BAB 2
TINJAUAN PUSTAKA

2.1 Game

Video game merupakan salah satu media hiburan yang paling popular untuk semua kalangan usia. Sejak pertama kali ditemukan sampai saat sekarang, teknologi game telah mengalami kemajuan yang terbilang sangat pesat. Hal ini ditandai dengan berkembangnya jenis, produk, alat, dan jenis interaksi game dengan penggunaan yang semakin beragam bentuknya.

2.1.1 Pengertian Game

Game merupakan sebuah bentuk seni dimana penggunanya, disebut dengan pemain (player), diharuskan membuat keputusan-keputusan dengan tujuan untuk mengelola sumber daya yang diperoleh dari kesempatan-kesempatan bermain (token) miliknya, untuk mencapai suatu tujuan tertentu. Video game adalah bentuk game yang interaksi utamanya melibatkan media video (dan biasanya melibatkan audio).

Berdasarkan representasi visualnya, game dapat dibedakan menjadi 2 jenis, yaitu game 2 dimensi (2D) dan 3 dimensi (3D). Game 2D adalah game, yang secara matematis, hanya melibatkan 2 elemen koordinat kartesius, yaitu x dan y, sehingga konsep kamera pada game 2D hanya menentukan “gambar” mana

pada game yang dapat dilihat oleh pemain. Sedangkan game 3D adalah game yang selain melibatkan elemen x dan y, juga melibatkan elemen z pada perhitungannya, sehingga konsep kamera pada game 3D benar-benar menyerupai konsep kamera pada kehidupan nyata, yaitu selain digeser (seperti pada game 2D), juga dapat diputar dengan sumbu tertentu.

2.1.2 Sejarah Singkat Game

Sejak pertama kali ditemukan sampai saat ini, game telah melewati beberapa fase evolusi penting dimana yang berperan besar terhadap perkembangan game yang sudah sangat pesat sekarang ini.

Pada jaman computer digital modern pertama dikembangkan, yaitu pada masa Perang Dunia II dan pada akhirnya baru tersedia secara komersil pada tahun 1950-an, computer berjalan dalam “batch mode”. Istilah ini untuk menggambarkan bagaimana computer digital tersebut bekerja, pengguna menulis programnya pada punch card lalu computer menjalankan dan menampilkan hasilnya pada suatu alat pencetak. Pada masa ini, untuk bermain game pada computer digital ini, setiiap pemain mengetikkan sesuatu pada suatu kartu data lalu memasukkannya pada computer dan para pemain melihat siapa pemenangnya pada alat pencetak tersebut. Pada masa setelahnya, tahun 1960-an, saat sistem operasi yang mendukung timesharing telah diciptakan, pengguna computer sudah dapat menggunakan computer secara interaktif pada terminal. Namun karena computer masih merupakan barang mahal bernilai jutaan dollar, saat itu belum ada pasar game, sehingga para game programmer membuat game hanya demi

kepuasan pribadi saja. Adapun game pada jaman tersebut ditulis menggunakan bahasa permrograman FORTRAN dan BASIC.

Pada tahun 1971, saat intel telah menciptakan microprocessor, muncullah game Pong, sebuah game arcade yang dioperasikan dengan memasukkan koin ke dalam mesinnya. Ada pula versi console Pong, yang bernama Magnavox Odyssey. Sejak saat itu, game mencetak sukses pertamanya dengan menemukan pasarnya sendiri. Mesin game console yang paling popular saat itu, Atari's 2600 dan Mattel's Intellivision, memungkinkan siapapun bisa membuat ROM cartridge yang berisi game tertentu pada mesin ini. Game pada jaman itu harus dapat dimainkan dengan menggunakan memori 4K, sehingga pemrograman yang dilakukan tidaklah semudah pemrograman berorientasi objek seperti yang kita kenal sekarang ini. Biasanya untuk sebuah game, sumber daya manusia yang dibutuhkan hanyalah programmer saja. Namun akibat proses produksi yang semakin cepat dan semakin tidak menghiraukan kualitas, game menjadi banyak mengandung bug, terlalu mirip satu sama lainnya, dan tidak begitu menarik lagi. Pada tahun 1983, pasar sudah mulai kehilangan minatnya terhadap game dan perusahaan game mulai bangkrut. Setelah masa awal perkembangan game ini, dunia game telah melalui beberapa fase, yaitu fase game PC, fase terlahirnya kembali game console, dan fase game online. Sekarang ketiga fase game tersebut sudah semakin banyak variannya dan masing-masing memiliki kelompok penggemarnya tersendiri.

2.1.3 Sejarah Game Pertama di Indonesia

Divinekids adalah game yang diakui dan mendapat award resmi dari MURI (Museum Rekor Indonesia) sebagai Game Pertama Indonesia pada tahun 2004 dan banyak majalah-majalah Game dan Komputer (Gamestation, PCGAME, Bobo, INDOPOS, Game21 dan lain sebagainya) . Sebenarnya sudah banyak game-game buatan orang-orang indonesia yang lain tetapi karena kekurangan syarat-syarat minimal tertentu seperti : Karakter menggunakan karakter asing, penyebaran yang kurang luas, bahasa yang digunakan bukan bahasa Indonesia, tidak terdaftar resmi Copyright ataupun Trademarknya dan lain sebagainya, karena itu tidak dipilih sebagai Game Pertama Indonesia Resmi.

Menurut sebagian pengamat game ini adalah kurangnya investigasi MURI dalam hal melihat adanya game-game tersebut sebelum memberikan gelar game pertama ke David Setiabudi. Terlebih mereka tidak meminta pendapat dari pihak yang memahami dunia game Indonesia itu sendiri.

Pengamat-pengamat yang berpendapat berbeda dengan MURI dan majalah-majalah PC dan Game, memposisikan sejarah game Indonesia sebagai berikut: Sejarah Game Indonesia menurut Gamedevid. Selain itu juga masih ada banyak game-game lain yang belum terdokumentasi dengan baik. Beberapa game buatan Elex Media Komputindo dengan copyright resmi dan penyebaran yang luas ada juga yang dibuat sebelum sebelum dirilisnya Divine Kids. Game ini dibuat dan dibagikan secara gratis melalui CD majalah2 game/ komputer dan melalui Internet.

Game-game Divinekids berjumlah 38 lebih dan terus bertambah (2006). Pembuatnya adalah David Setiabudi dan mahasiswa-mahasiswi bimbingannya. Pembuatan musik gamenya dibuat oleh Rama Aurora, tunanetra berbakat membuat musik digital.

David Setiabudi membuat game juga tahun 1992 tetapi tidak diakui sebagai game pertama Indonesia karena kurang memenuhi syarat. Ada juga game-game buatan Korea yang terkenal diterjemahkan ke bahasa Indonesia & game-game yang penyebaran kurang luas & game yang berkaraker asing atau berbahasa asing & tidak terdaftar Copyright secara resmi tidak terpilih menjadi Game Pertama Indonesia. Ada yang berpendapat asal tahun tertulis paling tua itu patut disebut Game Pertama Indonesia tidak perduli dengan buatan negara mana, atau tersebar di lokal saja. Sampai sekarang masih diperdebatkan kelompok tertentu.

2.1.4 Jenis-jenis Game

Berikut ini akan dijabarkan beberapa jenis game berdasarkan cara pembuatannya, cara pemasarannya, mesin yang menjalankannya. Jenis game diantaranya adalah :

1. Game PC

   Game yang dimainkan pada PC (Personal Computer) yang memiliki kelebihan yaitu memiliki tampilan antarmuka yang baik untuk input maupun output, output visual kualitas tinggi karena layar computer biasanya memiliki resolusi yang jauh lebih tinggi dibandingkan dengan layar televise biasa.

Kekurangannya adalah spesifikasi computer yang sangat bervariasai antara satu computer dengan computer lainnya menyebabkan beberapa game dapat ditampilkan dengan baik pada satu computer, tetapi tidak berjalan dengan baik pada computer yang lainnya.

2. Game Console

Game yang dijalankan pada suatu mesin spesifik yang biasanya tersedia di rumah pribadi, seperti Microsoft Xbox 360, Nintendo Wii dan sebagainya.

3. Game Arcade

Game yang dijalankan pada mesin dengan input dan output audio visual yang telah terintegrasi dan tersedia ditempat-tempat umum, seperti mal, bandara dan sebagainya.

4. Game Online

Game yang hanya dapat dimainkan secara online melalui LAN atau internet.

2.1.5 Genre Game

Game dibagi-bagi lagi ke dalam beberapa genre (jenis), diantaranya yaitu :

1. Action
2. Fighting
3. Education
4. Adventure

5. Racing
6. Strategy
7. Simulation
8. Puzzle
9. Flight
10. RPG (Roler Playing Game)

2.2 AI (Artificial Intelligence)

AI (Artificial Intelligence) atau kecerdasan buatan merupakan cabang ilmu dari ilmu komputer yang konsern dengan pengautomatisasi tingkah laku cerdas (Desiani Anita dan Arhami Muhammad, 2006). Pernyataan tersebut juga dapat dijadikan definisi dari AI. Definisi ini menunjukkan bahwa AI adalah bagian dari komputer sehingga harus didasarkan pada sound theoretical (teori suara) dan prinsip-prinsip aplikasi dari bidangnya. Prinsip-prinsip ini meliputi struktur data yang digunakan dalam representasi pengetahuan, algoritma yang diperlukan untuk mengaplikasikan pengetahuan tersebut, serta bahasa dan teknik pemrograman yang digunakan dalam mengimplementasikannya.

Namun, definisi di atas belum memadai karena pada kenyataannya istilah kecerdasan itu sendiri tidak didefinisikan atau dipahami dengan sangat baik. Kerap kali kita meyakini telah mengetahui tingkah laku cerdas saat melihatnya sendiri. Akan tetapi, keraguan juga akan muncul ketika orang lain menyatakan

kecerdasan dalam cara khusus menurut cara mereka yang cukup membantu dalam evaluasi kecerdasan program komputer. Dengan demikian, belum ada suatu definisi yang cocok untuk Artificial Intelligence itu sendiri.

Definisi AI ini menjadi masalah tersendiri yang harus dijawab : apakah kecerdasan tersebut merupakan suatu “single faculty”, atau apakah hanya sebuah nama untuk kumpulan perbedaan dan tidak ada hubungannya dengan kemampuan? Apa perlunya mempelajari AI? Apakah ada hubungannya dengan proses belajar, kreativitas dan intuisi? Dapatkah kecerdasan disimpulkan dari tingkah yang dapat diobservasi atau membutuhkan pembuktian dari suatu mekanisme internal yang khusus? Bagaimana pengetahuan direpresentasikan dalam jaringan saraf dari kehidupan? Semua pertanyaan tersebut merupakan pertanyaan yang mungkin belum terjawab dan tentunya akan membantu kita untuk terus berusaha menggeluti, mempelajari dan mencari metode yang dapat menjadi solusi bagi pertanyaan ini, dan nantinya akan membentuk suatu intisari AI modern. AI menawarkan media dan uji teori dari kecerdasan. Teori-teori tersebut dapat dinyatakan dalam bahasa program komputer, diuji dan dibuktikan melalui eksekusi pada komputer.

Karena tidak adanya definisi yang pasti tentang AI maka berikut ini adalah definisi yang diambil dari delapan buku teks (Tabel 2.1). Definisi tersebut dibagi dalam dua kotak utama, kotak bagian atas menunjukkan definisi berkaitan dengan proses pemikiran dan penalaran dan kotak yang satu menunjukkan tingkah laku.

Tabel 2.1 Beberapa Definisi AI

<table>
<tr>
<td>“The existing new effort to make computers think...machine with mind, in full and literal sense”

(Haugeland, 1985)</td>
<td>“The study of mental faculties through the use of computasional models” (Chaniak and McDermott, 1985)

“The study of the computation that make it possible to perceive, reason and act” (Winston, 1992)</td>
</tr>
<tr>
<td>“The art of creating machine that performs functions that require intelligence when performed by people” (Kurzwell, 1990)

“The study of how to make computers do things at which, at the moment, people are better” (Rich and Knight, 1991)</td>
<td>“A field of study that seeks to explain and emulate intelligent behavior in termof computational processes” (Schalkoff, 1990)

“The branch of computer science that is concerned with the automation of intelligent behaviour” (Luger and stubblefield, 1993)</td>
</tr>
<tr>
<td colspan="2">
<table>
<tr><td>System that thinks like humans</td><td>System that thinks rationally</td></tr>
<tr><td>System that acts like humans</td><td>System that acts rationally</td></tr>
</table>
</td>
</tr>
</table>

Dari beberapa definisi dari Tabel 1.1, definisi AI dapat dibagi dalam empat kategori, yaitu :

1. Sistem yang dapat berfikir seperti manusia “Thinking humanly”.
2. Sistem yang dapat bertingkah laku seperti manusia “Acting Humanly”.
3. Sistem yang dapat berfikir secara rasional “Thinking rationally”.
4. Sistem yang dapat bertingkah laku secara rasional “Acting rationally”.

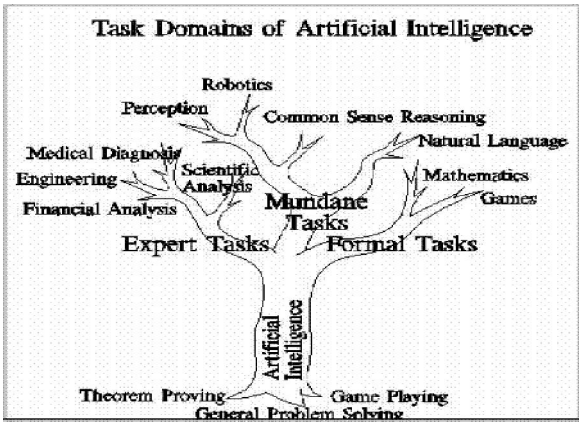
Secara historis, keempat pendekatan tersebut telah dilakukan. Pendekatan manusia haruslah merupakan suatu ilmu empiris, termasuk hipotesa dan konfirmasi percobaan dan pendekatan rasional meliputi kombinasi dari matematika dan rekayasa.

2.2.2 Area Aplikasi AI

Ada dua hal yang sangat mendasar mengenai penelitian-penelitian AI, yaitu knowledge representation (representasi pengetahuan) dan search (pelacakan). Para peneliti AI terus mengembangkan berbagai jenis teknik baru dalam menangani sejumlah permasalahan yang tergolong ke dalam AI seperti vision dan percakapan, pemrosesan bahasa alami dan permasalahan khusus seperti diagnosa medis.

AI seperti bidang ilmu lainnya juga memiliki sejumlah sub disiplin ilmu yang sering digunakan untuk pendekatan yang esensial bagi penyelesaian suatu masalah dan dengan aplikasi bidang AI yang berbeda. Berikut ini merupakan

sejumlah bidang terapan utama dari kontribusinya terhadap perkembangan AI yang digambarkan bentuk pohon berikut ini :



Gambar 2.1 Beberapa bidang terapan AI

Area aplikasi AI diantaranya yaitu (Desiani Anita dan Arhami Muhammad, 2006) :

1. Game Playing.
2. Penalaran Otomatis dan Pembuktian.
3. Sistem Pakar.
4. Pemahaman Bahasa Alami.
5. Perencanaan dan Robotik.
6. Jaringan Saraf Tiruan (JST).
7. Algoritma Genetika.

2.2.3 Perbandingan Kecerdasan Buatan dengan Kecerdasan Alamiah

Ada beberapa keuntungan kecerdasan buatan dibanding kecerdasan alamiah, yaitu :

1. Lebih permanen.
2. Memberikan kemudahan dalam duplikasi dan penyebaran.
3. Relatif lebih murah dari kecerdasan alamiah.
4. Konsisten dan teliti.
5. Dapat didokumentasi.
6. Dapat mengerjakan beberapa task dengan lebih cepat dan lebih baik dibanding manusia.

2.2.4 Perbedaan Komputasi AI dengan Proses Komputasi Konvensional

Tabel 2.2 Pemrosesan data komputer konvensional

Proses	Tugas
Kalkulasi	Mengerjakan operasi-operasi matematis seperti : +, =, x, :, atau mencari akar persamaan, menyelesaikan rumus/ persamaan
Logika	Mengerjakan operasi logika seperti and, or, invert
Penyimpanan	Menyimpan data dan gambar pada file
Retrieve	Mengakses data yang disimpan pada file
Translate	Mengonversi data dari satu bentuk ke bentuk yang lain
Sort	Memeriksa data dan menampilkan dalam urutan yang diinginkan
Edit	Melakukan perubahan, penambahan dan penghapusan data
Monitor	Mengamati event eksternal dan internal serta melakukan tindakan jika kondisi tertentu tercapai
Kontrol	Memberikan perintah atau mengendalikan peralatan luar

Perbandingan antara AI dengan pemrograman konvensional adalah seperti tabel berikut ini :

Tabel 2.3 Perbandingan AI dan pemrograman konvensional

Dimensi	AI	Pemrograman Konvensional
Processing	Simbolik	Algoritmik
Input	Tidak harus lengkap	Harus lengkap
Search	Heuristik	Algoritmik
Explanation	Tersedia	Tidak tersedia
Major interest	Pengetahuan	Data dan Informasi
Struktur	Terpisah antara kontrol dan pengetahuan	Kontrol terintegrasi dengan data
Output	Tidak harus lengkap	Harus lengkap
Maintenance dan update	Mudah karena menggunakan modul-modul	Umumnya susah dilakukan
Hardware	Workstation dan PC	Semua tipe

Kemampuan berfikir	Terbatas tetapi dapat ditingkatkan	Tidak ada

2.2.5 AI yang digunakan

2.2.5.1 Algoritma BFS dan DFS

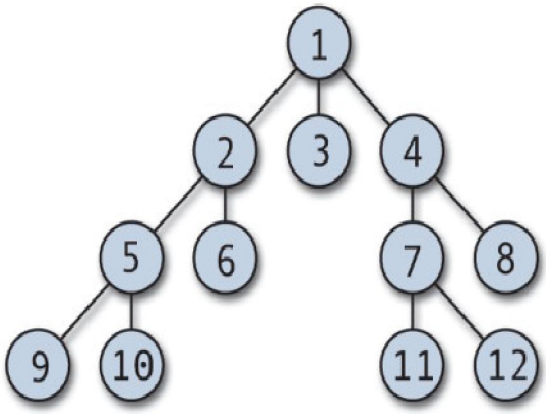
1. Pengertian Algoritma BFS (Breadth First-Search)

Breadth-first search adalah algoritma yang melakukan pencarian secara melebar yang mengunjungi simpul secara preorder yaitu mengunjungi suatu simpul kemudian mengunjungi semua simpul yang bertetangga dengan simpul tersebut terlebih dahulu. Selanjutnya, simpul yang belum dikunjungi dan bertetangga dengan simpul-simpul yang tadi dikunjungi , demikian seterusnya. Jika graf berbentuk pohon berakar, maka semua simpul pada aras d dikunjungi lebih dahulu sebelum simpul-simpul pad aras d+1. Algoritma ini memerlukan sebuah antrian q untuk menyimpan simpul yang telah dikunjungi. Simpulsimpul ini diperlukan sebagai acuan untuk mengunjungi simpul-simpul yang bertetanggaan dengannya. Tiap simpul yang telah dikunjungi masuk ke dalam antrian hanya satu kali. Algoritma ini juga membutuhkan Tabel Boolean untuk menyimpan simpul yang telah dikunjungi sehingga tidak ada simpul yang dikunjungi lebih dari satu kali.

Metode penelusuran ini memeriksa semua node (simpul) pohon pencarian, dimulai dari simpul akar. Simpul-simpul dalam tingkat diperiksa

seluruhnya sebelum pindah ke simpul di tingkat selanjutnya. Proses ini bekerja dari kiri ke kanan, baru bergerak ke bawah. Ini berlanjut sampai ke titik tujuan (goal).

Berikut gambar yang mengilustrasikan urutan simpul yang dikunjungi pada algoritma BFS:



Gambar 2.2 Ilustrasi urutan kunjungan simpul pada algoritma BFS

Dari gambar di atas, dapat dilihat bahwa dengan algoritma BFS, setiap simpul pada tingkat x dikunjungi lebih dahulu sebelum simpul pada tingkat di bawahnya (x+1).

a. Cara Kerja Algoritma BFS

Dalam algoritma BFS, simpul anak yang telah dikunjungi disimpan dalam suatu antrian. Antrian ini digunakan untuk mengacu simpul-simpul yang bertetangga dengannya yang akan dikunjungi kemudian sesuai urutan pengantrian. Untuk memperjelas cara kerja algoritma BFS beserta antrian yang digunakannya, berikut langkah-langkah algoritma BFS:

1. Masukkan simpul ujung (akar) ke dalam antrian
2. Ambil simpul dari awal antrian, lalu cek apakah simpul merupakan solusi
3. Jika simpul merupakan solusi, pencarian selesai dan hasil dikembalikan.
4. Jika simpul bukan solusi, masukkan seluruh simpul yang bertetangga dengan simpul tersebut (simpul anak) ke dalam antrian
5. Jika antrian kosong dan setiap simpul sudah dicek, pencarian selesai dan mengembalikan hasil solusi tidak ditemukan
6. Ulangi pencarian dari langkah kedua

b. Properti Algoritma BFS

Karena seluruh simpul yang telah dikunjungi harus disimpan, maka kompleksitas ruang algoritma BFS adalah O (jumlah simpul+jumlah sisi). Hal ini menunjukkan bahwa algoritma BFS memerlukan ruang besar dan tidak cocok untuk pemecahan masalah besar. Angka yang sama menunjukkan kompleksitas waktu terburuk, di mana setiap simpul harus diperiksa. Kompleksitas waktu terbaik, O(1), didapat jika solusi ditemukan pada pencarian pertama. Walaupun memiliki kompleksitas ruang dan waktu yang kurang baik, algoritma BFS dapat mencari solusi secara tuntas, berarti algoritma ini akan menemukan solusi terbaik dan terdekat bagaimanapun bentuk graf tersebut.

c. Aplikasi Algoritma BFS dalam Teori Graf

Karena algoritma BFS menggunakan graf sebagai media representasi persoalan, tidak sulit untuk mengaplikasikan algoritma ini dalam persoalan-persoalan teori graf.

a. Pengecekan Graf Bipartit

Graf bipartit adalah graf yang seluruh simpulnya dapat dipisah menjadi dua bagian, serta setiap sisi graf menghubungkan dua simpul yang masing-masing berada pada bagian yang berbeda, sehingga tidak ada simpul pada bagian yang sama yang bertetangga. BFS digunakan untuk mengecek apakah suatu graf merupakan graf bipartit atau bukan dengan memulai pencarian pada simpul manapun, lalu memberi nilai 0 dan 1 pada setiap simpul yang dikunjungi. Angka ini menandakan bagian di mana simpul tersebut berada. Contoh pemberian nilai pada graf: Jika simpul awal diberi nilai 0, maka seluruh simpul yang bertetangga dengannya diberi nilai 1, lalu seluruh simpul yang bertetangga dengan simpul bernilai 1 tersebut diberi nilai 0, dan seterusnya.

Dalam pemberian nilai ini, jika dicapai simpul yang telah dikunjungi sebelumnya, dicek apakah simpul tersebut berada pada tingkat yang sama dengan simpul ayahnya, dengan cara pengecekan nilai kedua simpul tersebut yang bertetangga tersebut. Jika simpul bertetangga dengan simpul yang bernilai sama dengannya, maka dapat disimpulkan bahwa graf bukan merupakan graf bipartit. Jika sampai akhir pemberian nilai hal tersebut tidak terjadi, maka graf tersebut merupakan graf bipartit.

b. Pencarian Komponen Terhubung dalam Graf

Dua buah komponen dinyatakan terhubung dalam graf jika terdapat lintasan antara keduanya. Dalam algoritma BFS, seluruh simpul yang dikunjungi pasti merupakan simpul yang terhubung dengan simpul akar, karena simpul-simpul tersebut hanya dikunjungi jika bertetangga dengan simpul ayahnya. Seluruh simpul yang terhubung dengan simpul akar juga pasti dikunjungi. Oleh karena itu, seluruh simpul yang dikunjungi dalam algoritma BFS dan masuk ke dalam pembentukan pohon BFS adalah komponen terhubung terbesar pada graf yang mencakup simpul awal (akar).

c. Pencarian Jarak Terpendek pada Graf Tak Berbobot

Pada graf tak berbobot, jarak terpendek antara dua simpul hanya ditentukan oleh jumlah simpul atau jumlah sisi yang berada di lintasan antara dua simpul tersebut. Untuk mendapatkan jarak terpendek seperti ini, algoritma BFS dapat digunakan untuk menyelesaikan persoalan. Pohon yang dibentuk dalam algoritma BFS memiliki sisi yang menghubungkan simpul-simpul pada setiap tingkat. Tidak mungkin ada sisi dalam graf yang terhubung dan berada di luar tingkat-tingkat pada pohon tersebut.

Hal ini menandakan bahwa pohon algoritma BFS menampilkan jarak terpendek antara simpul akar dan simpul-simpul lainnya dalam graf. Setiap simpul terhubung dengan akar, sehingga jarak terpendek dari simpul ke akar sama dengan tingkat simpul tersebut.

d. Pencarian Diameter Pohon

Diameter dari sebuah pohon adalah lintasan terpanjang dari seluruh jarak terpendek antara dua simpul pada suatu pohon. Dalam pencarian diameter pohon, digunakan algoritma BFS seperti yang telah diterangkan di atas untuk mendapatkan jarak terpendek dari setiap simpul. Hasil dari penghitungan jarak terpendek masing-masing tersebut digunakan bersama dengan satu variabel global yang menyatakan panjang maksimum untuk mendapatkan diameter pohon.

Berikut algoritma untuk mencari diameter pohon:

1. Inisialisasi panjang maksimum = 0
2. Lakukan algoritma BFS pada pohon
3. Masukkan hasil jarak terpendek pada suatu simpul ke
1. variabel temp
4. Jika panjang maksimum < temp, maka panjang maksimum = temp
5. Ulangi dari langkah 2 sampai jarak terpendek seluruh simpul telah dicari
6. Kembalikan panjang maksimum. Panjang maksimum yang dikembalikan merupakan diameter pohon yang dicari.

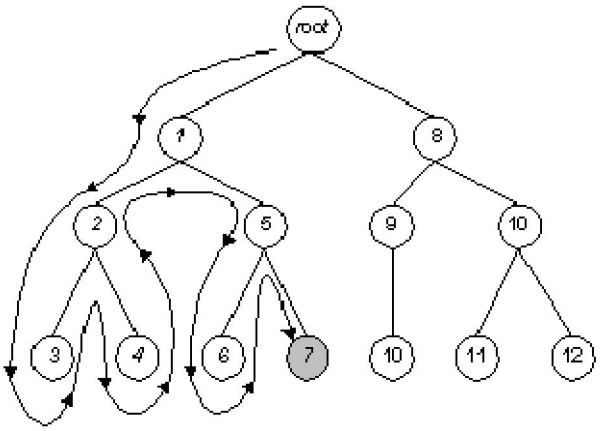
2. Pengertian Algoritma DFS (Depth First-Search)

Depth-first search (DFS) melakukan pencarian secara preorder. Mengunjungi anak suatu simpul sebelum simpul tetangganya. Berkaitan dengan mesin pencari, DFS ini cenderung mengindeks dokumen berdasarkan suatu link.

DFS adalah algoritma untuk melakukan traversal atau pencarian pada sebuah graf atau pohon. Dimulai dari simpul akar, pencarian akan dilakukan dan

mengeksplorasi sejauh mungkin pada tiap cabang sebelum akhirnya melakukan backtracking.

Algoritma pencarian mendalam (DFS) mencari solusi dengan mengunjungi simpul akar, lalu simpul-simpul yang bertetangga dengan simpul akar (setingkat di bawahnya), terus sampai simpul paling dalam pada bagian tersebut. Setelah itu, dicari simpul yang telah dikunjungi pada tingkat terdekat dan terdalam, lalu simpul yang bertetangga dengan simpul ini dikunjungi, demikian seterusnya sampai seluruh simpul telah dikunjungi. Berikut gambar yang mengiilustrasikan urutan simpul yang dikunjungi pada algoritma BFS:



Gambar 2.3 Ilustrasi urutan kunjungan simpul pada algoritma DFS

Dari gambar di atas, dapat dilihat bahwa dengan algoritma DFS, setiap anak simpul pertama yang bertetangga dengan simpul akar dikunjungi sampai tingkat terdalamnya lebih dahulu, lalu seluruh simpul pada subpohon tersebut, sebelum simpul lain yang juga bertetangga dengan simpul akar.

a. Cara Kerja Algoritma DFS

Dalam algoritma DFS, simpul yang telah dikunjungi disimpan dalam suatu tumpukan (stack). Antrian ini digunakan untuk mengacu simpul-simpul yang akan dikunjungi sesuai urutan tumpukan (masuk terakhir, keluar pertama) dan mempermudah proses runut-balik jika simpul sudah tidak mempunyai anak (simpul pada kedalaman maksimal). Untuk memperjelas cara kerja algoritma DFS beserta tumpukan yang digunakannya, berikut langkah-langkah algoritma DFS:

1. Masukkan simpul ujung (akar) ke dalam tumpukan
2. Ambil simpul dari tumpukan teratas, lalu cek apakah simpul merupakan solusi
3. Jika simpul merupakan solusi, pencarian selesai dan hasil dikembalikan.
4. Jika simpul bukan solusi, masukkan seluruh simpul yang bertetangga dengan simpul tersebut (simpul anak) ke dalam tumpukan
5. Jika tumpukan kosong dan setiap simpul sudah dicek, pencarian selesai dan mengembalikan hasil solusi tidak ditemukan
6. Ulangi pencarian dari langkah kedua

b. Properti Algoritma DFS

Berbeda dengan algoritma BFS yang hanya melakukan pencarian solusi pada graf sampai menemukannya, sehingga membutuhkan waktu yang lama, algoritma DFS menggunakan metode heuristik, sehingga solusi dapat ditemukan lebih cepat.

Kompleksitas ruang algoritma DFS juga jauh lebih rendah dari algoritma BFS, karena hanya menyimpan simpul-simpul pada suatu subpohon. Akan tetapi, kompleksitas waktu algoritma DFS sama dengan kompleksitas waktu algoritma BFS. Algoritma DFS juga dapat mencari solusi secara tuntas seperti algoritma BFS, berarti algoritma ini akan menemukan solusi terbaik dan terdekat bagaimanapun bentuk graf tersebut. Walaupun demikian, masih terdapat kelemahan algoritma DFS, yaitu jika pada graf yang sangat besar kedalaman graf terus bertambah dan tidak diketahui akhirnya, sehingga memori tidak cukup. Hal ini dapat diatasi dengan pengembangan algoritma DFS yaitu pencarian mendalam berulang (Iterative Deepening Search - IDS).

c. Aplikasi Algoritma DFS dalam Teori Graf

Seperti halnya algoritma BFS, algoritma DFS juga dapat digunakan dengan mudah pada teori graf. Bahkan banyak persoalan yang tidak dapat diselesaikan algoritma BFS yang dapat diselesaikan oleh algoritma DFS. Berikut contoh-contoh penggunaan algoritma BFS dalam
persoalan teori graf:

d. Pencarian Komponen Terhubung dalam Graf

Sama seperti pada algoritma BFS, dalam algoritma DFS, seluruh simpul yang dikunjungi juga pasti merupakan simpul yang terhubung dengan simpul akar, karena simpul-simpul tersebut hanya dikunjungi jika bertetangga dengan simpul ayahnya, sehingga seluruh simpul yang terhubung dengan simpul akar juga pasti dikunjungi.

Dalam hal pencarian komponen terhubung dalam graf, algoritma BFS dan DFS menghasilkan pencarian yang memiliki kecepatan dan kebutuhan memori sama, karena setiap simpul harus dikunjungi. Hasil pencarian juga sama, walaupun proses yang dilakukan berbeda. Kedua algoritma dapat menghasilkan komponen terhubung terbesar pada graf yang mencakup simpul awal (akar). Oleh karena itu, kedua algoritma dapat digunakan untuk menyelesaikan persoalan ini dengan mangkus.

e. Pengurutan Topologi

Dalam teori graf, pengurutan topologi dari graf DAG (graf berarah tanpa siklus) adalah pengurutan linear simpul-simpul di dalam graf yang bersesuaian dengan urutan di mana x dikunjungi sebelum y jika terdapat lintasan dari x ke y. Pengurutan topologi dapat diaplikasikan pada penjadwalan sekuensial suatu pekerjaan. Dengan menggunakan algoritma DFS, pengurutan topologi dilakukan dengan algoritma berikut:

1. Lakukan algoritma DFS pada graf G
2. Inisialisasi list kosong
3. Jika suatu simpul telah selesai dipanggil seluruh anaknya, masukkan simpul ke dalam list
4. Kembalikan list yang telah berisi semua simpul.

Urutan setiap simpul didapat dari posisinya dalam list, mulai dari awal list.

f. Pencarian Komponen Terhubung Kuat

Jika algoritma BFS dapat mencari komponen terhubung pada graf tak berarah, algoritma DFS dapat melakukan sesuatu yang lebih rumit, yaitu mencari komponen terhubung kuat pada graf berarah. Graf berarah disebut terhubung kuat jika untuk setiap pasangan simpul A dan B terdapat lintasan dari A ke B serta dari B ke A. Komponen terhubung kuat pada graf berarah merupakan subgraf terhubung kuat maksimal pada suatu graf. Komponen ini membentuk partisi dari graf.

Dalam pencarian komponen terhubung kuat, digunakan dua kali algoritma DFS, yaitu pada graf serta graf transpose dari graf tersebut. Berikut algoritma DFS yang digunakan dalam pencarian komponen terhubung kuat:

1. Lakukan algoritma DFS pada graf G
2. Cari graf transpose dari G: GT
3. Lakukan algoritma DFS pada graf GT secara terbalik
4. Kembalikan output hutan DFS yang terbentuk pada langkah 3 sebagai suatu komponen terhubung kuat

g. Penggunaan Algoritma DFS pada Aplikasi game yang dibangun

Algoritma DFS pada game ini digunakan pada class BatuJatuh.

2.3 Analisis Sistem

Analisis sistem yaitu proses penguraian suatu sistem yang akan dirancang oleh satu orang atau sekelompok orang, dimana informasi yang utuh akan diuraikan ke dalam bagian-bagian komponennya dengan maksud untuk mengidentifikasikan dan mengevaluasi permasalahan, kesempatan, hambatan yang terjadi dan kebutuhan yang diharapkan.

Langkah-langkah yang harus dilakukan untuk menganalisis sebuah sistem yaitu :

1. Mengidentifikasi masalah
2. Memahami kerja sistem yang ada atau yang akan dibangun
3. Menganalisis sistem.
4. Membuat laporan hasil analisis.

2.4 Tools yang Digunakan

2.4.1 OOP (Object Oriented Programming)

OOP (Object Oriented Programming) atau yang dikenal dengan Pemrograman Berorientasi Objek merupakan paradigma pemrograman yang berorientasikan kepada objek. Semua data dan fungsi di dalam paradigma ini dibungkus ke dalam kelas-kelas atau objek-objek.

Model data berorientasi objek dikatakan dapat memberi fleksibilitas yang lebih, kemudahan mengubah program, dan digunakan luas dalam teknik piranti lunak skala besar. Lebih jauh lagi, pendukung OOP mengklaim bahwa OOP lebih mudah dipelajari bagi pemula dibanding dengan pendekatan sebelumnya, dan pendekatan OOP lebih mudah dikembangkan dan dirawat.

Dengan menggunakan OOP maka dalam melakukan pemecahan suatu masalah kita tidak melihat bagaimana cara menyelesaikan suatu masalah tersebut (terstruktur) tetapi objek-objek apa yang dapat melakukan pemecahan masalah tersebut. Sebagai contoh anggap kita memiliki sebuah departemen yang memiliki manager, sekretaris, petugas administrasi data dan lainnya. Misal manager tersebut ingin memperoleh data dari bag administrasi maka manager tersebut tidak harus mengambilnya langsung tetapi dapat menyuruh petugas bag administrasi untuk mengambilnya. Pada kasus tersebut seorang manager tidak harus mengetahui bagaimana cara mengambil data tersebut tetapi manager bisa mendapatkan data tersebut melalui objek petugas adminiistrasi. Jadi untuk menyelesaikan suatu masalah dengan kolaborasi antar objek-objek yang ada karena setiap objek memiliki deskripsi tugasnya sendiri (Sumber : Raharjo.B, Heryanto. I, Arif Haryano, Mudah Belajar JAVA, Informatika, Bandung, 2009.).

Pemrograman orientasi-objek menekankan konsep berikut:

- Kelas (Class) — kumpulan atas definisi data dan fungsi-fungsi dalam suatu unit untuk suatu tujuan tertentu. Sebagai contoh 'class of dog' adalah suatu unit yang terdiri atas definisi-definisi data dan fungsi-fungsi yang

menunjuk pada berbagai macam perilaku/turunan dari anjing. Sebuah class adalah dasar dari modularitas dan struktur dalam pemrograman berorientasi objek. Sebuah class secara tipikal sebaiknya dapat dikenali oleh seorang non-programmer sekalipun terkait dengan domain permasalahan yang ada, dan kode yang terdapat dalam sebuah class sebaiknya (relatif) bersifat mandiri dan independen (sebagaimana kode tersebut digunakan jika tidak menggunakan OOP). Dengan modularitas, struktur dari sebuah program akan terkait dengan aspek-aspek dalam masalah yang akan diselesaikan melalui program tersebut. Cara seperti ini akan menyederhanakan pemetaan dari masalah ke sebuah program ataupun sebaliknya.

- Objek (Object) - membungkus data dan fungsi bersama menjadi suatu unit dalam sebuah program komputer. Objek merupakan dasar dari modularitas dan struktur dalam sebuah program komputer berorientasi objek.

- Abstraksi (Abstract) - Kemampuan sebuah program untuk melewati aspek informasi yang diproses olehnya, yaitu kemampuan untuk memfokus pada inti. Setiap objek dalam sistem melayani sebagai model dari "pelaku" abstrak yang dapat melakukan kerja, laporan dan perubahan keadaannya, dan berkomunikasi dengan objek lainnya dalam sistem, tanpa mengungkapkan bagaimana kelebihan ini diterapkan. Proses, fungsi atau metode dapat juga dibuat abstrak, dan beberapa teknik digunakan untuk mengembangkan sebuah pengabstrakan.

- Enkapsulasi (Encapsulation) - Memastikan pengguna sebuah objek tidak dapat mengganti keadaan dalam dari sebuah objek dengan cara yang tidak layak; hanya metode dalam objek tersebut yang diberi ijin untuk mengakses keadaannya. Setiap objek mengakses interface yang menyebutkan bagaimana objek lainnya dapat berinteraksi dengannya. Objek lainnya tidak akan mengetahui dan tergantung kepada representasi dalam objek tersebut.

- Polimorfisme (Polimorfism) melalui pengiriman pesan. Tidak bergantung kepada pemanggilan subrutin, bahasa orientasi objek dapat mengirim pesan; metode tertentu yang berhubungan dengan sebuah pengiriman pesan tergantung kepada objek tertentu di mana pesa tersebut dikirim. Contohnya, bila sebuah burung menerima pesan "gerak cepat", dia akan menggerakan sayapnya dan terbang. Bila seekor singa menerima pesan yang sama, dia akan menggerakkan kakinya dan berlari. Keduanya menjawab sebuah pesan yang sama, namun yang sesuai dengan kemampuan hewan tersebut. Ini disebut polimorfisme karena sebuah variabel tungal dalam program dapat memegang berbagai jenis objek yang berbeda selagi program berjalan, dan teks program yang sama dapat memanggil beberapa metode yang berbeda di saat yang berbeda dalam pemanggilan yang sama. Hal ini berlawanan dengan bahasa fungsional yang mencapai polimorfisme melalui penggunaan fungsi kelas-pertama.

- Inheritas (Inheritance)- Mengatur polimorfisme dan enkapsulasi dengan mengijinkan objek didefinisikan dan diciptakan dengan jenis khusus dari objek yang sudah ada - objek-objek ini dapat membagi (dan memperluas) perilaku mereka tanpa haru mengimplementasi ulang perilaku tersebut (bahasa berbasis-objek tidak selalu memiliki inheritas).

2.4.2 UML (Unified Modeling Language)

UML (Unified Modeling Language) adalah bahasa spesifikasi standar untuk mendokumentasikan, menspesifikasikan, dan membangun system (Flowler, 2006). Unified Modeling Language (UML) adalah himpunan struktur dan teknik untuk pemodelan desain program berorientasi objek (OOP) serta aplikasinya. UML adalah metodologi untuk mengembangkan sistem OOP dan sekelompok perangkat tool untuk mendukung pengembangan sistem tersebut. UML mulai diperkenalkan oleh Object Management Group, sebuah organisasi yang telah mengembangkan model, teknologi, dan standar OOP sejak tahun 1980-an. Sekarang UML sudah mulai banyak digunakan oleh para praktisi OOP. UML merupakan dasar bagi perangkat (tool) desain berorientasi objek dari IBM.



Gambar 2.4 Logo UML

UML adalah suatu bahasa yang digunakan untuk menentukan, memvisualisasikan, membangun, dan mendokumentasikan suatu sistem informasi. UML dikembangkan sebagai suatu alat untuk analisis dan desain berorientasi objek oleh Grady Booch, Jim Rumbaugh, dan Ivar Jacobson. Namun demikian UML dapat digunakan untuk memahami dan mendokumentasikan setiap sistem informasi. Penggunaan UML dalam industri terus meningkat. Ini merupakan standar terbuka yang menjadikannya sebagai bahasa pemodelan yang umum dalam industri peranti lunak dan pengembangan system.

UML menyediakan 10 macam diagram untuk memodelkan aplikasi berorientasi objek, yaitu :

1. Use Case Diagram untuk memodelkan proses bisnis.
2. Conceptual Diagram untuk memodelkan konsep-konsep yang ada di dalam aplikasi.
3. Sequence Diagram untuk memodelkan pengiriman pesan (message) antar objects.
4. Collaboration Diagram untuk memodelkan interaksi antar objects.

5. State Diagram untuk memodelkan perilaku objects di dalam sistem.
6. Activity Diagram untuk memodelkan perilaku Use Cases dan objects di dalam system.
7. Class Diagram untuk memodelkan struktur kelas.
8. Object Diagram untuk memodelkan struktur object.
9. Component Diagram untuk memodelkan komponen object.
10. Deployment Diagram untuk memodelkan distribusi aplikasi.

2.4.3 Java

Java adalah bahasa pemrograman yang disusun oleh James Gosling yang dibantu oleh rekan-rekannya seperti Patrick Naugton, Chris Warth, Ed Frank, dan Mike Sheridan di suatu perusahaan perangkat lunak yang bernama Susn Microsystems, pada tahun 1991 (Raharjo.B, Heryanto. I, Arif Haryano, 2009, hlm. 1-2). Bahasa pemrograman ini mula-mula diinisialisasi dengan nama “Oak”, namun oada tahun 1995 diganti namanya menjadi “Java”.

Alasan utama pembentukan bahasa Java adalah untuk membuat aplikasi-aplikasi yang dapat diletakkan diberbagai macam perangkat elektronik, seperti microwave oven dan remote control, sehingga Java harus bersifat porTabel atau yang sering disebut dengan platform independent (tidak bergantung pada platform). Itula yang menyebabkan dalam dunia pemrograman Java, dikenal adanya istilah ‘write once, run everywhere’, yang berarti kode program hanya ditulis sekali, namun dapat dijalankan di bawah platform manapun, tanpa harus melakukan perubahan kode program.

2.4.3.1 Java Versi Lama (Java 1)

Pada awalnya perilisannya, versi Java masih disebut dengan JDK (Java Development Kit). Dalam JDK, semua kebutuhan untuk pengembangan program dan eksekusi program masih tergabung jadi satu. Penamaan ini berlaku sampai dengan Java 1.1. Namun sekarang, setelah Java 1.2, Sun Mycrosystems menamainya dengan JSDK (Java Software Development Kit) dalam hal ini kebutuhan untuk pengembangan program dipisahkan dengan kebutuhan eksekusi. Bagian software yang digunakan untuk kebutuhan eksekusi program disebut dengan JRE (Java-Runtime Environment). Selanjutnya, Java 1.2 disederhanakan penamaannya menjadi “Java 2”.

2.4.3.2 Java 2

Sun Myrosystems telah mendefinisikan tiga buah edisi dari Java 2 yaitu sebagai berikut :

1. Java 2 Standard Edition (J2SE), yang digunakan untuk mengembangkan aplikasi-aplikasi desktop dan applet (aplikasi untuk Java yang dapal dijalankan di dalam browser web).
2. Java 2 Enterprise Edition (J2EE), merupakan superset dari J2SE yang memperbolehkan kita untuk mengembangkan aplikasi-aplikasi berskala besar(enterprise), yaitu dengan melakukan pembuatan aplikasi-aplikasi di sisi server dengan menggunakan EJBs (Enterprise JavaBeans), aplikasi web dengan menggunakan Servlet dan JSP (JavaServer Pages) dan teknologi

lainnya seperti CORBA (Common Object Request Broker Architecture) dan XML (Extensible Markup Language).

3. Java 2 Micro Edition (J2ME), merupakan subset dari J2SE yang digunakan untuk menangani pemrograman di dalam perangkat-perangkat kecil, yang tidak memungkinkan untuk mendukung implementasi J2SE secara penuh seperti pada teknologi mobile.

2.4.4 Greenfoot

Greenfoot merupakan sebuah IDE berbasis java yang dikhususkan untuk pembuatan game sederhana. Greenfoot diprakarsai oleh Michael Kolliung pada tahun 2003. Prototype yang pertama dibangun oleh Poul Henriksson dan Michael Kolling pada tahun 2003/2004. semenjak tahun 2005 pembuatan greenfoot ini dilanjutkan dengan melibatkan anggota kelompok dari University of Kent dan Deakin University.

Pada awalnya, greenfoot bertujuan untuk menarik minat anak-anak untuk belajar pemrograman, greenfoot ini cocok untuk anak usia 13 tahun ke atas. Cara penggunaannya pun cukup mudah. Selain berbentuk teks seperti editor lainnya, pada greenfoot ini terdapat perangkat alat bantu seperti class browser, editor dan compiler. Greenfoot juga mendukung bahasa Java secara utuh. Dengan adanya bantuan perangkat-perangkat tadi, konsep OOP pada greenfoot lebih mudah dimengerti.

Didalam greenfoot terdapat project yang dinamakan scenario. Di dalam project tersebut, kita bisa membuat dua macam tipe class dari library

greenfoot. Yaitu class world dan actor. World merupakan latar dari game yang akan dibuat. Untuk membuat class world, caranya adalah dengan meng-klik new sub class di bagian world. Maka akan mucul sebuah kotak dialog. Kemudian kita diminta untuk memberi nama untuk kelas world tersebut. Setelah itu kita dapat memilih gambar sebagai latar belakang class world yang akan kita buat. Di dalam greenfoot sendiri telah tersedia beberapa gambar untuk latar belakang, seperti gambar tanah, pasir, batu, air, dll.

Class actor adalah sebuah kelas yang berfungsi sebagai karakter pada game yang akan kita buat. Untuk membuat actor caranya adalah dengan meng-klik icon aktor, kemudian pilih new subclass. caranya hampir mirip dengan pembuatan class world. Pada class actor juga telah disediakan beberapa gambar, sepertii gambar animal, people, symbol, dll

2.4.5 Corel Draw

Corel draw 12 adalah sebuah aplikasi grafis berbasis vector. Format vector adalah gambar yang membentuk sejumlah objek garis dan objek kurva berdasarkanrumusan matematis. Format vector lebih banyak digunakan untuk membentuk objek buatan, seperti menggambar objek dua dimensi, yang lebih ditekankan ke dalam pembuatan objek garis, lingkaran, polygon dan persegi panjang. Sedangkan untukobjek tiga dimensi, lebih ditekankan ke dalam pembuatan : bola, kubus dan tabung.

Objek vector, banyak digunakan dalam pembuatan pengolahan teks dan logo.Software yang banyak berhubungan dengan format vector yaitu

Corel Draw, freehand, dan lain-lain. Ruang penyimpanan format vector relafitf lebih kecil dibandingkan dengan format bitmap.