkinkan solusi untuk masalah komputasi yang terdefinisi dengan baik. Sangat sulit membangun AI yang mampu menangani situasi yang tidak didefinisikan dengan cermat, hampir seperti skrip. Jenis AI yang mampu menangani situasi darurat, belajar, dan beradaptasi, sebagian besar masih dalam tahap pengembangan.

Dan jangan lupa bahwa untuk beberapa jenis permainan, kemampuan terbatas metode kecerdasan buatan saat ini masih kurang memadai, bahkan untuk masalah yang didefinisikan paling sempit sekalipun. Ambil contoh permainan strategi misalnya, permainan apa pun dalam seri *Civilization*, sebuah permainan strategi epik berbasis giliran. Dalam permainan ini, Anda memandu sebuah peradaban dari zaman Neolitikum hingga zaman antariksa, sambil terlibat dalam eksplorasi, ekspansi, peperangan, dan penelitian. Mirip dengan permainan strategi lainnya, pada titik tertentu Anda biasanya memiliki sejumlah besar "unit" (militer atau lainnya) yang tersebar di seluruh dunia, dan Anda perlu memberi tahu mereka semua apa yang harus dilakukan.

Bandingkan dengan Catur, Go, atau Dama, di mana Anda hanya memindahkan atau menempatkan satu unit setiap giliran. Fakta bahwa Anda memiliki begitu banyak unit sekaligus dalam gim seperti *Civilization* berarti jumlah kemungkinan gerakan dengan cepat menjadi sangat besar (Gambar 4.2). Jika Anda memiliki satu unit yang dapat Anda pindahkan ke sepuluh tempat berbeda (atau umumnya melakukan sepuluh tindakan berbeda), Anda memiliki faktor percabangan 10; jika Anda memiliki dua unit, Anda memiliki faktor percabangan 10 x 10 = 100; tiga unit, 10 x 10 x 10 = 1000 … Kita dengan cepat mencapai faktor percabangan jutaan, bahkan miliaran.

Gambar 4.2

Permainan dalam seri *Civilization* (Firaxis, 1991–2016) memungkinkan Anda memimpin sebuah peradaban melalui ribuan tahun ekspansi, penelitian, diplomasi, dan perang. Ruang kemungkinannya sangat luas, baik bagi komputer maupun manusia.

Dalam keadaan seperti itu, algoritma seperti minimax cepat rusak. Terlalu banyak tindakan potensial yang perlu dipertimbangkan, dan pencarian hampir tidak dapat mulai melihat konsekuensi dari masing-masing tindakan. Inilah alasan mengapa *Civilization*, yang utamanya merupakan permainan pemain tunggal, terkenal karena "AI-nya yang buruk"; unit yang dikendalikan komputer jarang berkoordinasi satu sama lain dan umumnya tampak bodoh. Untuk menawarkan tantangan yang kompetitif, permainan ini harus "curang" dengan memunculkan unit-unit dari udara di mana pemain tidak melihat. Situasi serupa terjadi dalam permainan strategi lainnya.

Game strategi real-time *StarCraft* adalah favorit untuk permainan kompetitif antarmanusia, dan telah ada kompetisi antar pemain AI sejak 2010. Terlepas dari semua upaya yang telah dilakukan, AI terbaik yang memainkan *StarCraft* hampir tidak bermain lebih baik daripada pemain pemula manusia. Kompleksitas permainan jumlah tindakan yang tersedia untuk diambil dengan dampak pada skala waktu yang berbeda benar-benar membanjiri metode AI kita saat ini.

Sejauh ini, kita hanya melihat pekerjaan menggunakan AI untuk bermain game atau mengendalikan karakter dalam game (dua tugas yang berkaitan erat, meskipun tidak sama). Seperti yang kita lihat, telah ada pekerjaan menggunakan AI untuk memainkan permainan papan klasik sejak sebelum ada komputer; baru-baru ini, semakin banyak peneliti telah mulai mengerjakan AI yang dapat bermain video game dan pendekatan baru untuk menciptakan perilaku karakter non-pemain yang menarik.

Tetapi metode AI dapat digunakan untuk lebih dari ini. Jika kita ingin mewujudkan visi permainan berbasis AI yang baru saja kita bahas, kita membutuhkan AI yang mampu beradaptasi dengan perilakunya, belajar dari kegagalan dan keberhasilan sebelumnya, memahami apa yang diketahui dan disukai pemain, menciptakan level dan permainan baru, serta bekerja sama dengan kita dalam merancang pengalaman. Dalam beberapa bab berikutnya, kita akan membahas beberapa upaya terbaru dalam menciptakan AI yang mampu melakukan hal-hal ini.

BAB 5
MENUMBUHKAN PIKIRAN DAN BELAJAR BERMAIN

Sejauh ini di buku ini, Anda telah membaca tentang beberapa jenis algoritma yang dapat digunakan untuk memainkan permainan dalam beberapa hal khususnya, algoritma minimax untuk permainan papan dan mesin berstatus terbatas serta pencarian A^* untuk bot FPS. Algoritma-algoritma ini dirancang dan diintegrasikan oleh manusia ke dalam sistem perangkat lunak kompleks yang kita sebut permainan video.

Membangun sistem semacam itu seringkali merupakan inti dari penciptaan AI: merakit berbagai komponen (algoritma) agar saling mendukung, menyetelnya agar bekerja dengan baik, menguji cara kerja produk akhir, lalu kembali dan mengerjakan ulang berbagai hal seperti Anda membangun sepeda, pompa air, atau sirkuit elektronik.

Membangun AI semacam itu adalah sebuah keahlian dan aktivitas yang relatif biasa saja, yang kurang menarik bagi kaum romantis yang terbuai oleh janji kecerdasan buatan yang belajar sendiri dan memutuskan sendiri. Selain itu, dan mungkin yang lebih penting, ini adalah proses yang padat karya dan karenanya mahal, yang ingin diotomatisasi oleh pengembang gim mana pun (atau siapa pun yang bergantung pada tingkat kecerdasan buatan tertentu dalam produknya).

Ide AI yang mengembangkan dirinya sendiri sehingga Anda tidak perlu memprogramnya cukup beri tahu apa saja hal yang harus dipelajarinya dengan baik terdengar jauh lebih menarik daripada AI yang dikodekan secara manual bagi seorang pecinta AI, maupun bagi orang yang berorientasi bisnis dan berfokus pada keuntungan finansial. Jadi, mari kita cari tahu bagaimana hal itu bisa dilakukan. Salah satu caranya adalah mencoba menciptakan sistem AI sebagaimana kita diciptakan: melalui evolusi Darwin.

5.1 GAGASAN YANG SANGAT SEDERHANA

Gagasan evolusi melalui seleksi alam tampak biasa-biasa saja dan hampir terbukti dengan sendirinya bagi kebanyakan orang di masyarakat Barat modern. Namun, lebih dari 150 tahun yang lalu, ketika Charles Darwin menerbitkan *The Origin of Species*, gagasan tersebut radikal, sesat, dan berbahaya. Hal itu juga tidak disadari oleh semua orang bahwa gagasan tersebut berhasil atau bahkan masuk akal. Karena gagasan inti evolusi melalui seleksi alam dapat dengan mudah tercampur dengan berbagai gagasan lain, mari kita coba meringkas konsep tersebut hingga ke dasarnya untuk melihat apakah kita dapat mereproduksinya di komputer.

Agar evolusi berhasil, Anda membutuhkan tiga unsur: variasi, hereditas (yang tidak sempurna), dan seleksi. Variasi berarti harus ada perbedaan di antara individu-individu. Hal ini secara implisit mengasumsikan bahwa ada hal-hal yang disebut individu kita belum membahasnya secara detail dan bahwa ada lebih dari satu individu. Kumpulan semua individu ini disebut "populasi". Keturunan berarti bahwa individu-individu dapat bereproduksi, baik sendiri atau bersama-sama dengan individu lain, dan bahwa keturunan yang dihasilkan dari

reproduksi ini entah bagaimana menyerupai "orang tua" mereka. Umumnya diasumsikan bahwa keturunan itu tidak sempurna, sehingga keturunannya bukan klon identik dari orang tua mereka; jika populasinya kecil dan orang tua dapat bereproduksi secara aseksual tanpa bercampur dengan orang tua lain, kondisi ini menjadi perlu.

Akhirnya, seleksi berarti bahwa beberapa individu dapat memiliki lebih banyak keturunan daripada yang lain, untuk beberapa alasan. Kita mengatakan bahwa individu yang dapat memiliki lebih banyak keturunan lebih "cocok" daripada yang lain; "kecocokan" individu dapat diperkirakan dengan berapa banyak cucu yang dimiliki individu tersebut. Mari kita pertimbangkan ini dalam konteks kelinci. Pertama, kita memiliki variasi.

Semua kelinci berbeda satu sama lain, dan bahkan jika Anda atau saya tidak dapat membedakan antara satu kelinci dan yang lain, mereka mungkin dapat membedakannya sendiri. Beberapa variasi ini bermakna secara fungsional; Misalnya, beberapa kelinci mungkin memiliki kaki yang lebih panjang sehingga mereka dapat berlari lebih cepat dan yang lainnya memiliki mata yang lebih tajam sehingga mereka dapat melihat rubah dari jarak yang lebih jauh. Kemudian, kita memiliki faktor keturunan.

Cetak biru untuk kelinci, seperti halnya semua hewan dan tumbuhan lainnya, ada pada DNA-nya. Kelinci (seringkali) mempraktikkan reproduksi seksual (bagaimanapun juga, mereka berkembang biak seperti kelinci) yang mengakibatkan DNA dari satu kelinci direkombinasikan dengan DNA kelinci lainnya. Biasanya juga ada beberapa perubahan kecil yang diperkenalkan pada DNA di setiap generasi; ini disebabkan oleh kesalahan transkripsi ketika untaian DNA disalin dalam proses pembelahan sel dan disebut mutasi.

Terakhir, kita memiliki seleksi. Ini dapat terjadi dalam banyak cara saya hanya tahu sedikit tentang apa yang membuat kelinci menarik satu sama lain tetapi bentuk seleksi yang jelas adalah bahwa kelinci yang ditangkap oleh rubah tidak dapat memiliki keturunan sebanyak mereka yang berlari lebih cepat darinya. Seleksi tidak hanya bergantung pada kelinci individu tetapi juga pada seluruh populasi: untuk berlari lebih cepat dari rubah, Anda sebenarnya tidak harus lebih cepat dari rubah, Anda hanya harus lebih cepat dari kelinci lainnya. Oleh karena itu, setiap peningkatan kecil pada kecepatan lari, taktik menghindar, atau penglihatan dapat meningkatkan kebugaran kelinci. Selama bergenerasi-generasi kelinci, kita mendapatkan kelinci yang sangat baik, atau setidaknya kelinci yang pandai berlari lebih cepat dari rubah.

Tentu saja, rubah juga mengalami evolusi melalui seleksi alam. Sementara variasi dan hereditas sangat mirip untuk populasi rubah dan populasi kelinci, seleksi bekerja agak berbeda. Rubah yang gagal menangkap kelinci akhirnya kelaparan dan menghentikan kehidupan rubahnya tanpa bereproduksi, sedangkan mereka yang menangkap dan memakan kelinci mungkin memperoleh nutrisi yang cukup untuk bertahan hidup dan memiliki keturunan. Tentu saja, apakah rubah menangkap kelinci tergantung pada rubah dan kelinci (dan mungkin kelinci lain dalam kawanan yang sama).

Jadi kebugaran rubah digabungkan dengan kebugaran kelinci dalam proses yang dikenal sebagai koevolusi; populasi rubah dan populasi kelinci memasuki "perlombaan senjata" di mana rubah mengembangkan taktik dan fitur tubuh yang lebih baik dan lebih baik untuk mengejar kelinci, dan kelinci mengembangkan cara yang lebih baik dan lebih baik untuk

menghindari rubah. Setelah beberapa generasi koevolusi kelinci dan rubah, beberapa kelinci masih ditangkap oleh rubah dan sebagian besar masih lolos.

Jika seekor kelinci dari seribu generasi yang lalu bertemu dengan rubah baru yang berkilau dari generasi terbaru, rubah tersebut hampir pasti menang, begitu pula sebaliknya. Perlombaan senjata koevolusi bertanggung jawab atas berbagai fenomena menarik di alam, termasuk kecepatan ekstrem cheetah dan gazel, paruh panjang burung kolibri, dan bentuk unik bunga yang diserbuki burung kolibri, yang menyembunyikan nektar berharga mereka jauh di dalam bunga.

Bab ini tidak akan membahas burung dan lebah secara harfiah. Saya berjanji untuk membahas cara menumbuhkan pikiran, jadi mari kita lihat bagaimana evolusi dapat diterapkan pada program komputer. Pertama, kita memiliki variasi. Bayangkan sebuah populasi program komputer yang berbeda; mereka berbeda dalam kode sumbernya, sehingga mereka juga berbeda dalam apa yang mereka lakukan. Dalam kasus yang paling sederhana, semua program ini acak pada awalnya. Kemudian kita memiliki hereditas. Kita dapat membuat keturunan dari sebuah program hanya dengan menyalinnya, dan kemudian membuat hereditas tersebut tidak sempurna dengan memperkenalkan beberapa mutasi (mengubah beberapa bagian kecil dari kode sumber).

Kita juga bisa menggabungkan kode sumber dari dua program induk, mengambil beberapa bagian dari yang satu dan beberapa dari yang lain, untuk membuat program turunan dalam proses yang dikenal sebagai crossover. Akhirnya, kita sampai pada seleksi. Kita cukup mengukur seberapa baik program tersebut dalam melakukan tugasnya dan menetapkan tingkat kebugaran yang lebih tinggi kepada program yang melakukan tugas tertentu dengan lebih baik.

Tugasnya bisa berupa apa pun yang Anda inginkan dari program komputer: mengurutkan daftar, melukis gambar, mungkin bermain gim. Berdasarkan pengukuran kebugaran ini, kita cukup membuang program yang buruk dan membuat salinan mutasi atau rekombinasi dari program yang baik. Dunia di sana seperti kode-makan-kode!

Apakah ini masuk akal bagi Anda? Jika tidak, Anda sepenuhnya memahami saya. Awalnya agak sulit untuk percaya bahwa kita dapat mengembangkan program karena ada argumen kuat mengapa hal itu tidak seharusnya berhasil. Program acak, misalnya, kemungkinan besar tidak akan terlalu baik dalam hal apa pun; bahkan, kemungkinan besar program tersebut tidak akan berjalan. Jadi, bagaimana Anda bisa memberikan nilai kebugaran yang masuk akal kepada populasi program komputer yang tidak berguna? Mengenai mutasi, memperkenalkan perubahan acak pada suatu program kemungkinan besar hanya akan memperburuknya, bahkan mungkin merusaknya sehingga tidak akan berfungsi sama sekali. Sulit membayangkan bagaimana hal ini dapat membuat program menjadi lebih baik.

Evolusi memang berhasil, tidak hanya di alam tetapi juga di komputer. Algoritma evolusioner, sebutan untuk algoritma yang didasarkan pada prinsip evolusi melalui seleksi (buatan), sering digunakan untuk berbagai tugas seperti peramalan deret waktu keuangan, pengendalian mesin jet, dan perancangan antena radar. Selain itu, beberapa AI terbaik untuk bermain gim setidaknya sebagian dibangun oleh evolusi, seperti yang akan kita lihat. Untuk

membantu memahami bagaimana proses yang tidak lazim ini sebenarnya bekerja, ada baiknya mempertimbangkan hal berikut.

Pertama, meskipun benar bahwa program yang dibangun secara acak biasanya sangat buruk dalam menyelesaikan tugas apa pun, belum tentu program tersebut benar-benar menyelesaikan tugas yang diberikan. Yang kita butuhkan untuk memulai evolusi adalah cara membedakan program mana yang sedikit kurang efektif dalam menyelesaikan tugasnya, program mana yang paling sedikit mengacaukannya dan memilihnya untuk direproduksi. Selama beberapa generasi, program-program tersebut kemudian dapat berubah dari sangat buruk menjadi hampir tanpa harapan, menjadi cukup buruk, menjadi setengah buruk, menjadi lumayan, menjadi cukup baik, menjadi baik, menjadi sangat baik.

Agar hal ini terjadi, kita memerlukan fungsi kebugaran, suatu cara untuk menetapkan kebugaran pada program, yang dapat menangkap semua nuansa ini. Inilah salah satu alasan mengapa game sangat bagus untuk penelitian AI: biasanya mudah untuk mengukur kinerja pemain dengan sangat tepat melalui skor atau peringkat melawan pemain lain. Saya akan menjelaskan nanti di bab ini bagaimana fungsi kebugaran yang baik membantu saya mengembangkan pengemudi game balap yang mengemudi lebih baik daripada saya.

Kedua, memang benar bahwa perubahan acak pada program yang ditulis dalam bahasa pemrograman standar seperti Java, Python, atau C++ cenderung menghancurkan program tersebut; sebagian besar perubahan kode mengakibatkan program tidak berjalan sama sekali, seperti menghilangkan satu tongkat acak di menara Jenga kemungkinan akan menyebabkan menara itu runtuh, atau menghilangkan satu buah acak dalam permainan Catur dalam keadaan permainan tingkat lanjut akan mengubah keseimbangan permainan sepenuhnya.

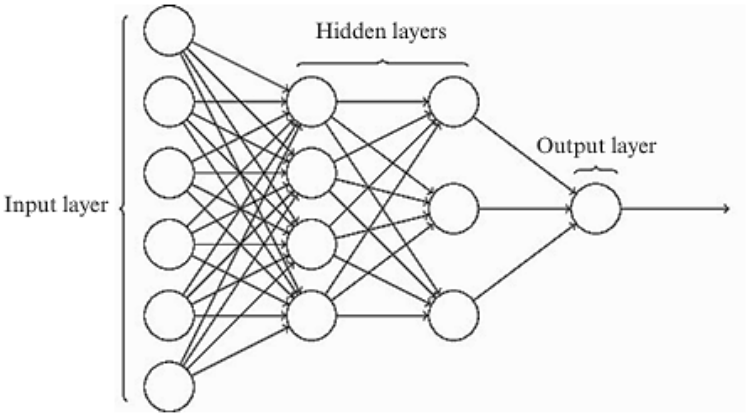
Kita tidak perlu menggunakan bahasa-bahasa ini saat kita mengembangkan program. Memilih representasi yang benar untuk program Anda adalah bagian yang sangat penting untuk membuat evolusi berhasil. Untuk banyak jenis program, kita sekarang memiliki representasi di mana sebagian besar mutasi kecil pada program tidak membawa bencana, dan banyak yang sebenarnya bermanfaat bagi kebugaran program. Secara khusus, cara yang baik untuk merepresentasikan program-program ini adalah sebagai jaringan saraf.

5.2 OTAK YANG SANGAT KECIL

Seperti banyak konsep lain dalam kecerdasan buatan, Anda dapat melihat (dan membicarakan) jaringan saraf dari perspektif romantis atau pragmatis. Dari perspektif romantis, jaringan saraf adalah simulator otak kecil, yang meniru fungsi inti sirkuit saraf otak. Dari perspektif pragmatis, jaringan saraf hanyalah sistem persamaan nonlinier, yang menerapkan transformasi geometris pada data masukan.

Gambar 5.1 mengilustrasikan jaringan saraf sederhana. Jaringan ini tersusun menjadi empat lapisan: lapisan masukan (dengan enam simpul), dua lapisan tersembunyi (masing-masing dengan empat dan tiga neuron), dan lapisan keluaran (hanya dengan satu neuron). Setiap simpul (sering disebut "neuron" dengan analogi neuron biologis di otak) termasuk dalam lapisan tertentu dan terhubung ke semua neuron di lapisan berikutnya. Jenis jaringan

saraf ini disebut jaringan umpan maju, karena nilai-nilai diumpankan (atau disebarkan) maju dari satu lapisan ke lapisan berikutnya.



Gambar 5.1

Gambar ini mengilustrasikan jaringan saraf tiruan yang sangat sederhana. Jaringan ini tersusun menjadi empat lapisan: lapisan masukan (dengan enam simpul), dua lapisan tersembunyi (masing-masing dengan empat dan tiga neuron), dan lapisan keluaran (hanya dengan satu neuron). Setiap simpul (sering disebut "neuron" dengan analogi neuron biologis di otak) termasuk dalam lapisan tertentu dan terhubung ke semua neuron di lapisan berikutnya. Jenis jaringan saraf tiruan ini disebut jaringan umpan maju, karena nilai-nilai diumpankan (atau disebarkan) maju dari satu lapisan ke lapisan berikutnya.

Anda menggunakan jaringan saraf umpan maju dengan menetapkan nilai ("input"), yang disebut aktivasi, ke neuron di lapisan masukan. Aktivasi ini kemudian disebarkan ke lapisan berikutnya melalui koneksi antar neuron, dan koneksi tersebut memiliki nilai (disebut "bobot") itu sendiri. Ketika sebuah aktivasi berpindah dari satu neuron ke neuron lain, aktivasi tersebut dikalikan dengan bobot koneksi antar neuron. Karena semua neuron dalam satu lapisan terhubung ke semua neuron di lapisan berikutnya, aktivasi sel di suatu lapisan (misalnya, lapisan tersembunyi pertama) adalah jumlah semua aktivasi neuron di lapisan sebelumnya (misalnya, lapisan masukan) dikalikan dengan bobot koneksi dari lapisan tersebut.

Dan kemudian semua ini terjadi lagi ketika aktivasi diteruskan ke lapisan berikutnya. Dan seterusnya. Jika gagasan bahwa banyak aktivasi neuron, yang sebenarnya hanyalah angka, dikalikan dengan angka lain tidak menjelaskan apa pun, coba bayangkan jaringan saraf seperti sistem pipa. Sejenis cairan (misalnya, rum) dimasukkan ke dalam lapisan masukan dan kemudian dialirkan dari neuron ke neuron melalui pipa-pipa dengan diameter bervariasi. Pipa yang lebih tebal secara alami membawa lebih banyak cairan, sehingga neuron dengan pipa tebal dari neuron aktivasi tinggi menerima lebih banyak cairan dan mampu mengalirkannya lebih banyak. Ketebalan pipa di sini berhubungan dengan bobot koneksi.

Hampir sesederhana itu, tetapi ada satu detail penting lagi: setiap kali aktivasi neuron dihitung, ia melewati fungsi nonlinier (seperti tangen hiperbolik atau fungsi linear yang disearahkan). Hal ini meningkatkan kapasitas komputasi jaringan saraf, tetapi memahaminya

tidak terlalu penting untuk memahami apa yang terjadi pada tingkat konseptual. Selain itu, memang sesederhana itu. Setidaknya dasar-dasarnya.

Gagasan dasar aktivasi yang diteruskan dari neuron melalui koneksi dengan bobot yang bervariasi adalah sama di hampir semua jenis jaringan saraf, bahkan jaringan yang memiliki koneksi dengan kekuatan yang bervariasi, berputar kembali pada dirinya sendiri, dan berbagi bobot dengan koneksi lain. Bahkan jaringan yang sangat besar yang digunakan dalam apa yang sekarang disebut "pembelajaran mendalam" dan yang mungkin memiliki lusinan lapisan dan jutaan koneksi pada intinya hanyalah sistem persamaan atau, jika Anda lebih suka, sistem pipa.

Konstruksi komputasi sederhana ini ternyata sangat berguna dan serbaguna; secara matematis, jaringan yang cukup besar dapat mengaproksimasi fungsi apa pun. Jaringan saraf dapat diajarkan untuk mengenali wajah, mengendarai mobil, menggubah musik, menerjemahkan teks, dan sebagainya. Ya, mereka juga dapat diajarkan untuk bermain gim. Tetapi pertama-tama mereka perlu diajarkan, atau dilatih, artinya semua bobot koneksi perlu ditetapkan. Karena bobot koneksi menentukan apa yang dapat dilakukan jaringan saraf, struktur jaringan saraf yang sama dengan nilai koneksi yang berbeda dapat bekerja dengan baik dalam melakukan hal-hal yang sama sekali berbeda, seperti mengkonjugasikan kata kerja bahasa Prancis, bermain sepak bola, atau menemukan cacat pada lembaran baja. Jaringan saraf dengan bobot koneksi acak biasanya tidak dapat melakukan apa pun dengan baik.

Jadi, bagaimana cara melatih jaringan saraf? Pada dasarnya ada dua cara. Pertama, melalui algoritma evolusioner, seperti yang telah saya jelaskan; perubahan kecil pada program di sini mengacu pada perubahan kecil pada bobot koneksi. Saya akan menjelaskan bagaimana evolusi dapat digunakan untuk melatih jaringan saraf untuk bermain gim di bagian selanjutnya. Cara penting lainnya untuk melatih jaringan saraf adalah dengan membuat perubahan kecil sebagai respons terhadap setiap kesalahan yang dibuat jaringan dan mengirimkan koreksi ini secara mundur dalam jaringan saraf dari lapisan keluaran ke lapisan masukan. Ini disebut backpropagation dan kita akan melihat bagaimana hal itu dapat digunakan di bagian selanjutnya.

5.3 KELANGSUNGAN HIDUP YANG TERCEPAT

Ketika saya memulai program Kuliah saya pada tahun 2004, rencana saya adalah menggunakan algoritma evolusioner untuk melatih jaringan saraf untuk mengendalikan robot. Robot-robot ini akan diberi penghargaan karena melakukan hal-hal seperti mengikuti robot lain, tidak menabrak dinding, memecahkan labirin, dan sebagainya. Karena jaringan saraf yang mendapatkan lebih banyak imbalan akan mampu berkembang biak, pada akhirnya saya akan memiliki populasi jaringan saraf yang berperilaku cukup baik. Setidaknya itulah rencananya.

Saat itu, peneliti lain telah berhasil mengajarkan jaringan saraf pengendali robot untuk melakukan hal-hal ini, tetapi saya akan melakukannya... dengan lebih baik! Saya punya beberapa ide tentang menghubungkan beberapa jaringan saraf dan melatihnya satu per satu, dan hal-hal semacam itu. Namun, ketika saya mulai mencoba mengajarkan jaringan saraf untuk mengendalikan robot-robot ini, saya menyadari bahwa ini adalah pekerjaan yang sangat sulit. Robot-robot itu lambat dan sering menempatkan diri mereka dalam situasi yang harus

saya selamatkan, belum lagi saya harus berhadapan dengan ban yang aus, kabel yang putus, dan sebagainya. Sepertinya saya tidak akan pernah membuat kemajuan nyata dengan cara itu. Jadi saya meninggalkan dunia nyata dan beralih ke gim video.

Ide saya adalah saya dapat menggunakan gim video (alih-alih robot) sebagai lingkungan untuk menguji algoritma saya. Gim video memiliki sebagian besar kualitas yang diinginkan dari masalah robotika dan kekurangan beberapa kualitas yang sangat tidak diinginkan. Khususnya, Anda tidak perlu membangun robot mahal atau rintangan dunia nyata, Anda dapat mempercepat gim sehingga pengujian berlangsung jauh lebih cepat (ribuan kali lebih cepat dalam banyak kasus) daripada waktu nyata, dan ketika terjadi kesalahan, Anda cukup memulai ulang gim. Anda tidak perlu membersihkan piksel setelah karakter gim Anda mogok, dan Anda tidak perlu membayar uang untuk membangun yang baru.

Saya memutuskan untuk memulai dengan gim balap karena memiliki kurva kesulitan yang bagus: relatif mudah untuk mempelajari dasar-dasarnya cukup injak pedal gas untuk melaju lurus ke depan tetapi kemudian segalanya menjadi lebih rumit ketika Anda juga harus melewati tikungan dan menyalip mobil lain sambil menghindari tabrakan. Dan ternyata ada banyak hal yang harus dipelajari tentang cara terbaik mengemudikan mobil dalam balapan mobil; jika tidak, tidak akan ada kompetisi internasional besar yang berulang tentang hal ini. Jadi saya mengembangkan gim balap mobil sederhana dan cara mengemudikan mobil dengan jaringan saraf.

Cara jaringan saraf digunakan untuk mengemudikan mobil cukup sederhana: ia menghubungkan masukan jaringan saraf dengan apa yang "dilihat" pengemudi dan keluarannya ke roda kemudi dan pedal. Dalam salah satu pengaturan saya, saya menggunakan delapan neuron di lapisan masukan: enam terhubung ke sensor pendeteksi jarak simulasi yang mengembalikan jarak ke tepi lintasan terdekat atau mobil lain, masing-masing di sepanjang enam arah berbeda, satu ke speedometer, dan satu ke sensor yang mengembalikan sudut relatif terhadap lintasan. Jaringan saraf tersebut memiliki satu lapisan tersembunyi yang terdiri dari enam neuron, dan terakhir lapisan keluaran hanya memiliki dua neuron, yang terhubung ke akselerator/rem dan kemudi.

Ambil contoh jaringan saraf dengan bobot koneksi acak dan tempatkan untuk mengendalikan mobil, dan ia tidak akan melakukan apa pun atau melakukan sesuatu yang agak membosankan, seperti keluar jalur dan menabrak. Ambil contoh seratus jaringan tersebut dengan bobot koneksi acak yang berbeda, dan beberapa di antaranya akan melakukan hal-hal yang lebih menarik atau bermanfaat daripada yang lain. Untuk mengubahnya menjadi algoritma evolusi, kita hanya perlu fungsi kebugaran dan cara melakukan seleksi dan mutasi.

Dalam kasus ini, fungsi kebugarannya sangat sederhana: seberapa jauh mobil melaju dalam 30 detik. Dengan menggunakan populasi seratus jaringan, algoritma evolusi yang saya gunakan akan mencoba semua jaringan dan menghapus lima puluh jaringan yang berkinerja terburuk (mengemudi dalam jarak terpendek dalam waktu yang dialokasikan); kemudian akan menggantinya dengan salinan dari lima puluh jaringan yang berkinerja terbaik, tetapi menambahkan mutasi dalam bentuk perubahan acak pada beberapa bobot koneksi dalam jaringan ini.

Proses sederhana ini bekerja dengan sangat baik. Dalam belasan generasi, saya akan memiliki jaringan saraf yang dapat melaju dengan cukup baik, dan dalam seratus generasi saya biasanya akan mendapatkan jaringan saraf yang dapat melaju lebih baik daripada saya! Dengan demikian, saya sekali lagi bisa merasakan kekalahan dalam permainan oleh program AI yang saya kembangkan sendiri (pengalaman yang sangat saya rekomendasikan), tetapi berbeda dengan program bermain Catur dari tugas kuliah saya beberapa tahun sebelumnya, saya tidak benar-benar menentukan bagaimana program tersebut harus menyelesaikan tugasnya, melainkan bagaimana program tersebut harus belajar menyelesaikan tugasnya.

Ketika saya mengatakan prosesnya berjalan lancar, ada kendala. Proses ini memang berjalan lancar ketika melatih jaringan saraf untuk mengemudi di trek balap tertentu. Ketika menggunakan jaringan saraf yang sama dan menempatkannya di mobil di trek balap yang berbeda, hasilnya tidak berjalan dengan baik: sebagian besar mobil gagal berbelok saat seharusnya, dan keluar jalur.

Saya agak bingung dengan hal ini, tetapi kemudian saya menyadari bahwa trek tempat saya melatih jaringan saraf untuk mengemudi cukup terbatas dalam tantangan yang ditawarkannya; misalnya, trek tersebut hanya berisi belokan kiri. Maka saya menerapkan pola latihan baru: setiap kali jaringan belajar mengemudi dengan baik di satu trek, saya menambahkan trek balap baru yang berbeda ke fungsi kebugaran, sehingga kebugaran jaringan saraf akan bergantung pada cara mengemudinya di beberapa trek.

Pendekatan ini bekerja dengan sangat baik, dan relatif cepat evolusi menghasilkan jaringan saraf yang dapat mengemudi dengan mahir di trek apa pun yang dapat saya pikirkan, meskipun secara umum mereka adalah pengemudi yang sedikit lebih berhati-hati daripada jaringan yang telah berevolusi di satu trek saja.

Lalu bagaimana dengan balapan di hadapan mobil lain? Tak heran, jika Anda mengambil jaringan saraf yang telah dilatih untuk mengemudi di lintasannya sendiri tanpa gangguan dari pesaing yang mengganggu dan menempatkannya dalam balapan kompetitif dengan mobil lain, kekacauan pun terjadi. Karena jaringan tersebut belum pernah bertemu mobil lain sebelumnya, ia tidak tahu cara menghindari tabrakan, atau bahwa menghindarinya adalah ide yang bagus, atau bahkan apa itu tabrakan antar mobil. Hal ini dapat diperbaiki dengan melatih jaringan dengan kehadiran mobil lain dan biasanya menghasilkan perilaku mengemudi yang cukup baik tergantung pada fungsi kebugarannya. Ketika kita tidak lagi balapan sendirian, kita perlu kembali dan memikirkan kembali fungsi kebugarannya.

Fungsi kebugaran yang bekerja sangat baik untuk belajar balapan sendirian di lintasan adalah dengan mengukur seberapa jauh mobil melaju di lintasan dalam 30 detik. Namun dalam balapan mobil yang benar-benar kompetitif, yang terpenting adalah posisi Anda di depan atau di belakang mobil lain jadi mungkin fungsi kebugaran seharusnya mencerminkan hal itu. Ini akan membuat situasinya lebih mirip koevolusi di dunia alami, di mana kebugaran individu dalam satu spesies (atau kelompok, secara lebih umum) sebagian bergantung pada spesies lain seperti contoh kelinci dan rubah di awal bab ini. Dan tampaknya strategi yang berhasil dalam balapan mobil, seperti dalam permainan lainnya, sebagian bergantung pada cara pengemudi lain mengemudi.

Untuk melihat apa yang mungkin terjadi jika fungsi kebugaran mencerminkan struktur hadiah balapan mobil yang sebenarnya, saya mengubah kebugaran menjadi relatif, posisi di depan atau di belakang mobil lain di akhir balapan. Dengan sangat cepat, proses evolusi menemukan bahwa strategi yang layak adalah menjadi sangat agresif dan mendorong mobil lain keluar jalur. Perilaku ini tampaknya lebih mudah dipelajari daripada belajar mengemudi cepat, menghindari tabrakan, dan mendapatkan waktu putaran yang baik.

Atau mungkin jaringan yang mempelajari strategi non-agresif akan terdesak oleh jaringan saraf yang mempelajari strategi agresif, dan oleh karena itu menerima kebugaran yang lebih rendah. Dalam kasus apa pun, komposisi fungsi kebugaran dapat dengan mudah digunakan di sini sebagai tombol untuk menaikkan atau menurunkan agresivitas dalam jaringan yang telah berkembang, sesuatu yang pasti berguna saat membuat karakter menarik dalam permainan.

5.4 UJI COBA DAN KESALAHAN PADA KECEPATAN

Komputasi evolusioner dapat digambarkan sebagai proses uji coba dan kesalahan yang masif. Proses ini tampaknya sangat boros, semua jaringan saraf yang sedikit lebih buruk daripada jaringan saraf terbaik di setiap generasi dibuang begitu saja. Tak satu pun informasi yang mereka temui dalam "kehidupan" singkat mereka tersimpan. Namun, proses evolusi melalui seleksi berhasil, baik di alam (seperti yang kita buktikan sendiri) maupun di dalam program komputer.

Tetapi adakah cara lain yang bisa kita pelajari dari pengalaman untuk menciptakan AI yang efektif, mungkin dengan melestarikan lebih banyak informasi? Masalah belajar melakukan suatu tugas hanya dengan umpan balik berkala tentang seberapa baik kinerja Anda disebut masalah pembelajaran penguatan, yang mengimpor beberapa terminologi dari psikologi behavioris (jenis psikologi di mana psikolog meminta tikus menarik tuas dan berlarian dalam labirin) ke ilmu komputer. Pada dasarnya, ada dua pendekatan umum untuk memecahkan masalah ini. Pendekatan yang kurang umum adalah menggunakan beberapa bentuk algoritma evolusioner.

Pendekatan yang lebih umum adalah menggunakan beberapa bentuk pemrograman dinamis perkiraan, seperti algoritma *Q-learning*. Anda dapat memikirkannya seperti ini: sementara komputasi evolusioner memodelkan jenis pembelajaran yang terjadi di beberapa masa hidup, *Q-learning* (dan algoritma serupa) memodelkan jenis pembelajaran yang terjadi selama masa hidup. Alih-alih belajar berdasarkan satu nilai kebugaran di akhir upaya untuk melakukan tugas (seperti yang dilakukan evolusi), *Q-learning* dapat belajar dari banyak peristiwa saat tugas dilakukan. Alih-alih membuat perubahan acak pada seluruh jaringan saraf (seperti yang terjadi dalam evolusi), dalam *Q-learning*, perubahan dilakukan ke arah tertentu sebagai respons terhadap imbalan positif atau negatif.

Dalam *Q-learning*, jaringan saraf menerima masukan yang mewakili apa yang "dilihat" oleh agen, seperti jaringan kendali mobil yang telah berevolusi yang saya jelaskan di bagian sebelumnya. Jaringan juga menerima masukan yang menjelaskan tindakan apa yang sedang

dipertimbangkan oleh agen; dalam domain balap mobil, bisa berupa belok kiri, belok kanan, akselerasi, dan rem (atau beberapa kombinasi).

Keluarannya adalah *Q-values*, yang merupakan estimasi seberapa baik suatu tindakan tertentu dalam suatu keadaan (situasi) tertentu. Jadi, alih-alih memetakan masukan sensor ke tindakan, jaringan memetakan masukan sensor dan tindakan ke *Q-values*. Cara jaringan saraf ini digunakan untuk melakukan sesuatu, seperti mengemudikan mobil, adalah setiap kali perlu mengambil keputusan, ia menguji semua tindakan yang mungkin dan memilih tindakan dengan *Q-values* tertinggi dalam keadaan saat itu.

Jelas, jaringan saraf perlu dilatih sebelum dapat digunakan; jaringan yang mengeluarkan *Q-values* acak tidak akan memenangkan perlombaan atau memecahkan masalah lainnya. Ide dasar melatih jaringan saraf menggunakan *Q-learning* adalah membandingkan nilai prediksi dari tindakan yang diambil dalam suatu keadaan dengan nilai aktual dari tindakan yang diambil dalam keadaan tersebut, seperti yang diamati setelah tindakan tersebut diambil. Jika nilai aktual berbeda dari nilai prediksi, jaringan saraf akan sedikit disesuaikan menggunakan algoritma propagasi balik. Misalnya, kita tidak tahu apakah berbelok kiri di persimpangan merupakan ide yang baik. Jadi kami mencobanya, dan melihat apa yang terjadi. Setelah kami tahu apa yang terjadi, kami memperbarui keyakinan kami tentang nilai belok kiri di persimpangan tersebut.

Namun, bagaimana kami tahu nilai sebenarnya dari melakukan tindakan tertentu dalam kondisi tertentu? Hal itu bergantung pada umpan balik, atau penguatan, yang diterima agen dari dunia. Misalnya, ketika mengajarkan jaringan saraf untuk balapan mobil, Anda dapat memberinya umpan balik positif (hadiah) setiap kali mencapai tujuan atau mungkin setiap kali melewati sebagian lintasan, dan umpan balik negatif (hukuman) setiap kali keluar lintasan atau menabrak mobil lain. Jika umpan balik lebih tinggi, atau lebih rendah, dari yang diharapkan jaringan, maka algoritma *backpropagation* digunakan untuk sedikit mendorong jaringan saraf ke arah umpan balik, sehingga memberikan estimasi yang lebih baik tentang nilai tindakan tersebut saat menghadapi kondisi serupa berikutnya. Inti dari algoritma *Q-learning* adalah memperbarui jaringan saraf secara konstan sehingga menjadi lebih baik dalam memperkirakan seberapa baik tindakan yang berbeda yang akan dilakukan dalam kondisi tertentu berdasarkan hadiah yang didapatnya sesekali.

Masalah dengan prosedur yang baru saja saya uraikan adalah Anda ingin dapat mengetahui seberapa baik tindakan tersebut meskipun tidak langsung mendapatkan imbalan. Misalnya, jika Anda belajar mengemudi mobil, Anda ingin belajar bahwa sedikit saja keluar dari lintasan balap adalah ide yang buruk, meskipun Anda harus terus melaju selama beberapa detik sebelum benar-benar keluar dari lintasan dan menerima imbalan negatif (tabrakan!). Anda juga ingin tahu bahwa jika Anda berada di garis start, berakselerasi itu baik meskipun akan membutuhkan waktu yang cukup lama hingga Anda benar-benar menyelesaikan balapan dan mendapatkan imbalan positif.

Demikian pula, jika Anda bermain Tetris, Anda ingin tahu bahwa menumpuk balok sehingga Anda akhirnya dapat melewati beberapa jalur sekaligus adalah ide yang bagus, meskipun mungkin tergoda untuk mendapatkan keuntungan jangka pendek dengan melewati

satu jalur saja. Dalam kehidupan nyata, Anda mungkin terkadang berada dalam kondisi di mana tindakan tertentu, misalnya memesan minuman lagi, memberikan imbalan jangka pendek. Namun, Anda mungkin telah mempelajari bahwa tindakan tersebut dapat memiliki nilai negatif karena hukuman jangka panjang berupa mabuk keesokan harinya dan meningkatkan risiko hukuman jangka panjang berupa berakhir sebagai pecandu alkohol. Dalam pembelajaran penguatan, hal ini disebut masalah penugasan kredit, dan seperti yang mungkin Anda duga, ini merupakan masalah yang sangat sulit.

Dalam *Q-learning*, cara standar untuk mendekati masalah penugasan kredit adalah dengan belajar dari imbalan yang diharapkan. Jadi, setiap kali suatu tindakan dilakukan, jika tidak ada imbalan atau hukuman nyata dari dunia nyata, ia menyesuaikan estimasi jaringan saraf tentang nilai tindakan yang baru saja dilakukan berdasarkan estimasinya sendiri tentang nilai tindakan terbaik di kondisi berikutnya. Jaringan saraf pada dasarnya bertanya pada dirinya sendiri apa yang menurutnya seharusnya menjadi penguatannya. Kedengarannya gila, tetapi mengingat jaringan tersebut sesekali mendapatkan bala bantuan nyata dari dunia nyata (atau permainan), prosedur ini seharusnya berhasil secara teori.

Dalam praktiknya, untuk waktu yang lama, cukup sulit untuk membuat *Q-learning* bekerja dengan andal pada masalah-masalah kompleks. Namun, pada tahun 2015, sekelompok peneliti di perusahaan riset AI DeepMind yang berbasis di London berhasil membuat *Q-learning* memainkan sejumlah gim arkade klasik dari konsol Atari 2600, seperti *Missile Command* dan *Pac-Man*. Dibutuhkan daya komputer yang besar untuk melatih jaringan-jaringan ini, lebih dari sebulan waktu komputer per gim, tetapi jaringan saraf dalam banyak kasus belajar bermain lebih baik daripada manusia.

Jadi, jenis algoritma mana yang lebih baik untuk belajar bermain gim: *Q-learning* atau evolusi? Secara teori, *Q-learning* seharusnya mampu mengeksploitasi lebih banyak informasi karena dapat menggunakan penguatan yang lebih sering, dan juga dapat membuat perubahan terarah pada bobot jaringan saraf, sedangkan evolusi hanya membuat perubahan acak. Namun, tampaknya evolusi memiliki lebih banyak kebebasan untuk menciptakan strategi yang tidak secara langsung bergantung pada imbalan, dan evolusi juga mampu mengubah struktur jaringan, bukan hanya bobotnya. Anda dapat berargumen bahwa *Q-learning* bereaksi terhadap umpan balik yang diterimanya dengan menyesuaikan strateginya secara bertahap, sementara evolusi dengan berani mengusulkan strategi baru yang lengkap dan mengujinya secara keseluruhan.

Dalam praktiknya, kedua metode ini dapat bekerja dengan baik ketika seorang praktisi yang terampil menerapkannya. Namun, kita masih berada pada titik di mana sebagian besar metode pembelajaran tidak bekerja dengan baik secara langsung. Dalam jangka panjang, kita dapat mencari inspirasi dari alam: hewan (termasuk kita) belajar selama dan lintas masa hidup, dan kemungkinan besar kita juga membutuhkan kedua jenis pembelajaran tersebut untuk menciptakan sistem yang mampu mempelajari tugas-tugas yang sangat rumit secara mandiri.

BAB 6
GAME BELAJAR DARI ANDA SAAT ANDA MEMAINKANNYA

6.1 HAL YANG PERLU DIPELAJARI DALAM GAME

Seperti yang telah kita bahas di bab-bab sebelumnya, kita belajar dari game cara memainkannya dan hampir pasti juga keterampilan lainnya. Kita juga dapat mengembangkan algoritma yang belajar bermain game. Namun, mari kita balik pernyataan ini. Bisakah game belajar dari kita? Dan jika ya, apa yang bisa dipelajarinya? Bisakah kita mengembangkan algoritma yang menggunakan interaksi kita dengan game untuk mempelajari tentang kita?

Saat Anda bermain game, Anda terus-menerus memberikan informasi kepada game tersebut. Anda menekan tombol dan memutar stik konsol. Dalam banyak game, Anda juga memasukkan teks. Anda terus-menerus membuat pilihan: pergi ke arah ini atau itu, menanggapi karakter tersebut secara positif atau negatif dalam percakapan, menyerang musuh itu atau tidak (dan menggunakan senjata apa). Beberapa pilihan bersifat kompleks dan diekspresikan sepanjang permainan, seperti kepribadian dan karakteristik lain dari karakter yang Anda mainkan atau bentuk dan orientasi politik negara yang Anda pimpin; pilihan lainnya terjadi dalam skala subdetik, seperti kapan tepatnya harus melompat dari platform agar tidak jatuh ke celah. Semua ini adalah informasi yang dapat diekspresikan dengan baik menggunakan angka dan simbol lainnya.

Untuk permainan yang diimplementasikan di komputer (termasuk permainan komputer standar serta versi digital permainan papan dan kartu), hal ini praktis karena itulah kehebatan komputer: menyimpan dan memproses informasi. Sangat mungkin bagi sebuah permainan komputer untuk menyimpan semua masukan yang pernah Anda berikan dan kemudian menggunakan algoritma cerdas untuk menganalisisnya. Saat ini, hampir semua perangkat yang kita gunakan untuk bermain (komputer, ponsel pintar, konsol permainan) terhubung ke Internet.

Dengan koneksi Internet, sangat mungkin bagi sebuah permainan untuk "menelepon ke rumah" dan mengirimkan semua data yang telah dikumpulkannya dari Anda saat Anda memainkannya, baik dalam bentuk mentah maupun agregat, ke server perusahaan pembuat permainan tersebut. Pengembang permainan kemudian dapat menjalankan semua jenis algoritma pada data tersebut untuk mengetahui hal-hal tentang Anda dan seluruh populasi pemainnya. Faktanya, sangat banyak mungkin sebagian besar? video game terbaru sudah melakukan hal ini.

Namun, hal-hal apa saja yang dapat dipelajari permainan dari Anda?

Apa yang Akan Anda Lakukan?

Sama seperti Anda dapat belajar dari sebuah gim cara memainkannya, gim tersebut juga dapat belajar dari para pemainnya bagaimana gim tersebut dimainkan. Dengan melihat riwayat bagaimana para pemain telah memainkan gim tersebut, kita dapat mengetahui apa yang biasanya dilakukan para pemain dalam setiap situasi. Informasi ini dapat digunakan untuk membuat AI yang memainkan gim seperti pemain "rata-rata" hanya dengan mengambil

tindakan yang paling sering dilakukan dalam setiap situasi. Untuk melihat bagaimana hal ini dapat dilakukan, bayangkan gim tersebut hanya menyimpan daftar panjang semua situasi yang pernah dialami pemain (dalam gim) dan tindakan yang dilakukan pemain dalam setiap situasi.

Mari kita asumsikan kita dapat menggambarkan situasi tersebut dengan beberapa angka; misalnya, koordinat pemain di dunia gim, kesehatan saat ini, posisi relatif karakter non-pemain (NPC) terdekat, dan sebagainya. Setelah kita menyimpan semua data ini dalam daftar panjang, menjadi mudah untuk membuat agen AI yang dapat memainkan gim seperti yang dilakukan pemain manusia. Di setiap titik waktu, cukup perhatikan situasi yang dihadapi karakter agen, temukan situasi tersebut dalam daftar panjang situasi yang dihadapi pemain, dan ambil tindakan yang diambil pemain tersebut. Sederhana dan elegan, bukan?

Ada dua masalah dengan solusi sederhana ini. Pertama, daftar semua situasi yang dihadapi pemain bisa bertambah panjang, sangat panjang jika Anda mencatat di mana pemain berada, katakanlah, sepuluh kali per detik dan pemain bermain selama sepuluh jam, dan sangat sangat panjang jika Anda ingin belajar bukan hanya dari satu pemain, tetapi mungkin ratusan atau jutaan. Panjangnya daftar tersebut menjadi masalah bukan hanya untuk menyimpannya dalam memori komputer, tetapi juga untuk dapat mencari salah satu situasi ini dengan cepat. Anda tentu tidak ingin melihat jutaan tindakan tersimpan yang berbeda setiap kali ingin mengetahui apa yang harus dilakukan. Kita membutuhkan cara yang lebih ringkas untuk menyimpan riwayat permainan lengkap seorang pemain (atau beberapa pemain).

Masalah lainnya adalah meskipun Anda menghabiskan sepuluh jam bermain sebuah game, Anda hampir pasti belum mengalami setiap kemungkinan situasi dalam game tersebut. Faktanya, meskipun Anda memiliki ratusan pemain dalam daftar Anda, Anda akan kehilangan banyak situasi potensial. Untuk setiap permainan yang tidak sepenuhnya sepele, jumlah kemungkinan status permainan yang berbeda akan menjadi angka yang sangat besar, mungkin lebih besar dari jumlah bintang di alam semesta. Anda juga memerlukan cara bagi agen Anda untuk menangani situasi yang tidak dihadapi pemain. Jadi, kita memerlukan cara untuk menggeneralisasi.

Untungnya, ternyata Anda dapat menggunakan algoritma backpropagation untuk melatih jaringan saraf untuk memprediksi apa yang akan dilakukan pemain. Ya, ini adalah metode yang sama yang saya jelaskan di bab sebelumnya ketika saya berbicara tentang belajar mengendarai mobil melalui coba-coba. Perbedaannya adalah di sini, kita menggunakan backpropagation bukan untuk pembelajaran penguatan tetapi untuk pembelajaran terawasi. Dalam pembelajaran terawasi, Anda memiliki daftar "instans", di mana setiap instans memiliki sejumlah fitur yang menggambarkan berbagai aspek instans, dan nilai target.

Saat belajar memainkan permainan seperti manusia, setiap contoh akan terdiri dari fitur-fitur yang menggambarkan situasi di mana agen pemain berada dan tindakan apa yang diambil pemain dalam situasi tersebut. Backpropagation kemudian digunakan untuk melatih jaringan saraf untuk mereproduksi daftar ini. Ingat bahwa dalam pembelajaran penguatan, algoritma backpropagation mengubah bobot jaringan saraf tergantung pada apakah tindakan yang diputuskan jaringan mengarah pada hasil yang baik atau buruk; dalam pembelajaran

terawasi, ia mengubah bobot tergantung pada apakah tindakan yang diputuskan jaringan saraf sama dengan apa yang diputuskan manusia.

Dengan menggunakan prinsip sederhana ini, jaringan saraf dapat dilatih untuk memprediksi tindakan apa yang akan diambil pemain dalam setiap situasi, biasanya dengan akurasi yang sangat baik. Keuntungan besar dari ini adalah bahwa jaringan saraf jauh lebih kecil daripada daftar panjang situasi dan tindakan yang digunakan untuk melatihnya, dan jauh lebih cepat untuk "meminta" jaringan saraf untuk suatu tindakan daripada mencari keadaan dalam tabel besar. Jaringan saraf seperti itu biasanya juga memiliki kemampuan yang cukup baik untuk melakukan generalisasi, artinya mereka dapat memberikan jawaban atas apa yang akan dilakukan agen dalam situasi yang tidak pernah benar-benar dihadapi pemain berdasarkan apa yang dilakukan pemain dalam situasi serupa.

AI dalam game menggunakan berbagai metode untuk mempelajari dan menyesuaikan diri dengan gaya bermain pemain, di antaranya:

1. Machine Learning dan Analisis Data
   Game modern mengadopsi Machine Learning (ML) untuk merekam dan mempelajari pola bermain pemain. Setiap kali bermain, AI mengumpulkan data tentang cara pemain bergerak, menyerang, dan bereaksi dalam situasi berbeda. Berdasarkan informasi ini, AI dapat mengatur tingkat kesulitan atau menyediakan tantangan yang lebih sesuai dengan kemampuan pemain.
2. Adaptive AI: Tantangan yang Menyesuaikan Diri
   Adaptive AI memungkinkan musuh dalam game belajar dan beradaptasi dengan pola permainan pemain. Misalnya, jika pemain sering memakai strategi bertahan, AI akan mencoba mengidentifikasi celah dalam pertahanan tersebut. Jika pemain mengikuti strategi tertentu, AI akan berusaha mengantisipasi dan memberikan tantangan baru agar permainan tetap menarik. Contoh penerapan Adaptive AI dapat ditemukan di game seperti Alien: Isolation, di mana musuh utama belajar dari cara pemain bersembunyi dan menyesuaikan taktiknya.
3. Neural Networks dan Prediksi Perilaku
   Dengan menggunakan Neural Networks, AI mampu memprediksi langkah pemain selanjutnya berdasarkan data yang telah dikumpulkan. Misalnya, dalam game balap, AI bisa memperkirakan rute yang akan diambil oleh pemain dan menyesuaikan pergerakan lawan untuk memberikan kompetisi yang lebih seimbang.

Di masa depan, kecerdasan buatan (AI) dalam dunia permainan diperkirakan akan semakin maju dan canggih. Teknologi seperti Deep Learning akan memungkinkan AI beradaptasi dengan lebih cepat, menciptakan pengalaman bermain yang semakin personal, dan bahkan mengubah alur cerita sesuai dengan keputusan yang diambil pemain. Seiring dengan kemajuan AI, kemungkinan munculnya NPC yang mampu berinteraksi layaknya manusia sejatinya semakin terbuka, memberikan pengalaman bermain yang lebih realistis dan imersif. Perkembangan AI dalam dunia game telah melampaui sekadar musuh yang berperilaku mengikuti skenario tetap menjadi teknologi yang mampu belajar dan menyesuaikan diri

dengan gaya bermain pemain. Masa depan AI menjanjikan pengalaman bermain yang makin nyata dan disesuaikan secara personal bagi setiap pemain.

6.2 TOKOH ATAU KARAKTER DALAM GAME

Bagi pengembang gim, penting untuk mengetahui siapa yang memainkan gim mereka: aspek apa yang mereka kuasai dan kurang kuasai, aspek apa yang mereka sukai dan tidak sukai, dan secara umum apa yang akan mereka lakukan dalam gim. Di luar dunia gim, pemasar menggunakan istilah seperti analisis kelompok sasaran dan segmentasi pasar ketika berbicara tentang mengidentifikasi dan mengkarakterisasi calon pelanggan suatu produk, sehingga perusahaan pembuat produk tersebut tahu cara menjual atau meningkatkannya. Dalam gim, kita berbicara tentang analisis tipe pemain.

Idenya adalah bahwa pemain gim dapat dikelompokkan ke dalam kelompok yang berbeda, atau tipe pemain, di mana pemain dari masing-masing tipe berperilaku serupa dan memiliki preferensi yang serupa. Upaya awal dan sangat berpengaruh untuk mengidentifikasi arketipe pemain dilakukan pada tahun 1980an oleh Richard Bartle, seorang pelopor gim multipemain daring.

Bartle mengembangkan pengamatannya terhadap pemain dalam gim daring berbasis teks *MUD* dan menetapkan empat tipe pemain: berprestasi, yang suka mengumpulkan poin dan maju dalam gim; penjelajah, yang gemar menjelajahi ruang permainan dan sistem aturannya, serta menemukan tempat baru atau menciptakan cara bermain baru; sosialisator, yang tertarik pada permainan daring karena kesempatan untuk bergaul dan mengobrol dengan orang lain; dan terakhir, pembunuh, yang senang menyakiti karakter dalam permainan pemain lain.

Jelas, tipologi ini paling cocok untuk jenis permainan yang dirancangnya: permainan multipemain daring. Meskipun kategori pencapaian dan penjelajah mudah diterapkan pada *Super Mario Bros.* dan *Angry Birds* (dan mungkin Catur, meskipun tidak jelas apa arti eksplorasi dalam permainan semacam itu), kategori sosialisator dan pembunuh tidak masuk akal untuk permainan satu atau dua pemain. Kemungkinan besar Anda perlu menemukan tipologi yang berbeda untuk setiap permainan atau setidaknya untuk setiap genre permainan. Untungnya, kita memiliki alat untuk melakukan ini sekarang, mengingat semua data yang dikumpulkan permainan tentang kita dan pemrosesan data modern serta teknik pembelajaran mesin. Dengan kata lain, permainan dapat mempelajari tipologi pemain dari pemain.

Pada tahun 2009, beberapa kolega saya di Universitas TI Kopenhagen (ITU), Alessandro Canossa, Anders Drachen, dan Georgios Yannakakis, berhasil mendapatkan harta karun berupa data pemain. Melalui kolaborasi dengan penerbit gim video Square Enix Eropa, mereka memperoleh akses ke data yang dikumpulkan dari sekitar satu juta pemain yang memainkan *Tomb Raider: Underworld* di XBox 360. Gim dalam waralaba *Tomb Raider* adalah gim aksi-petualangan di mana Anda memainkan satu karakter (petualang Lara Croft) dan menavigasi dunia tiga dimensi sambil memecahkan teka-teki dan melawan orang jahat dan terkadang monster (gambar 6.1).

Para pengembang telah memasukkan fungsionalitas dalam kode sehingga setiap kali pemain menyelesaikan level, gim tersebut menghubungi server Square Enix dan mengunggah sebagian informasi tentang bagaimana pemain memainkan level tersebut. Informasi ini mencakup berapa banyak waktu yang dihabiskan karakter pemain di berbagai bagian level, berapa banyak harta karun yang ditemukan, berapa banyak musuh yang dibunuh, dan seberapa sering pemain menggunakan sistem bantuan permainan, di antara hal-hal lainnya.

Ini adalah ide baru dan belum teruji pada tahun 2009 (pada tahun 2018, akan sulit menemukan permainan yang dirilis secara komersial yang tidak "menelepon" pengembang dengan informasi tentang cara memainkannya), dan oleh karena itu datanya agak kotor dan banyak pekerjaan yang diperlukan untuk membuatnya sedemikian rupa sehingga algoritma pembelajaran mesin dapat digunakan di dalamnya.

Gambar 6.1

Menyeimbangkan diri di tepian di *Tomb Raider*: *Underworld* (Crystal Dynamics, 2008).

Data yang telah dibersihkan dan ditata ulang dimasukkan ke dalam algoritma yang dikenal sebagai peta pengorganisasian mandiri. Ini adalah jenis jaringan saraf. Seperti yang dibahas di bab sebelumnya, ini adalah struktur komputasi yang terinspirasi oleh otak manusia, tetapi cara kerjanya agak berbeda dari jaringan pengemudi mobil yang dibahas di sana. Peta pengorganisasian mandiri mengambil sejumlah besar data dan memisahkan instans ke dalam kelompok yang berbeda sehingga instans dalam satu kelompok semaksimal mungkin serupa dan instans dalam kelompok yang berbeda sebisa mungkin berbeda satu sama lain.

Dalam bahasa pembelajaran mesin, ini disebut pengelompokan dan merupakan bentuk pembelajaran tanpa pengawasan (berbeda dengan pembelajaran terawasi atau pembelajaran penguatan). Anda tidak tahu sebelumnya berapa banyak kelompok yang akan Anda dapatkan; ini tergantung pada data dan, sampai batas tertentu, bagaimana Anda mengonfigurasi peta pengorganisasian mandiri. Dalam hal ini, setiap instans mewakili satu pemain dan berisi informasi yang dipilih dengan cermat tentang hal-hal apa yang telah dilakukan pemain selama permainan. Keluarlah empat kelompok data, yang mewakili empat jenis pemain.

Sekadar mengetahui bahwa ada empat tipe pemain tidak banyak memberi tahu kita. Sebagai pengembang, kita ingin tahu apa yang diwakili oleh tipe-tipe pemain tersebut dengan

kata lain, bagaimana pemain dari satu tipe berbeda dari tipe lainnya. Maka, tim mengamati sejumlah pemain representatif dari setiap tipe dan membandingkan seberapa banyak mereka telah melakukan setiap jenis tindakan.

Mereka mengidentifikasi empat tipe: veteran, yang jarang mati, mengumpulkan sebagian besar harta karun, dan umumnya memainkan permainan dengan sangat baik; pemecah masalah, yang jarang menggunakan sistem bantuan atau petunjuk apa pun, bermain lambat, dan lebih suka memecahkan semua teka-teki permainan sendiri; pelari, yang menyelesaikan permainan dengan sangat cepat tetapi sering meminta bantuan dan cenderung lebih sering mati; dan pasifis, yang pandai memecahkan teka-teki permainan tetapi buruk dalam elemen pertempuran dan tampaknya berusaha menghindarinya.

Tipologi ini jelas sangat berbeda dari Bartle, yang dapat dimengerti mengingat kita berhadapan dengan tipe permainan yang sangat berbeda dengan populasi pemain yang berbeda. Yang cukup menarik adalah para pengembang game di Square Enix tidak mengantisipasi keberadaan tipe pemain pasifis ketika mereka mengembangkan game tersebut, dan mereka terkejut mengetahui bahwa game tersebut dimainkan dengan cara yang tidak mereka "inginkan". Meskipun mengetahui tipe pemain seperti apa yang memainkan game Anda jelas bermanfaat, mungkin akan lebih bermanfaat lagi jika mengetahui apa yang akan dilakukan para pemain dalam game tersebut. Biasanya, Anda ingin para pemain Anda tetap bermain selama mungkin, karena pemain yang puas akan merekomendasikan game Anda kepada teman dan mungkin membeli game Anda berikutnya.

Hal ini juga umum terjadi pada game gratis yang awalnya gratis tetapi mengharuskan pembayaran semi-wajib untuk peningkatan agar dapat terus bermain. Bagi pengembang game semacam itu, sangat penting untuk dapat memprediksi pemain mana yang akan tetap bermain (dan akhirnya membayar) dan mana yang mungkin berhenti bermain. Mengapa? Karena ketika Anda mengetahui aspek desain mana yang membuat orang tetap bermain dan membayar, Anda dapat menyempurnakan game Anda untuk menghasilkan lebih banyak uang. Selain itu, sebagai pengembang game, Anda mungkin hanya tertarik untuk memahami para pemain Anda.

Tugas selanjutnya untuk tim yang sama, di mana saya sekarang menjadi anggota (saya baru saja pindah ke ITU untuk menduduki jabatan fakultas pertama saya), seperti halnya Tobias Mahlmann (salah satu mahasiswa Kuliah kami), adalah mencoba mempelajari aturan yang akan memprediksi perilaku pemain di kemudian hari dalam permainan dari perilaku pemain di awal permainan. Salah satu hal yang kami coba pelajari untuk diprediksi adalah level tertinggi yang akan diselesaikan pemain dari tujuh level dalam permainan.

Secara teoritis, ada banyak metode pembelajaran terawasi yang dapat digunakan untuk mempelajari cara memprediksi hal ini, tetapi beberapa lebih cocok daripada yang lain. Kami mencoba beberapa metode ini untuk memprediksi setelah level berapa pemain akan berhenti bermain. Salah satu metode yang paling berhasil adalah induksi pohon keputusan, sebuah metode yang juga memiliki keuntungan karena hasilnya mudah dipahami oleh manusia. Metode ini menghasilkan pohon keputusan, yang dapat dianggap sebagai daftar panjang

aturan jika-maka yang saling terkait. Berikut adalah contoh dari apa yang dipelajari algoritma tersebut:

```
IF Rewards on level 2 <18.5
THEN IF Time in Flushtunnel <9858: 2
ELSE (Time in Flushtunnel ≥9858): 3
ELSE (Rewards on level 2 ≥18.5): 7
```

Dengan kata lain, jika Anda mengumpulkan skor rendah di level 2 dan hanya menghabiskan sedikit waktu di Flush Tunnel (area di level 2), Anda akan berhenti bermain setelah level 2 dan tidak akan pernah menyelesaikan level 3. Jika tidak, Anda akan berhenti bermain setelah level 3. Namun, jika Anda mengumpulkan skor tinggi di level 2, Anda akan menyelesaikan seluruh permainan.

Ini tentu terdengar seperti aturan yang sangat konyol dan sewenang-wenang. Aturan ini tampak sama masuk akalnya dengan astrologi, dan bukan jenis aturan yang Anda harapkan akan dibuat oleh perancang gim manusia sungguhan. Namun, meskipun konyol, aturan ini dibangun di atas bukti empiris yang kuat: akurasi prediksinya mencapai 76,7 persen setelah diuji pada puluhan ribu pemain. Ini berarti bahwa meskipun memang ada beberapa orang yang mendapatkan hadiah rendah di level 2 dan kemudian terus menyelesaikan seluruh permainan, secara statistik hal ini kecil kemungkinannya.

Meskipun mungkin terdengar menghina akal sehat bahwa jumlah waktu yang dihabiskan di terowongan menjadi indikasi yang menentukan apakah seorang pemain akan menyerah setelah level 2 atau 3, hal ini tampaknya memang demikian berdasarkan semua data ini. Mungkin hasil yang paling penting adalah akurasi prediksinya yang sangat tinggi. Hal ini menunjukkan bahwa kita manusia sebenarnya cukup mudah ditebak, bahkan ketika kita bermain gim.

Siapakah Anda di Luar Gim?

Sejauh ini kita telah melihat bahwa gim dapat belajar dari permainan Anda, tipe pemain seperti apa Anda dan bagaimana Anda akan bermain di masa mendatang. Namun, Anda bukan hanya seorang pemain gim. Anda adalah manusia seutuhnya, dengan harapan, impian, ketakutan, tata krama, teman, dan kebiasaan. Tidak ada alasan untuk percaya bahwa seluruh diri Anda yang lain lenyap begitu Anda bersandar di sofa dan meraih pengontrol Xbox; Anda tetaplah diri Anda, meskipun untuk sesaat Anda menjadi Mario, Master Chief, atau Lara Croft. Sekarang pertanyaannya adalah, apakah ada bagian dari diri Anda yang lain yang terpancar dalam permainan Anda? Apa yang bisa dipelajari permainan ini tentang diri Anda yang sebenarnya dari analisis cara bermain Anda?

Pada tahun 2013, Alessandro dan saya memiliki seorang mahasiswa magister yang ambisius, Josep Martinez, dan kami sedang mencari topik untuk tesisnya. Alessandro baru-baru ini membaca karya Stephen Reiss, seorang psikolog kepribadian yang telah merancang model untuk mengkategorikan motif hidup seseorang, yaitu, apa yang memotivasi mereka dalam hidup. Reiss mengidentifikasi enam belas motif hidup yang luas (dalam urutan abjad):

penerimaan, rasa ingin tahu, makan, keluarga, kehormatan, idealisme, kemandirian, ketertiban, fisik, kekuasaan, asmara, menabung, sosial, status, ketenangan, dan balas dendam. Masing-masing motif ini memiliki beberapa subkategori, dan tersedia kuesioner yang telah teruji dengan baik untuk menilai motif hidup.

Kami bertanya-tanya apakah motif yang dimiliki orang-orang dalam kehidupan nyata juga diekspresikan dalam permainan. Jika ya, motif yang mana? Dan dalam permainan yang mana? Seperti banyak orang lainnya, Alessandro, Josep, dan saya terpesona oleh *Minecraft*, gim dunia terbuka yang menggemparkan dunia sejak tahun 2010. Ketika pertama kali dirilis, sebagai versi beta yang penuh bug, *Minecraft* adalah gim yang cukup unik, kini ada banyak tiruannya bukan hanya karena grafisnya yang khas blok, tetapi juga karena kebebasan tak tertandingi yang diberikannya kepada pemain.

Gim ini sekarang menjadi fenomena global yang digunakan untuk segala hal, mulai dari pembuatan machinima (film animasi yang dibuat di dalam gim video), pendidikan, hingga pengujian algoritma AI. *Minecraft* dapat digambarkan sebagai persilangan antara gim peran dan versi digital Lego (gambar 6.2). Ketika Anda tiba di dalam gim, Anda tidak memiliki apa-apa, dan Anda harus bergegas merakit beberapa peralatan agar Anda dapat membangun tempat berlindung sebelum malam tiba dan monster mulai berkeliaran di daratan. Namun, untuk membuat peralatan ini, Anda membutuhkan material, dan untuk mendapatkannya, Anda perlu menambang tanah.

Setelah membuat peralatan yang lebih canggih, Anda akan dapat menambang lebih dalam untuk material yang lebih eksotis sehingga Anda dapat membangun bangunan dan mekanisme yang lebih canggih. Dengan waktu dan usaha yang cukup, Anda dapat membangun apa pun yang Anda inginkan. Mencari video *Minecraft* di YouTube akan menghasilkan ribuan contoh replika bangunan dan kendaraan terkenal yang dibuat oleh pemain (bahkan *Starship Enterprise*). Ada juga alur cerita di *Minecraft*, termasuk misi-misi yang cukup umum seperti dalam permainan peran, tetapi sepenuhnya opsional untuk mengikuti alur cerita ini dan menyelesaikan misi-misi tersebut; banyak pemain tidak melakukannya.

Gambar 6.2

Dunia kubus *Minecraft* (Mojang, 2011). Permainan ini sebagian besar berpusat pada penambangan kubus untuk mendapatkan material sehingga Anda dapat membangun sesuatu dari kubus lain.

Hampir semua permainan menawarkan sejumlah gaya bermain yang berbeda, tetapi *Minecraft* menawarkannya lebih banyak daripada kebanyakan permainan lainnya. Saya rasa bisa dikatakan bahwa ada lebih banyak cara berbeda untuk bermain *Minecraft* daripada cara bermain *Tomb Raider: Underworld*. Jelas bahwa gaya bermain yang berbeda ini mencerminkan motivasi dalam permainan yang berbeda: beberapa orang termotivasi untuk menyelesaikan misi, yang lain mengekspresikan diri dengan membangun bangunan megah, dan yang lainnya dengan mengumpulkan sumber daya langka. Namun, apakah motivasi-motivasi ini ada hubungannya dengan motif kehidupan nyata Anda? Apakah seseorang yang paling peduli dengan keluarganya bermain berbeda dari seseorang yang perhatian utamanya adalah meraih kesuksesan dalam kehidupan profesional? Kami memutuskan untuk mencari tahu.

Josep mengirimkan kuesioner, dengan pertanyaan yang diambil dari Profil Motivasi Reiss, kepada 100 pemain *Minecraft*; Kuesioner-kuesioner ini digunakan untuk menyusun profil setiap pemain berdasarkan motivasi mereka. Ia kemudian meminta setiap pemain untuk menunjukkan berkas log *Minecraft* mereka. Berkas kecil ini disimpan secara otomatis oleh permainan, berisi lebih dari enam ratus variabel, termasuk hal-hal seperti berapa jam pemain telah bermain, berapa banyak bijih redstone yang telah ditambang (redstone digunakan untuk membuat sirkuit seperti listrik), dan seberapa jauh mereka telah bepergian dengan pig (pilihan transportasi yang sering terabaikan). Setelah mengekstrak dan membersihkan data ini, kami menjalankan analisis korelasi dari semua kombinasi variabel permainan yang berpotensi relevan dan motif hidup.

"Korelasi" adalah cara untuk mengatakan bahwa, secara statistik, dua hal saling berkaitan. Hal ini tidak selalu berarti bahwa yang satu menyebabkan yang lain: jika penjualan payung dan jumlah jam yang dihabiskan untuk menonton TV berkorelasi sepanjang minggu dalam setahun, keduanya mungkin disebabkan (setidaknya sebagian) oleh cuaca buruk. Dua variabel dapat berkorelasi negatif atau positif. Begitu pula, misalnya, merokok berkorelasi negatif dengan umur panjang: ketika salah satu tinggi, yang lain rendah. (Dalam hal ini, masuk akal untuk berasumsi bahwa yang satu menyebabkan yang lain.)

Kami menemukan bahwa semua motif hidup berkorelasi signifikan dengan beberapa variabel dalam permainan. Namun, beberapa hanya berkorelasi dengan beberapa variabel (sangat sedikit sehingga mungkin bergantung pada kebetulan), sementara yang lain berkorelasi dengan sejumlah besar variabel dalam permainan, dan beberapa korelasinya begitu kuat sehingga hampir tidak ada ruang untuk keraguan.

Di antara motif hidup yang paling berkorelasi adalah rasa ingin tahu, menabung, balas dendam, dan kehormatan, sedangkan yang tampaknya tidak banyak diekspresikan dalam permainan adalah romansa, ketenangan, dan aktivitas fisik. Dalam beberapa kasus, korelasi ini masuk akal secara intuitif bagi seseorang yang mengetahui permainan; dalam kasus lain, korelasi ini tidak terduga dan cukup lucu. Orang-orang yang sangat termotivasi oleh rasa ingin tahu dalam kehidupan nyata cenderung membuat banyak obor dan peralatan batu dalam permainan, yang masuk akal karena ini adalah cara paling hemat biaya untuk menjelajahi sebagian besar dunia permainan. Mereka yang termotivasi oleh menabung cenderung

menggunakan material yang murah dan sederhana dalam bangunan dan peralatan yang mereka bangun.

Pemain yang pendendam tampaknya lebih sering berhenti bermain dan memulainya kembali (mungkin dari penyimpanan sebelumnya) yang disebut "rage quit" dalam istilah gamer. Pemain yang sangat termotivasi oleh kemandirian dalam kehidupan nyata menunjukkan hal ini dalam permainan dengan menolak menyelesaikan misi dalam alur cerita permainan; khususnya, hal ini berkorelasi kuat dengan tidak mencoba menyelesaikan misi terakhir. Ekspresi menarik lainnya dari motif hidup adalah orang-orang yang sangat membutuhkan ketenangan membangun lebih banyak pagar di sekitar tempat tinggal mereka. Tampaknya orang yang Anda mainkan saat bermain *Minecraft* adalah Anda dalam beberapa hal yang sangat penting.

Hasil ini dapat dilihat berdasarkan studi oleh Nick Yee, yang saat itu berada di Universitas Stanford, dan rekan-rekannya, yang menyelidiki bagaimana pemain mengekspresikan kepribadian mereka (alih-alih motif hidup) dalam permainan peran multipemain daring *World of Warcraft* (gambar 6.3). Yee menggunakan kuesioner kepribadian Big Five, yang mengelompokkan ciri-ciri kepribadian ke dalam lima kategori: Keterbukaan, Kehati-hatian, Ekstroversi, Keramahan, dan Neurotisisme. Ada banyak korelasi dalam data ini juga, dan dia dapat melihat, misalnya, bahwa pemain yang teliti lebih cenderung mengumpulkan berbagai jenis barang dan lebih kecil kemungkinannya meninggal karena kecelakaan, bahwa pemain dengan keterbukaan tinggi lebih banyak menjelajahi dunia permainan, dan bahwa pemain yang ekstrovert (tidak mengherankan) memiliki lebih banyak interaksi sosial dalam permainan.

Sebuah kelompok yang dipimpin oleh Pieter Spronck dan termasuk Shoshanna Tekofsky di Universitas Tilburg juga menemukan efek serupa dalam permainan yang berbeda seperti permainan strategi epik *Civilization* dan penembak orang pertama *Battlefield 4*. Misalnya, dimungkinkan untuk memprediksi jenis kelamin dan usia dengan akurasi yang relatif baik dari cara orang bermain *Battlefield 4.*

Gambar 6.3

World of Warcraft (Blizzard, 2004) adalah permainan peran daring multipemain masif; sebagian besar permainan berkomunikasi dengan pemain lain melalui teks atau obrolan suara.

Secara keseluruhan, gambaran yang kita dapatkan dari penelitian ini, serta banyak studi lain tentang topik ini, adalah bahwa Anda mengekspresikan diri Anda cukup banyak saat bermain gim. Jika gim tersebut mau, ia dapat mengetahui tidak hanya siapa Anda dalam gim dan bagaimana Anda akan bermain di masa mendatang, tetapi juga banyak hal tentang siapa Anda di luar gim. Hal ini memunculkan banyak peluang menarik tidak hanya bagi pengembang gim tetapi juga bagi psikolog dan ilmuwan sosial lainnya yang ingin memahami bagaimana manusia berfungsi.

Penelitian ini juga memunculkan sejumlah pertanyaan rumit. Beberapa tahun yang lalu, saya berbicara di sebuah konferensi yang dihadiri oleh sejumlah orang dari dinas keamanan dan lembaga pemerintah lainnya. Salah satu hal yang saya bicarakan adalah seberapa banyak Anda dapat mengetahui tentang pemain dari perilaku mereka dalam gim. Untuk memicu percakapan, saya menyarankan bahwa mungkin saja untuk mengetahui informasi yang sangat sensitif tentang pemain, seperti pandangan politik, seksualitas, riwayat penggunaan dan penahanan narkoba, atau status kesehatan mereka. Saya berharap akan mendapat beberapa reaksi khawatir, tetapi orang-orang ini malah mengangguk termenung, seolah berkata, "Itu ide yang menarik."

Wajar untuk mengatakan bahwa kekhawatiran saya tentang potensi penggunaan pemodelan pemain untuk tujuan jahat tidak berkurang sejak saat itu. Terutama mengingat kekhawatiran tentang seberapa banyak informasi pribadi kita dikumpulkan oleh layanan keamanan, perusahaan jejaring sosial, penyedia internet, dan segala macam operator curang yang menjual layanan mereka kepada penawar tertinggi, saya pikir penting untuk menyadari bahwa permainan kita adalah cara lain kita meninggalkan jejak digital yang kaya. Perbedaannya, mungkin, adalah ketika kita mengunggah di jejaring sosial, kita sadar bahwa kita berbagi informasi tentang diri kita sendiri; ketika kita bermain gim, hal ini tidak terlihat jelas karena kita yakin kita hanya bertindak di dalam dunia gim. Namun, seperti yang telah kita lihat, kita membawa banyak diri kita ke dunia itu.

BAB 7
MENGOTOMATISKAN KREATIVITAS

Ada Lovelace secara luas dianggap sebagai programmer pertama di dunia. Ia adalah orang pertama yang menulis program untuk Mesin Analitik Charles Babbage, sebuah komputer mekanis yang sangat ambisius namun tak pernah terwujud, pada pertengahan abad ke-19. Ia juga termasuk orang pertama yang menunjukkan potensi mesin komputasi yang sungguh luar biasa. Namun, menurutnya, "Mesin Analitik sama sekali tidak berpretensi untuk menciptakan apa pun. Ia dapat melakukan apa pun yang kita tahu bagaimana memerintahkannya untuk dilakukan."

Dalam satu setengah abad terakhir, kita telah menyaksikan kemajuan pesat dalam komputasi, khususnya sejak penemuan komputer digital yang sebenarnya. Namun, yang mengejutkan, banyak yang masih mempercayai hal-hal seperti berikut: Meskipun kita dapat membuat komputer bermain gim, memprediksi apa yang akan dilakukan pemain, dan bahkan mengaitkan perilaku pemain tertentu dengan karakteristik kepribadian, komputer tidak akan pernah dapat merancang gim itu sendiri. Untuk itu, kita membutuhkan kreativitas manusia karena komputer pada dasarnya tidak akan pernah dapat menciptakan sesuatu yang tidak diprogramkan oleh kita, manusia, untuk dilakukan terlebih dahulu.

Ini sepenuhnya salah dan merupakan salah satu mitos paling berbahaya yang tersebar luas tentang komputasi dan kecerdasan buatan. Meskipun "mengotomatiskan kreativitas" mungkin terdengar seperti sebuah kontradiksi bagi sebagian orang, kreativitas sebenarnya tidak lebih atau kurang dapat diotomatisasi dibandingkan kemampuan kognitif manusia lainnya. Dalam bab ini, saya membahas beberapa cara di mana metode kecerdasan buatan dapat digunakan untuk melakukan hal-hal yang akan disebut "kreatif" jika dilakukan oleh manusia, khususnya dalam hal merancang gim. Saya juga akan melihat bagaimana kita dapat menggunakan kecerdasan buatan untuk meningkatkan kreativitas kita sendiri dalam tugas-tugas desain semacam itu. Namun, pertama-tama, mari kita mundur satu dekade.

Pada tahun 2006, saya telah menempuh program Kuliah selama dua tahun, dan saya telah menerbitkan beberapa makalah tentang cara-cara mengembangkan jaringan saraf untuk mengendarai mobil atau memainkan gim lain. Makalah-makalah ini diterima dengan baik oleh komunitas riset, tetapi tidak inovatif. Saya telah menunjukkan bahwa neuroevolusi dapat bekerja dengan baik untuk jenis gim ini, tetapi pada akhirnya, yang saya lakukan hanyalah mengambil metode yang terkenal dalam robotika dan menunjukkan cara membuatnya bekerja untuk jenis gim tertentu. Saya bertanya-tanya apa langkah selanjutnya dalam proyek doktoral saya. Salah satu ide yang saya pertimbangkan adalah mencoba menggunakan informasi yang lebih kompleks, seperti data visual mentah, sebagai input ke jaringan saraf tiruan, tetapi ini tampaknya kurang menarik. Namun suatu hari, ketika saya sedang berpikir di kamar mandi, saya mendapat ide lain. Algoritma evolusioner ternyata sangat berguna untuk menciptakan agen (yang diimplementasikan sebagai jaringan saraf tiruan) yang dapat memainkan sebuah

permainan. Tetapi bisakah Anda menggunakan prinsip yang sama, evolusi, untuk menciptakan bagian lain dari permainan misalnya, level?

Saya menyampaikan ide ini kepada teman saya Renzo De Nardi, yang menganggapnya menarik dan setuju untuk membantu. Karena ada konferensi yang sesuai dan tenggat waktunya tinggal seminggu lagi, kami pikir kami akan punya cukup waktu untuk menyempurnakan konsep, menulis kode, merancang dan menjalankan eksperimen, serta menulis makalah. (Optimisme yang tak berdasar dan kemauan untuk bekerja sepanjang malam merupakan aset yang berguna saat menempuh pendidikan doktoral.) Kami memilih gim balap yang sama yang saya buat untuk eksperimen saya sebelumnya dalam mengembangkan jaringan saraf untuk mengendarai mobil karena kami sudah tahu cara kerja kode tersebut.

Kami langsung menghadapi dua masalah: bagaimana merepresentasikan lintasan balap agar evolusi dapat mencari lintasan yang bagus secara efektif dan bagaimana membuat fungsi kebugaran yang secara akurat memberi tahu kita seberapa "bagus," atau menyenangkan, sebuah lintasan balap.

Masalah pertama tidak sepenuhnya sepele tetapi juga tidak terlalu sulit untuk dipecahkan. Kami merepresentasikan lintasan menggunakan teknik yang disebut *b-splines*, di mana lintasan dapat dideskripsikan dengan serangkaian angka yang menentukan bagaimana lintasan tersebut melengkung. Jadi, seperti halnya jaringan saraf, "genom" lintasan balap hanyalah daftar angka.

Masalah kedua jauh lebih rumit dan langsung memunculkan masalah mendasar dalam estetika. Bagaimana kita tahu bahwa lintasan balap, atau jenis level permainan lainnya, atau jenis konten permainan apa pun, itu bagus? Jika kita mencoba sedikit lebih spesifik, bagaimana kita bisa menulis kode program yang secara otomatis mengevaluasi lintasan balap dan mengembalikan angka yang sesuai dengan seberapa seru, atau menarik, atau menghiburnya lintasan balap tersebut menurut manusia saat bermain gim balap? Sekilas, ini tampak seperti tugas yang mustahil.

Bagaimana kita bisa tahu apa yang akan dipikirkan manusia tentang level permainan tanpa kehadiran manusia itu? Kita harus mensimulasikan manusia secara keseluruhan dan menanyakan simulasi apa yang dipikirkannya sesuatu yang, secara halus, jauh di luar kemampuan teknis kita. Jika Anda masih belum yakin betapa sulitnya hal ini, bayangkan menulis sebuah program yang akan melihat lukisan dan memberinya skor antara 1 dan 10 yang mencerminkan seberapa besar seorang kritikus seni profesional akan menyukai lukisan tersebut. Sulit membayangkan harus mulai dari mana. Masalah seperti ini terkadang disebut *AI-complete problems*, yang mencerminkan gagasan bahwa pertama-tama kita perlu mengembangkan AI tingkat manusia secara umum untuk dapat menyelesaikannya.

Seperti halnya banyak soal sulit lainnya, ternyata Anda bisa membuat banyak kemajuan jika Anda tidak terlalu peduli untuk menyelesaikan semuanya dengan tepat, dan hanya mencoba mendapatkan perkiraan kasar. Dalam kasus kami, kami mempelajari teori desain gim dan juga memainkan beberapa gim balap sendiri untuk melihat apakah kami dapat menemukan beberapa aturan sederhana yang menunjukkan bahwa satu lintasan balap lebih

baik daripada yang lain. Kami menemukan aturan heuristik berikut untuk menentukan apa yang membuat lintasan balap yang baik:

- Harus memiliki tingkat kesulitan yang tepat.
- Harus memiliki berbagai jenis tantangan di sepanjang putaran, seperti beberapa tikungan tajam dan beberapa tikungan halus.
- Pada titik tertentu di lintasan, seharusnya memungkinkan untuk melaju dengan sangat cepat.

Bagaimana kita bisa mengukur apakah suatu lintasan memiliki sifat-sifat ini? Nah, cara paling sederhana adalah dengan mengendarai lintasan dan melihat apa yang terjadi. Kita bisa melihat apakah pengemudi berhasil menyelesaikan putaran, perbedaan kecepatan minimum dan maksimum di sepanjang lintasan (yang menunjukkan adanya berbagai jenis tantangan), dan kecepatan maksimum yang dicapai. Kita kemudian dapat membuat fungsi kebugaran yang mencerminkan ketiga nilai ini, yang memungkinkan algoritma evolusioner untuk mencari lintasan dengan ketiga properti yang telah kita sebutkan.

Masalah yang tersisa adalah seseorang perlu mengemudikan mobil. Kita tidak bisa membiarkan manusia sungguhan mengemudi karena algoritma evolusioner perlu mencoba ribuan atau bahkan puluhan ribu lintasan berbeda dengan sedikit variasi, dan manusia terlalu lambat untuk itu dan juga mudah lelah. Kita membutuhkan pemain buatan untuk mengemudikan lintasan kita guna mengevaluasi kualitasnya.

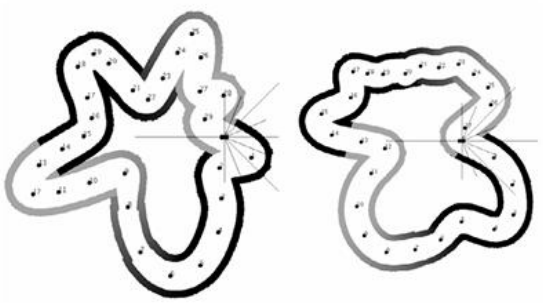
Untungnya, saya telah mengembangkan jaringan saraf untuk mengemudikan mobil dalam gim balap khusus ini, sehingga kita dapat menggunakan driver jaringan saraf tersebut untuk menguji lintasan. Lebih baik lagi, kami dapat melatih jaringan saraf untuk mengemudi seperti kami, sehingga kami dapat mengembangkan trek balap yang sesuai dengan gaya mengemudi kami, menggunakan kombinasi algoritma evolusioner dan metode backpropagation yang saya jelaskan di bab sebelumnya. Renzo dan saya melatih beberapa jaringan saraf untuk mengemudi seperti kami.

Sesuai dengan itu, jaringan yang dilatih berdasarkan data dari permainan saya mengemudi dengan cepat dan ugal-ugalan, sementara jaringan yang dilatih berdasarkan permainan Renzo mengemudi dengan lambat dan cermat. (Perhatikan bahwa ini hanya mencerminkan gaya mengemudi kami dalam gim balap; dalam kehidupan nyata, saya sebenarnya tidak memiliki SIM.) Kami kemudian mengembangkan trek balap agar sesuai dengan masing-masing gaya mengemudi kami. Hasilnya ditunjukkan pada Gambar 7.1.

Sejak itu, saya telah bekerja dengan berbagai tim untuk menerapkan ide umum ini menciptakan level permainan baru dengan evolusi, menggunakan agen yang memainkan level tersebut untuk mengevaluasinya ke berbagai permainan dan jenis level. Misalnya, kami menunjukkan bahwa kami dapat secara otomatis membuat peta yang seimbang untuk *StarCraft* dan level untuk *Super Mario Bros*.

Kami menyebut ide umum ini "pembuatan konten prosedural berbasis pencarian", karena konten permainan dihasilkan melalui proses pencarian dalam hal ini, berdasarkan evolusi buatan. Yang dibutuhkan hanyalah cara yang baik untuk merepresentasikan konten

permainan dan fungsi kebugaran yang baik (meskipun, seperti yang telah kita lihat, bagian ini bisa rumit).



Gambar 7.1
Lintasan balap yang berevolusi dari jaringan saraf.

Gagasan memandang kreativitas sebagai pencarian dalam ruang artefak potensial bukanlah hal baru; gagasan ini telah dibahas panjang lebar oleh, misalnya, filsuf Inggris Margaret Boden. Ada juga banyak contoh pendekatan berbasis pencarian untuk menghasilkan musik, gambar, dan sebagainya. Dengan memandang kreativitas sebagai pencarian, menjadi jelas bahwa kreativitas sama terotomatisasinya dengan upaya lain yang biasanya membutuhkan pemikiran manusia: sama sekali tidak mudah, tetapi jelas bukan tidak mungkin.

7.1 DEWA ANGKA ACAK

Gagasan untuk membuat beberapa bagian permainan secara otomatis, melalui algoritma, juga bukanlah hal baru. Bahkan, gagasan ini hampir setua permainan video itu sendiri. Dahulu kala, ketika daya komputasi merupakan sumber daya langka yang hanya tersedia di komputer mainframe yang harus Anda gunakan bersama ratusan atau ribuan komputer lain, dan bahkan mainframe tersebut memiliki kecepatan pemrosesan yang jauh lebih rendah dan kapasitas penyimpanan yang jauh lebih kecil daripada ponsel saat ini, menghemat byte adalah hal yang sangat penting.

Lingkungan inilah yang digunakan Michael Toy dan Glenn Wichmann untuk menciptakan *Rogue* pada tahun 1980 (gambar 7.2). Toy dan Wichmann, yang sedang kuliah di Universitas California di Santa Cruz, adalah penggemar berat permainan peran pena dan kertas *Dungeons and Dragons* yang berpengaruh. Biasanya, kampanye *Dungeons and Dragons* membutuhkan seorang master dungeon khusus yang menjalankan permainan dan memainkan berbagai peran NPC di dunia permainan tempat sebuah tim yang terdiri dari satu atau beberapa pemain bertualang. Toy dan Wichmann bertanya-tanya bagaimana mereka bisa menciptakan permainan komputer yang sedikit mirip dengan *Dungeons and Dragons* tetapi bisa dimainkan sendiri, melawan komputer. Menerjemahkan mekanisme pertarungan dari *Dungeons and Dragons* ke dalam kode cukup mudah. Masalahnya terletak pada petualangannya yang dalam *Dungeons and Dragons*, cukup sering terjadi di dungeon. Dalam *Dungeons and Dragons*, dungeon-dungeon ini dibuat oleh master dungeon atau dibeli dari buku yang dijual oleh penerbit game.

Menyelesaikan dungeon melibatkan menavigasi labirin untuk menemukan jalan keluar; mengumpulkan item; mengelola kesehatan, makanan, dan uang; dan melawan (dan/atau melarikan diri dari) monster. Untuk game *Rogue* yang mereka ciptakan, mereka tidak ingin membuat ruang bawah tanah sendiri karena mereka menciptakan game tersebut terutama untuk diri mereka sendiri dan akan lebih menyenangkan untuk dikejutkan oleh ruang bawah tanah tersebut. Bagaimanapun, mereka tidak memiliki ruang disk untuk menyimpan banyak ruang bawah tanah untuk game tersebut, sehingga membuatnya secara manual praktis mustahil.

Kebutuhan konon merupakan ibu dari segala penemuan, sehingga Toy dan Wichmann terpaksa menciptakan cara untuk menghasilkan ruang bawah tanah secara otomatis. Setiap kali game *Rogue* baru dimulai, ruang bawah tanah yang benar-benar baru dibuat, dan prosedur ini cepat bahkan pada komputer era 1980an. Algoritmenya, yang agak disederhanakan di sini, bekerja sebagai berikut.

Pertama, bagi ruang bawah tanah menjadi beberapa segmen; lalu buat ruangan di semua segmen ruang bawah tanah; tandai ruangan pertama yang dikunjungi; lalu terus buat koridor dari ruangan yang dikunjungi (dipilih secara acak) ke ruangan yang belum dikunjungi (dipilih secara acak) hingga semua ruangan ditandai sebagai telah dikunjungi. Ini akan menciptakan sejumlah ruangan yang dihubungkan oleh koridor, sehingga memungkinkan untuk berpindah dari ruangan mana pun ke ruangan lainnya.

Gambar 7.2

Rogue (A. I. Design, 1980), roguelike orisinal, memiliki persyaratan perangkat keras yang sederhana karena dikembangkan untuk komputer yang daya komputasinya lebih rendah daripada kulkas Anda. Wajah tersenyum mewakili karakter pemain.

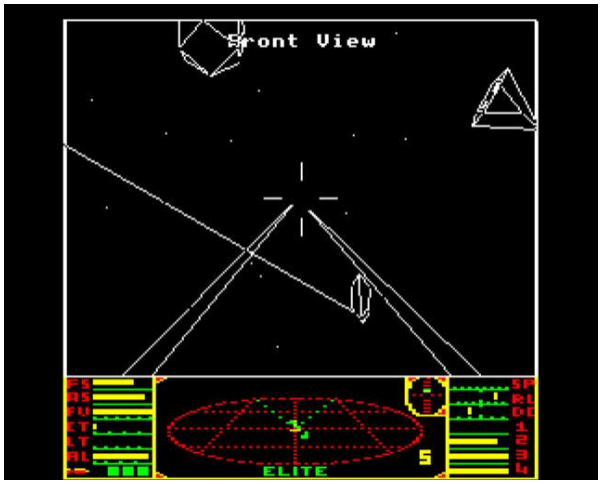
Yang tersisa hanyalah menambahkan item dan monster; ini sebagian besar tersebar secara acak di dalam ruangan.

Hasil dari proses ini adalah setiap permainan *Rogue* terasa baru. Setiap kali Anda mulai memainkannya, ada ruang bawah tanah baru yang harus Anda masuki, ramuan baru untuk dipecahkan, monster baru untuk dibunuh. Ini berarti bahwa untuk menjadi mahir dalam memainkannya, Anda perlu mempelajari strategi untuk bermain dengan baik, alih-alih hanya

menghafal tata letak ruang bawah tanah. Ini adalah jenis permainan yang secara kualitatif berbeda dibandingkan dengan banyak jenis permainan lainnya.

Meskipun *Rogue* bukanlah permainan pertama yang menyertakan level yang dihasilkan secara prosedural (perbedaan itu mungkin dimiliki oleh *Beneath Apple Manor* yang sedikit lebih awal dan lebih sederhana), popularitasnya mendorong terciptanya genre permainan baru yang disebut "roguelike" untuk menghormati permainan penting tersebut. Menurut definisi, setiap sesi permainan roguelike menghadirkan serangkaian level baru untuk dimainkan. Banyak juga yang merupakan game role-playing seperti *Rogue*, misalnya, game open source klasik *NetHack*, *Dwarf Fortress* yang rumit tanpa akhir, atau seri *Diablo* terlaris dari Blizzard.

Namun, banyak game lain yang tidak umum dianggap sebagai roguelike justru membangun ide untuk menghasilkan level, atau dunia, di awal setiap sesi permainan. Ini termasuk beberapa game yang sangat terkenal; platformer sensasional indie *Spelunky*, game strategi epik dalam seri *Civilization*, atau game sandbox terlaris *Minecraft*; semua game ini tidak akan mungkin hadir dalam format seperti saat ini tanpa pembuatan prosedural.



Gambar 7.3
Mendekati stasiun luar angkasa di *Elite* (Acornsoft, 1984).

Bahasa Indonesia: Permainan lain dari tahun 1980an yang mempelopori pembuatan konten prosedural adalah *Elite* (gambar 7.3), sebuah permainan perdagangan dan pertempuran luar angkasa tempat pemain menjelajahi galaksi dengan ribuan planet, membeli dan menjual barang dan benda, melawan bajak laut luar angkasa, dan mengambil misi. Skala permainan ini besar menurut standar apa pun, tetapi yang lebih mengesankan adalah perangkat keras yang sangat terbatas yang dijalankannya.

Saya memainkannya di Commodore 64 saya, komputer rumah populer dengan memori 64 kilobyte. Bagaimana mungkin dunia permainan yang begitu besar dapat masuk ke dalam memori 65.536 karakter, terutama mengingat bahwa mesin permainan dan grafik perlu disimpan di sana juga? Jawabannya adalah bahwa setiap sistem bintang dihasilkan saat Anda mengunjunginya, termasuk nama dan posisi planet dan stasiun luar angkasa, harga komoditas,

dan lokasi pesawat luar angkasa. Namun tidak seperti *Rogue*, permainan tidak berubah dengan setiap sesi permainan, dan jika Anda kembali ke sistem bintang yang telah Anda tinggalkan, Anda akan mendapati tampilannya sama seperti saat Anda meninggalkannya. Ini karena *Elite* menggunakan nilai benih tertentu untuk generator angka acak untuk setiap sistem bintang, yang secara tepat menentukan keluaran algoritma generatif.

Alih-alih menyimpan ribuan sistem bintang, *Elite* hanya menyimpan nomor setiap sistem bintang dan menggunakannya untuk meregenerasi sistem bintang sesuai kebutuhan. Ide ini sangat berpengaruh dalam pengembangan game dan digunakan dalam berbagai peran di banyak game untuk meregenerasi hal-hal seperti vegetasi sesuai permintaan. Contoh terbaru yang menonjol dari game yang menyimpan galaksi lengkap sebagai nilai benih, yang menciptakan ruang raksasa untuk dijelajahi pemain, adalah *No Man's Sky* (gambar 7.4).

Gambar 7.4

Dalam *No Man's Sky* (Hello Games, 2016), semua planet dihasilkan secara prosedural, termasuk flora, fauna, dan geologinya.

Jadi, apakah ini berarti masalah pembuatan konten prosedural sudah terpecahkan? Jauh dari itu. Masalahnya dapat diilustrasikan oleh Dewa Angka Acak. Saat memulai permainan Rogue, sebagian besar waktu Anda akan menemukan ruang bawah tanah dengan tingkat kesulitan yang wajar, tetapi terkadang Anda bisa mendapatkan lonjakan kesulitan yang gila atau rangkaian panjang dengan sedikit tantangan. Selain itu, beberapa ruang bawah tanah memang (jauh) lebih baik daripada yang lain. Jadi, sudah menjadi hal yang umum untuk menyalahkan hasil yang tidak adil dalam *Rogue* (atau roguelike lainnya) pada Dewa Angka Acak, yang jelas-jelas ingin menjatuhkan pemain dengan memunculkan karakternya tepat di sebelah naga tingkat tinggi, atau di ruangan yang hampir mustahil untuk keluar, atau hal lain seperti itu.

Masalahnya adalah sangat sulit untuk memastikan bahwa jenis algoritma yang digunakan oleh permainan seperti *Rogue* sangat akurat. Dengan kata lain, sulit untuk memastikan bahwa level yang keluar di akhir selalu berada pada tingkat kesulitan yang tepat, atau seimbang, atau terkadang bahkan dapat dimainkan. Hal ini, tentu saja, membatasi jenis permainan yang dapat kita hasilkan levelnya secara prosedural dengan menggunakan metode ini.

Di sinilah pendekatan berbasis pencarian menjadi lebih unggul. Dengan fungsi kebugaran yang dirancang dengan baik berdasarkan agen yang memainkan level tersebut, kita dapat memasukkan syarat bahwa level tersebut memang seimbang, dapat dimainkan, dan memiliki tingkat tantangan yang tepat. Hal ini memungkinkan pembuatan konten permainan secara prosedural dalam rentang permainan yang jauh lebih luas, yang membawa kita lebih dekat ke visi.

7.2 MENJADI PRIBADI

Visi yang saya uraikan di bab 4 juga mencakup bahwa permainan ini entah bagaimana beradaptasi dengan Anda dan menciptakan konten baru yang tidak hanya bagus secara umum, tetapi juga dirancang khusus untuk Anda: apa yang Anda sukai, apa yang Anda kuasai, dan cara Anda bermain. Dengan menggunakan pembuatan konten prosedural berbasis pencarian, kita dapat menciptakan level yang dapat dimainkan oleh agen buatan yang dilatih dengan gaya bermain pemain tertentu, sehingga kita setidaknya secara tidak langsung dapat mengadaptasi level baru tersebut dengan keterampilan dan gaya bermain pemain manusia. Namun, bagaimana kita dapat menciptakan level yang diadaptasi dengan preferensi manusia, level yang akan disesuaikan untuk menciptakan jenis pengalaman tertentu dalam diri pemain?

Inilah pertanyaan yang saya dan teman sekaligus kolega saya, Georgios Yannakakis, ajukan pada tahun 2009. Georgios telah mengerjakan metode untuk memodelkan pengalaman pemain untuk tesis Kuliah-nya dan beberapa tahun setelahnya. Ia telah mengembangkan metode berbasis pembelajaran mesin untuk memprediksi apa yang akan dipikirkan pemain tentang bagian tertentu dari sebuah permainan. Saya telah mengerjakan pendekatan berbasis pencarian untuk pembuatan konten prosedural, seperti yang dijelaskan di atas. Kami berpikir bahwa harus ada cara untuk menggabungkan kedua ide ini untuk secara otomatis menghasilkan konten gim yang akan menciptakan pengalaman spesifik bagi pemain.

Kami merekrut seorang mahasiswa magister, Chris Pedersen, untuk proyek ini dan mulai mengumpulkan data. Karena kami membutuhkan banyak data, kami ingin menggunakan gim yang sudah dikenal banyak orang, jadi kami senang menemukan *Infinite Mario*, tiruan sumber terbuka dari platformer klasik Nintendo, *Super Mario Bros*. Kami memodifikasi generator level agar dapat membuat level berdasarkan parameter (Gambar 7.5).

Parameter ini menentukan properti seperti seberapa besar lubang di tanah seharusnya dan seberapa jauh musuh harus muncul. Dengan memvariasikan parameter ini, kami dapat membuat generator menghasilkan berbagai jenis level, beberapa sebagian besar kosong, beberapa dengan banyak musuh untuk dikalahkan, yang lain dengan tantangan lompatan yang rumit, dan sebagainya. Kami membuat beberapa ratus level dengan pengaturan parameter yang sangat berbeda dan melanjutkan untuk mencoba membuat orang bermain gim untuk kami.

Gambar 7.5

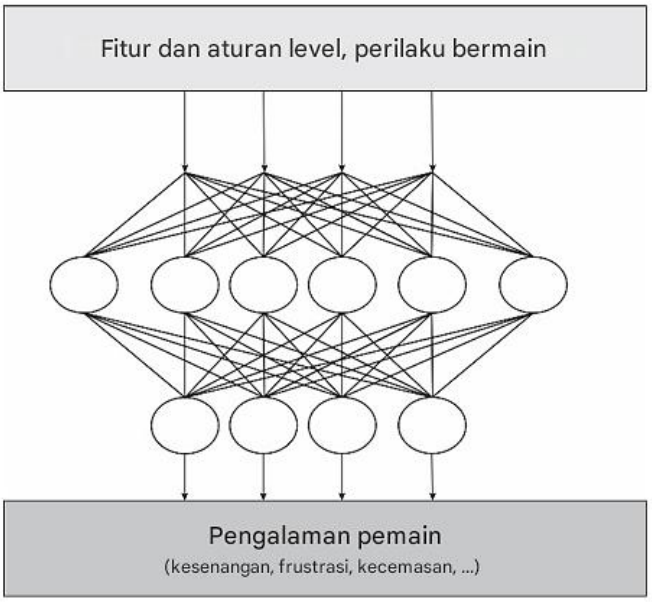
Bagian dari level yang dihasilkan dalam kerangka kerja AI *Mario* menggunakan algoritma evolusioner.

Meskipun kebanyakan orang senang bermain gim, mengajak ratusan orang bermain gim Anda agar Anda dapat mengumpulkan data tentang mereka sama sekali tidak mudah. Terkadang Anda harus membayar orang untuk bermain gim. Namun, untuk percobaan ini, cukup dengan mengganggu teman-teman kami menggunakan Twitter, Facebook, dan email (sejauh yang saya tahu, tidak ada yang menghapus pertemanan saya karena perilaku seperti ini).

Setiap orang ditugaskan untuk memainkan setidaknya dua level gim. Kedua level tersebut dibuat menggunakan nilai parameter yang berbeda sehingga terasa berbeda untuk dimainkan. Setiap orang memainkan pasangan level yang berbeda. Kami mencatat semua yang dilakukan pemain saat bermain, dan setelah setiap pasangan level, kami mengajukan serangkaian pertanyaan kepada mereka, Manakah dari dua level yang baru saja dimainkan yang lebih menantang? Mana yang lebih menghibur? Mana yang lebih membuat frustrasi?

Setelah mengumpulkan data dari lebih dari tujuh ratus pemain, kami mulai mencoba membuat model pengalaman pemain. Kami mendefinisikan jaringan saraf (gambar 7.6) yang akan mengambil parameter dari dua level berbeda sebagai input, bersama dengan beberapa data tentang gaya bermain pemain, seperti seberapa sering pemain melompat, seberapa banyak pemain berlari, dan berapa banyak musuh yang dikalahkan pemain.

Tiga output dari jaringan saraf mewakili preferensi pemain: mana dari dua level yang mereka anggap lebih menantang, lebih menghibur, dan lebih membuat frustrasi. Setelah kami memiliki data, melatih jaringan saraf untuk memprediksi preferensi pemain secara akurat menjadi mudah. Kami sekarang memiliki model preferensi pemain yang, mengingat dua level dalam permainan dan gaya bermain tertentu, dapat memprediksi mana dari dua level yang akan disukai pemain di masing-masing dari tiga dimensi ini.



Gambar 7.6

Diagram jaringan saraf tiruan yang menggunakan parameter desain level dan gaya bermain sebagai input, dan output memprediksi pengaruh pemain. Dengan menjaga gaya bermain tetap konstan dan mengoptimalkan pengaruh pemain yang diinginkan, kita dapat mengetahui jenis level apa yang kemungkinan akan menyebabkan pengalaman tertentu pada pemain.

Langkah selanjutnya adalah menggunakan model ini untuk menghasilkan level baru. Untuk bagian proyek ini, kami menghadirkan mahasiswa Kuliah baru kami yang menjanjikan, Noor Shaker, yang mulai mengerjakan apa yang kami sebut generator konten prosedural berbasis pengalaman. Ternyata lebih mudah dari yang diharapkan: mengingat jaringan saraf yang kami latih dapat memprediksi level mana yang disukai orang tertentu, kami dapat menggunakannya sebagai fungsi kebugaran. Anda cukup mengembangkan parameter level untuk memaksimalkan seberapa banyak jaringan saraf memprediksi pemain akan menikmati level tersebut. Setelah Anda memiliki parameter level ini, Anda memasukkannya ke dalam generator level standar, yang menghasilkan level yang ternyata cukup persis seperti yang Anda minta. (Seperti biasa, deskripsi ini mengabaikan sejumlah detail teknis dan keputusan desain yang rumit, tetapi secara konseptual inilah yang terjadi.)

Fitur bagus dari prosedur ini adalah Anda dapat mengoptimalkan masing-masing dari tiga dimensi preferensi secara terpisah atau dalam beberapa kombinasi. Jadi, misalnya, Anda dapat mencari level yang akan sangat menghibur dan seminimal mungkin membuat frustrasi bagi pemain tertentu. Tentu saja, Anda juga dapat mencari level yang minimal menghibur dan maksimal membuat frustrasi (jika itu yang Anda inginkan).

7.3 MENJADI LEBIH UMUM

Sejauh ini, saya telah membahas konten game secara abstrak, tetapi hanya memberikan contoh level game (jika kita menghitung trek balap dan ruang bawah tanah

sebagai level). Hal lain apa yang bisa kita hasilkan? Nah, sangat umum bagi game untuk menggunakan pembuatan prosedural vegetasi seperti pohon dan rumput, serta fitur alami lainnya seperti awan dan air. Menghasilkan konten "latar belakang" seperti itu pada dasarnya adalah masalah yang sudah terpecahkan, dan ada perangkat lunak yang akan mengurusnya untuk Anda jika Anda mau.

Alasan mengapa hal-hal seperti semak dan awan mudah dihasilkan adalah karena mereka tidak perlu terlalu banyak berinteraksi dengan keseluruhan game dan mekanismenya. Awan berbentuk aneh atau pohon yang aneh mungkin akan membuat beberapa orang mengernyitkan dahi, tetapi tidak akan membuat game Anda tidak dapat dimainkan. Tetapi bagaimana jika kita melihat ke arah lain, menghasilkan konten yang merupakan inti dari game dan berinteraksi dengan segalanya? Bisakah kita menghasilkan aturan game? Mungkin bahkan game yang lengkap?

Pada tahun 2008, Cameron Browne menyelesaikan tesis Kuliah-nya tentang topik ini. Dia telah melakukan Kuliah-nya sebagian besar pada akhir pekan dan malam hari sambil bekerja sebagai pengembang perangkat lunak dan juga menikmati hobinya yang lain: merancang permainan papan dan menulis buku tentang merancang permainan papan. (Saya mengenalnya dan dapat memastikan bahwa dia terkadang tidur juga.) Untuk Kuliah-nya, dia telah merancang bahasa *Ludi* dan sistem pembangkitan permainan, khususnya difokuskan pada apa yang disebut permainan rekombinasi: permainan dengan papan biasa dan buah dari hanya beberapa jenis, seperti Checkers, Go, Hex, dan Othello.

Bahasa Ludi memungkinkan permainan seperti itu untuk direpresentasikan hanya dalam beberapa baris kode, di mana satu baris mendefinisikan ukuran dan bentuk papan, baris lain mendefinisikan bagaimana dan apakah buah dapat ditangkap, dan seterusnya. Kode ini dapat diperlakukan sebagai genom, sehingga aturan permainan dapat dibuat dengan algoritma evolusi. Untuk menjalankan algoritma evolusi, Cameron memasok sistem Ludi dengan lusinan permainan rekombinasi yang ada, sebagian besar permainan klasik, untuk berfungsi sebagai populasi awal.

Dia juga merancang fungsi kebugaran yang akan mengevaluasi kualitas permainan melalui memainkannya dan mengukur sejumlah properti playthrough, seperti seberapa sering keunggulan berubah dan seberapa awal dalam permainan dimungkinkan untuk memprediksi siapa yang akan menang (umumnya dianggap hal yang baik jika Anda dapat memprediksi ini selambat mungkin). Dengan representasi dan fungsi kebugaran yang ditentukan, Ludi dapat mulai mengembangkan permainan. Ini tentu saja merupakan proses yang sangat lambat karena sistem perlu memainkan setiap permainan yang muncul berkali-kali melawan dirinya sendiri. Tetapi hasilnya sepadan dengan penantian. Secara khusus, satu permainan, *Yavalath* (gambar 7.7), sangat baru dan bagus sehingga penerbit game tertarik untuk menjual permainan tersebut sebagai set kotak di toko-toko. Ini mungkin adalah permainan komersial pertama yang sepenuhnya dihasilkan komputer di dunia.

Gambar 7.7

Yavalath (Nestorgames, 2007) dirancang oleh sistem Ludi, yang juga dirancang oleh Cameron Browne.

Sejauh yang saya ketahui, Cameron mendapatkan semua royalti dari penjualan game, tanpa ada satu pun uang yang masuk ke Ludi.

Pada saat Cameron mengerjakan Ludi, saya juga sedang mengembangkan ide-ide saya sendiri untuk membuat aturan permainan. Namun, tidak seperti Cameron, yang sistemnya bekerja pada jenis permainan papan tertentu, saya menargetkan permainan arkade sederhana bergaya *Pac-Man*. Saya mengambil aksioma bahwa permainan ini akan berlangsung di dunia permainan dua dimensi dan bahwa pemain mengendalikan agen yang dapat bergerak dan berinteraksi dengan berbagai "benda". Cara saya berpikir tentang hal ini adalah bahwa berbagai benda tersebut bisa berupa musuh, makanan, bonus, teman, ranjau, atau hal lainnya, semuanya bergantung pada bagaimana mereka berinteraksi satu sama lain dan dengan pemain. Misalnya, di *Pac-Man*, pelet menghilang saat berinteraksi dengan agen pemain dan meningkatkan skor, dan agen pemain menghilang jika berinteraksi dengan hantu. Jadi saya memutuskan untuk mengembangkan aturan tentang bagaimana hal-hal ini berinteraksi. Saat merancang fungsi kebugaran, saya terinspirasi oleh gagasan Raph Koster bahwa kesenangan dalam permainan berasal dari pembelajaran, seperti yang dijelaskan secara rinci dalam bab 2.

Saya ingin mengembangkan permainan yang dapat dipelajari, tetapi tentu saja saya tidak dapat menggunakan manusia yang sebenarnya dalam fungsi kebugaran. Sebaliknya, saya menggunakan algoritma evolusioner lain di dalam fungsi kebugaran, di dalam algoritma evolusioner utama yang akan mencoba belajar memainkan permainan. Permainan di mana algoritma dapat membuat peningkatan cepat mendapat skor kebugaran yang tinggi. Sayangnya, eksperimen saya tidak menghasilkan permainan hit baru yang menguasai App Store. Yang saya dapatkan adalah sejumlah contoh permainan yang menurut fungsi kebugaran masuk akal tetapi karena berbagai alasan tidak menarik untuk dimainkan, atau terkadang bahkan tidak dapat dimainkan. Ternyata, membuat aturan gim video jauh lebih sulit daripada membuat aturan untuk gim papan seperti gim rekombinasi.

Salah satu alasannya adalah agar fungsi kebugaran berfungsi dengan baik, kita membutuhkan agen AI yang tidak hanya mampu memainkan gim aneh apa pun yang diberikan oleh algoritma evolusi, tetapi juga memainkannya dengan baik dan dengan cara yang mirip manusia. Perhatikan bahwa jika Anda merupakan bagian dari fungsi kebugaran, Anda pasti akan menemukan beberapa gim yang sangat aneh yang dihasilkan oleh mutasi acak dan persilangan. Ini merupakan masalah penelitian yang belum terpecahkan.

Bersama berbagai mahasiswa dan kolaborator, saya terus berupaya mengembangkan algoritma yang dapat menciptakan aturan untuk gim video. Dalam satu proyek, kami menggunakan Bahasa Deskripsi Gim Video sebagai representasi dan mencoba mengembangkan gim yang dapat dikuasai dengan baik oleh agen yang baik dan yang buruk oleh agen yang buruk dengan kata lain, gim yang memiliki kedalaman keterampilan. Proyek ini memiliki beberapa keberhasilan terbatas, karena mampu menciptakan aturan baru untuk gim teka-teki sederhana bergaya *Sokoban*. Orang lain juga telah mengerjakan hal ini, misalnya Adam Smith telah mengusulkan penggunaan pemrograman logika untuk menciptakan aturan gim, dan Mike Cook telah lama mengerjakan ANGELINA, sebuah sistem multifaset yang tidak hanya dapat menghasilkan aturan tetapi juga berbagai aset gim, dengan hasil yang menarik (gambar 7.8).

Mencoba menciptakan sistem yang dapat menciptakan gim bukan hanya tentang membangun sistem teknis yang dapat melakukan hal-hal luar biasa. Sejak awal, kecerdasan buatan telah memiliki tujuan ganda untuk menciptakan sistem yang dapat menyelesaikan tugas-tugas yang tampaknya membutuhkan kecerdasan dan untuk memahami prinsip-prinsip di balik kecerdasan yang sudah ada di dunia, misalnya, di dalam diri kita. Secara lebih umum, salah satu hal yang Anda pelajari sebagai ilmuwan komputer adalah bahwa Anda tidak benar-benar memahami tugas sampai Anda telah menulis kode program yang dapat menyelesaikan tugas itu. Ini karena hanya merancang dan mengimplementasikan algoritma yang memecahkan tugas memaksa Anda untuk melihat tugas itu dengan cukup detail. Ini berlaku bahkan untuk hal yang biasa seperti pengurutan: mempelajari dan mengimplementasikan algoritma pengurutan memaksa Anda untuk memahami pengurutan secara mendalam. Saya yakin Anda secara intuitif tahu cara mengurutkan kaus kaki atau pensil atau koin, tetapi kecuali Anda telah mengambil kelas ilmu komputer, Anda mungkin belum banyak berpikir tentang aturan mana yang Anda ikuti ketika Anda mengurutkan sesuatu, dan bagaimana pengurutan dapat dibuat lebih efisien.

Pengamatan ini bahkan lebih relevan untuk desain gim, sebuah upaya kompleks yang (meskipun tidak sempurna) dapat kita latih untuk dilakukan orang, tetapi kita belum memahaminya sedalam pemahaman kita terhadap tugas-tugas seperti menyortir, membaca teks, atau mengemudi mobil. Oleh karena itu, merancang dan mengimplementasikan sistem yang dapat menjalankan beberapa aspek desain gim, meskipun dalam lingkungan yang sangat terbatas, merupakan salah satu cara mempelajari desain gim.

Gambar 7.8

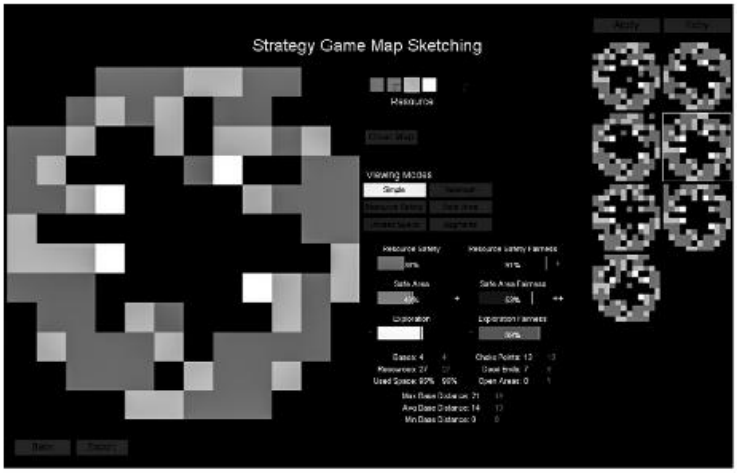
To That Sect (Michael Cook, 2014) dirancang oleh sistem ANGELINA, yang dirancang oleh Michael Cook.

7.4 BERKREASI BERSAMA

Di masa mendatang, kita tidak akan memiliki sistem AI yang dapat merancang game lengkap dari awal dengan kualitas, atau setidaknya konsistensi kualitas, yang dapat dicapai oleh tim pengembang game manusia. Desainer manusia tidak akan kehilangan pekerjaan dalam waktu dekat. Namun, ada sejumlah masalah yang telah diatasi dengan metode AI secara mengesankan, seperti yang telah kita lihat. Dalam banyak kasus, kekuatan desainer manusia dan algoritma saling melengkapi, alih-alih saling menggantikan.

Hal ini menunjukkan bahwa kita dapat membangun sistem di mana manusia berkolaborasi dengan algoritma AI untuk membuat game misalnya, dengan menggunakan algoritma untuk ide, umpan balik, penyempurnaan, dan pengujian permainan otomatis. Secara khusus, hal ini dapat dilakukan melalui pengembangan alat desain game berbantuan AI dengan inisiatif campuran. Ini adalah sistem di mana pengguna manusia dan AI dapat mengambil inisiatif dalam hal penyuntingan game dan di mana AI dapat memberikan saran, umpan balik, dan pembuatan otomatis terbatas untuk pengguna manusia.

Salah satu contoh berpengaruh dari sistem semacam itu adalah Tanagra karya Gillian Smith, yang kini menjadi profesor di Worcester Polytechnic Institute. Tanagra adalah editor untuk gim platform yang menggunakan pemecahan kendala untuk menghasilkan seluruh level atau sebagian level. Pengguna dapat membuat level sepenuhnya bebas, tetapi kapan saja dapat menggunakan alat ini untuk menghasilkan level yang benar-benar baru atau sekadar meregenerasi bagian tertentu dari suatu level. Generator level memastikan bahwa setiap level dapat dimainkan.



Gambar 7.9
Mengedit sketsa peta dengan Sentient Sketchbook.

Terinspirasi oleh sistem ini, Antonios Liapis, saat itu seorang mahasiswa Kuliah Georgios Yannakakis dan saya, mulai mengerjakan sistem yang kami sebut Sentient Sketchbook. Idenya adalah untuk menyatukan beberapa jenis dukungan desain berbasis AI dalam sebuah sistem yang akan membantu menciptakan level untuk permainan strategi. Dalam Sentient Sketchbook, pengguna mengerjakan "sketsa peta," representasi yang agak abstrak dari peta permainan strategi (lihat gambar 7.9).

Saat pengguna mengedit peta, alat tersebut memberikan umpan balik tentang ukuran-ukuran seperti seberapa seimbang peta tersebut, seberapa dekat sumber daya dengan lokasi pangkalan, dan seberapa terlindunginya pangkalan. Beberapa kuantitas ini divisualisasikan sebagai pengukur dan beberapa sebagai overlay pada peta yang sebenarnya, misalnya, menampilkan secara grafis sumber daya mana yang berada di bawah kendali lokasi pangkalan tertentu. Alat tersebut juga terus-menerus memberikan saran kepada pengguna tentang cara-cara untuk meningkatkan peta. Saran-saran ini dihasilkan oleh algoritma evolusioner yang berjalan di latar belakang, di setiap titik waktu, dimulai dari desain peta saat ini dan mengajukan pertanyaan-pertanyaan seperti ini: Bagaimana jadinya jika peta sedikit lebih seimbang, atau memiliki lebih sedikit area yang tidak dapat dilewati, atau mungkin sumber daya yang lebih sulit dikendalikan? Pengguna tentu saja bebas untuk mengabaikan saran apa pun; Sentient Sketchbook akan terus mencoba menebak-nebak maksud perancang.

Alat seperti Tanagra dan Sentient Sketchbook dan beberapa alat lain yang mengikuti jejak mereka menunjukkan jalan menuju penggunaan metode AI yang lebih luas dalam desain dan pengembangan gim. Kombinasi algoritma dan perancang manusia dalam hampir semua kasus lebih baik daripada keduanya sendiri-sendiri. Saya memperkirakan kita akan segera melihat alat serupa terintegrasi ke dalam mesin gim arus utama seperti Unity dan Unreal. Tetapi akankah kita juga melihat metode AI modern diterapkan dalam gim, alih-alih untuk desain dan pengembangan, dalam waktu dekat?

BAB 8
MENDESAIN UNTUK AI

8.1 SEJARAH DESAIN GAME DALAM AI

Di bab 4, saya mengajukan pertanyaan mengapa jenis metode kecerdasan buatan yang telah kita lihat begitu banyak contohnya dalam buku ini tidak lebih banyak digunakan dalam gim. Saya menguraikan beberapa kemungkinan alasan untuk hal ini. Salah satu alasan yang saya sebutkan adalah bahwa pengembangan gim ternyata merupakan industri yang menghindari risiko karena sifat bisnis yang berorientasi pada hit dan teknologinya mungkin belum cukup matang. Sekarang, setelah membahas beberapa bab terakhir tentang metode AI untuk bermain gim, memodelkan pemain, dan menghasilkan konten, kita akan membahas kembali pertanyaan tersebut. Kali ini kita berfokus pada peran desain gim dalam memungkinkan AI dan, sebaliknya, AI dalam memungkinkan desain gim.

Dulu ketika saya masih seorang mahasiswa Kuliah yang naif dan terlalu antusias, dan bahkan ketika saya masih seorang postdoc yang sedikit kurang naif dan terlalu antusias, saya mencoba dengan agak naif untuk melakukan perubahan. Ketika saya bertemu dengan seorang desainer atau pengembang game di sebuah konferensi, saya akan mencoba meyakinkannya bahwa game baru perusahaannya berpotensi menang banyak dengan menggunakan beberapa metode AI baru yang canggih ini. Biasanya tanggapan yang saya dapatkan adalah tidak, bahkan game mereka sama sekali tidak membutuhkan AI saya; game tersebut berfungsi dengan baik apa adanya.

Misalnya, meskipun kita dapat melatih jaringan saraf untuk mengendarai mobil lebih cepat atau menyediakan lawan yang lebih menantang dalam game pertarungan, hal ini tidak diperlukan karena lebih mudah untuk memanipulasi kecepatan tertinggi mobil yang dikendalikan komputer secara artifisial atau menabrak kotak petarung karakter nonpemain (NPC) hingga mereka mendapatkan performa yang diinginkan. Pada dasarnya, mengapa memperkenalkan AI yang kompleks ketika Anda bisa saja menipu? Dan, bagaimanapun, game akan menjadi membosankan jika musuhnya terlalu sulit karena kesenangan datang dari mengalahkan mereka.

Memang benar bahwa kita dapat menggunakan adaptasi daring, mungkin melalui pembelajaran penguatan, untuk membuat karakter game yang belajar dari perilaku Anda dalam permainan peran dan memperbarui perilakunya sendiri agar sesuai dengan apa yang Anda lakukan; Namun, hal ini berisiko sangat nyata merusak keseimbangan permainan yang telah disetel dengan cermat dan membuat permainan tidak dapat dimainkan. Tentu, kita bisa membangun algoritma pembuatan level yang memungkinkan pasokan level multipemain kompetitif baru tanpa batas untuk first-person shooter; tetapi permainan ini sudah memiliki beberapa level yang bagus dan sebagian besar pemain lebih suka memainkan level yang sudah mereka ketahui. Saya merasa sikap ini sangat konservatif dan menjengkelkan, tetapi setelah beberapa saat, saya harus mengakui bahwa dalam banyak kasus, mereka benar. Banyak

permainan tidak akan benar-benar mendapat manfaat dari AI tingkat lanjut karena dirancang untuk tidak membutuhkan AI apa pun.

Biar saya jelaskan. Sebagian besar genre video game saat ini berakar pada permainan yang dikembangkan pada tahun 1980an dan awal 1990an. Era-era ini menyaksikan perkembangan platformer, role-playing game, puzzle game, turn-based dan real-time strategy, team sports game, first-person shooter dan third-person shooter, simulasi konstruksi dan manajemen, game balap, dan sebagainya. Meskipun tentu saja telah ada inovasi desain sejak tahun 2000 misalnya, penemuan arena pertempuran daring multipemain (MOBA) seperti *League of Legends* dan permainan kotak pasir seperti *Minecraft*, genre permainan baru ini berevolusi dari genre sebelumnya.

Pada tahun 1980an dan awal 1990an, kecerdasan buatan jauh kurang maju dibandingkan saat ini. Meskipun algoritma fundamental di balik pembelajaran mendalam modern, backpropagation, telah ditemukan, pemahamannya masih jauh kurang dibandingkan saat ini, dan banyak penemuan yang membuat jaringan saraf bekerja dengan sangat baik belum ada. Pencarian pohon Monte Carlo belum ada, dan meskipun algoritma evolusi merupakan bidang penelitian yang aktif, kemajuan besar telah dicapai sejak saat itu. Namun, yang terpenting adalah bahwa daya komputer sangat terbatas saat itu. Tergantung pada apa yang Anda ukur, laptop Anda saat ini setidaknya puluhan ribu kali lebih cepat daripada komputer yang dirancang untuk menjalankan game-game yang mendefinisikan genre seperti *DOOM* dan *Dune 2*, dan ponsel cerdas Anda lebih cepat daripada superkomputer tercepat di tahun 1980an. Selain itu, kemampuan untuk menjalankan jaringan saraf pada kartu grafis (GPU) belum ada saat itu; penemuannya telah menambahkan beberapa kali lipat kecepatan untuk pembelajaran mendalam khususnya. Ketika gim-gim yang kemudian mendefinisikan seluruh genre dikembangkan, menggabungkan kecerdasan buatan mutakhir bukanlah sebuah pilihan.

Saya rasa tujuan desain untuk platform-platform awal bukanlah untuk memiliki (hanya) musuh yang bergerak maju mundur dalam pola yang dapat diprediksi. Rasanya juga mustahil bahwa hal itu dianggap baik dalam gim role-playing awal jika NPC mengucapkan kalimat-kalimat yang sama sepanjang waktu dan memaksa pemain untuk menavigasi pohon dialog yang rumit dan, agaknya, pembuatan level dalam gim roguelike awal tidak dimaksudkan untuk sangat tidak menentu dan mengabaikan keterampilan dan preferensi pemain. Sebaliknya, beginilah seharusnya karena keterbatasan teknis, dan kemudian sisa gim dirancang untuk mengakomodasi kekurangan-kekurangan ini.

Sebagai contoh lain, saya menjelaskan algoritma di balik musuh tipikal dalam gim tembak-menembak orang pertama dengan menjelaskan "rentang hidupnya" tujuh detik. Mengapa hanya tujuh detik? Game tembak-menembak orang pertama (FPS) pada awalnya dirancang tanpa karakter persisten untuk menutupi kesederhanaannya. Jika Anda berinteraksi dengan karakter di *DOOM* selama satu menit, pemrogramannya yang sederhana akan terasa sangat jelas bagi setiap pemain. Namun, jika musuh hanya muncul di layar selama beberapa detik, tidak akan ada cukup petunjuk bagi Anda tentang seberapa cerdasnya (atau tidak)

musuh tersebut. Dan game tembak-menembak orang pertama di kemudian hari sangat dipengaruhi oleh para pelopor genre ini, seperti *DOOM* (Gambar 8.1).

Gambar 8.1

DOOM (id Software, 1993) adalah salah satu game tembak-menembak orang pertama yang orisinal dan berpengaruh besar dalam perkembangan genre game ini.

Dengan kata lain, gim video pada era itu dirancang berdasarkan ketiadaan AI. Hal ini menyebabkan sejumlah pilihan desain yang tidak akan dibuat seandainya AI yang lebih baik tersedia. Misalnya, pertarungan bos dirancang berdasarkan pola tindakan berulang yang perlu dipecahkan oleh pemain, alih-alih berdasarkan bos yang benar-benar berusaha mengakali pemain, dan dialog dalam gim peran dirancang berdasarkan serangkaian pilihan dialog yang tetap, alih-alih berdasarkan NPC yang memiliki basis pengetahuan dinamis tentang dunia yang dapat ditanyakan oleh karakter pemain dengan cara yang sewenang-wenang.

Untuk alasan yang sama, penskalaan kesulitan dalam gim biasanya diimplementasikan dengan memberi musuh yang dikendalikan komputer lebih banyak atau lebih sedikit sumber daya, pada dasarnya curang alih-alih memodelkan keterampilan pemain dan mengadaptasi kedalaman pengambilan keputusan karakter yang dikendalikan komputer. Pilihan desain ini kemudian mendefinisikan genre gim ketika desainer lain menirunya dan pemain mulai mengharapkannya. Melanggar konvensi genre memang mungkin, tetapi ini mungkin melibatkan penciptaan genre baru. Menciptakan permainan peran yang tidak memiliki pohon dialog tetap, seperti yang dilakukan peneliti AI Michael Mateas dan pengembang game Andrew Stern dalam game drama hubungan inovatif mereka, *Façade*, kini dianggap sebagai penciptaan jenis permainan baru, alih-alih upaya memperbaiki aspek desain permainan peran yang telah rusak sejak awal. Mengingat kehati-hatian (yang dapat dibenarkan) dari sebagian besar pengembang dan penerbit game besar, tidak mengherankan jika kemajuan pesat dalam metode AI baru-baru ini hampir tidak tercermin dalam pengembangan game. Game yang ada saat ini tidak membutuhkan AI tingkat lanjut karena memang dirancang untuk tidak membutuhkannya.

8.2 POLA DESAIN GAME BERBASIS AI

Bagi seseorang seperti saya, yang sangat peduli dengan kecerdasan buatan dan game, pertanyaan yang muncul adalah bagaimana mengubahnya. Kemajuan dalam metode AI menjanjikan untuk memungkinkan game baru yang menakjubkan, tetapi karena praktik desain dan pengembangan yang konservatif, hal ini belum terwujud. Jadi, bagaimana kita bisa merancang game yang benar-benar membutuhkan metode AI tingkat lanjut?

Itulah pertanyaan yang saya ajukan bersama beberapa rekan kerja pada suatu hari di bulan Januari yang dingin di loteng Schloss Dagstuhl, sebuah kastil di Jerman, tempat kami menyelenggarakan seminar tentang masa depan AI dalam game. Kami memutuskan untuk menyelidiki berbagai peran yang dapat dimainkan AI dalam game, mencoba menemukan contoh dari game yang terkenal maupun yang kurang dikenal yang menggunakan AI sedemikian rupa sehingga Anda perlu berinteraksi dan memahaminya agar dapat memainkan game dengan baik. Kami mencoba mengkategorikan ini ke dalam pola desain. Pola desain yang kami hasilkan, beberapa di antaranya sebagai berikut, dapat menjadi inspirasi untuk membayangkan lebih banyak cara mendesain di sekitar AI.

AI Divisualisasikan: Dalam pola desain ini, cara kerja internal algoritma AI diperlihatkan kepada pemain, dan pemain dapat menggunakan informasi tersebut dalam permainan. Dengan kata lain, pemain dapat melihat bagaimana satu atau beberapa NPC berpikir dengan melihat ke dalam pikirannya. Contohnya adalah permainan siluman *Third Eye Crime*, di mana Anda ditugaskan untuk mengakali penjaga keamanan.

Perilaku penjaga didorong oleh teknik AI yang disebut peta hunian, yang menciptakan model ke mana penjaga harus menjelajah selanjutnya saat mereka mencari Anda. Triknya di sini adalah bahwa peta hunian ini terlihat oleh pemain melalui tata letaknya di peta permainan. Akibatnya, pemain dapat melihat keadaan pikiran para penjaga (gambar 8.2). Agar dapat memainkan permainan dengan baik, pemain perlu memahami sistem AI untuk memprediksi apa yang akan dilakukan NPC.

Gambar 8.2

Dalam *Third Eye Crime* (Moonshot Games, 2014), warna-warna di tanah memberi sinyal kepada pemain di mana para penjaga saat ini dapat melihat dan ke mana mereka akan melihat selanjutnya, menawarkan pemain pandangan ke dalam pikiran musuh.

AI sebagai Panutan: Banyak algoritma yang mendasari perilaku NPC relatif sederhana dan mudah diprediksi. Alih-alih mencoba membuat algoritma ini lebih mirip manusia, salah satu ide desain gim yang menarik adalah membuat manusia berperilaku lebih mirip algoritma tersebut. *Spy Party* adalah gim dua pemain asimetris, di mana satu pemain harus mengidentifikasi seorang pemain manusia dalam sekelompok NPC dan pemain lainnya berusaha untuk berbaur sebisa mungkin agar tidak teridentifikasi oleh pemain pertama saat menjalankan misi yang telah ditugaskan kepadanya. Berbaur paling baik dicapai dengan mencoba meniru pola pergerakan dan pengambilan keputusan NPC (gambar 8.3).

Gambar 8.3

Sebuah adegan dari *Spy Party* (Chris Hecker, 2009) menampilkan sejumlah NPC di sebuah bar, dan seorang pemain harus mencoba untuk berbaur dengan mereka.

Dengan kata lain, satu pemain perlu memahami cara kerja algoritma yang menggerakkan perilaku NPC melalui observasi agar dapat meniru perilaku tersebut, dan pemain lainnya perlu memahami perilaku yang sama agar dapat mengenali manusia yang mengganggu. Salah satu cara untuk melihat mekanika gim ini adalah sebagai bentuk uji Turing terbalik. Konsep dasar di balik uji Turing sangat menarik dan mungkin banyak mekanika permainan menarik lainnya yang dapat dikembangkan di atasnya.

AI sebagai Peserta Pelatihan: Permainan dewa (atau permainan simulator manajemen, jika Anda menginginkan nama yang lebih umum untuk genre ini) *Black and White* menempatkan pemain dalam peran dewa setempat, yang dengan berbagai cara memengaruhi kehidupan penduduk desa yang sebagian besar malang (gambar 8.4). Cara terpenting untuk memengaruhi penduduk desa adalah melalui makhluk raksasa, yang bertindak sebagai representasi Anda di dunia. Anda tidak dapat mengendalikan makhluk ini secara langsung; sebaliknya, Anda harus mengajarinya cara berinteraksi dengan penduduk desa.

Anda melakukannya dengan memberi penghargaan dan hukuman atas tindakannya dan dengan menunjukkan kepadanya melalui contoh apa yang harus dilakukan. Perilaku makhluk tersebut digerakkan oleh algoritma pembelajaran mesin, yang belajar dari tindakan Anda secara langsung saat Anda bermain. Untuk memainkan permainan ini dengan baik, Anda perlu menguasai seni melatih makhluk tersebut, yang sedikit mirip dengan belajar melatih anjing: Anda dapat melakukannya tanpa memahami apa yang sebenarnya terjadi di kepala anjing.

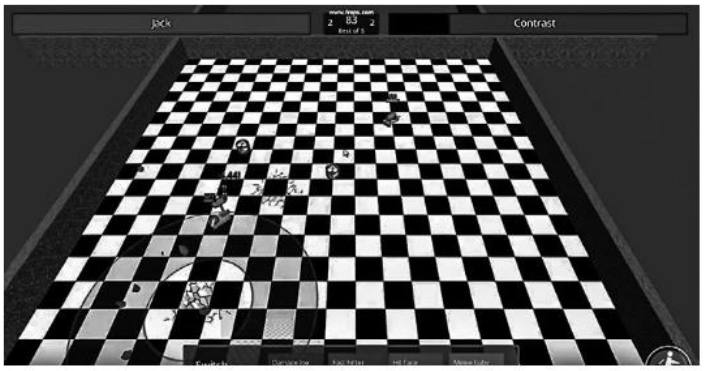
Gambar 8.4

Makhluk-makhluk raksasa dalam *Black and White* (Lionhead Studios, 2001) dapat menuruti perintah Anda, tetapi hanya jika Anda melatih mereka dengan baik.

Bahasa Indonesia: Pendekatan lain terhadap pola khusus ini adalah membangun gim tempat Anda melatih agen yang kemudian bersaing atau bertarung satu sama lain, sedikit seperti mekanik pelatihan dari seri *Pokémon* tetapi dengan pembelajaran mesin yang sebenarnya, bukan mekanik kemajuan bergaya permainan peran sederhana. Salah satu contohnya adalah NERO (*NeuroEvolution of Robotic Operatives*), gim berbasis penelitian oleh Ken Stanley, sekarang di University of Central Florida dan Uber AI Labs.

Dalam gim itu, Anda melatih pasukan tentara mini dengan merancang berbagai tugas untuk mereka dan memutuskan jenis perilaku apa yang akan mereka beri hadiah. Gim penelitian lain dari tim saya, *EvoCommander*, didasarkan pada ide yang sama untuk melatih agen untuk melakukan perintah pemain, tetapi alih-alih melatih banyak agen, Anda melatih sejumlah "otak" (jaringan saraf terpisah) untuk robot simulasi (gambar 8.5). Saat bermain melawan pemain lain, Anda mengendalikan robot secara tidak langsung dengan memilih otak mana yang harus digunakan pada setiap titik waktu.

AI Dapat Diedit: Anda juga dapat merancang permainan dengan mengedit langsung instruksi untuk algoritma yang mengendalikan perilaku agen. Permainan papan *RoboRally* adalah bukti kemungkinan menciptakan permainan yang sangat sukses dengan mekanika semacam itu. Dalam *RoboRally,* setiap pemain secara bergiliran memilih instruksi yang harus dijalankan robotnya pada giliran tersebut. Meskipun "pemrograman" di sini sederhana, memprediksi perilaku yang dihasilkan sangat menantang karena semua robot pemain menjalankan program mereka secara paralel.



Gambar 8.5

Pohon keluarga otak di *EvoCommander* (Daniel Jallov, 2015). Sebelum pertandingan, Anda memilih otak terlatih mana yang akan dibawa ke medan perang.

Contoh yang lebih canggih adalah mode editor jaringan *Galactic Arms Race*, gim berbasis riset lain karya tim Ken Stanley. *Galactic Arms Race* adalah gim tembak-menembak luar angkasa orang ketiga yang dibangun di atas bentuk unik pembuatan konten prosedural berbasis pencarian (gambar 8.6). Senjata dalam gim ini dikendalikan oleh jaringan saraf, yang menentukan perilaku partikel yang ditembakkan oleh pesawat luar angkasa pemain. Pemain dapat mengumpulkan dan membuang senjata di seluruh dunia gim, dan kapan saja, mereka dapat beralih di antara beberapa senjata yang telah dilengkapi. Senjata dibuat melalui algoritma evolusi kolaboratif di mana semua pemain gim bertindak sebagai fungsi kebugaran; senjata baru adalah turunan dari senjata yang paling sering dipilih pemain.

Hal ini sendiri merupakan penggunaan teknik AI yang sangat menarik dalam gim, meskipun lebih berperan sebagai latar belakang karena pemain tidak perlu memahami algoritma evolusi pembangkit senjata untuk memainkan gim. Pola desain AI-is-editable diperkenalkan dalam sebuah ekstensi gim, yang memungkinkan pengeditan manual jaringan saraf yang mendefinisikan senjata. Struktur jaringan saraf umumnya sulit dipahami manusia, yang berarti mode penyuntingan ini tidak cocok untuk semua orang. Namun, bagi sebagian pemain, penyuntingan jaringan saraf untuk mencoba mendapatkan perilaku senjata yang diinginkan merupakan permainan teka-teki yang menarik.

Gambar 8.6

Senjata yang berevolusi dalam *Galactic Arms Race* (Evolutionary Games, 2009).

AI Dipandu: Ide lain tentang cara merancang permainan berbasis AI sehingga pemain perlu berinteraksi dan memahaminya adalah dengan mengendalikan karakter permainan oleh algoritma AI. Namun, hal ini tidak sempurna, baik karena Anda membatasi apa yang dapat dilakukan algoritma atau karena tugas yang diminta untuk dilakukan oleh karakter permainan terlalu rumit. Pemain kemudian perlu bertindak sebagai pemandu atau manajer bagi para agen, memberi mereka perintah tingkat tinggi atau membimbing mereka melalui operasi yang tidak dapat mereka lakukan sendiri. Contoh yang sangat baik dari pola desain ini adalah seri permainan *The Sims* yang sangat sukses.

Permainan ini paling tepat digambarkan sebagai simulator kehidupan atau rumah boneka virtual, tempat Anda mengendalikan sekelompok karakter saat mereka menjalani hidup. Anda perlu membuat semua keputusan hidup besar untuk mereka, seperti di mana membangun rumah, tetapi dalam banyak kasus Anda juga perlu membantu dengan tugas-tugas kecil, seperti memastikan ada panci dan wajan yang tersedia untuk memasak. Tetapi karakter juga memiliki suara. Game *The Sims* menampilkan sistem AI yang kompleks yang mengendalikan agen, sehingga mereka tidak hanya melakukan tindakan otonom seperti pergi ke kamar mandi dan memasak makan malam tetapi juga menjalin persahabatan dan jatuh cinta (gambar 8.7).

Bermain game adalah tindakan penyeimbangan yang konstan antara pemain dan sistem AI. Yang terpenting, game ini sering mengomunikasikan keadaan sistem AI-nya melalui gelembung pikiran kecil di atas kepala karakter, yang memungkinkan pemain untuk memahami apa yang terjadi.

Gambar 8.7
Pertemuan romantis di *The Sims 4* (Maxis, 2013).

Tentu saja, ini hanyalah sebagian kecil dari sekian banyak kemungkinan cara AI dapat digunakan dalam peran nyata dalam gim video. Dan saya hanya menyebutkan satu pola yang melibatkan pembuatan prosedural dan tidak ada yang dibangun di atas pemodelan pemain. Cukup jelas bahwa ada ruang desain yang luas dan belum dieksplorasi di luar sana, dengan banyak ide desain gim baru yang tersedia bagi mereka yang melihat melampaui genre yang sudah mapan dan prasangka tentang peran apa yang dapat dan tidak dapat dimainkan AI.

BAB 9
KECERDASAN UMUM DAN PERMAINAN SECARA UMUM

9.1 KECERDASAN BUATAN PADA KOMPUTER PLAYER

Salah satu komponen vital dalam sebuah permainan adalah keberadaan kecerdasan buatan atau AI (artificial intelligence). AI dianggap sebagai elemen penting dalam pengembangan game karena berperan dalam membuat permainan menjadi lebih dinamis dan terstruktur. Game AI sendiri merupakan penerapan berbagai metode, proses, dan algoritma kecerdasan buatan yang diaplikasikan dalam pembuatan dan pengembangan game. Tiga perspektif utama dalam game AI meliputi sudut pandang metode dari komputer, pengalaman pengguna manusia, dan interaksi pemain. AI kerap dipakai dalam game yang sangat bergantung pada interaksi dengan pemain, sehingga peran AI menjadi krusial dalam meningkatkan daya tarik dan ketertarikan pemain terhadap game tersebut. Dari penjelasan ini dapat disimpulkan bahwa penggunaan AI dalam game dibutuhkan untuk menambah tingkat tantangan, membuat permainan lebih dinamis, serta mendukung realisme dalam game. Hal ini berujung pada pengalaman bermain yang lebih menyenangkan dan minat pengguna yang semakin meningkat.

Khusus pada game RPG yang memiliki berbagai kemungkinan jalur dan skenario, diperlukan metode AI yang mampu mengambil keputusan secara efektif. Salah satu metode yang diterapkan adalah *Support Vector Machine* (SVM),. Metode SVM diketahui dapat mempercepat proses pengambilan keputusan, namun sangat tergantung pada jumlah data pelatihan yang tersedia. SVM menggunakan data pelatihan untuk menentukan hyperplane terbaik dalam memisahkan dua kelas keputusan. Selain SVM, metode decision tree juga sering digunakan untuk mengatur perilaku pemain komputer dalam game. Metode ini membangun pohon keputusan dari data pelatihan yang nantinya menjadi dasar dalam pengambilan keputusan berikutnya. Alternatif lain adalah rulebase, yang berbeda dengan SVM dan decision tree karena menggunakan sekumpulan aturan berupa kondisi dan aksi untuk menentukan perilaku.

Berdasarkan paparan tersebut, tujuan utama pengembangan game adalah sebagai sarana hiburan. Game jenis RPG populer karena menawarkan banyak kemungkinan yang mempengaruhi pencapaian tujuan permainan. Kecerdasan buatan menjadi faktor penting dalam meningkatkan tingkat hiburan dengan menjadikan gameplay lebih dinamis dan menarik bagi pemain. Oleh karenanya, penelitian ini bertujuan membandingkan metode AI SVM, decision tree, dan rulebase dalam mengelola perilaku pemain komputer pada game RPG untuk menemukan metode yang paling sesuai dalam pengambilan keputusan.

Mari kita kembali ke pertanyaan tentang apa sebenarnya kecerdasan itu, yang saya bahas di bab 3 tanpa mencapai kesimpulan yang memuaskan. Karena Anda membaca ini, Anda jelas belum menyerah membaca buku ini, tetapi Anda mungkin sedikit kecewa dengan saya karena tampaknya saya tidak dapat memberikan jawaban yang lugas. Yah, saya hanya jujur. Sejauh mana kecerdasan umum itu ada, masih banyak perdebatan. Saya tidak akan memaksakan pandangan tertentu kepada Anda karena saya pikir masih banyak pekerjaan, baik filosofis maupun empiris, yang harus dilakukan untuk memahami pertanyaan ini dengan lebih baik. Namun, yang tampaknya kita semua sepakati adalah bahwa beberapa sistem kecerdasan buatan memiliki penerapan yang lebih luas daripada yang lain dalam arti bahwa sistem

tersebut dapat melakukan tugas yang lebih beragam dan bahwa kualitas sistem AI yang diinginkan adalah bersifat generik, alih-alih spesifik. Tidak ada yang salah dengan sistem AI yang hanya dapat melakukan satu hal jika kita hanya mencoba merancang solusi untuk masalah tertentu. Namun, jika kita mencoba membuat kemajuan ilmiah dalam menciptakan kecerdasan buatan, maka penting bagi kita untuk membangun sistem yang dapat melakukan berbagai hal yang berbeda misalnya, memainkan berbagai permainan.

Sekitar waktu saya menyelesaikan Kuliah saya, saya pikir permainan balap mobil kecil yang saya buat untuk eksperimen saya dengan jaringan saraf yang berkembang cukup bagus dan orang lain mungkin ingin menggunakannya untuk eksperimen mereka sendiri, jadi saya memutuskan untuk menyediakan kodenya. Saat saya melakukannya, saya memutuskan untuk memulai sebuah kompetisi. Para peneliti, mahasiswa, dan siapa pun dapat mengirimkan agen terbaik mereka, dan mereka akan bersaing satu sama lain. Sama seperti dalam balap mobil dunia nyata, mobil yang menyelesaikan lintasan tercepat akan menang. Juga seperti dalam balap mobil dunia nyata, tabrakan adalah bagian yang paling menyenangkan untuk ditonton.

Saya dengan cepat mendapatkan beberapa lusin pesaing dari seluruh dunia, mengirimkan pengontrol berdasarkan beberapa teknik AI yang sangat berbeda. Pemenangnya menggunakan teknik yang disebut *logika fuzzy* untuk bernalar tentang cara mengemudi terbaik, tetapi ada beberapa agen bagus berdasarkan pembelajaran penguatan dan algoritma evolusi.

Melihat betapa suksesnya kompetisi ini, saya memutuskan untuk menjalankannya lagi, tetapi kali ini saya bekerja sama dengan beberapa peneliti Italia, Pier Luca Lanzi dan Daniele Loiacono di Politecnico di Milano, untuk mengembangkannya menjadi gim balap 3D yang lebih mumpuni bernama *TORCS*. Kompetisi ini berlangsung selama tujuh tahun, dengan partisipasi berkelanjutan dari berbagai universitas, dan dalam beberapa kasus, para penghobi dan perusahaan swasta, di seluruh dunia (gambar 9.1).

Beberapa tahun kemudian, saya memulai kompetisi AI lain berdasarkan *Infinite Mario*, klon sumber terbuka dari *Super Mario Bros*. yang saya sebutkan sebelumnya. Mahasiswa saya, Sergey Karakovskiy, dan saya membangun kembali *Infinite Mario* menjadi tolok ukur AI dan meminta orang-orang mengirimkan agen AI terbaik mereka yang memainkan Mario. Dengan beberapa minggu tersisa sebelum kompetisi berakhir, seorang mahasiswa Kuliah muda bernama Robin Baumgarten mengirimkan sebuah agen berdasarkan algoritma A^*. Agen tersebut sangat efektif. Ia menyelesaikan semua level yang dapat dihasilkan oleh generator level kami dengan sempurna dan memenangkan kompetisi.

Gambar 9.1

TORCS, 2014. (Gambar milik Libre Game Wiki.)

Hal ini sedikit mengecewakan bagi kami, karena kami membayangkan telah membangun masalah AI yang sulit, tetapi ternyata agen yang didasarkan pada algoritma yang begitu sederhana dan terkenal justru mengalahkannya. Dalam upaya menyelamatkan kompetisi untuk putaran berikutnya, kami berupaya membuat generator level yang lebih kejam. Saat kompetisi berikutnya kami jalankan, generator level menciptakan level-level dengan jalan buntu yang sering terjadi, yang harus dilewati Mario untuk keluar. Ini adalah tantangan yang tidak dapat diatasi oleh agen A^* Baumgarten; sebaliknya, kompetisi berikutnya dimenangkan oleh agen kompleks bernama REALM, yang menggabungkan algoritma evolusioner dengan sistem berbasis aturan dan, sebagai salah satu bagian penting dari campuran tersebut, algoritma A^* yang mirip dengan milik Baumgarten.

Tentu saja, saya bukan orang pertama yang menjalankan kompetisi AI berbasis gim. Kompetisi untuk pemain AI Catur, Dama, dan Go telah berlangsung selama beberapa dekade. Dalam dunia gim video, telah ada kompetisi AI yang telah lama berjalan berdasarkan gim arkade klasik seperti *Ms. Pac-Man*, first-person shooter seperti *DOOM*, dan gim teka-teki fisika seperti *Angry Birds*. Salah satu kompetisi paling aktif saat ini adalah kompetisi *StarCraft*, yang berkisar pada gim di mana agen terbaik yang dikirimkan tetap tidak memiliki peluang melawan pemain manusia yang baik.

Di sebagian besar kompetisi ini, setidaknya yang berlanjut selama beberapa tahun, terdapat kemajuan yang jelas. Agen balap yang dikirimkan ke Kompetisi Balap Mobil Simulasi 2012 benar-benar berlari mengelilingi agen yang dikirimkan ke kompetisi 2008, dan agen yang dikirimkan ke Kompetisi AI Mario 2011 menyelesaikan level yang tidak dapat diselesaikan oleh agen yang dikirimkan ke kompetisi 2009. Ini semua bagus dan tampaknya menunjukkan bahwa kompetisi ini memacu kemajuan dalam AI bermain gim.

Namun, jika melihat kiriman dari setiap tahun, Anda dapat melihat tren yang mengkhawatirkan: secara umum, semakin sedikit algoritma AI umum dalam kiriman selanjutnya. Kiriman untuk edisi pertama Simulated Car Racing Competition terdiri dari agen yang menggunakan algoritma yang relatif umum yang dapat digunakan untuk memainkan game lain dengan perubahan kecil. Pada tahun-tahun berikutnya dari kompetisi, agen semakin

disesuaikan dengan tugas memainkan game balap khusus ini, termasuk mekanisme yang dibuat dengan susah payah untuk mengganti gigi, mempelajari bentuk lintasan, memblokir mobil yang menyalip, dan sebagainya.

Faktanya, algoritma pembelajaran mesin secara umum digunakan dalam peran yang semakin sedikit dalam kiriman tahun-tahun berikutnya dibandingkan dengan yang ada di awal kompetisi. Algoritma AI tingkat lanjut diturunkan ke peran pendukung. Peningkatan kinerja agen tidak benar-benar disebabkan oleh peningkatan apa pun dalam algoritma yang mendasarinya tetapi untuk rekayasa spesifik game yang lebih baik. Perkembangan serupa dapat diamati dalam kompetisi AI *Mario*.

Mengenai kompetisi *Star-Craft*, agen yang menang cenderung berupa strategi yang dibuat dengan rumit dan hanya sedikit menggunakan apa yang biasa kita sebut AI, seperti algoritma pencarian atau pembelajaran mesin. Yang terpenting, agen-agen ini sangat spesifik. Agen yang dikirimkan ke Kompetisi AI *Mario* tidak dapat mengendalikan mobil dalam Kompetisi Balap Mobil Simulasi atau membangun markas dan memimpin pasukan dalam *StarCraft*. Agen *StarCraft* tidak dapat mengendarai mobil atau memainkan *Super Mario Bros.*, dan sebagainya. Bukan hanya agen tersebut akan memainkan game-game ini dengan buruk; tetapi juga tidak dapat memainkannya sama sekali: status game direpresentasikan dengan sangat berbeda untuk setiap game. Status game *StarCraft* tidak masuk akal bagi agen yang memainkan *Mario*, dan keluaran agen tersebut (seperti berlari dan melompat) tidak masuk akal bagi game *StarCraft*.

Ini bukan masalah yang hanya terjadi pada kompetisi-kompetisi ini. Saya telah menyebutkan bahwa DeepMind melatih jaringan saraf untuk memainkan beberapa lusin gim Atari klasik. Ini mungkin tampak seperti contoh AI gim yang lebih umum, jika bukan karena fakta bahwa setiap jaringan saraf dilatih untuk memainkan satu gim saja. Jaringan saraf yang dilatih untuk memainkan *Space Invaders* tidak dapat memainkan *Pac-Man*, *Montezuma's Revenge*, atau gim Atari lainnya setidaknya tidak dapat memainkannya lebih baik daripada monyet di depan mesin tik, tetapi dengan joystick, bukan mesin tik.

Ada beberapa upaya untuk melatih jaringan saraf agar dapat memainkan lebih dari satu gim, sejauh ini dengan keberhasilan yang terbatas. Hal yang sama berlaku untuk agen gim terkenal lainnya dari DeepMind, AlphaGo. Ia sangat mahir bermain Go, tetapi hanya bisa bermain Go. Ia tidak bisa memainkan apa pun, bahkan Catur.

9.2 BERMAIN GIM VIDEO UMUM

Mari kita kembali ke pertanyaan tentang pengembangan kecerdasan buatan umum, atau setidaknya kecerdasan buatan yang agak umum. Tampaknya semua upaya pengembangan agen AI yang dapat memainkan gim individual ini mungkin tidak akan membawa kita lebih dekat ke tujuan ini. Dalam kasus terburuk, ini bahkan bisa menjadi dua langkah maju dan satu langkah mundur: kita terus menghabiskan sumber daya untuk memahami dan mengeksploitasi dinamika gim individual alih-alih mencoba menciptakan agen yang dapat menunjukkan kecerdasan yang lebih umum. Cara terbaik untuk menunjukkan kecerdasan yang lebih umum adalah dengan menggunakan agen yang sama, tanpa atau

dengan sedikit pelatihan ulang, untuk menyelesaikan beberapa tugas berbeda, seperti memainkan beberapa gim yang berbeda.

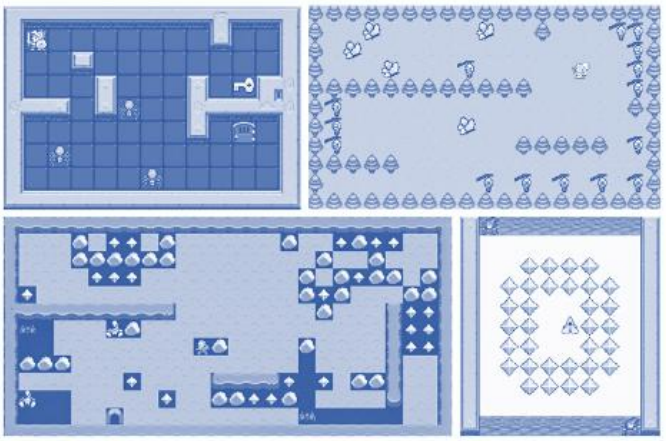
Bagaimana Anda memastikan bahwa para peneliti berupaya menciptakan agen yang memiliki kapasitas bermain gim yang lebih umum? Anda menciptakan sebuah kompetisi! Itulah yang dipikirkan sekelompok dari kami pada tahun 2013 ketika kami mulai mengerjakan Kompetisi *General Video Game AI* (GVG-AI) (gambar 9.2). Idenya adalah mengadakan kompetisi di mana Anda tidak dapat menyesuaikan agen Anda dengan gim tertentu, jadi Anda harus membuatnya setidaknya agak umum. Kami pikir kami perlu merancang kompetisi sehingga Anda tidak tahu gim apa yang akan dimainkan agen Anda.

Setiap kali kami mengadakan kompetisi, kami membutuhkan game-game baru yang belum pernah dilihat orang sebelumnya (meskipun game-game tersebut mungkin mirip atau versi dari game-game terkenal). Untuk itu, kami membutuhkan cara untuk membuat game-game ini dengan mudah, jadi kami mulai dengan merancang bahasa pemrograman baru khusus untuk membuat game-game bergaya arcade klasik. Tom Schaul memimpin pengembangan bahasa ini, yang kami sebut *Video Game Description Language* (VGDL). Diego Perez-Liebana kemudian memimpin pengembangan perangkat lunak kompetisi.

Sejauh ini, kami telah menyelenggarakan kompetisi GVG-AI beberapa kali per tahun sejak 2014. Setiap acara kompetisi menguji semua agen yang dikirimkan pada satu set sepuluh permainan baru, yang harus dibuat sendiri untuk setiap kompetisi. Hingga saat ini, lebih dari seratus permainan telah dibuat, banyak di antaranya merupakan versi atau terinspirasi oleh permainan arkade dari tahun 1980an.

Para peserta tidak tahu permainan mana yang akan diuji agen mereka hingga setelah mereka mengirimkan agen mereka, memastikan bahwa mereka memfokuskan energi mereka untuk meningkatkan kapasitas bermain agen secara umum, alih-alih untuk menyesuaikannya dengan permainan tertentu. Saat ini, agen terbaik hanya dapat menang dengan andal di kurang dari setengah permainan yang ada, menunjukkan bahwa masih ada banyak ruang untuk perbaikan.

Jika seseorang membuat agen yang dapat menang di semua permainan yang ada dalam kompetisi GVG-AI, akankah kita menyebut agen tersebut "secara umum cerdas"? Belum tentu. Perangkat lunak GVG-AI memberi agen AI akses ke model maju, atau simulator permainan, yang memudahkan Anda merencanakan tindakan dengan mensimulasikan apa yang akan terjadi jika Anda menjalankan rencana Anda.



Gambar 9.2

Empat permainan berbeda dalam kerangka kerja GVG-AI: *Zelda, Butterflies, Boulder Dash*, dan *Solar Fox*. Antarmuka yang sama berarti agen yang sama dapat memainkan semua permainan dalam kerangka kerja tersebut, tetapi dengan tingkat keahlian yang berbeda-beda.

Misalnya, versi agen A^* yang memenangkan kompetisi *AI Mario* sangat bergantung pada model maju. Umumnya, Anda tidak akan memiliki akses ke model tersebut sebagai manusia yang memainkan gim arkade klasik, dan "dunia nyata" terkenal tidak memiliki model maju. Kami sedang mengembangkan versi baru kompetisi ini, yang tidak memberikan kemungkinan ini kepada agen, melainkan memberi mereka waktu singkat untuk beradaptasi dengan setiap gim.

Selain itu, gim yang dapat diekspresikan dalam versi VGDL saat ini terbatas pada jenis gim yang akan Anda temukan di komputer rumah atau ruang arkade awal 1980an, dan bahkan saat itu pun beberapa jenis gim tidak ada (misalnya, tidak ada gim berbasis teks). Di masa mendatang yang belum ditentukan, kami berharap VGDL atau bahasa pemrograman penerusnya akan mampu mengekspresikan ragam gim yang jauh lebih luas. Kami juga berharap bahwa di masa mendatang, gim-gim ini dapat dibuat secara otomatis, sehingga akan jauh lebih mudah untuk membuat gim baru guna menguji agen AI.

Meskipun proyek GVG-AI hanya langkah kecil menuju pemecahan, atau bahkan perumusan yang tepat, masalah bermain gim secara umum, saya percaya bahwa proyek ini sangat relevan untuk memahami kecerdasan secara umum. Seperti yang telah kita lihat dalam buku ini, gim sangat beragam, dan gim menantang kapasitas kognitif kita dengan cara yang baru mulai kita pahami. Jika suatu saat nanti kita menciptakan agen yang dapat belajar dengan cepat untuk memainkan semua gim video atau bahkan hanya gim yang paling baik desainnya (katakanlah 100 gim paling populer di setiap platform distribusi utama, seperti Steam atau iOS App Store) dengan keterampilan yang mirip dengan manusia yang bermain gim, maka saya pikir kita akan mencapai kecerdasan umum buatan. Setidaknya, kita akan sangat memajukan pemahaman kita tentang apa itu kecerdasan dan apa yang bukan.

DAFTAR PUSTAKA

Agbo, F. J., Olaleye, S. A., Bower, M., & Oyelere, S. S. (2023). Examining the relationships between students' perceptions of technology, pedagogy, and cognition: the case of immersive virtual reality mini games to foster computational thinking in higher education. *Smart Learning Environments*, *10*(1), 16.

Alam, A. (2022, April). A digital game based learning approach for effective curriculum transaction for teaching-learning of artificial intelligence and machine learning. In *2022 international conference on sustainable computing and data communication systems (ICSCDS)* (pp. 69-74). IEEE.

Ali, S., Kumar, V., & Breazeal, C. (2023). AI audit: a card game to reflect on everyday AI systems. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 37, No. 13, pp. 15981-15989).

Aronson, J. K. (2022). Artificial intelligence in pharmacovigilance: an introduction to terms, concepts, applications, and limitations. *Drug safety*, *45*(5), 407-418.

Bharathi, G. P., Chandra, I., Sanagana, D. P. R., Tummalachervu, C. K., Rao, V. S., & Neelima, S. (2024). AI-driven adaptive learning for enhancing business intelligence simulation games. *Entertainment Computing*, *50*, 100699.

Bui, T. H., & Nguyen, V. P. (2023). The impact of artificial intelligence and digital economy on Vietnam's legal system. *International Journal for the Semiotics of Law-Revue internationale de Sémiotique juridique*, *36*(2), 969-989.

Cao, L. (2022). A new age of AI: Features and futures. *IEEE Intelligent Systems*, *37*(1), 25-37.

Chen, C. H., & Chang, C. L. (2024). Effectiveness of AI-assisted game-based learning on science learning outcomes, intrinsic motivation, cognitive load, and learning behavior. *Education and Information Technologies*, *29*(14), 18621-18642.

Correia, A., & Lobo, V. (2024). Enhancing critical thinking in education through AI-driven gamification: The development and impact of the Adaptive Critical Thinking Enhancement System (ACTES). In *EDULEARN24 Proceedings* (pp. 7768-7777). IATED.

De Cremer, D., & Kasparov, G. (2021). AI should augment human intelligence, not replace it. *Harvard Business Review*, *18*(1), 1-8.

de Vicente-Yagüe-Jara, M. I., López-Martínez, O., Navarro-Navarro, V., & Cuéllar-Santiago, F. (2023). Writing, creativity, and artificial intelligence: ChatGPT in the university context. *Comunicar: Media Education Research Journal*, *31*(77), 45-54.

Dillon, S., & Schaffer-Goddard, J. (2023). What AI researchers read: The role of literature in artificial intelligence research. *Interdisciplinary Science Reviews*, *48*(1), 15-42.

Dohn, N. B., Kafai, Y., Mørch, A., & Ragni, M. (2022). Survey: Artificial intelligence, computational thinking and learning. *KI-Künstliche Intelligenz*, *36*(1), 5-16.

Druga, S., & Ko, A. J. (2021, June). How do children's perceptions of machine intelligence change when training and coding smart programs?. In *Proceedings of the 20th annual ACM interaction design and children conference* (pp. 49-61).

Dyulicheva, Y. Y., & Glazieva, A. O. (2021). Game based learning with artificial intelligence and immersive technologies: An overview. *CS&SE@ SW*, 146-159.

Eun, S. J., Kim, E. J., & Kim, J. Y. (2022). Development and evaluation of an artificial intelligence–based cognitive exercise game: A pilot study. *Journal of environmental and public health*, *2022*(1), 4403976.

Filipović, A. (2023). The role of artificial intelligence in video game development. *Kultura polisa*, *20*(3), 50-67.

Fitria, T. N. (2021, December). Artificial intelligence (AI) in education: Using AI tools for teaching and learning process. In *Prosiding seminar nasional & call for paper STIE AAS* (pp. 134-147).

Henry, J., Hernalesteen, A., & Collard, A. S. (2021). Teaching artificial intelligence to K-12 through a role-playing game questioning the intelligence concept. *KI-Künstliche Intelligenz*, *35*(2), 171-179.

Hoffmann, C. H. (2022). Is AI intelligent? An assessment of artificial intelligence, 70 years after Turing. *Technology in Society*, *68*, 101893.

Hsu, T. C., & Chen, M. S. (2025). Effects of students using different learning approaches for learning computational thinking and AI applications. *Education and Information Technologies*, *30*(6), 7549-7571.

Hu, X., Liu, Y., Huang, J., & Mu, S. (2022). The effects of different patterns of group collaborative learning on Fourth-Grade students' creative thinking in a digital artificial intelligence course. *Sustainability*, *14*(19), 12674.

Huang, X., & Qiao, C. (2024). Enhancing computational thinking skills through artificial intelligence education at a STEAM high school. *Science & Education*, *33*(2), 383-403.

Ilgun Dibek, M., Sahin Kursad, M., & Erdogan, T. (2025). Influence of artificial intelligence tools on higher order thinking skills: a meta-analysis. *Interactive Learning Environments*, *33*(3), 2216-2238.

Kanervisto, A., Bignell, D., Wen, L. Y., Grayson, M., Georgescu, R., Valcarcel Macua, S., ... & Hofmann, K. (2025). World and human action models towards gameplay ideation. *Nature*, *638*(8051), 656-663.

Kang, H. (2023). Artificial intelligence and its influence in adult learning in China. *Higher Education, Skills and Work-Based Learning*, *13*(3), 450-464.

Lee, S., Mott, B., Ottenbreit-Leftwich, A., Scribner, A., Taylor, S., Park, K., ... & Lester, J. (2021, May). AI-infused collaborative inquiry in upper elementary school: A game-based learning approach. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 35, No. 17, pp. 15591-15599).

Li, W. (2022). Analysis of piano performance characteristics by deep learning and artificial intelligence and its application in piano teaching. *Frontiers in Psychology*, *12*, 751406.

Louis, M., & ElAzab, M. (2023). Will AI replace teacher?. *International Journal of Internet Education*, *22*(2), 9-21.

Ma, J., Zhang, Y., Bin, H., Wang, K., Liu, J., & Gao, H. (2022, July). The Development of Students' Computational Thinking Practices in AI Course Using the Game-Based Learning: A Case Study. In *2022 International Symposium on Educational Technology (ISET)* (pp. 273-277). IEEE.

Mitchell, M. (2021). Why AI is harder than we think. *arXiv preprint arXiv:2104.12871*.

Muthmainnah, Ibna Seraj, P. M., & Oteir, I. (2022). Playing with AI to investigate human-computer interaction technology and improving critical thinking skills to pursue 21st century age. Education Research International, 2022(1), 6468995.

Nader, K., Toprac, P., Scott, S., & Baker, S. (2024). Public understanding of artificial intelligence through entertainment media. *AI & society*, *39*(2), 713-726.

Nalbant, K. G., & Aydın, S. (2023). Development and transformation in digital marketing and branding with artificial intelligence and digital technologies dynamics in the Metaverse universe. *Journal of Metaverse*, *3*(1), 9-18.

Nyholm, S. (2024). Artificial intelligence and human enhancement: can AI technologies make us more (artificially) intelligent?. *Cambridge Quarterly of Healthcare Ethics*, *33*(1), 76-88.

Pitychoutis, K. M., & Al Rawahi, A. (2024). Smart teaching: The synergy of multiple intelligences and artificial intelligence in English as a foreign language instruction. In *Forum for Linguistic Studies* (Vol. 6, No. 6, pp. 249-260).

Rapaka, A., Dharmadhikari, S. C., Kasat, K., Mohan, C. R., Chouhan, K., & Gupta, M. (2025). Revolutionizing learning– A journey into educational games with immersive and AI technologies. *Entertainment Computing*, *52*, 100809.

Reddy, V. S., Kathiravan, M., & Reddy, V. L. (2024). Revolutionizing animation: unleashing the power of artificial intelligence for cutting-edge visual effects in films. *Soft Computing-A Fusion of Foundations, Methodologies & Applications*, *28*(1).

Saritepeci, M., & Yildiz Durak, H. (2024). Effectiveness of artificial intelligence integration in design-based learning on design thinking mindset, creative and reflective thinking skills: An experimental study. *Education and Information Technologies*, *29*(18), 25175-25209.

Siddiqi, S. H., Shukla, V. K., Bhardwaj, A. B., & Gaur, D. (2021, September). Analyzing psychological gamers profile through progressive gaming and artificial intelligence. In *2021 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO)* (pp. 1-5). IEEE.

Sidji, M., Smith, W., & Rogerson, M. J. (2024). Human-ai collaboration in cooperative games: A study of playing codenames with an llm assistant. *Proceedings of the ACM on Human-Computer Interaction*, *8*(CHI PLAY), 1-25.

Su, J., & Yang, W. (2024). AI literacy curriculum and its relation to children's perceptions of robots and attitudes towards engineering and science: An intervention study in early childhood education. *Journal of Computer Assisted Learning*, *40*(1), 241-253.

Valavanidis, A. (2023). Artificial intelligence (ai) applications. *Department of Chemistry, National and Kapodistrian University of Athens, University Campus Zografou*, *15784*.

Villareale, J., Harteveld, C., & Zhu, J. (2022). " I Want To See How Smart This AI Really Is": Player Mental Model Development of an Adversarial AI Player. *Proceedings of the ACM on Human-Computer Interaction*, *6*(CHI PLAY), 1-26.

Vorobeva, D., El Fassi, Y., Costa Pinto, D., Hildebrand, D., Herter, M. M., & Mattila, A. S. (2022). Thinking skills don’t protect service workers from replacement by artificial intelligence. *Journal of Service Research*, *25*(4), 601-613.

Wagan, A. A., Khan, A. A., Chen, Y. L., Yee, P. L., Yang, J., & Laghari, A. A. (2023). Artificial intelligence-enabled game-based learning and quality of experience: A novel and secure framework (B-AIQoE). *Sustainability*, *15*(6), 5362.

Yim, I. H. Y., & Su, J. (2025). Artificial intelligence (AI) learning tools in K-12 education: A scoping review. *Journal of Computers in Education*, *12*(1), 93-131.

Zhai, X., Chu, X., Chai, C. S., Jong, M. S. Y., Istenic, A., Spector, M., ... & Li, Y. (2021). A Review of Artificial Intelligence (AI) in Education from 2010 to 2020. *Complexity*, *2021*(1), 8812542.

Zhao, Y., Cheng, Y., Ding, S., Fang, Y., Cao, W., Liu, K., & Cao, J. (2024, June). Magic Camera: An AI Drawing Game Supporting Instantaneous Story Creation for Children. In *Proceedings of the 23rd Annual ACM Interaction Design and Children Conference* (pp. 738-743).

Pengaruh AI dalam Permainan Game dan berfikir cerdas

Dr. Ir. Agus Wibowo, M.Kom, M.Si, MM

BIO DATA PENULIS

Penulis memiliki berbagai disiplin ilmu yang diperoleh dari Universitas Diponegoro (UNDIP) Semarang. dan dari Universitas Kristen Satya Wacana (UKSW) Salatiga. Disiplin ilmu itu antara lain teknik elektro, komputer, manajemen dan ilmu sosiologi. Penulis memiliki pengalaman kerja pada industri elektronik dan sertifikasi keahlian dalam bidang Jaringan Internet, Telekomunikasi, Artificial Intelligence, Internet Of Things (IoT), Augmented Reality (AR), Technopreneurship, Internet Marketing dan bidang pengolahan dan analisa data (komputer statistik).

Penulis adalah pendiri dari Universitas Sains dan Teknologi Komputer (Universitas STEKOM) dan juga seorang dosen yang memiliki Jabatan Fungsional Akademik Lektor Kepala (Associate Professor) yang telah menghasilkan puluhan Buku Ajar ber ISBN, HAKI dari beberapa karya cipta dan Hak Paten pada produk IPTEK. Sejak tahun 2023 penulis tercatat sebagai Dosen luar biasa di Fakultas Ekonomi & Bisnis (FEB) Universitas Diponegoro Semarang. Penulis juga terlibat dalam berbagai organisasi profesi dan industri yang terkait dengan dunia usaha dan industri, khususnya dalam pengembangan sumber daya manusia yang unggul untuk memenuhi kebutuhan dunia kerja secara nyata.

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 978-634-7227-38-6 (PDF)
9 786347 227386