BAB 2
TINJAUAN PUSTAKA

2.1 Permainan (*Game*)

Permainan atau *game* merupakan sebuah aktivitas rekreasi dengan tujuan bersenangsenang ataupun untuk mengisi waktu luang. Permainan biasanya dilakukan secara sendiri-sendiri ataupun secara kelompok, dan merupakan salah satu media hiburan yang paling popular untuk semua kalangan usia. Sejak pertama kali ditemukan sampai saat sekarang, teknologi permainan atau *game* telah mengalami kemajuan yang terbilang sangat pesat. Hal ini ditandai dengan berkembangnya jenis, produk, alat, dan jenis interaksi permainan atau *game* dengan penggunaan yang semakin beragam bentuknya.

2.1.1 Pengertian *Game*

pengertian dan definisi *game* menurut beberapa ahli [3]:

1. Menurut John C Beck & Mitchell Wade, *game* adalah penarik perhatian yang telah terbukti. *Game* adalah lingkungan pelatihan yang baik bagi dunia nyata dalam organisasi yang menuntut pemecahan masalah secara kolaborasi
2. Menurut Agustinus Nilwan, *game* merupakan permainan komputer yang dibuat dengan teknik dan metode *animasi*.
3. Menurut Ivan C. Sibero, *game* merupakan aplikasi yang paling banyak digunakan dan dinikmati para pengguna media elektronik saat ini.
4. Menurut Fauzi A, *game* merupakan suatu bentuk hiburan yang seringkali dijadikan sebagai penyegar pikiran dari rasa penat yang disebabkan oleh aktivitas dan rutinitas kita
5. Menurut Samuel Henry, *game* merupakan bagian tak terpisahkan dari keseharian anak, sedangkan sebagian orang tua menuding *game* sebagai penyebab nilai anak turun, anak tak mampu bersosialisasi, dan tindakan kekerasan yang dilakukan anak

6. Menurut Andik Susilo, *game* adalah salah satu candu yang susah dihilangkan, bahkan ada yang mengatakan bahwa candu *game online* setara dengan narkoba
7. Menurut John Naisbitt, *game* merupakan sistem partisipatoris dinamis karena game memiliki tingkat penceritaan yang tidak dimiliki film
8. Menurut Albert Einstein, *game* adalah bentuk investigasi paling tinggi
9. Menurut Wijaya Ariyana & Deni Arifianto, *game* merupakan salah satu kebutuhan yang menjadi masalah besar bagi pengguna komputer, karena untuk dapat memainkan *game* dengan nyaman, semua komponen komputernya harus memiliki kualitas yang baik, terutama VGA *card*-nya

2.1.2 Klasifikasi Permainan (*Game Classification*)

Klasifikasi permainan berdasarkan pembatasan usia dari pemainnya, menurut ESRB (*Entertainment Software Rating Board*) adalah sebagai berikut [8]:

1. ***Early Childhood (3+)*** memiliki isi yang mungkin cocok untuk 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.

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) [6].

2.1.3 Jenis-jenis *Game*

Jenis - jenis *game* berdasarkan cara pembuatannya, cara pemasarannya, mesin yang menjalankannya. Jenis game diantaranya adalah [8]:

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 komputer yang sangat bervariasai antara satu komputer dengan komputer lainnya menyebabkan beberapa *game* dapat ditampilkan dengan baik pada satu komputer, tetapi tidak berjalan dengan baik pada komputer 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.

2.1.4 Genre Game

Game dibagi-bagi lagi ke dalam beberapa *genre* (jenis), *Game* dari tahun-tahun semakin banyak saja jenisnya, teknologi yang dipakaipun semakin canggih. Nah 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 diantaranya yaitu [4] :

1. *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*.

2. *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*.

3. *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*.

4. *RPG* (*Role Playing Game*)

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*.

5. *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*.

6. *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*.

7. *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 : *GTA & Bully*.

8. *FPS* (*First Person Shooter*)

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*.

9. *TPS* (*Third Person Shooter*)

Third Person Shooter adalah tipe *game* yang mirip dengan FPS, tapi bedanya tipe *game* ini mengunakan sudut pandang orang ke-3, sehingga kita bisa melihat seluruh tubuh karakter yang dimainkan, contohnya : *Gears of Wars*.

10. *Music Game*

Music Game adalah tipe *game* yang menuntut pemainnya untuk menekan tombol sesuai dengan tombol yang ada di layar dengan diiringi dengan musik, contohnya : *Guitar Hero & Dance Dance Revolution*.

11. *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*.

12. *RTS* (*Real Time Strategy*)

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 & Conquer*.

13. *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*.

14. *Racing Game*

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

15. *Education* (edukasi)

Game edukasi merupakan paket software yang menciptakan kemampuan pada lingkungan *game* yang diberikan sebagai alat bantu untuk memotivasi atau membantu siswa untuk melalui prosedur *game* secara teliti untuk mengembangkan

kemampuannya. *Developer* yang membuatnya, harus memperhitungkan berbagai hal agar *game* ini benar-benar dapat mendidik, menambah pengetahuan dan meningkatkan ketrampilan yang memainkannya. Target segmentasi pemain harus pula disesuaikan dengan tingkat kesulitan dan *design* visual ataupun animasinya.

2.1.5 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 [6]:

1. *Side Scrolling*

Side Scrolling adalah 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 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 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*).

2.2 Kecerdasan Buatan

Artificial Intelligence (AI) atau kecerdasan buatan merupakan cabang dari ilmu komputer yang berhubungan dengan pengautomatisasian tingkah laku cerdas 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. Beberapa perspektif, AI dapat dipandang sebagai [7]:

1. Perspektif kecerdasan, AI adalah bagaimana membuat mesin yang cerdas dan dapat melakukan hal-hal yang sebelumnya hanya dapat dilakukan manusia.

2. Perspektif bisnis, AI adalah sekelompok alat bantu *(tools)* yang berdayaguna dan metodologi yang menggunakan alat-alat bantu tersebut untuk menyelesaikan masalah-masalah bisnis.

3. Perspektif pemrograman, AI meliputi studi tentang pemrograman simbolik, pemecahan masalah, dan proses pencarian *(search)*.

4. Perspektif penelitian:

a. Riset tentang AI dimulai pada awal tahun 1960-an, percobaan pertama adalah membuat program permainan catur, membuktikan teori, dan *general problem solving*.

b. *Artificial intelligence* adalah nama pada akar dari studi area.

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.

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

a. Mundane task

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

1. Persepsi (*vision* & *speech*).
2. Bahasa alami (*understanding, generation & translation*).

3. Pemikiran yang bersifat *commonsense*.
4. *Robot control*.

b. Formal task

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

1. Permainan/*games*.
2. Matematika (geometri, logika, kalkulus, integral, pembuktian).

c. Expert task

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

1. Analisis finansial
2. Analisis medikal
3. Analisis ilmu pengetahuan
4. Rekayasa (desain, pencarian, kegagalan, perencanaan, manufaktur)

Aplikasi *Artificial Intelegent* memiliki dua bagian utama, yaitu :

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

2.2.1 Pengertian Kecerdasan Buatan / AI (*Artificial Intelligence*)

Definisi AI yang disampaikan oleh beberapa ahli. Para ahli mendefinisikan AI secara berbeda-beda tergantung pada sudut pandang mereka masing-masing. Ada yang fokus pada logika berpikir manusia saja, tetapi ada juga yang mendefinisikan AI secara lebih luas pada tingkah laku manusia. Stuart Russel dan Peter Norvig mengelompokan definisi AI dalam empat kategori berdasarkan beberapa *textbook* yang berbeda, yaitu [7]:

1. ***Thinking Humanly*** : *The Cognitive Modeling Approach*

Pendekatan ini dilakukan dengan dua cara sebagai berikut :

a. Introspeksi : mencoba menangkap pemikiran-pemikiran kita sendiri pada saat kita berfikir. Tetapi, seseorang psikologi barat mengatakan "*how do you know that you understand ?*" yang berarti bagaimana anda tahu bahwa anda mengerti ?, karena pada saat anda menyadari pemikiran anda ternyata pemikiran tersebut sudah lewat dan digantikan kesadaran anda. Dengan demikian definisi ini terkesan mengada-ada dan tidak mungkin dilakukan.
b. Eksperimen-eksperimen psikologi.

2. ***Acting Humanly*** : *The Turing Test Approach*
Tahun 1950, Alan Turing merancang suatu ujian bagi computer berintelijensia apakah komputer tersebut mampu mengelabuhi seorang
manusia yang menginterogasinya melalui *teletype* (komunikasi berbasis teks jarak jauh). Jika *interrogator* tidak dapat membedakan yang tidak dapat membedakan yang diinterogasi adalah manusia atau komputer, maka komputer berintelijensia tersebut lolos dari *turing test.* Komputer tersebut harus memiliki kemampuan berikut :
a. *Natural Language Processing.*
b. *Knowledge Representation.*
c. *Automated Reasoning.*
d. *Machine Learning.*
e. *Computer Vision.*
f. *Robotics.*
Turing test sengaja menghindari interaksi fisik antara *interrogator* dan komputer karena simulasi fisik manusia tidak memerlukan intelinjensia.
3. ***Thinking Rationally*** : *The Laws of Thought Approach*
Terdapat dua masalah dalam pendekatan ini, yaitu :
a. Tidak mudah untuk membuat pengetahuan informal dan menyatakan pengetahuan tersebut ke dalam *formal term* yang diperlukan oleh notasi logika, khususnya ketika pengetahuan tersebut memiliki kepastian kurang dari 100%. b. Terdapat perbedaan besar antara dapat memecahkan masalah "dalam perinsip" dan memecahkannya "dalam dunia nyata".

4. ***Acting Rationally*** : *The Rational Agent Approach*

Membuat inferensi yang logis merupakan bagian dari suatu *rational agent*. Hal ini disebabkan satu-satunya cara untuk melakukan aksi secara rasional adalah dengan menalar secara logis. Dengan menalar secara logis, maka bisa didapatkan kesimpulan bahwa asik yang diberikan akan mencapai tujuan atau tidak. Jika mencapai tujuan, maka *agent* dapat melakukan aksi

berdasarkan kesimpulan tersebut. *Thinking humanly* dan *acting humanly* adalah dua definisi dalam arti yang sangat luas. Sampai saat ini, pemikiran manusia yang diluar rasio yakni refleks dan intuitif (berhubungan dengan perasaan), belum dapat ditirukan sepenuhnya oleh komputer. Dengan demikian, kedua definisi ini dirasa kurang tepat untuk saat ini. Jika kita menggunakan definisi ini maka banyak produk komputasi cerdas saat ini yang tidak layak disebut sebagai produk AI. Definisi *thinking rationally* terasa lebih sempit daripada *acting rationally*. Oleh karena itu, definisi AI yang paling tepat untuk saat ini adalah *acting rationally* dengan pendekatan *rational agent*. Hal ini berdasarkan pemikiran bahwa komputer bisa melakukan penalaran secara logis dan juga bisa melakukan aksi secara rasional berdasarkan hasil penalaran tersebut.

2.3 AI yang digunakan

Algoritma yang akan digunakan pada aplikasi adalah algoritma *A-Star* dan berikut beberapa penjelasan.

2.3.1 Algoritma A* (*A Star*)

Algoritma ini merupakan algoritma *Best First Search* yang menggabungkan *Uniform Cost Search* dan *Greedy Best First Search*. Biaya yang diperhitungkan didapat dari biaya sebenarnya ditambah dengan biaya perkiraan. Dalam notasi matematika dituliskan sebagai : f(n) = g(n) + h(n). Dengan perhitungan biayaseperti ini, algoritma A* (A Star) adalah *complete* dan optimal.

Pada pencarian *rute* kasus sederhana, dimana tidak terdapat halangan pada peta, A* bekerja secepat dan seefisien BFS. Pada kasus peta dengan halangan, A* dapat menemukan solusi rute tanpa „terjebak" oleh halangan yang ada.

Pencarian menggunakan algoritma A* mempunyai prinsip yang sama dengan algoritma BFS, hanya saja dengan dua faktor tambahan.

1. Setiap sisi mempunyai "*cost*" yang berbeda-beda, seberapa besar *cost* untuk pergi dari satu simpul ke simpul yang lain.

2. *Cost* dari setiap simpul ke simpul tujuan bisa diperkirakan. Ini membantu pencarian, sehingga lebih kecil kemungkinan kita mencari ke arah yang salah. *Cost* untuk setiap simpul tidak harus berupa jarak. *Cost* bisa saja berupa waktu bila kita ingin mencari jalan dengan waktu tercepat untuk dilalui. Sebagai contoh, bila kita berkendaraan melewati jalan biasa bisa saja merupakan jarak terdekat, tetapi melewati jalan tol biasanya memakan waktu lebih sedikit.

Algoritma A* bekerja dengan prinsip yang hampir sama dengan BFS, kecuali dengan dua perbedaan, yaitu :

1. Simpul-simpul di *list* "terbuka" diurutkan oleh *cost* keseluruhan dari simpul awal ke simpul tujuan, dari *cost* terkecil sampai *cost* terbesar. Dengan kata lain, menggunakan *priority queue* (antrian prioritas). *Cost* keseluruhan dihitung dari *cost* dari simpul awal ke simpul sekarang (*current node*) ditambah *cost* perkiraan menuju simpul tujuan.

2. Simpul di *list* "tertutup" bisa dimasukkan ke list "terbuka" bila jalan terpendek (*cost* lebih kecil) menuju simpul tersebut ditemukan. Karena *list* "terbuka" diurutkan berdasarkan perkiraan *cost* keseluruhan, algoritma mengecek simpul-simpul yang mempunyai perkiraan *cost* yang paling kecil terlebih dahulu, jadi algoritmanya mencari simpul-simpul yang kemungkinan mengarah ke simpul tujuan. Karena itu, lebih baik perkiraan *cost*-nya, lebih cepat pencariannya. *Cost* dan perkiraannya ditentukan oleh kita sendiri. Bila *cost*-nya adalah jarak, akan menjadi mudah.

Cost antara simpul adalah jaraknya, dan perkiraan *cost* dari suatu simpul ke simpul tujuan adalah penjumlahan jarak dari simpul tersebut ke simpul tujuan. Atau agar lebih mudahnya bisa ditunjukkan seperti berikut ini.

$$f(n) = g(n) + h(n)$$

dengan :

$f(n)$ = fungsi evaluasi

$g(n)$ = biaya *(cost)* yang sudah dikeluarkan dari keadaan sampai keadaan n

$h(n)$ = estimasi biaya untuk sampai pada suatu tujuan mulai dari n

Perhatikan bahwa algoritma ini hanya bekerja bila *cost* perkiraan tidak lebih besar dari *cost* yang sebenarnya. Bila *cost* perkiraan lebih besar, bisa jadi jalan yang ditemukan bukanlah yang terpendek.

Node dengan nilai terendah merupakan solusi terbaik untuk diperiksa pertama kali pada $g(n) + h(n)$. Dengan fungsi *heuristic* yang memenuhi kondisi tersebut, maka pencarian dengan algoritma A* dapat optimal[5].

2.3.2 Kompleksitas Algoritma *A-Star*

Kompleksitas waktu dari algoritma A* tergantung dari *heuristic*nya. Dalam kasus terburuk (*worst case*), jumlah simpul yang diekspansi bisa eskponensial dalam solusinya (jalan tependek). Akan tetapi, kompleksitasnya bisa berupa polinomial bila fungsi heuristik *h* bertemu kondisi berikut:

$$| h(x) - h^*(x) | = O(\log h^*(x))$$

Dimana h^* adalah heuristik optimal, atau *cost* pasti untuk menuju tujuan dari *x*. Dengan kata lain, kesalahan (*error*) dari *h* tidak boleh tumbuh lebih cepat dari algoritma "*perfect heuristic*" h^* yang mengembalikan jarak sebenarnya dari *x* menuju tujuan[7].

2.3.3 Efisien Waktu Algoritma *A-Star*

Fungsi *heuristic* H(n), algoritma A* dapat memfokuskan pencarian pada *node-node* yang berada pada arah yang mendekati *node* tujuan. Kemudian pencarian diterminasikan pada waktu *node* tujuan diperiksa. Hal ini dapat meminimalisasikan jumlah *node* yang harus diperiksa dan arena waktu yang diperlukan untuk mendapatkan jalur berbanding lurus dengan jumlah *node* yang diperiksa, maka waktu pencarian dapat diminimalisasikan. Walaupun jumlah *node* yang diperiksa dapat diminimalisasikan, algoritma A* mempunyai kasus terburuk.

Pada kasus ini, sebagian besar ataupun keseluruhan *node* pada jalan diperiksa, sehingga algoritma A* bekerja seperti algoritma dijkstra atau BFS *(Best-First-Search).* Ada dua hal yang dapat menyebabkan keadaan terburuk ini, yaitu keadaan sepadan dan jika jalur yang dicari tidak ditemukan[7].

2.3.4 Penerapan Algoritma A* pada *Game*

Setiap permainan memiliki aturan main. Hal ini mempermudah upaya menghasilkan ruang pencarian dan memberikan kebebasan pada para peneliti dari bermacam-macam ambisi dan kompleksitas sifat serta kurangnya struktur permasalahan. Papan konfigurasi yang digunakan untuk memainkan permainan ini mudah direpresentasikan pada komputer dan tidak memerlukan bentuk yang kompleks. Permainan dapat menghasilkan sejumlah besar pencarian ruang. Hal ini cukup besar dan kompleks sehingga membutuhkan suatu teknik yang tangguh untuk menentukan alternatif pengeksplorasian ruang permasalahan. Teknik ini dikenal dengan nama *heuristic* dan merupakan area utama dari penelitian tentang AI. Banyak hal yang biasanya dikenal sebagai kecerdasan tampaknya berada dalam *heuristic* yang digunakan oleh manusia untuk menyelesaikan permasalahannya [7].

2.4 *Object Oriented Programming* (OOP)

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.

OOP 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.

Pemrograman orientasi-objek menekankan konsep sebagai berikut :

a. Kelas (*Class*) — kumpulan atas definisi data dan fungsi-fungsi dalam suatu unit untuk suatu tujuan tertentu. 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.
b. 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.
c. 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.

d. 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.
e. 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
f. 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).

2.5 UML (*Unified Modeling Language*)

UML adalah bahasa spesifikasi standar untuk mendokumentasikan, menspesifikasikan, dan membangun sebuah sistem. 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. UML menyediakan 10 macam diagram yang dapat digunakan untuk memodelkan aplikasi berorientasi objek yang dibuat, yaitu [10]:

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 Case* 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

2.6 *Adobe Flash CS* 5

Adobe flash merupakan aplikasi multiguna yang dapat dimanfaatkan untuk berbagai macam kebutuhan. Dengan berbagai fitur yang ada didalamnya, anda dapat menggambar, membuat animasi, hingga membuat permainan yang luar biasa.

Program ini memiliki fitur yang memungkinkan anda membuar animasi yang lebih baik dan menarik. Dalam *flash* terdapat teknik-teknik membuat *animasi*, fasilitas *action script*, filter, *custom easing*, dan dapat memasukan video lengkap dengan fasilitas *playback* flv. Dengan fitur tersebut maka dapat menjadi mudah membuat kerangka game dan menjalankannya dengan kode *script*.

Fitur terbaru pada *Adobe Flash CS* 5 sangat membantu Anda dalam membuat beragam animasi, diantaranya kemudahan penggunaan *Adobe Bridge* dan *Version Cue, Bounding Box, Adobe Device* Central, deteksi *Active Content*. Peningkatan dukungan video *Quicktime*, serta peningkatan dalam *ActionScript*. Dengan adanya fasilitas yang lengkap dan bervariasi, Anda dapat lebih nyaman bekerja dengan *Adobe Flash* CS5 ini dan mendapat hasil yang lebih optimal [1].

2.7 Action Script

Action script merupakan suatu *programming language* (bahasa pemograman) yang dapat ditambahkan pada *flash document* (baik itu pada frame, movie clip, atau button) untuk dapat membuat suatu animasi yang lebih interaktif. *Action script* seperti halnya pada pemograman C, C++ dan Java memiliki sifat yang sangat sensitive (*case-sensitive*) artinya penulisan huruf sangat berpengaruh. Ada beberapa aturan yang harus di perhatikan ketika Anda hendak menggunakan *action script*:

ActionScript merupakan bahasa *scripting* yang terdapat di dalam program Flash. Tujuan penggunaan *ActionScript* ialah untuk mempermudah pembangunan suatu aplikasi atau *animasi*. Biasanya 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* juga merupakan sebuah kumpulan dari *action*, *function*, *event*, dan *event handler* yang memungkinkan untuk dikembangkan oleh para developer untuk membuat *Flash movie* atau *animasi* yang lebih kompleks dan lebih interaktif. Selain itu *ActionScript* juga dapat mengubah kebiasaan linier pada *Flash*. Sebuah *ActionScript* dapat menghentikan sebuah movie atau animasi di frame tertentu lalu berulang ke *frame* sebelumnya

atau *frame* mana saja tergantung masukan yang diberikan oleh *user*. Bahasa *ActionScript* pada *Flash* hingga saat ini telah mengalami perkembangan dari versi 1, versi 2, dan versi 3. menjelaskan bahwa bahasa *ActionScript* awalnya berasal dari *ActionScript* 1.0 yang dirilis pertama kali pada tahun 2000 di Macromedia Flash 5 (saat Macromedia belum diakuisisi oleh Adobe) yang merupakan pengembangan dari *Action* di Macromedia Flash 4 dan masih digunakan hingga Flash MX atau Flash 6. Bahasa *scripting* ini berisi semua kode dan perintah lainnya yang berbasis web pengembang bahasa, seperti Macromedia Director Lingo dan Sun Java. Namun kecepatan dan kekuatannya sangat pendek.

Pada Macromedia Flash MX 2004 atau yang dikenal juga sebagai Flash 7 dirilis ActionScript 2.0. Versi ini tetap digunakan hingga Macromedia Flash 8. Kelebihan ActionScript 2.0 dibandingkan dengan ActionScript 1.0 ialah memiliki kemampuan compile time *checking*, *strict-typing* pada variabel, dan *class-based syntax*. *ActionScript* 2.0 juga didasarkan pada ECMA *Script* yang merupakan standar untuk bahasa pemrograman yang dikembangkan oleh Asosiasi Produsen Komputer Eropa. ECMA Script juga merupakan dasar yang digunakan oleh JavaScript. *ActionScript* 3.0 baru mulai digunakan pada Adobe Flash CS3 atau Flash 9 hingga yang paling terbaru adalah Adobe Flash CS5. *ActionScript* 3.0 ini merupakan restrukturisasi fundamental dari model pemrograman sebelumnya. Penggunaannya yang luas terutama dalam pengembangan Rich Internet Application (RIA) dengan hadirnya Flex yang menawarkan hal serupa dengan AJAX, JavaFX, dan Microsoft Silverlight. Flex memungkinkan pengembang untuk membangun suatu aplikasi yang membutuhkan Flash Player. Namun Flash juga menawarkan interface yang lebih visual untuk mengembangkan aplikasi sehingga lebih cocok untuk membangun aplikasi game.

2.8 *Corel Draw* X5

Corel Draw merupakan salah satu aplikasi pengolah gambar berbasis *vector* yang banyak dipakai oleh pengguna PC. Karena berbagai kemudahan dan keunggulan yang dimiliki oleh *corel draw*, maka *corel draw* sering dimanfaatkan

untuk desktop *publishing*, percetakan, dan bidang lain yang memerlukan pemrosesan visual.

Keunggulan mengolah gambar berbasis vector adalah ukuran hasil akhir yang dapat ditekan seminimal mungkin namun dengan kualitas yang tidak kalah dengan gambar berbasis raster atau bitmap.

Sebenarnya ada banyak sekali software desain grafis dan pengolah gambar berbasis vector yang bisa kita gunakan,misalnya saja diantaranya adalah *corel draw*, *photoshop*, *freehand*, adobe ilustrator dll. Namun *corel draw* merupakan program paling populer yang sering dimanfaatkan. Selain karena tampilannya yang user friendly dan mudah dipelajari, *corel draw* juga mempunyai berbagai keunggulan-keunggulan lain yang sangat bermanfaat.