BAB 2
LANDASAN TEORI

2.1 Permainan (*Game*)

Game merupakan kata yang berasal dari bahasa inggris yang berarti permainan. *Game* atau permainan ini pun tidak sembarangan dimana di dalamnya terdapat berbagai aturan yang harus dipahami oleh penggunaannya. Dalam *game* juga perlu adanya skenario agar alur permainan pun jelas dan terarah. Skenario di sini bisa meliputi setting *map, level,* alur cerita bahkan efek yang ada dalam *game* tersebut [2].

2.1.1 Permainan Video (Video Game)

Permainan *video* (*video game*) adalah permainan yang menggunakan interaksi antarmuka pengguna melalui gambar yang dihasilkan oleh piranti video. Permainan video umumnya menyediakan sistem penghargaan, misalnya *score* yang dihitung berdasarkan tingkat keberhasilan yang dicapai dalam menyelesaikan tugas-tugas yang ada di dalam permainan video tersebut. Sistem elektronik yang digunakan untuk menjalankan permainan video disebut *platform*, contohnya adalah komputer pribadi (*personal computer*) ataupun konsol permainan (*game console*).

1. **Permainan Komputer (*Computer Game*)**

Permainan komputer adalah permainan video yang di mainkan pada komputer pribadi, dan bukan pada konsol permainan. Permainan komputer telah berevolusi dari sistem grafis sederhana menjadi sangat kompleks dan mutakhir.

Permainan online (*online game*) adalah jenis permainan video atau permainan komputer dengan menggunakan jaringan internet sebagai media interaksinya.

Permainan online terdiri dari dua unsur utama, yaitu *server* dan *client*. *Server* adalah penyedia layanan *gaming* yang merupakan basis agar *client –client* yang terhubung dapat memainkan permainan dapat melakukan komunikasi dengan baik. Suatu *server* pada prinsipnya hanya melakukan administrasi permainan dan menghubungkan *client-client*. Sedangkan client adalah pengguna

permainan dan memakai kemampuan *server*. Contoh permainan online adalah *Ragnarok Online, DoTA, World of Warcraft*.

2. Konsol Permainan (*Game Console*)

Konsol permainan adalah sebuah mesin elektronik yang dirancang khusus untuk memainkan permainan video. Perangkat keluarannya biasa berupa monitor komputer atau televisi dan alat masukan utamanya berupa sebuah pengendali. Konsol permainan yang pertama kali dibuat adalah *Atari*, kemudian dilanjutkan dengan *Nintendo* yang sukses pada tahun 1985-1989. Konsol permainan modern sekarang ini antara lain adalah *Playstation* buatan perusahaan *Sony*. Untuk *Console* yang berbentuk kecil dan dapat dibawah kemana-mana biasa disebut *Portable Console*.

2.1.2 Klasifikasi Permainan (*Game*)

Sejarah *game* tidak lepas dengan namanya *platform*, dari yang awalnya hanya mesin *Arcade* (Mesin ding dong) hingga sampai sekarang ini yang sudah mencapai era *Mobile Games*. Untuk lebih jelasnya, berikut ini adalah klasifikasi *game* berdasarkan jenis *Platform* yang di gunakan [3]:

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 didesain 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).
2. *PC Games* (*Personal Computer*) yaitu *video game* yang dimainkan menggunakan *Personal Computers*.
3. *Console Games* yaitu *video game* yang dimainkan menggunakan *console* tertentu seperti playstation *XBOX, Nintendo,* dan lan-lain.
4. *Handheld Gam*e yaitu *game* yang dimainkan di *console* khusus *video game* yang dapat dibawa kemana-mana, contoh *Nintendo DS* dan *Sony PSP*.
5. *Mobile Games* yaitu *game* yang biasa dimainkan khusus untuk *handphone* atau *PDA*.

Klasifikasi permainan berdasarkan pembatasan usia dari pemainnya, menurut **ESRB** (***Entertainment Software Rating Board***) adalah sebagai berikut [4]:

1. ***Early Childhood*** **(3+)** memiliki isi yang mungkin cocok untu usia 3 keatas. Tidak mengandung materi yang tidak pantas seperti adegan kekerasan.
2. ***Everyone*** **(6+)** memiliki konten yang mungkin cocok untuk usia 6 keatas. Judul-judul dalam kategori ini mungkin berisi sedikit kartun, fantasi atau kekerasan ringan dan tidak jarang menggunakan bahasa yang ringan.
3. ***Everyone*** **(10+)** memiliki konten yang mungkin cocok untuk usia 10 keatas. Judul-judul dalam kategori ini mungkin berisi lebih banyak kartun, fantasi atau kekerasan ringan, bahasa ringan dan juga minimal tema sugestif.
4. ***Teen*** **(13+)** memiliki konten yang mungkin cocok untuk usia 13 keatas. Judul-judul dalam kategori ini mungkin mengandung kekerasan, tema sugestif, humor kasar, sedikit darah, simulasi perjudian, dan tidak jarang menggunakan bahasa yang kuat.
5. ***Mature*** **(17+)** memiliki konten yang mungkin cocok untuk orang usia 17 dan lebih tua. Judul-judul dalam kategori ini mungkin berisi intens kekerasan, darah, konten seksual dan bahasa yang kuat. Mature (17+) memiliki konten yang mungkin cocok untuk orang usia 17 dan lebih tua. Judul judul dalam kategori ini mungkin berisi intens kekerasan, darah, konten seksual dan bahasa yang kuat.
6. ***Adults Only*** **(18+)** memiliki konten yang hanya boleh dimainkan oleh orang-orang 18 tahun keatas. Judul-judul dalam kategori ini mungkin termasuk adegan yang intens, kekerasan dan juga mengandung konten seksual.
7. **RP (*Rating Pending*)** telah disampaikan kepada ESRB dan sedang menunggu penilaian akhir. (Simbol ini hanya muncul dalam iklan sebelum permainan dibebaskan).

2.1.3 Sudut Pandang Permainan (*Game View Point*)

Sebuah *game* biasanya mempunyai sudut pandang permainan tersendiri disesuaikan berdasarkan genre *game* yang diambil. Berikut beberapa macam sudut pandang permainan yang biasa digunakan [5]:

1. ***Side Scrolling***

Side Scrolling adalah jenis sudut pandang permainan yang terlihat dari samping dan memungkinkan karakter utama untuk bergerak dari kiri ke kanan serta memungkinkan background pada *game* seolah-olah bergeser mengikuti pergerakan karakter utama.

2. ***Top Down***

Top Down adalah jenis sudut pandang permainan yang memungkinkan karakter utama bermanuver ke empat arah namun cara permainannya sendiri bergeser dari bawah ke atas, dan biasanya *game* yang menggunakan sudut pandang permainan jenis ini adalah *shooter game*.

3. ***Isometric***

Isometric adalah jenis sudut pandang permainan yang memungkinkan permainan terlihat diantara sisi *Side Scrolling* dan juga *Top Down*, dan biasanya diterapkan pada *game* dengan genre RTS (*Real Time Strategy*).

2.1.4 Jenis Permainan (*Genre Game*)

Game dari tahun-tahun semakin banyak saja jenisnya, teknologi yang dipakaipun semakin canggih. Dari ratusan atau bahkan ribuan *game* yang telah bermunculan di dunia, *game-game* tersebut dikelompokkan menjadi beberapa genre. Pada dasarnya, genre-*genre game* yang beredar di pasaran dapat dikelompokkan sebagai berikut [5]:

a. ***Advergames***

Advergames adalah tipe *game* yang dikembangkan untuk tujuan periklanan.

b. ***Action Game***

Action Game adalah tipe *game* dengan fitur utama berupa banyaknya aksi di mana pemain harus memiliki keterampilan reaksi yang cepat untuk menghindari musuh atau menghindari rintangan. Pengembang *game* tipe ini perlu

memastikan *game* yang dibuat dioptimasi sehingga pemain memiliki pengalaman bermain yang baik, yang tidak terganggu oleh *delay* proses yang lama, contohnya: *Metal Gear Solid*.

c. *Adventure Game*

Adventure Game adalah tipe *game* yang umumnya membuat pemain harus berjalan mengelilingi suatu tempat yang terkondisi, seperti sebuah istana, gua yang berkelok, dan planet yang jauh. Pemain melakukan navigasi suatu area mencari pesan-pesan rahasia, memperoleh obyek yang memiliki kemampuan yang bervariasi, bertempur dengan musuh, dan lain-lain. Untuk membuat *game* ini, diperlukan perencanaan yang akurat sehingga memiliki alur cerita yang menarik bagi pemain, contohnya : *Tomb Raider*, *Prince of Persia*, *Hitman*, dan lain-lain.

d. *Fighting Game*

Fighting Game adalah tipe *game* yang intinya harus menjatuhkan lawan tandingnya, entah itu dengan pukulan, tendangan, *combo*, maupun dengan jurus special, contohnya : *Tekken*, *Mortal Kombat*, *Naruto*, dan lain-lain.

e. FPS (*First Person Shooter*)

FPS atau *First Person Shooter* adalah tipe *game* yang mengunakan sudut pandang orang pertama untuk membidik atau membunuh musuh, sehingga kita hanya melihat tangannya saja dan tidak melihat tubuh karakter yang dimainkan, contohnya : *Call of Duty*, *Half Life*, *Crysis*, dan lain-lain.

f. RPG (*Role Playing Game*)

RPG atau *Role Playing Game* adalah tipe *game* yang seringkali berupa multi karakter *game* di mana setiap pemain memiliki karakter dengan kemampuan, kekuatan, dan kelemahan yang spesifik. Para pemain saling berkompetisi, berinteraksi, dan bertempur satu sama lain. Tampilan grafis yang khas untuk setiap karakter pemain ditambah dengan *storyline* yang mendebarkan akan sangat menarik dan memberikan pengalaman yang berbeda di dalam bermain, contohnya : *Final Fantasy*, *Warcraft*, dan lain-lain.

g. *Platform Game*

Platform Game adalah tipe *game* yang mengharuskan pemain mengarahkan suatu obyek dengan melalui berbagai tahap atau tingkatan area

untuk menyerang musuh dan menghindar terhadap serangan. Tipe *game* ini sedikit serupa dengan *action game*, tetapi aksinya tidak secepat *action game*. Teknik *collision detection* sangat sering dimanfaatkan pada tipe *game* ini, contohnya : *Sonic The Hedgehog*, *Mario Bros*, *Metal Slug*.

h. *Puzzle Game*

Puzzle Game adalah tipe *game* yang umumnya membuat pemain menggunakan kemampuan berpikirnya sebagai pengganti keterampilan reaksi yang cepat karena terdapat rahasia yang perlu dipecahkan. *Game* ini lebih bersifat statis dibanding *action game*. Pembuatan *game* tipe ini seringkali ditunjang dengan AI (*Artificial Intelligence*), contohnya : *7 Wonders of The Ancient World*.

i. *Racing Game*

Racing Game adalah tipe *game* yang tujuannya adalah mencapai garis *finish* dari suatu *race*, dalam *game* ini biasanya pemain dapat memilih dan membeli kendaraan, mendandani, melakukan *upgrade* mesin, contohnya: *Need For Speed*.

j. *Simulation Game*

Simulation Game adalah tipe *game* yang memberikan pengalaman atau interaksi sedekat mungkin dengan keadaan yang aslinya, meskipun terkadang keadaan tersebut masih eksperimen atau bahkan fiktif, tapi ada penekanan khusus pada detail dan pengalaman realistik menggunakan keadaan tersebut, contohnya : *Gran Turismo*.

k. *Sandbox Game*

Sandbox Game adalah tipe *game* yang umumnya ditujukan untuk menjelajahi suatu kota atau tempat dan bebas berinteraksi dengan objek disekitarnya. Mungkin *game* ini bisa di bilang mirip RPG tapi *Sandbox* sangat berbeda dengan RPG *game*, perbedaannya adalah tidak adanya peningkatan *level* karakternya, contohnya : *Grand Theft Auto* dan *Bully*.

l. RTS (*Real Time Strategy*)

RTS atau *Real Time Strategy* adalah tipe *game* yang mengharuskan pemainnya menggunakan taktik dan strategi untuk jeli dalam melihat setiap peluang, kelemahan musuh dan bijaksana dalam menggunakan sumber daya yang ada, contohnya : *Command and Conquer*.

m. *Sport Game*

Sport Game adalah tipe *game* yang berupa kompetisi antara dua pemain atau lebih, di mana pemain dapat berupa individual atau tim. Contoh *game* tipe ini antara lain sepakbola, bola basket, tenis, dan bilyard. Tergantung seberapa cepat permainan yang terjadi, aplikasi *game* perlu dioptimalkan, contohnya : *Pro Evolution Soccer*.

2.2 Pengertian 2D dan 3D

Seiring perkembangan dunia *game* dan teknologi grafis, maka berkembang teruslah model-model yang terus diperkenalkan pada dunia *game* ini. Mulai dari *game* Dua Dimensi (2D) hingga game Tiga Dimensi (3D), bahkan ada beberapa game yang sudah memakai teknologi *Augmented Reality* atau *Virtual Reality*. Hal ini sedikit banyaknya mempengaruhi seluruh pembuatan *game* dari perusahaan-perusahaan pembuat *game*. Bahkan dari *vendor-vendor* penyedia kartu grafis guna memfasilitasi dan mendukung agar *game-game* yang keren seperti 3D dapat dan mampu berjalan pada komputer.

Terdapat perbedaan yang mendasar antara *Game* 2D dan 3D beserta dari sudut pandang yang lainnya:

1. *Game* 2D

Game dua dimensi dilihat dari keadaan ruangnya yang hanya memiliki dua sisi (X dan Y). Sedangkan untuk gambarnya sendiri dapat menggunakan *Vector* maupun *Bitmap*. Untuk membuat animasi bergerak (berjalan, melompat, berlari, dll) kita harus membuat gambar satu persatu yang disebut dengan (*frame*). Kerealisasian gerakan ditentukan dari gambar yang dibuat, jumlah gambar (*frame*) yang digunakan, serta hitungan gambar per detik (*frame per second*). *Game* dua dimensi ini memiliki 2 konsep, yaitu:

a. *Static View*, dimana semua objek berada pada satu bidang dan gerakan karakter utama hanya terbatas pada bidang itu saja.
b. *Side Scrolling View*, dimana objek-objek dan gambar latar akan terus bergerak ke kanan dan ke kiri sesuai dengan kecepatan gerakan karakter

yang dimainkan. Contoh: *Pac-man*, *Space Invader*, *Mario Bros*, dan *game-game* sederhana lainnya.

2. *Game* 3D

Setelah munculnya *game* 2D, berbagai perusahaan berlomba-lomba untuk mengembangkan teknologi terbaru, sehingga muncullah tampilan 3D datar (*3D Plaine*) hal ini sering membuat bingung karena sering disebut *game* 2D tapi mereka tidaklah sama. *Game* seperti ini bukan 2D tapi tidak juga full 3D. Biasanya *gameplay*-nya memang mirip *game* 2D dimana pemain hanya bisa bergerak secara *horizontal* dan *vertical* namun beberapa gambarnya di *render* secara 3D. Teori grafik seperti ini disebut dengan 2.5D atau pseudo-3D sedangkan pada istilah *game* lebih dikenal dengan *isometric/diametric* atau bahkan *trimetric projection*.

Berbeda dengan *game* 2D dalam *game* 3D anda akan menemui tiga sisi (X,Y,Z). *Game* bertipe ini menggunakan 2 macam tipe pemodelan:

- **a.** *3d object/model*: ini merupakan model/ object 3 dimensi yang nantinya akan anda jadikan sebagai karakter utama, bangunan, *object-object* seperti senjata, musuh, permukaan tanah, pohon, bukit, dll. *Object* 3d seperti ini bisa dibuat dengan menggunakan program seperti "3D S.Max","Maya", "Hash", dll.
- **b.** *2d graphic*: gambar 2d juga berperan dalam membuat *game* bertipe 3D fungsi gambar 2d antara lain:
    1. sebagai texture untuk *object*
    2. sebagai latar belakang (langit, pemandangan, dll)
    3. sebagai meteran untuk nyawa, gambar untuk *speedometer* (untuk *game racing*), dll.

Sedangkan untuk membuat animasi dalam 3D dapat digunakan *bones* (tulang) yang layaknya manusia dapat digerakkan. Selain itu animasi tetap juga dapat menggunakan *bitmap* (gambar 2d) untuk animasi pada *texture* air, api, dll.

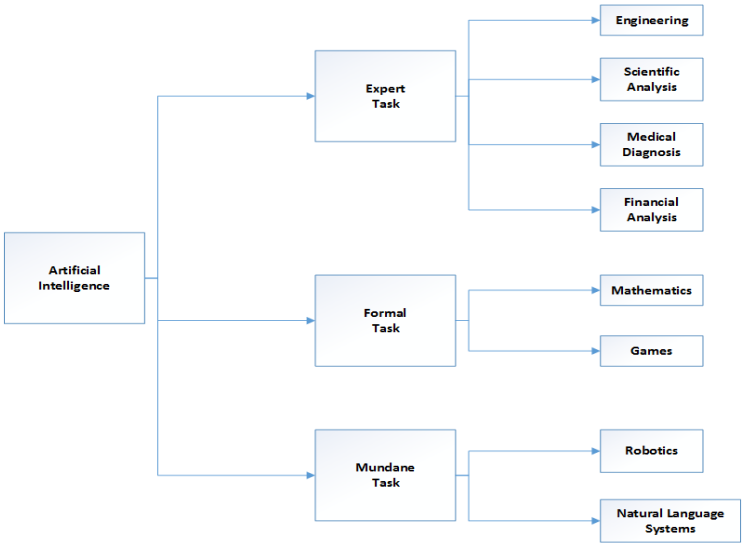
2.3 Kecerdasan Buatan

Kecerdasan buatan atau *Artificial Intelligence* (AI) merupakan cabang dari ilmu komputer yang berhubungan dengan pengautomatisan tingkah laku cerdas. Kecerdasan buatan didasarkan pada teori suara (*sound theoretical*) dan prinsip-prinsip aplikasi dari bidangnya [6]. 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 mengimplenetasikannya.

Berdasarkan sudut pandang, AI dapat dipandang sebagai berikut:

1. Sudut pandang kecerdasan, AI adalah bagaimana membuat mesin yang cerdas dan dapat melakukan hal-hal yang sebelumnya hanya dapat dilakukan manusia.
2. Sudut pandang bisnis, AI adalah sekelompok alat bantu yang berdayaguna dan metodologi yang menggunakan alat-alat bantu tersebut untuk menyelesaikan masalah-masalah bisnis.
3. Sudut pandang pemrograman, AI meliputi studi tentang pemrograman simbolik, pemecahan masalah dan proses pencarian.
4. Sudut pandang penelitian :
    a. Riset tentang AI dimulai pada awal tahun 1960-an, percobaan pertama adalah membuat program permainan catur, membuktikan teori general *problem solving*.
    b. AI adalah nama pada akar dari studi area.

Kecerdasan buatan memiliki sejumlah sub disiplin ilmu yang sering digunakan untuk pendekatan yang esensial bagi penyelasaian suatu masalah dan dengan aplikasi bidang AI yang berbeda. Berikut gambar bidang-bidang tugas dari AI yaitu:



Gambar 2.1 Bidang-bidang tugas (*task domains*) dari AI [6]

Aplikasi penggunaan AI dapat dibagi ke dalam tiga kelompok, yaitu:

1. *Expert Task*

   AI dibentuk berdasarkan pengalaman dan pengetahuan yang dimiliki oleh para ahli. Penggunaan ini dapat membantu para ahli untuk menyampaikan ilmu-ilmu yang dimiliki. Contohnya adalah:

   a. Analisis finansial.
   b. Analisis medikal.
   c. Analisis ilmu pengetahuan.
   d. Rekayasa (desain, pencarian, kegagalan, perencanaan, manufaktur).

2. *Formal Task*

   AI digunakan untuk melakukan tugas-tugas formal yang selama ini manusia biasa lakukan dengan lebih baik. Contohnya adalah:

   a. *Game*.
   b. Matematika (geometri, logika, kalkulus, integral).

3. *Mundane Task*

   Secara harfiah *mundane* adalah keduniaan. AI digunakan untuk melakukan hal-hal yang sifatnya duniawi atau melakukan kegiatan yang dapat membantu manusia. Contohnya adalah :

a. Persepsi
b. Bahasa alami.
c. Robot control.

Aplikasi kecerdasan buatan memiliki dua bagian utama yaitu :

1. Basis Pengetahuan (*Knowledge Base*): berisi fakta-fakta, teori, pemikiran dan hubungan antara satu dengan lainnya.
2. Motor Inferensi (*Inference Engine*): kemampuan menarik kesimpulan berdasarkan pengalaman.

2.3.1 Perbandingan Kecerdasan Buatan dengan Kecerdasan Alamiah

Ada beberapa keuntungan dari kecerdasan buatan dibandingkan dengan kecerdasan alamiah yaitu [6]:

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

2.3.2 Teknik Dasar Pencarian

Pada dasarnya teknik pencarian (*searching*) dapat dibagi menjadi 2 kelompok besar, yaitu pencarian buta (*blind search*) dan pencarian terbimbing (*heuristic search*). Untuk mengukur performa metode pencarian, terdapat empat kriteria yang dapat dilakukan [7] yaitu:

a. *Completeness*: apakah metode tersebut menjamin adanya solusi jika solusinya ada?
b. *Time Complexity*: berapa lama waktu yang diberikan untuk menemukan solusi tersebut.
c. *Space Complexity*: berapa banyak memori yang dibutuhkan untuk menemukan solusi tersebut.

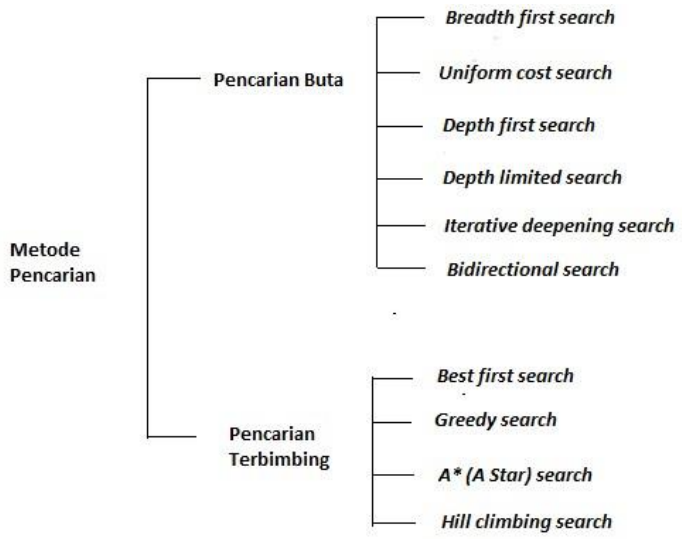
d. *Optimaly*: apakah metode tersebut menjamin menemukan solusi terbaik jika terdapat beberapa solusi yang berbeda.

2.3.3 Algoritma Pencarian

Permasalahan pencarian dapat diselesaikan dengan beberapa metode yaitu [6]:

1. Metode pencarian yang pertama adalah metode sederhana yang berusaha mencari kemungkinan penyelesaian yang disebut juga pencarian buta.
2. Metode yang lebih kompleks yang akan mencari jarak terpendek. Metode ini adalah *British Museum Procedure*, *Branch and Bound*, *Dynamic Programming*, *Best First Search*, *Greedy Search*, A* (*A Star*) dan *Hill Climbing Search*. Metode-metode ini digunakan pada saat perjalanan untuk mencari kemungkinan menjadi perhitungan.

Metode pencarian sangat penting untuk menyelesaikan permasalahan karena setiap *state* atau keadaan menggambarkan langkah-langkah untuk menyelesaikan permasalahan. Dalam sebuah permainan, metode pencarian akan menentukan apa yang harus dilakukan dimana setiap state menggambarkan kemungkinan posisi pada suatu saat. Metode pencarian adalah bagian dari kesimpulan dimana setiap *state* menggambarkan hipotesis dalam sebuah rangkaian deduktif.



Gambar 2.2 Bagan Metode Pencarian [6]

Pada gambar 2.2 di atas, metode penelusuran dibagi menjadi dua golongan, yakni pencarian buta (*blind search*) dan pencarian terbimbing (*heuristic search*).

2.3.4 Pencarian Buta (*Blind Search*)

Pencarian buta (*Blind Search*) adalah pencarian solusi tanpa adanya informasi yang dapat mengarahkan pencarian untuk mencapai *goal state* dari *current state* [6]. Informasi yang ada hanyalah definisi *goal state* itu sendiri sehingga algoritma dapat mengenali *goal state*. Apabila tidak ada informasi maka pencarian buta dalam kerjanya akan memeriksa *node-node* secara tidak terarah dan kurang efisien untuk kebanyakan kasus karena banyaknya *node* yang dikembangkan. Beberapa contoh algoritma yang termasuk *blind search* adalah *Breadth First Search, Uniform Cost Search, Depth First Search, Depth Limited Search, Interative Deepening Search* dan *Bidirectional Search* [6].

2.3.5 Pencarian Terbimbing (*Heuristic Search*)

Kata heuristik berasal dari kata kerja bahasa Yunani, *heuriskein*, yang berarti mencari atau menemukan. Dalam dunia pemrograman, sebagian orang menggunakan kata heuristik sebagai lawan kata dari algoritmik, dimana kata heuristik ini diartikan sebagai suatu proses yang mungkin dapat menyelesaikan suatu masalah tetapi tidak ada jaminan bahwa solusi yang dicari selalu dapat ditemukan. Di dalam mempelajari metode-metode pencarian ini, kata heuristik diartikan sebagai suatu fungsi yang memberikan suatu nilai berupa biaya perkiraan (estimasi) dari suatu solusi [6].

Pencarian terbimbing (*heuristic search*) mempunyai informasi tentang biaya untuk mencapai *goal state* dari *current state*. Pencarian terbimbing dapat melakukan pertimbangan untuk mengembangkan atau memeriksa node-node yang mengarah ke *goal state*. Pencarian terbimbing untuk menghitung cost ke goal stete digunakan fungsi heuristik. Fungsi heuristik berbeda dari pada algoritma dimana heuristik lebih merupakan perkiraan untuk membantu algoritma dan tidak harus valid setiap waktu. Beberapa contoh algoritma pencarian yang menggunakan

metode *heuristic search* adalah *Best First Search*, *Greedy Search*, A* (*A Star*) dan *Hill Climbing Search* [6].

2.3.6 AI yang digunakan

Artificial Intelligence yang digunakan pada dalam *game* ini menggunakan metode *Greedy Best First Search*. Metode atau algoritma greedy merupakan salah satu metode yang merancang dan mendesain suatu algoritma. Metode ini digunakan untuk mendapatkan solusi yang optimal dari suatu permasalahan. Permasalahan ini akan diselesaikan dengan langkah per langkah.

1. Ciri-ciri Algoritma Greedy
    a. Mempunyai persoalan optimasi (*optimization problems*) yaitu persoalan yang menuntut pencarian solusi optimum.
    b. Persoalan optimasi ada dua macam antara lain:
        1. Maksimasi (*maximization*)
        2. Minimasi (*minimization*)
    c. Solusi optimum adalah solusi yang bernilai minimum atau maksimum dari sekumpulan alternatif solusi.
    d. Mempunyai elemen persoalan optimasi:
        1. kendala (*constraints*).
        2. fungsi objektif atau fungsi optimasi.
    e. Algoritma greedy membentuk solusi langkah per langkah (*step by step*).
    f. Terdapat banyak pilihan yang perlu dieksplorasi pada setiap langkah solusi. Oleh karena itu, pada setiap langkah harus dibuat keputusan yang terbaik dalam menentukan pilihan. Keputusan yang telah diambil pada suatu langkah tidak dapat diubah lagi pada langkah selanjutnya.
    g. Pendekatan yang digunakan di dalam algoritma greedy adalah membuat pilihan yang bisa memberikan perolehan terbaik, yaitu dengan membuat pilihan optimum lokal (*local optimum*) pada setiap langkah dengan harapan bahwa sisanya mengarah ke solusi optimum global (global optimum).
    h. Algoritma greedy adalah algoritma yang memecahkan masalah langkah per langkah, pada setiap langkah:

1. Mengambil pilihan yang terbaik yang dapat diperoleh pada saat itu tanpa memperhatikan konsekuensi ke depan.
2. Berharap bahwa dengan memilih optimum lokal pada setiap langkah akan berakhir dengan optimum global.

i. Pada setiap langkah diperoleh optimum lokal. Jika algoritma berakhir, maka optimum lokal menjadi optimum global.

2. Prinsip Utama Algoritma Greedy

Prinsip utama algoritma greedy adalah mengambil keputusan yang paling optimal untuk setiap langkah tanpa memperhatikan konsekuensi pada langkah selanjutnya. Kemudian saat pengambilan nilai optimum lokal atau solusi pada setiap langkah, diharapkan tercapai optimum global yaitu tercapainya solusi optimum yang melibatkan keseluruhan langkah dari awal sampai akhir.

3. Elemen Algoritma Greedy

Elemen-elemen yang digunakan dalam penerapan algoritma greedy antara lain:

a. Himpunan Kandidat
   Himpunan yang berisi kemungkinan-kemungkinan yang bisa menjadi solusi dari permasalahan.
b. Himpunan Solusi
   Himpunan yang berisi kandidat yang telah terpilih sebagai solusi.
c. Fungsi Seleksi
   Fungsi untuk melakukan seleksi terhadap kandidat agar menghasilkan solusi yang diharapkan.
d. Fungsi Kelayakan
   Fungsi untuk memastikan bahwa solusi yang dipilih memenuhi syarat.
e. Fungsi Objektif
   Memilih solusi paling optimal dari himpunan solusi.

2.4 Tools yang digunakan

2.4.1 OOP (*Object Oriented Programming*)

Metodologi berorientasi objek adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnya. Metodologi berorientasi objek merupakan suatu cara bagaimana sistem perangkat lunak dibangun melalui pendekatan objek secara sistematis. Metode berorientasi objek didasarkan pada penerapan prinsip-prinsip pengelolaan kompleksitas. Metode berorientasi objek meliputi rangkaian aktivitas analisis berorientasi objek, perancangan berorientasi objek, pemrograman berorientasi objek dan pengujian berorientasi objek.

Pada saat ini, metode berorientasi objek banyak dipilih karena metodologi lama banyak menimbulkan masalah seperti adanya kesulitan pada saat mentransformasi hasil dari satu tahap pengembangan ke tahap berikutnya, misalnya pada metode pendekatan terstruktur, jenis aplikasi yang dikembangkan saat ini berbeda dengan masa lalu. Aplikasi yang dikembangkan saat ini sangat beragam (aplikasi bisnis, *real-time, utility*, dan sebagainya) dengan *platform* yang berbeda-beda, sehingga menimbulkan tuntutan kebutuhan metodologi pengembangan yang dapat mengakomodasi ke semua jenis aplikasi tersebut [8].

Keuntungan menggunakan metodologi berorientasi objek adalah sebagai berikut:

1. Meningkatkan produktivitas

   Karena kelas dan objek yang ditemukan dalam suatu masalah masih dapat dipakai ulang untuk masalah lainnya yang melibatkan objek tersebut (*reusable*).
2. Kecepatan pengembangan

   Karena sistem yang dibangun dengan baik dan benar pada saat analisis akan menyebabkan berkurangnya kesalahan pada saat pengkodean.
3. Kemudahan pemeliharaan

   Karena dengan model objek, pola-pola yang cenderung tetap dan stabil dapat dipisahkan dan pola-pola yang mungkin sering berubah-ubah.

4. Adanya konsistensi

   Karena sifat pewarisan dan penggunaan notasi yang sama pada saat analisis, perancangan maupun pengkodean.

5. Meningkatkan kualitas perangkat lunak

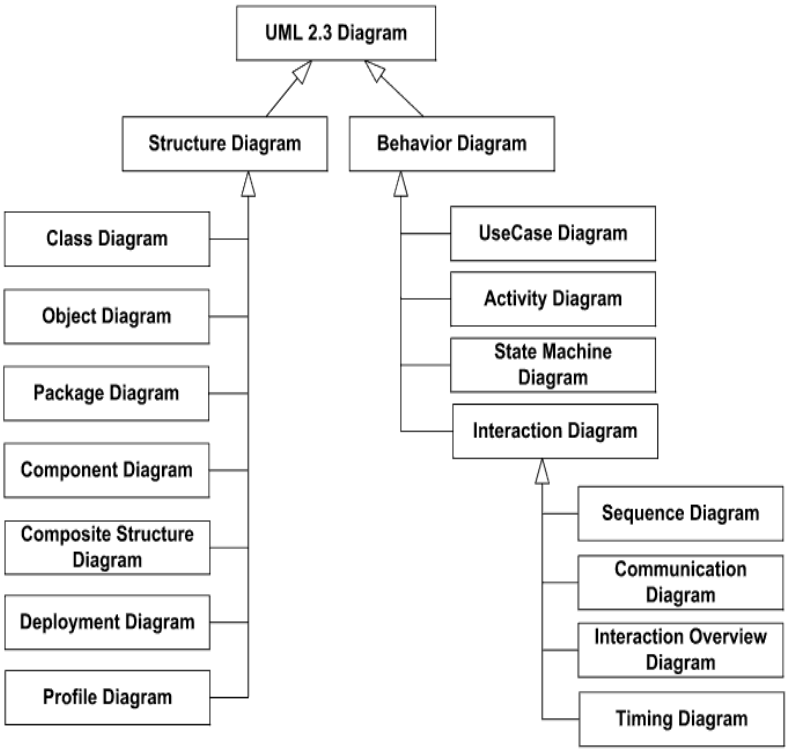
   Karena pendekatan pengembangan lebih dekat dengan dunia nyata dan adanya konsistensi pada saat pengembangannya, perangkat lunak yang dihasilkan akan mampu memenuhi kebutuhan pemakai serta mempunyai sedikit kesalahan.

2.4.2 UML (*Unified Modeling Language*)

UML (*Unified Modeling Language*) adalah salah satu standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan requirement, membuat analisis & desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek [8].

Pada perkembangan teknik pemrograman berorientasi objek, muncullah sebuah standarisasi bahasa pemodelan untuk pembangunan perangkat lunak yang dibangun dengan menggunakan teknik pemrograman berorientasi objek, yaitu *Unified Modeling Language* (UML). UML muncul karena adanya kebutuhan pemodelan visual untuk mendefinisikan, menggambarkan, membangun, dan dokumentasi dari sistem perangkat lunak. UML merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung.

Pada UML 2.3 terdiri dari 13 macam diagram yang dikelompokan dalam 3 kategori. Pembagian kategori dan macam-macam diagram tersebut dapat dilihat pada gambar 2.3 dibawah.



Gambar 2.3 Kategori Diagram UML [8]

Berikut ini penjelasan singkat dari pembagian kategori tersebut.

a. *Structure diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan.
b. *Behaviors diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem
c. *Interaction diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan interaksi sistem dengan sistem lain maupun interaksi antar subsistem pada suatu sistem.

Penjelasan tentang masing-masing diagram akan dilakukan pada sub bab-sub bab berikutnya.

2.4.3 *Class Diagram*

Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi [8].

a. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas
b. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas

Kelas-kelas yang ada pada struktur sistem harus dapat melakukan fungsi-fungsi sesuai dengan kebutuhan sistem sehingga pembuat perangkat lunak atau programmer dapat membuat kelas-kelas di dalam program perangkat lunak sesuai dengan perancangan diagram kelas. Struktur struktur kelas yang baik pada diagram kelas sebaiknya memiliki jenis-jenis berikut:

a. Kelas main
b. Kelas yang menangani tampilan sistem (*view*)
c. Kelas yang diambil dari pendefinisian *use case* (*controller*)
d. Kelas yang diambil dari pendefinisian data (*model*)

2.4.4 *Use Case Diagram*

Use case atau diagram use case merupakan pemodelan untuk kelakuan (*behavior*) sistem yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem yang akan dibuat. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem dan siapa saja yang berhak menggunakan fungsi-fungsi itu [8].

Syarat penamaan pada *use case* adalah nama didefinisikan sesimpel mungkin dan dapat dipahami. Ada dua hal utama pada *use case* yaitu pendefinisian apa yang disebut aktor dan *use case*.

a. Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem yang akan dibuat diluar sistem yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.
b. *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.

2.4.5 *Activity Diagram*

Diagram aktivitas atau *activity* diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada

perangkat lunak. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem.

Diagram aktivitas juga banyak digunakan untuk mendefiniskan hal-hal berikut:

a. Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan
b. Urutan atau pengelompokan tampilan dari sistem/user *interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan
c. Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya
d. Rancangan menu yang ditampilkan pada perangkat lunak.

2.4.6 *Sequence Diagram*

Diagram sekuen menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan message yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambar diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu. Membuat diagram sekuen juga dibutuhkan untuk melihat skenario pada *use case* [8].

Banyaknya diagram sekuen yang harus digambar adalah minimal sebanyak pendekatan pendefinisian use case yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada diagram sekuen sehingga semakin banyak *use case* yang didefinisikan maka diagram sekuen yang dibuat juga semakin banyak.

2.4.7 Adobe Flash CS6

Adobe Flash (dahulu bernama *Macromedia Flash*) adalah salah satu perangkat lunak komputer yang merupakan produk unggulan *Adobe Systems*. Adobe Flash digunakan untuk membuat gambar vektor maupun animasi gambar

tersebut. Berkas yang dihasilkan dari perangkat lunak ini mempunyai file extension .swf dan dapat diputar dipenjelajah web yang telah dipasangi Adobe Flash Player. Flash menggunakan bahasa pemrograman bernama ActionScript yang muncul pertama kalinya pada Flash 5 [9].

Sebelum tahun 2005, Flash dirilis oleh Macromedia. Flash 1.0 diluncurkan pada tahun 1996 setelah Macromedia membeli program animasi vektor bernama Future Splash. Versi terakhir yang diluncurkan dipasaran dengan menggunakan nama 'Macromedia' adalah Macromedia Flash 8. Pada tanggal 3 Desember 2005 Adobe System mengakuisisi Macromedia dan seluruh produknya, sehingga nama Macromedia Flash berubah menjadi Adobe Flash.

Adobe Flash merupakan sebuah program yang didesain khusus oleh *Adobe* dan program aplikasi standar *authoring tool* profesional yang digunakan untuk membuat animasi dan *bitmap* yang sangat menarik untuk keperluan pembangunan situs web yang interaktif dan dinamis. Flash didesain dengan kemampuan untuk membuat animasi 2 dimensi yang andal dan ringan sehingga Flash banyak digunakan untuk membangun dan memberikan efek animasi pada website, CD Interaktif dan yang lainnya. Selain itu, aplikasi ini juga dapat digunakan untuk membuat animasi logo, *movie, game,* pembuatan navigasi pada situs web, tombol animasi, banner, menu interaktif, interaktif form isian, *e-card, screen saver,* dan pembuatan aplikasi-aplikasi web lainnya. Dalam Flash, terdapat teknik-teknik membuat animasi, fasilitas *action script*, *filter*, *custom easing* dan dapat memasukan video lengkap dengan fasilitas playback FLV. Keunggulan yang dimiliki oleh Flash ini adalah ia mampu diberikan sedikit code pemrograman baik yang berjalan sendiri untuk mengatur animasi yang ada di dalamnya atau digunakan untuk berkomunikasi dengan program lain seperti HTML, PHP dan *Database* dengan pendekatan XML, dapat dikolaborasikan dengan web, karena mempunyai keunggulan antara lain kecil dalam ukuran file *output*-nya.

Movie-movie Flash memiliki ukuran file yang kecil dan dapat ditampilkan dengan ukuran layar yang dapat disesuaikan dengan keinginan. Aplikasi Flash merupakan sebuah standar aplikasi industri perancangan animasi web dengan peningkatan pengaturan dan perluasan kemampuan integrasi yang lebih baik.

Banyak fitur baru dalam Flash yang dapat meningkatkan kreativitas dalam pembuatan isi media yang kaya dengan memanfaatkan kemampuan aplikasi tersebut secara maksimal. Fitur-fitur baru ini membantu pengembang lebih memusatkan perhatian pada desain yang dibuat secara cepat, bukannya memusatkan pada cara kerja dan penggunaan aplikasi tersebut. Flash juga dapat digunakan untuk mengembangkan secara cepat aplikasi-aplikasi web yang kaya dengan pembuatan script tingkat lanjut. Didalam aplikasinya juga tersedia sebuah alat untuk men-*debug script*. Dengan menggunakan code hint mempermudah dan mempercepat pembuatan dan pengembangan isi ActionScript secara otomatis.

2.4.8 ActionScript 3.0

ActionScript merupakan bahasa *scripting* yang terdapat di gunakan Flash yang bertujuan untuk mempermudah pembangunan suatu aplikasi atau animasi. Biasaynya semakin kompleks animasi pada Flash, maka akan semakin banyak memakan *frame*. Dengan ActionScript, penggunaan *frame* tersebut dapat dikurangi, bahkan dapat membuat animasi yang kompleks hanya dengan satu *frame* saja.

ActionScript awalnya berasal dari ActionScript 1.0 yang dirilis pada tahun 2000 di Macromedia Flash 5. Bahasa *scripting* ini berisi semua kode dan perintah lainnya yang berbasis web pengembang bahasa, namun kecepatan dan kekuatannya sangat pendek. Selanjutnya ActionScript 2.0, versi ini digunakan pada Macromedia Flash MX 2004 hingga Macromedia Flash 8. Kelebihannya ialah memiliki kemampuan *compile time checking*, *strict-typing*, dan *class-based syntax*. ActionScript 3.0 baru mulai digunakan pada Adobe Flash CS3 yang merupakan restrukturisasi fundamental dari model pemrograman sebelumnya. Banyak kelebihan dari AS3 seperti penggunaannya yang luas terutama dalam pengembangan *Rich Internet Application* (RIA) dan mendukung pemrograman berorientasi objek.

Berikut merupakan pembaharuan ActionScript 3.0, yaitu :

1. Dapat memeriksa jenis-jenis informasi yang ada di *compile time* dan *runtime prototipe*.

2. Peningkatan kinerja dari sistem pewarisan berbasis kelas yang terpisah dari sistem pewarisan berbasis *prototipe*.
3. Dukungan untuk paket-paket, *namespaces*, dan ekspersi reguler.
4. Mengkompilasi ke tipe baru sepenuhnya dari *bytecode*, tidak kompatibel dengan Actionscript 1.0 dan 2.0 *bytecode*.
5. Revisi Flash Player API disusun menjadi paket.
6. Integrasi ECMAScript untuk XML (E4X) untuk keperluan pengolahan XML.

Dapat dibilang AS3 merupakan Full OOP sedangkan AS2 masih procedural & struktural biasa. AS3 dirancang untuk beberapa tujuan, yaitu sebagai berikut:

1. Keamanan, bahasa ini mendukung keamanan pengetikan sehingga pengembang dapat menulis jelas (tidak ambigu) mudah *maintainable code*.
2. Kesederhanaan, bahasa ini cukup intuitif bagi pengembang untuk bisa membaca dan menulis program tanpa terus-menerus berkonsultasi dengan referensi manual.
3. Kinerja, bahasa memungkinkan pengembang untuk menulis program yang kompleks dengan efisien dan responsif.
4. Kompabilitas, bahasa yang pendek menyediakan kompabilitas ke belakang dan ke depan jalan dan tumpang tindih yang signifikan dengan standar industri.

2.4.9 Adobe Photoshop

Adobe Photoshop adalah *software* yang dibuat oleh perusahaan Adobe System, yang di khususkan untuk pengeditan foto atau gambar dan pembuatan *effect*. Perangkat lunak ini banyak digunakan oleh Fotografer Digital dan perusahaan iklan sehingga dianggap sebagai pemimpin pasar (*market leader*) untuk perangkat lunak pengolah gambar. Meskipun pada awalnya Photoshop dirancang untuk menyunting gambar untuk cetakan berbasis-kertas, Photoshop yang ada saat ini juga dapat digunakan untuk memproduksi gambar untuk *World Wide Web*. Beberapa versi terakhir juga menyertakan aplikasi tambahan, Adobe ImageReady, untuk keperluan tersebut.

Photoshop juga memiliki hubungan erat dengan beberapa perangkat lunak penyunting media, animasi, dan *authoring* buatan-Adobe lainnya. File format asli Photoshop, PSD, dapat diekspor ke dan dari Adobe ImageReady. Adobe Illustrator, Adobe Premiere Pro, After Effects dan Adobe Encore DVD untuk membuat DVD profesional, menyediakan penyuntingan gambar *non-linear* dan layanan *special effect* seperti *background*, tekstur, dan lain-lain untuk keperluan televisi, film, dan situs web [10].