BAB II

LANDASAN TEORI

II.1 Definisi Permainan

Seiring dengan majunya dunia hiburan, salah satu area *entertainment* yang cukup banyak melibatkan ilmuwan dan seniman adalah industri permainan (*gaming industry*). Dulunya permainan merupakan salah satu aspek hiburan yang minor, hanya sebagai selingan saja. Dan dianggap tidak menghasilkan sesuatu. Namun kini hal itu mulai berubah. Perubahan itu ditandai dengan munculnya berbagai konsul yang cukup bervariasi yang menunjukkan bahwa dunia permainan selalu mengalami perkembangan.

Menurut Agustinus Nilwan dalam bukunya *"Pemrograman Animasi dan Game Profesional"* terbitan Elex Media Komputindo, *game* merupakan permainan komputer yang dibuat dengan teknik dan metode animasi. Artinya jika ingin mendalami pengunaan animasi haruslah memahami pembuatan game. Atau jika ingin membuat game, maka haruslah memahami teknik dan metode animasi, sebab keduanya saling berkaitan.

Namun sebelum akhirnya memahami bagaimana pembuatan "*game*" harus diketahui dahulu arti dari kata "*game*" itu sendiri kemudian menentukan karakteristik dari seluruh permainan yang ada. *Games are a fundamental part of human existence. The parlance of games has insinuated itself into our language to refer to activities that are not truly games.* (Chris Crawford, 1982). *Games* adalah bagian fundamental dari eksistansi manusia. Secara bahasa "*games*" mempunyai

makna sindiran yang mengacu pada aktifitas yang bukan "*games*" sesungguhnya. Menurutnya apabila ingin memahami tentang "*games*" dan desain "game" yang harus dilakukan pertama kali ialah menentukan pokok orientasinya. Kemudian menentukan seluruh karakteristik "*game*" yang ada.

II.1.1 Klasifikasi Permainan

Secara umum, permainan yang ada pada saat ini dapat diklasifikasikan berdasarkan pada jenis *platform* dan *genre* permainannya. Berdasarkan jenis *platform*, permainan digolongkan menjadi:

1. *Arcade games*, yaitu yang sering disebut ding-dong di Indonesia, biasanya berada di daerah / tempat khusus dan memiliki box atau mesin yang memang khusus di design untuk jenis video games tertentu dan tidak jarang bahkan memiliki fitur yang dapat membuat pemainnya lebih merasa "masuk" dan "menikmati", seperti pistol, kursi khusus, sensor gerakan, sensor injakkan dan stir mobil (beserta transmisinya tentunya).
2. *PC games*, yaitu permainan yang dimainkan menggunakan *Personal Computers*.
3. *Consule games*, yaitu *video games* yang dimainkan menggunakan konsul tertentu, seperti *Playstation* 2, *Playstation* 3, XBOX 360, dan Nintendo Wii.
4. *Handheld games*, yaitu yang dimainkan di console khusus video game yang dapat dibawa kemana-mana, contoh Nintendo DS dan Sony PSP.

5. *Mobile games*, yaitu yang dapat dimainkan atau khusus untuk *mobile phone* atau PDA.

Sedangkan berdasarkan *genre* permainannya, dikelompokkan menjadi:

1. Aksi – *Shooting*, (tembak-tembakan , atau hajar-hajaran bisa juga tusuk-tusukan, tergantung cerita dan tokoh di dalamnya), video game jenis ini sangat memerlukan kecepatan refleks, koordinasi mata-tangan, juga timing, inti dari permainan jenis ini adalah tembak, tembak dan tembak. Seperti : *Counter Strike, Spy Hunter, 1942, Double Dragon*, dan sebagainya.
2. *Fighting* (pertarungan), inti dari permainan ini adalah penguasaan jurus (hafal caranya dan lancar mengeksekusinya), hal yang sangat penting adalah pengenalan karakter dan timing. Dan berbeda seperti game Aksi pada umumnya yang umumnya hanya melawan Artificial Intellegence atau istilah umumnya melawan komputer saja, pemain jenis fighting game ini baru teruji kemampuan sesungguhnya dengan melawan pemain lainnya. Contohnya *Seri Street Fighter, Tekken, Mortal Kombat, Soul Calibur* dan *King of Fighter*.
3. Aksi – Petualangan. Pada perkembangannya permainan jenis ini mengalami perkembangan yang sangat jauh, di tahun 2000 an, jenis ini cenderung untuk memiliki visual 3D dan sudut pandang orang ke-tiga. Contohnya *Tomb Rider, Grand Theft Auto* dan *Prince of Persia*.
4. Petualangan. Bedanya dengan jenis *video game* aksi-petualangan, refleks dan kelihaian pemain dalam bergerak, berlari, melompat hingga memecut

atau menembak tidak diperlukan di sini. Pada permainan ini lebih menekankan pada jalan cerita dan kemampuan berpikir pemain dalam menganalisa tempat secara visual, memecahkan teka-teki maupun menyimpulkan rangkaian peristiwa dan percakapan karakter hingga penggunaan benda-benda tepat pada tempat yang tepat. Termasuk didalamnya:

a. Petualangan dengan teks atau sistem tunjuk dan klik, contoh: *Kings Quest, Space Quest, Heroes Quest, Monkey Island, Sam and Max.*

b. Novel atau film interaktif, seperti game "dating" yang banyak beredar di jepang, *Dragons Lair* dan *Night Trap*.

5. Simulasi, Konstruksi dan manajemen. Video permainan jenis ini seringkali menggambarkan dunia di dalamnya sedekat mungkin dengan dunia nyata dan memperhatikan dengan detil berbagai faktor. Seperti membangun rumah, gedung hingga kota. Dunia kehidupan rumah tangga sampai bisnis membangun konglomerasi, dari jualan limun pinggir jalan hingga membangun laboratorium kloning. sehingga membuat pemain harus berpikir untuk mendirikan, membangun dan mengatasi masalah dengan menggunakan dana yang terbatas. Contoh: *Sim City, The Sims, Tamagotchi.*

6. *Role Playing*. Sesuai dengan terjemahannya yaitu bermain peran, memiliki penekanan pada tokoh/peran perwakilan pemain di dalam permainan, yang biasanya adalah tokoh utamanya yang sering dimainkan, karakter tersebut dapat berubah dan berkembang ke arah yang diinginkan pemain (biasanya

menjadi semakin hebat, semakin kuat, semakin berpengaruh, dan sebagainya) dalam berbagai parameter yang biasanya ditentukan dengan naiknya level, baik dari status kepintaran, kecepatan dan kekuatan karakter, senjata yang semakin sakti, ataupun jumlah teman. Secara kebudayaan, pengembang game Jepang biasanya membuat *Role Playing Game* (RPG) ke arah cerita linear yang diarahkan seolah karakter kita adalah tokoh dalam cerita itu, seperti *Final Fantasy, Dragon Quest* dan *Xenogears*. Sedangkan pengembang game RPG Eropa, cenderung membuat karakter kita bebas memilih jalan cerita sendiri secara non-linear, seperti *Ultima, Never Winter Nights, baldurs gate, Elder Scroll,* dan *Fallout*.

7. *Strategy*. Pada permainan ini memerlukan keahlian berpikir dan memutuskan setiap gerakan secara hati-hati dan terencana. Pemain game strategi melihat dari sudut pandang lebih meluas dan lebih kedepan dengan waktu permainan yang biasanya lebih lama dan santai dibandingkan action-games. Permainan jenis ini terbagi atas:

   a. *Real time Strategy,* permainan berjalan dalam waktu sebenarnya dan serentak antara semua pihak dan pemain harus memutuskan setiap langkah yang diambil saat itu juga berbarengan mungkin saat itu pihak lawan juga sedang mengeksekusi strateginya. Contoh: *Starcraft, Warcraft* , dan *Command and Conquer*.

   b. *Turn based Strategy* , game yang berjalan secara bergiliran, misal saat kita mengambil keputusan, saat itu pihak lawan menunggu,

begitu pula sebaliknya, layaknya catur. Contoh: *Front Mission, Super robot wars, Final Fantasy tactics, Heroes of might and magic, Master of Orion*.

8. *Puzzle*. Inti permainan ini mengenai pemecahan teka-teki, baik itu menyusun balok, menyamakan warna bola, memecahkan perhitungan matematika, melewati labirin, sampai mendorong-dorong kota masuk ke tempat yang seharusnya. Contoh *Tetris, Minesweeper, Bejeweled, Sokoban dan Bomberman*.
9. Simulasi kendaraan. Perainan jenis ini memberikan pengalaman atau interaktifitas sedekat mungkin dengan kendaraan yang aslinya, muskipun terkadang kendaraan tersebut masih eksperimen atau bahkan fiktif, tapi ada penekanan khusus pada detil dan pengalaman realistik menggunakan kendaraan tersebut.
10. Olahraga. Biasanya permainannya diusahakan serealistik mungkin walau kadang ada yang menambah unsur fiksi seperti NBA JAM. Contohnya pun jelas, *Seri Winning Eleven, seri NBA, seri FIFA, John Madden NFL, Lakers vs Celtics, Tony hawk pro skater*, dan lain-lain.

II.2 Definisi dan Konsep Permainan NIM

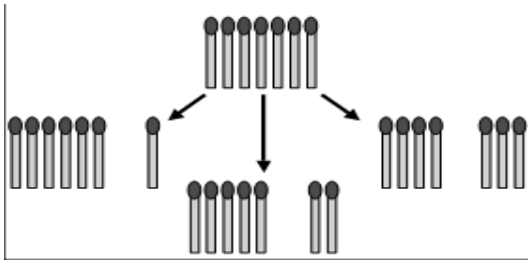
Nim merupakan salah satu permainan tradisional yang saat ini mulai jarang dimainkan. Pada faktanya, di Indonesia banyak orang yang tidak mengetahui apa itu permainan nim. Hal ini wajar, mengingat asal dari permainan itu sendiri. Dalam permainan berbasis komputer, nim termasuk permainan yang bersifat

Single Player mode yang artinya permainan dapat dimainkan sendiri dengan komputer sebagai lawan mainnya.

Nim biasanya dimainkan sebagai *misere game*, dimana pemain yang mengambil objek terakhir maka ialah yang kalah. Nim juga dapat dimainkan secara normal, yaitu pemain yang mengambil objek terakhir menang. Hal ini dikatakan normal karena mengikuti konvensi, sedangkan Nim biasanya tidak mengikuti konvensi tersebut.

Nim telah secara matematika dipecahkan untuk berbagai banyaknya tumpukan dan objek. Telah ada sebuah metode untuk menentukan pemain mana yang akan menang dan strategi apa yang bisa digunakan untuk memenangkan permainan ini. Kunci dari teori permainan adalah *binary digital sum* dari ukuran tumpukan, yaitu jumlah dalam biner dengan mengabaikan semua kelebihan dari penjumlahan tiap digitnya. Operasi ini dikenal sebagai *exclusive or* (xor). Dalam permainan kombinatorial operasi ini biasanya dinamakan nim-sum.

Adapun cara bermain Nim, yaitu misal diasumsikan dengan 7 batang korek, pemain bermain secara bergiliran, setiap pemain dapat mengambil 1, 2, atau 3 batang korek, dan pemain yang terakhir mengambil batang korek maka ialah yang kalah.



Gambar II.1 Ilustrasi Permainan Nim

II.2.1 Sejarah Permainan Nim

Nim merupakan suatu permainan strategi matematika yang dimainkan oleh dua orang pemain dimana setiap pemain harus memindahkan objek secara bergiliran dari tumpukan yang berbeda. Permainan ini sebenarnya sudah ada sejak zaman dahulu. Kabarnya permaiann ini merupakan permaian dari Cina yang bernama *Tsyanshidzi* atau "mengambi batu". Asal kata "Nim" itu sendiri sebenarnya berasal dari German yaitu *nimm* yang artinya "*take*" atau "ambil". Di Eropa permainan nim ini sudah ada sejak awal abad ke-16. Salah seorang professor matematika dari Harvard, Charles Bouton, ialah yang mengembangkan teori permainan nim ini.

II.3 *Combinatorial Game Theory*

Teori Permainan Kombinatorial (*Combinatorial Game Theory*) adalah salah satu teori matematika yang hanya mempelajari permainan dengan dua pemain yang memiliki posisi dimana setiap pemain bermain bergantian untuk memperoleh kemenangan.

Dalam teori matematika terdapat teori yang disebut teori permainan (*Game Theory*). Perlu diketahui bahwa antarteori permainan kombinatorial dan teori permainan ini berbeda. Teori permainan digunakan pada teori ekonomi dan saat ini mulai banyak digunakan di bidang akademink lainnya seperti biologi, psikologi, sosiologi, dan lain-lain. Hal yang memebedakannya ialah pada teori permainan, permainan yang dipelajari sebagian besar merupakan permainan yang memiliki *imperfect information*, sehingga tidak semua pemain mengetahui

tindakan dari pemain lainnya. Selain itu pemain bermain secara bersama-sama. Sedangkan pada teori permainan kombinatorial, permainan dijalankan secara bergiliran.

Pada permainan kombinatorial, permainan hanya akan berakhir dengan satu kemenangan bagi satu orang pemain atau seri. Sepanjang permainan, pemain yang mendapat giliran memindahkan objek pertama kali disebut sebagai *Player1* dan yang mendapat giliran kedua disebut sebagai *Player2*. Pada umumnya, yang disebut sebagai permainan kombinatorial ialah yang memiliki sayarat seperti :

1. Permainan memiliki tepat 2 pemain.
2. Pada umumnya pemain yang melakukan langkah terakhir ialah yang menang (seperti permainan nim).
3. Pemain bermain secara bergliran.
4. Permianan memiliki akhir, tidak berlangsung terus menerus.
5. Tidak ada informasi yang disembunyikan dari pemain.
6. Permainan tidak berdasarkan pada keberuntungan.

II.3.1 Sejarah *Combinatorial Game*

Teori Kombinatorial, ditemukan pada awal abad 20, sesuai dengan analisis rekursif dari permainan kombinatorial, yaitu 2 pemain memiliki posisi dimana setiap pemain bermain bergantian untuk memperoleh kemenangan. Dalam aplikasinya, teori ini digunakan untuk menganalisis strategi dari permainan yang memiliki perfect information, yaitu hanya satu orang pemain yang melangkah tiap

waktu dan masing-masing pemain mengetahui setiap tindakan dari pemain yang bergerak sebelum pemain tersebut pada setiap titik.

Dua yayasan yang bekerja di lapangan partizan *games* adalah *Conway's On Numbers and Games and Berkelamp, Conway, dan Guy's Winning Ways for Your Mathematical Plays*. Pembelajaran mengenai impartial games dimulai dengan Zermelo's theorem. Theorem1 (Teori Zermelo, [Mortimer Brown.2006]), *In a combinatorial game, either one of the players has a formal strategy that guarantees a win, or both players have formal strategies that guarantee at least a draw.*

II.4 Definisi *Artificial Intelligence* (AI)

Menurut John McCarthy, 1956, Kecerdasan Buatan :

Untuk mengetahui dan memodelkan proses-proses berpikir manusia dan mendesain mesin agar dapat menirukan perilaku manusia.

Definisi kecerdasan buatan apabila ditinjau dari berbagai macam sudut:

1. Sudut pandang kecerdasan

   Bahwa kecerdasan buatan akan membuat mesin menjadi 'cerdas' atau mampu melakukan seperti yang dilakukan manusia.

2. Sudut pandang penelitian

   Kecerdasan buatan merupakan suatu pembelajaran tentang bagaimana membuat komputer agar dapat melakukan sesuatu sebaik yang dikerjakan manusia.

Domain yang sering dibahas oleh para peneliti antara lain:

a. *Mundane Task*, meliputi: persepsi (*vision and speech*), bahasa alami (*understanding, generation and translation*), pemikiran yang bersifat *commonsense*, bobot kontrol.

b. *Formal Task* meliputi: permainan atau *games*, matematika (geometri, logika, kalkulus, integral, pembuktian).

c. *Expert Task*, maliputi: Analisis finansial, alisis medikal, analisis ilmu pengetahuan, rekayasa (desain, pencarian kegagalan, perencanaan manufaktur).

3. Sudut pandang bisnis

Kecerdasan buatan adalah kumpulan peralatan sangat *powerfull* dan metodologis dalam menyelesaikan masalah-masalah bisnis.

4. Sudut pandang pemrograman

Kecerdasn buatan meliputi studi tentang pemrograman simbolik, penyelesaian masalah (*problem solving*) dan pencarian (*searching*). Bagian utama yang dibutuhkan untuk melakukan aplikasi kecerdasan buatan:

a. Basic pengetahuan (*knowledge base*), yang berisi fakta-fakta, teori, pemikiran dan hubungan satu dengan lainnya.

b. Motor inferensi (*inference Engine*), yaitu kemampuan untuk menarik kesimpulan berdasarkan pengalaman.

II.4.1 Sejarah AI

Pada tahun 1950-an, seorang pionir AI dan ahli matematika Ingggris, Alan Turing melakukan sebuah percobaan (*Turing Test*) dengan meletakkan sebuah komputer melalui terminalnya pada jarak yang jauh. Pada ujung yang satu ada terminal dengan *software* AI dan pada ujung lainnya ada terminal dengan seorang operator. Operator tidak mengetahui jika di ujung terminal lainnya sudah dipasang perangkat lunak AI. Mereka berkomunikasi dengan cara memberikan respon terhadap pertanyaan yang diajukan oleh operator, sehingga operator mengira bahwa ia sedang berkomunikasi dengan operator lainnya yang berada di terminal lain.

Ia beranggapan bahwa jika mesin dapat membuat seseorang percaya bahwa dirinya mampu berkomunikasi dengan orang lain, maka dapat dikatakan bahwa mesin tersebut cerdas (seperti layaknya manusia). Dari sini kemudian berkembanglah beberapa program kecerdasan buatan.

Beberapa program kecerdasan buatan yang dibuat mulai tahun 1956-1966, antara lain:

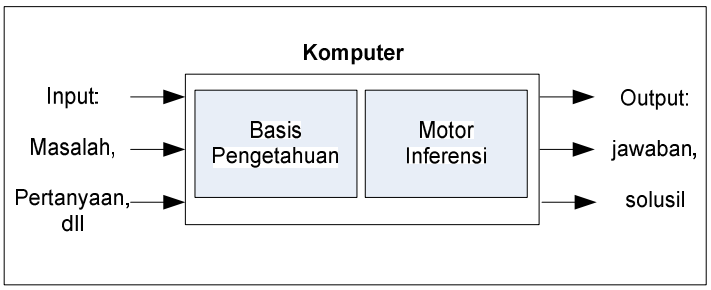
1. *Logic Theorist*, diperkenalkan pada *Dartmouth Conference*, yang mana program ini dapat membuktikan teorema-teorema matemtika.
2. Sadsam, diprogram oleh Robert K. Lindsay pada tahun 1960. Program ini dapat mengetahui kalimat sederhana yang ditulis dalam bahasa inggris dan mampu memberikan jawaban dari fakta-fakta yang didengar dalam sebuah percakapan.

3. ELIZA, diprogram oleh Joseph Weizenbaum (1967). Program ini mampu melakukan terapi terhadap pasien dengan memberikan beberapa pertanyaan.

II.4.2 Konsep AI

Berdasarkan pengertian yang telah diuraikan di atas bahwa Kecerdasan Buatan (*Artificial Inteligence*) merupakan bagian dari ilmu komputer yang mempelajari bagaimana membuat mesin (komputer) dapat melakukan pekerjaan seperti dan sebaik yang dilakukan oleh manusia bahkan lebih baik daripada yang dilakukan manusia. Sama halnya seperti manusia, agar komputer dapat bertindak seperti dan sebaik yang dilakukan manusia, maka komputer harus diberi bekal pengetahuan, dan kemampuan menalar. Semakin banyak pengetahuan yang dimilikinya tentu diharapkan manusia akan lebih mampu dalam menyelesaikan permasalahan, manusia juga diberi akal untuk dapat melakukan penalaran, dan mengambil kesimpulan berdasarkan pengetahuan dan pengalaman yang telah dimilikinya. Karena itu, pada kecerdasan buatanpun akan diberikan beberapa metod untuk membekali komputer dengan pengetahuan dan pengalaman tersebut agar komputer dapat menjadi mesin yang pintar.

Dengan demikian, sistem yang menggunakan kecerdasan buatan, akan mencoba untuk memberikan output berupa solusi dari suatu masalah berdasarkan kumpulan pengetahuan yang ada.



Gambar II.2 Penerapan Konsep Kecerdasan Buatan

Pada gambar II.2, input yang diberikan pada sistem yang menggunakan kecerdasan buatan berupa masalah. Pada sistem harus dilengkapi dengan sekumpulan pengetahuan yang terdapat pada basis pengetahuan. Sistem juga memiliki motor inferensi yang mampu mengambil kesimpulan berdasrkan fakta atau pengetahuan yang ada. Dan *output* yang diberikan berupa solusi masalah sebagai hasil dari inferensi.

Pada prinsipnya, secara umum untuk membangun sebuah sistem yang dapat menyelesaikan masalah, ada empat hal yang perlu dipertimbangkan:

1. Mendefinisikan masalah dengan tepat yang mencakup spesifikasi yang tepat mengenai keadaan awal dan solusi yang diharapkan.
2. Menganalisis masalah tersebut serta mencari beberapa teknik penyelesaian masalah yang sesuai.
3. Mempresentasikan pengetahuan yang perlu untuk menyelesaikan masalah tersebut.
4. Memilih teknik penyelesaian masalah yang terbaik.

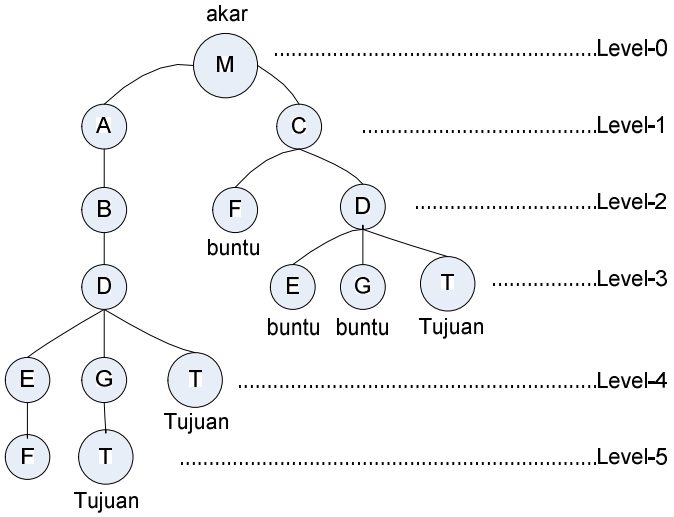
Secara umum, untuk mendeskripsikan suatu masalah yang baik, maka harus:

1. Mendefinisikan suatu ruang keadaan

2. Menetapkan satu atau lebih keadaan awal
3. Menetapkan satu atau lebih tujuan
4. Menetapkan kumpulan aturan

Dalam mempresentasikan masalah ke dalam ruang keadaan ada beberapa cara, yaitu antara lain dengan menggunakan *graph* keadaan, dan pohon pelacakan. Pada *graph* keadaan, graph terdiri dari *nodes* yang menunjukkan keadaan yaitu keadaan awal dan keadan baru yang akan dicapai dengan menggunakan operator. *Nodes* dalam ruang keadan saling dihubungkan dengan menggunakan busur yang diberi panah untuk menunjukkan arah dari suatu keadaan ke keadaan berikutnya. Biasanya bentuk graph seperti ini agak sulit jika diterapkan dalam sebuah *software*.

Sedangkan pohon pelacakan atau struktur pohon digunakan untuk menggambarkan keadaan secara hirarkis, biasanya digunakan untuk menghindari kemungkinan adanya proses pelacakan suatu *node* secara berulang. Struktur pohon juga terdiri dari beberapa *node*. *Node* yang terletak pada level 0 disebut dengan “akar” yang menunjukkan keadaan awal yang biasanya merupakan objek. Pada *node* akar terdapat beberapa percabangan yang terdiri dari beberapa *node sucessor* atau yang disebut “anak”. Sedangkan *node* yang tidak memiliki anak disebut *node* “daun”, dapat berupa tujuan yang diharapkan (*goal*).



Gambar II.3 Struktur Pohon

II.4.3 *Game Playing* dalam AI

Permainan merupakan fasilitas yang sangat menarik dalam komputer. Ide permainan dimunculkan pertama kali oleh Claude Shannon (1950) yang menulis *paper* tentang mekanisme pembuatan program permainan catur. Beberapa tahun kemudian, Alan Turing mendeskripsikan program permainan catur namun ia sendiri belum pernah membuat rancangan program. Baru pada awal tahun 1960-an Arthur Samuel mencoba untuk membuat program catur tersebut.

Dalam ilmu kecerdasan buatan ada beberapa alasan mengapa permainan dapat menjadi domain yang baik untuk di eksplore, yaitu karena:

1. Mudah untuk menentukan ukuran kesuksesan dan kegagalannya.
2. Ruang keadaannya mudah direpresentasikan.
3. Operator-operator yang digunakan tidak terlalu banyak.
4. Sebagian besar permainan dapat dimodelkan dengan mudah.

5. Sangat mungkin untuk dibandingkan dengan kemampuan manusia.

Selain itu, dalam permainan kecerdasan buatan juga terdapat beberapa ciri umum, yaitu:

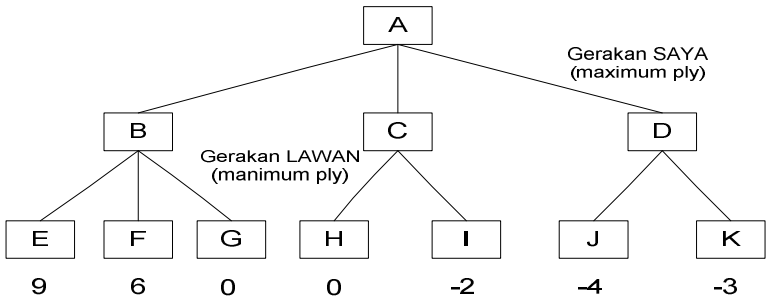
1. Terdapat 2 pemain
2. Kesempatan pemain bergantian
3. *Zero-sum*: kerugian seorang pemain adalah keuntungan pemain lain
4. *Perfect information*: pemain mengetahui semua informasi *state* dari *game*
5. Tidak mengandung probabilistik (seperti dadu)

II.5 Minimax

Dalam bidang kecerdasan buatan salah satu teknik permainan yang terkenal adalah Minimax. Teorema Minimax (von Neumann, 1928), bahwa:

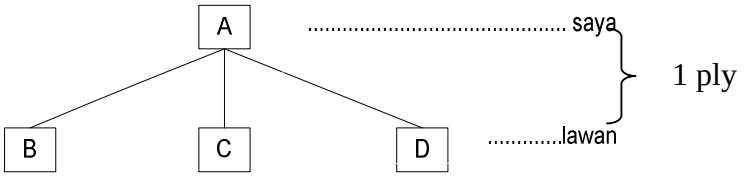
"A strategy tells what an agent will do in every possible situation and strategies may be pure (deterministic) or mixed (probabilistic)"

Metode Minimax menggunakan *depth-first search* dengan kedalaman terbatas. Fungsi evaluasi yang digunakan adalah fungsi evaluasi statis, dengan mengasumsikan bahwa lawan akan membuat langkah terbaik yang mungkin djalankan.

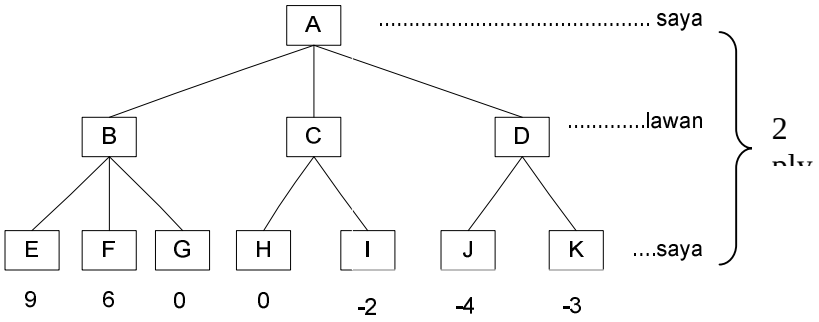


Gambar II.4 Pohon Pelacakan Minimax

Pada Minimax, dikenal istilah *ply* atau gerakan ‘saya’ dan ‘lawan’. Gambar II.5 menunjukkan permainan dengan 1 *ply search*, sedangkan Gambar II.6 menunjukkan permainan dengan 2 *ply search*.



Gambar II.5 *1-ply search*



Gambar II.6 *2-ply search*

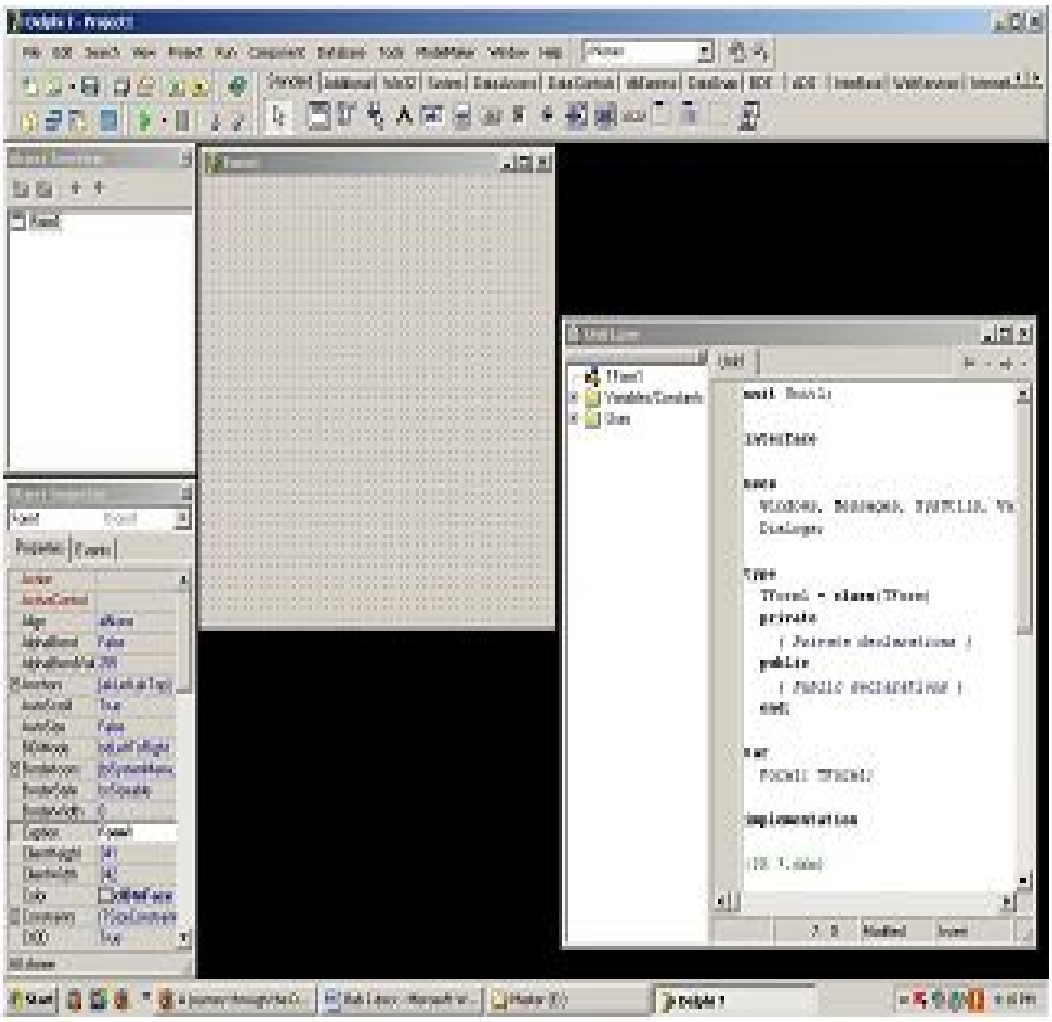
```
Minimax (StatusSaya, Kedalaman, Pemain)
IF (Kedalaman==Max)
        RETURN static (StatusSaya, Pemain)
Bangkitkan successor S[1..n]
IF (Pemain==Saya)
        RETURN max  of Minimax (S[i], kedalaman+1,
   Lawan)
ELSE
        RETURN min  of Minimax (S[i], kedalaman+1,
   Saya)
```

II.6 Borland Delphi

Borland Delphi adalah sebuah alat pegembangan aplikasi-aplikasi untuk sistem operasi Microsoft Windows. Dan sangat mudah digunkan untuk membuat suatu program yang berbasiskan GUI (*Graphical User Interface*) atau *console*. Delphi juga merupakan bahasa pemrograman pertama yang memecahkan batasan antara bahasa tingkat tinggi dan pengembangan aplikasi dengan cepat (*Rapid Aplication Development*/RAD). RAD merupakan perangkat pengembang yang mampu dengan mudah dan cepat menghasilkan program aplikasi.

II.6.1 IDE Delphi

Borland Delphi juga memiliki lingkungan pengembangan terintegrasi (*Integrated Development Environment*/IDE) dari Borland Delphi itu sendiri. pada IDE terdapat beberapa fasilitas untuk melakukan desain, pengkodean maupun *debugging*.



Gambar II.7 IDE Delphi 7

IDE Delphi memiliki beberapa windows yang mempunyai fungsi tersendiri, diantaranya:

1. *Windows* utama

Window utama terdiri dari menu utama, *toolbar* dan *component pallete*. Window utama berfungsi untuk memilih operasi-operasi tertentu seperti menyimpan objek, buka, dan lain-lain.

2. Component pallete

Component pallete berfungsi untuk meletakkan berbagai komponen untuk digunakan pada aplikasi Delphi, seperti tombol, textbox, radiobutton, checkbox, dan lain-lain.

3. Object Inspector

Berfungsi untuk menampilkan *properties* dan *event* untuk komponen yang sedang dipilih dan memperbolehkan kita untuk mengganti nilai propertinya atau memilih respon terhadap suatu event yang terjadi.

4. Object treeview

Sebuah diagram pohon yang menggambarkan hubungan logis antara komponen visual dan nonvisual yang terletak pada form, dan module, atau frame.

5. Form designer

Merupakan suatu objek yang dapat dipakai sebagai tempat untuk merancang program aplikasi.

6. Code editor

Merupakan tempat dimana kode program dituliskan.

II.6.2 Sejarah Borland Delphi

Dimulai pada awal 1993, anggota *Borland International Incoporation* di bawah pimpinan Andes Hejlberg mencoba merancang pemrograman berbasis Windows yang lebih mudah yang kemudian diberi nama Visual Foo dan sampai sekarang sudah diganti menjadi Delphi.

Pada tahun 1995, Borland Dephi dibuat. Kemudian pada bulan Maret 1996, *Borland International Incorporation* meluncurkan Delphi 2.0 yang lebih dikhususkan untuk Sistem Operasi Windows 95 dan Windows NT, setelah itu dipromosikan menjadi bahasa pemrograman terbaik yang pernah ada.

Selanjutnya *Borland International Incorporation* merilis Delphi 3.0 dan Delphi 4.0 berturut-turut pada tahun 1997 dan 1998. Tahun 1999, Borland Delphi 5.0 diluncurkan hingga sekarang mengalami perkebangan menjadi Borland Delphi 6.0 dan versi terbaru Delphi versi 7.0 dengan penambahan fitur-fitur baru dan fasilitas yang canggih.

II.6.3 Kelebihan Borland Delphi

Beberapa kelebihan yang dapat diambil apabila menggunakan Borland Delphi, antara lain:

1. Delphi mendukung Pemrograman Beroientasi Object (*Object Orented Programming*)
2. Pengembangan aplikasi secara cepat (RAD)
3. Menggunakan bahasa tingkat tinggi

4. Hasil dari proses kompilasi berupa sebuah file yang dapat dieksekusi sehingga mempermudah dalam pendistribusian program dan mengurangi banyanya file pendukung
5. Delphi menyediakan banyak sekali komponen yang dapat digunakan. Komponen juga dapat bersumber dari pihak ketiga yang biasanya disertai dengan dokumentasi, source code dan lain-lain, yang sifatnya komersil atau free.
6. Mendukung banyak database server (MySQL, SQL Server, Interbase, Oracle dan sebagainya) sehingga dapat mempermudah dalam membuat aplikasi database.

II.7 Definisi dan Sejarah *Unified Modeling Laguage* (UML)

Pada kenyataannya pendapat orang tentang *Unified Modeling Laguage* (UML) itu berbeda-beda satu sama lain. Hal ini dikarenakan oleh sejarahnya sendiri dan oleh perbedaan persepsi tentang apa yang membuat sebuah proses rancang bangun perangkat lunak efektif. Menurut Martin Fowler dalam bukunya yanng berjudul *"UML Distilled Edisi 3 Panduan Singkat Bahasa Pemodelan Objek Standar"* ditulis bahwa UML adalah keluarga notasi grafis yang didukung oleh meta-model tunggal, yang membantu pendeskripsian dan desain sistem perangkat lunak, khususnya sistem yang dibangun menggunakan pemrograman berorientasi objek (OO).

Bahasa pemodelan grafis telah ada di industri perangkat lunak sejak lama. Pemicu utama di balik semuanya adalah bahwa bahasa pemrograman berada pada

tingkat abstraksi yang tidak terlalu tinggi untuk memfasilitasi diskusi tentang desain.

Meskipun pada faktanya, bahasa pemodelan grafs ini telah ada sejak lama, masih terdapat banyak pertentangan di dalam industri perangkat lunak tentang perannya. Pertentangan-pertentangan ini berpengaruh pada bagaiman orang-orang memandang tentang UML itu sendiri.

UML merupakan standar yang relatif terbuka yang dikontrol oleh *Object Management Group* (OMG), yaitu sebuah konsorsium terbuka yang terdiri dari banyak perusahaan. OMG dibentuk untuk membuat standar-standar yang mendukung interoperabilitas, khususnya interoperabilitas sistem berorientasi objek. OMG mungkin lebih dikenal dengan standar-standar CORBA (*Common Object Request Broker Architecture*). UML lahir dari penggabungan banyak bahsa pemodelan grafis berorientasi objek yang berkembang pesat pada akhir 1980-an dan awal 1990-an.

II.7.1 Cara Menggunakan UML

Pada inti peran UML dalam pengembangan perangkat lunak, orang-orang memiliki cara yang berbeda dalam penggunaannya. Ada 3 cara orang-orang menggunakan UML, yaitu dengan: sketsa, blueprint, dan bahasa pemrograman. Dalam penggunaan UML sebagai sketsa, developer menggunakan UML untuk membantu menyampaikan aspek dari sebuah sistem. Sama halnya dengan blueprint, kita dapat menggunakan sketsa dalam sebuah forward engineering atau reverse engineering. forward engineering menggambar sebuah diagram UML

sebelum membuat kode, sedangkan reverse engineering membuat diagram dari kode yang sudah ada untuk membantu memahaminnya. sketsa biasanya sedikit informal dan dinamis

Sketsa juga berguna di dokumen, di mana fokusnya lebih pada penyampaian ide daripada keutuhan. Peranti yang digunakan dalam pembuatan sketsa adalah alat gambar ringan dan biasanya orang-orang tidak terlalu mengikuti aetiap aturan dalam UML.dan sebagian besar diagram UML yang terdapat dalam buku-buku adalah sketsa.

Kebalikannya, UML sebagai blueprint menyampaikan suatu keutuhan. Perbedaan antar blueprint dansketsa terletak pada fakta bahwa sketsa dibuat tidak lengkap, hanya menunjukkan informasi penting sedangkan blueprint cenderung komprehensif dan acapkali bertujuan meringaks pemrograman menjadi aktivitas yang simpel dan sedikit mekanis. Dalam kata lain, sketsa cenderung bersifat eksploratif sedangkan blueprint bersifat definitif.

Dan UML sebagai bahasa pemrograman, dalam lingkungan ini, developer menggambar diagram UML yang digabungkan langsung menajdi kode yang dapat dijalankan dan UML menjadi source code.

II.7.2 Diagram-Diagram pada UML

Pada UML 2 terdiri dari 1 jenis diagram resmi. Berikut diagramnya yang disajikan dalam bentuk tabel II.1.

1. *Activity Diagram*

   Menggambarkan rangkaian aliran dari aktifitas, digunakan untuk mendeskripsikan aktifitas yang dibentuk dalam suatu operasi sehingga dapat juga digunakan untuk aktifitas lainnya seperti *use case* atau interaksi.

2. *Class Diagram*

   Adalah deskripsi kelompok obyek-obyek dengan properti, perilaku dan relasi yang sama. Sehingga dengan adanya *class diagram* dapat memberikan pandangan global atas sebuah sistem. Hal tersebut tercermin dari *class-class* yang ada dan relasinya satu dengan yang lainnya. Sebuah sistem biasanya mempunyai beberapa *class diiagram. Class diagram* sangat membantu dalam visualisasi struktur kelas dari suatu sistem.

3. *Componen Diagram*

   Merupakan bagian fisik dari sebuah sistem, karena menetap di komputer tidak berada di benak para analis. Komponen merupakan implementasi *software* dari sebuah atau lebih *class. Component* dapat berupa *source code*, komponen biner, atau *executable component*. Sebuah component berisi informasi tentang *logic class* atau *class* yang diimplentasikan sehingga membuat pemetaan dari *logical view* ke *component view*. Sehingga *component diagram* merepresentasikan duania *real* yaitu *component software* yang mengandung *component*, *interface* dan *relationship*.

4. *Collaboration Diagram*

   Menggambarkan kolaborasi dinamis *seperti sequence diagram*.

5. *Deployment Diagram*

   Menggambarkan tata letak sebuah sistem secara fisik, menampakkan bagian-bagian *software* yang berjalan pada bagian-bagian *hardware*, menunjukkan hubungan komputer dengan perangkat satu sama lain dan jenis hubungannya.

6. *Use Case Diagram*

   Adalah abstraksi dari interaksi antara sistem dan aktor. *Use case* bekerja dengan cara mendeskripsikan tipe interaksi antara *user* sebuah sistem dengan dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai.

7. *Sequence Diagram*

   Digunakan untuk menggambarkan perilaku pada sebuah skenario.

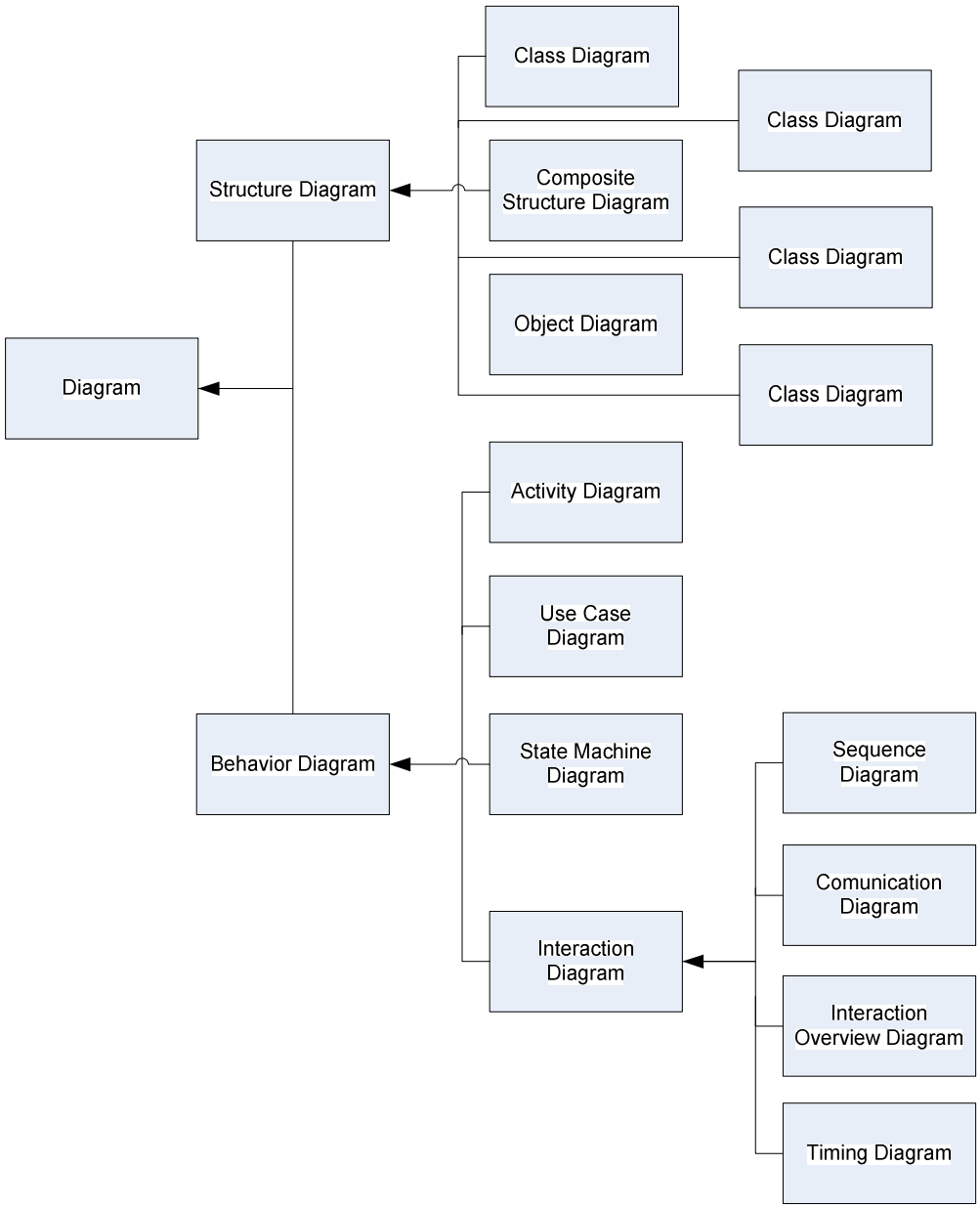
8. *State Diagram*

   Menggambarkan semua *state* (kondisi) yang dimiliki oleh suatu objek dari suatu *class* dan keadaan yang menyebabkan state berubah.

Tabel II.1 Diagram UML

Diagram	Kegunaan
Activity	*Behavior* prosedural dan paralel
Class	*Class*, fitur, an hubungan-hubungan
Communication	Interaksi antar objek; penekanan pada

	jalur
Component	Struktur dan koneksi komponen
Composite stucture	Dekomposisi *runtime* sebuah *class*
Deployment	Pemindahan artifak ke *node*
Interaction overview	Campuran *sequence* dan *activity diagram*
Object	Contoh konfigurasi dari contoh-contoh
Package	Struktur hirarki *compile-time*
Sequence	Interaksi antar objek; penekanan pada *sequence*
State machine	Bagaimana objek mengubah even selama aktif
Timing	Interaksi antar objek; penekanan pada *timing*
Use case	Bagaiman pengguna berinteraksi dengan sebuah sistem



Gambar II.8 Klasifikasi Jenis Diagram UML

II.7.3 Kelebihan UML

Kelebihan dari UML (*Unified Modeling Languang*) adalah tidak hanya terbatas untuk object oriented software development saja tetapi juga lebih dari itu, UML dapat diterapkan untuk pengembangan software yang lainnya seperti data

modeling. UML dapat digunakan untuk menggambarkan perkembangan secara lengkap dari relasi dan objek *database* relasional dari kebutuhan bisnis hingga physical data model.