BAB 2
TINJAUAN PUSTAKA

2.1 Teori Umum

Teori pokok yang menjadi teori landasan bagi teori lainnya yang terdapat di dalam skripsi ini adalah sebagai berikut :

2.1.1 Game

Game diambil dari bahasa Inggris yang diterjemahkan yanSg artinya permainan. Di dalam era perkembangan era perkembangan teknologi yang pesat ini, seperti halnya permainan kartu, catur dan lain-lainnya dapat ditemui melalui dunia *virtual* atau yang biasa kita mainkan di dalam komputer.

Pengertian game komputer menurut ahli [4] :

Menurut Chris Crawford, seorang komputer game designer mengemukakan bahwa game, pada intinya adalah sebuah interaktif, aktifitas yang berpusat pada sebuah pencapaian, ada pelaku aktif, ada lawan anda.

Andrew Rollings dan Ernest Adams dalam bukunya yang berjudul *Andrew Rollings and Ernest Adams on Game Design* membagi *genre* pada *game* sebagai berikut : [5]

A. *Action*

Action game mencakup berbagai macam gaya permainan, baik 3D ataupun 2D. tapi semua game dalam genre ini memiliki satu kesamaan, yaitu mengutamakan gerak/sentakan *(twitch)* dala permainannya. Keterampilan kunci yang diuji adalah waktu reaksi dan koordinasi tangan-mata di bawah tekanan. *Action game* cenderung lebih sederhana dibandingkan jenis *game* lainnya karena membutuhkan respon yang cepat dari *player*. Sebagai aturan umum, hubungan ini dapat dinyatakan sebagai suatu proporsionalitas terbalik antara kecepatan permainan dan kompleksitas. *Fighting game* dan *FPS (First Person Shooter)* termasuk dalam *genre* ini.

B. *Strategy*

Asal usul dari *genre* ini adalah *board games.* Jika ada format *game* yang terdekat dengan bentuk *pre-computer* asli merupakan *strategy game. Strategi game* dapat memiliki banyak aturan.

C. *Role-playing*

Role Playing Games memiliki ragam jenis yang menarik, meliputi *arcade-style game* yang simple, seperti seri *Diablo,* sampai tipe petualangan yang lamban seperti *Anachronox.* Ada dua hal yang hampir dimiliki oleh semua *Role Playing Games,* yaitu :

a. *Player* yang dapat dikonfigurasi dan ditingkatkan dengan pengalaman.
b. Cerita yang kuat.

D. *Sports*

Berbeda dengan jenis game lainnya yang mengambil tempat di dunia yang sedikit diketahui oleh pemainnya, *sports games* meniru dunia yang diketahui dengan baik oleh pemainnya, pertandingan olahraga seperti di dunia nyata. *Sports games* memiliki sejumlah kecil jenis game yang memungkinkan perbandingan langsung dengan dunia nyata.

E. *Vehicle Simulation*

Jenis game ini mencoba untuk menciptakan rasa mengemudi atau menerbangkan, baik kendaran nyata ataupun imajiner. Dalam simulasi kendaraan nyata, salah satu tujuan utamannya adalah, membuat pengalaman yang seakan-akan nyata bagi pemain. Pemain dapat diasumsikan mengetahui banyak hal tentang mesin yang disimulasikan, dan ingin merasakan pengalaman yang setidak-tidaknya mirip secara visual dengan mengedalikan mesin yang sebenarnya.

F. *Construction and Management Simulation*

Jenis *game* ini adalah *game* mengenai proses. Tujuan pemain bukanlah untuk mengalahkan musuh, tetapi untuk membuat sesuatu dalam koteks proses yang berkelanjutan. Semakin baik pemain memiliki pemahaman dan kemampuan mengontrol proses, semakin baik hasil pembangunan yang diperoleh.

G. *Adventure*

Sebuah *game* petualangan bukanlah suatu kompetisi atau simulasi. Sebuah *game* petualangan juga tidak menawarkan proses untuk dikelola atau lawan harus dikalahkan melalui strategi dan taktik. Sebaliknya, sebuah *game* petualangan adalah sebuah cerita interaktif tentang karakter yang dikendalikan oleh pemain.

H. *Artificial Life*

Jenis ini adalah cabang penelitian ilmu computer seperti *artificial intelligence. Artificial life (A-Life)* melibatkan proses pemodelan biologis dan seringkali untuk mensimulasikan siklus kehidupan makhluk hidup. Peniliti *A-life* berharap untuk menemukan cara-cara baru menggunakan komputer dengan menggunakan mekanisme biologis, seperti mutasi dan seleksi alam, bukan secara algoritmik. Secara khusus, *A-life* adalah studi tentang sifat yang baru, kualitas tak terduga atau perilaku yang timbul dari interaksi system yang kompleks.

I. *Puzzle*

Puzzle game adalah permainan mengenai pemecahan teka-teki, terkadang tanpa memasukan teka-teki tersebut ke dalam sebuah alur cerita atau tujuan yang lebih besar. *Puzzle game* harus bersifat menantang (tapi tdak terlalu susah), menarik secara visual dan menyenangkan untuk dimainkan.

2.1.2 GAZA

September 1993 Deklarasi Prinsip pada Pemerintahan Sendiri Pengaturan disediakan untuk masa transisi dari pemerintahan sendiri Palestina di Tepi Barat dan Jalur Gaza. Dalam serangkaian perjanjian yang ditandatangani antara Mei 1994 dan September 1999, Israel dipindahkan ke keamanan Otoritas Palestina dan tanggung jawab sipil bagi banyak penduduk Palestina di wilayah Tepi Barat dan Jalur Gaza. Negosiasi untuk menentukan status permanen dari Tepi Barat dan Jalur Gaza terhenti menyusul pecahnya intifada dari bulan September 2000. Pada bulan April 2003, Quartet (AS, Uni Eropa, PBB, dan Rusia) diundang untuk penyelesaian akhir konflik, tahun 2005 berdasarkan langkah timbal balik oleh kedua pihak yang mengarah ke dua negara, Israel dan Palestina yang demokratis. Setelah kematian pemimpin Palestina

Yasir ARAFAT pada akhir tahun 2004, Mahmud Abbas terpilih sebagai presiden pada Januari 2005. Sebulan kemudian, Israel menyetujui Komitmen dalam upaya untuk memindahkan proses perdamaian ke depan. Pada bulan September 2005, Israel secara sepihak menarik semua pemukim dan tentara dan dibongkar fasilitas militer di Jalur Gaza dan menarik pemukim dan tentara tugaskan dari empat kecil pemukiman Tepi Barat utara. Meskipun demikian, Israel masih mengontrol maritim, udara, dan akses lainnya ke Jalur Gaza, Israel juga memberlakukan zona terbatas di sepanjang perbatasan di Gaza [6].

2.1.3 Game Side Scrolling

Side-scrolling game atau sisi-scroller adalah permainan video di mana tindakan gameplay dipandang dari sudut kamera side-view, dan karakter layar umumnya bergerak dari sisi kiri layar ke kanan. Permainan ini menggunakan teknologi komputer layar bergulir. Bergerak dari grafis tunggal layar atau flip-layar untuk grafis bergulir.

Techopedia menjelaskan format *Side Scrolling* paling sering diemplementasikan dalam genre permainan platform, seperti permainan action dengan karakter berjalan, memanjat dan melompat melalui serangkaian tingkat berurutan maju. Salah satu contoh permainan side scrolling paling popular adalah:, *Contra, Super Mario.* Walaupun grafis dan pengolahan *game* telah maju, dan video game juga telah berkembang melampaui perspektif *sides scrolling*. Namun, beberapa permainan masih dirancang dengan unsur-unsur *side scrolling* atau sebagai *side scroller* game [7].

2.1.4 Multimedia

Dalam *Multimedia: Making It Works, Tay Vaughan* mengatakan bahwa multimedia adalah kombinasi dari teks, grafik, animasi, dan video yang dikemas oleh komputer.

Elemen – elemen yang terdapat dalam multimedia adalah [8]:

A. Teks

Merupakan dasar penyampaian informasi, juga merupakan media paling sederhana yang direpresentasikan dengan *typeface* (jenis huruf) yang beragam agar harmonis dengan elemen media lainnya.

B. Citra Diam (gambar)

Merupakan representasi special dari objek yang disusun sebagai matriks nilai numeric yang mereprensentasikan setiap titik/*pixel.*

C. Suara (Audio)

Merupakan fenomena fisik yang dihasilkan oleh pengantar materi.

Audio terdiri dari 3 jenis, yaitu :

1. Ucapan *(Speech)* : Suara orang berbicara
2. Musik *(music)* : hasil pendengaran alat music
3. Efek Suara *(Sound effect)* : suara lainnya, seperti : petir, gelas pecah, tembakan

D. Animasi

Merupakan penayangan *frame-frame* gambar secara cepat untuk menghasilkan kesan gerakan..

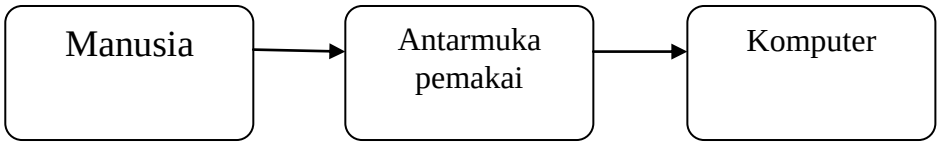
E. Video

Video sama seperti animasi tetapi disimpan dalam format khusus yang dapat menyimpan adegan dunia nyata atau rekaan dengan komputer.

2.1.5 Interaksi Manusia dan Komputer

Menurut Ben Shneiderman, Interaksi Manusia dan Komputer (IMK) atau *Human-Computer Interaction* (HCI) adalah disiplin ilmu yang berhubungan dengan perancangan, evaluasi dan implementasi system komputer interaktif untuk digunakan oleh manusia, serta studi fenomena – fenomena besar yang berhubungan dengannya

Fokus pada IMK adalah perancangan dan evaluasi antarmuka pemakain *(user interface).* Antarmuka pemakai adalah bagian sistem komputer yang memungkinkan manusia berinteraksi dengan komputer.



Gambar 2.1 Fokus pada Interaksi Manusia dan Komputer

Dalam IMK, terdapat 8 aturan emas *(golden rules)* yang digunakan dalam perancangan antarmuka pemakai. Penggunaan delapan aturan emas ini berasal dari prinsip-prinsip yang didapatkan dari pengalaman-pengalaman dan direvisi selama lebih dari 2 dekade. Delapan aturan emas tersebut adalah sebagai berikut [9]:

A. Memungkinkan pengguna *shortcut*

Menambahkan fitur untuk pemakai yang sudah ahli bias memperkaya desain antarmuka dan meningkatkan kualitas system.

B. Memberikan umpan balik yang informatif

Setiap tindakan yang dilakukan pemakai, diharapkan adanya respon dari sistem. Untuk tindakan yang sering terjadi dan tidak membutuhkan banyak aksi, respon yang ada bisa dibuat secara sederhana, sedangkan tindakan yang jarang dilakukan dan membutuhkan banyak aksi harus lebih ditonjolkan.

C. Merancang dialog yang memberikan penutupan (keadaan akhir)

Urutan aksi harus disusun ke dalam kelompok awal, tengah dan akhir. Suatu umpan balik yang informative pada akhir pekerjaan sebaiknya dibuat untuk

mengindikasikan bahwa pekerjaan telah selesai dan siap untuk melanjutkan ke aksi berikutnya.

D. Memberikan pencegahan kesalahan dan penanganan kesalahan yang sederhana

Sebisa mungkin dibuat system yang tidak akan menghasilkan kesalahan yang serius jika digunakan oleh pemakai. Jika pemakai membuat kesalahan, system tersebut harus dapat mendeteksi kesalahan dan menawarkan instruksi yang sederhana, konstruktif, dan spesifik untuk perbaikan.

E. Memungkinkan pembalikan aksi yang mudah

Aksi yang telah dilakukan harus dapat dikembalikan ke keadaan awal dengan mudah.

F. Mendukung pusat kendali internal

Menjadikan pemakai sebagai pemberi aksi bukan sebagai orang yang merespon aksi.

G. Mengurangi beban ingatan jangka pendek

Karena keterbatasan manusia dalam pemrosesan informasi pada ingatan jangka pendek, sebaiknya tampilan dibuat sederhana, beberapa halaman dijadikan satu, frekuensi pergerakan *window* dikurangi dan harus ada waktu yang cukup untuk mempelajari kode-kode, singkatan, serta urutan aksi. Informasi seperti singkatan atau kode sebaiknya juga tersedia.

H. Berusaha untuk konsisten

Urutan aksi yang konsisten diperlukan pada situasi yang sama. Konsistensi juga harus digunakan pada *prompt*, menu dan layar bantu. Warna tampilan, kapitalisasi, huruf juga harus konsisten.

Interaksi dengan orang lain dapat dilakukan dalam *game* yang bersifat multiplayer game. Dalam game ini kita tidak bermain sendiri tetapi kita bermain dengan banyak orang yang berasal dari latar belakang yang berbeda. Interaksi yang dilakukan selain komunikasi adalah kerjasama dalam menyelesaikan *game*. Beberapa *game* yang bersifat team seperti dotA *(Defense of the ancient*) dan CS (*Counter Strike*) sangat mementingkan team work bukan skill individu. Hal lain yang perlu kita

perhatikan juga adalah seberapa besar pengaruh game terhadap gamer. Beberpa *game* memang dapat memberikan pengaruh yang buruk terhadap orang yang memainkannya. Gamer yang sering memainkan *game* yang memiliki unsur kekerasan biasanya memiliki karakter yang keras seperti g*ame* yang dimainkan. Untuk menghidari hal ini, seorang *gam*e *developer* harus mampu memikirkan seberapa besar efek yang ditimbulkan *game* yang dibuatnya. Selain itu, seorang player harus memilki kemampuan untuk tidak terpengaruh oleh game yang dimainkan.

2.1.6 AI (*Artificial Intelligence*)

Menurut William Jhon Teahan, *Artificial Intelligence* merupakan studi tentang bagaimana membangun sistem komputer dengan kecerdasan dengan beberapa cara.

Kecerdasan buatan atau *Artificial Intelligence* dimunculkan oleh seorang professor dari Massachussetts Instititute of Technology yang bernama *John McCartchy* pada tahun 1956 pada konferensi Darmouth yang dihadiri oleh para peneliti A.I. Pada konferensi tersebut juga didefinisikan tujuan utaman dari kecerdasan buatan, yaitu mengetahui dan memodelkan proses-proses berpikir manusia dan mendesain mesin agar dapat menirukan kelakuan manusia tersebut.

Terdapat banyak teknik yang digunakan oleh para *game* developer dalam hal penggunaan *Artificial Intelligence* (AI) dalam suatu *game.* Beberapa teknik yang popular adalah: [10]

A. *Randomization*,

Dalam teknik ini setiap tindakan diberi label atau nomor lalu AI secara acak akan mengambil tindakan berdasar pada daftar tindakan tersebut,

B. *Simple Rules*,

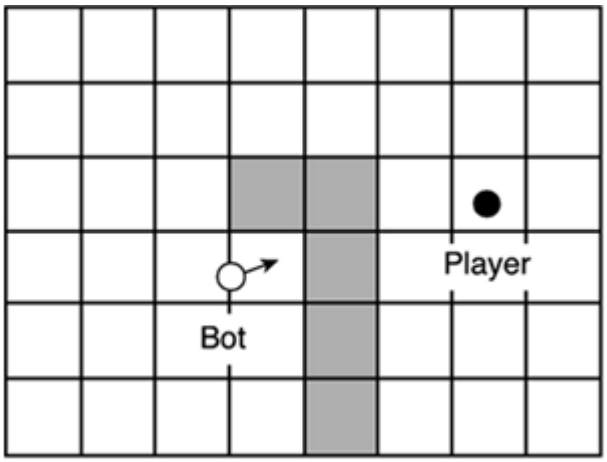
Dalam teknik ini, AI akan memilikh tindakan berdasar pada aturan yang telah dirancang untuk menentukan tindakan yang harus diambil oleh AI jika kondisi tertentu terjadi.

C. *Pattern*,

Dalam teknik ini AI akan mengambil tindakan sama seperti pola yang telah dirancang pengembang

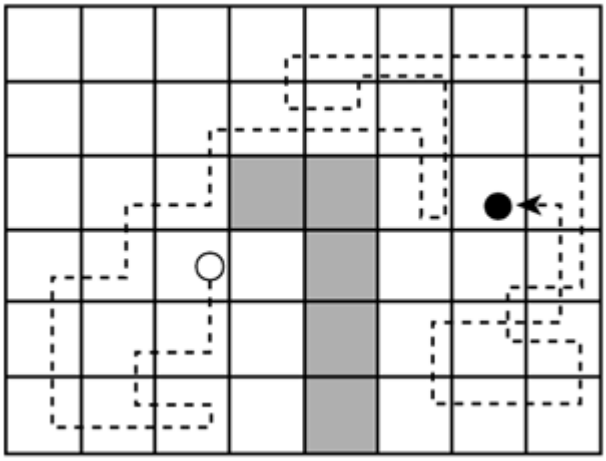
2.1.7 Path finding (Pencarian Jalur)

Beberapa jalan pencarian awal jika lingkungan datar tanpa hambatan, maka pencarian jalan tidak ada masalah : musuh itu hanya bisa bergerak dalam garis lurus ke tujuan tapi ketika ada hambatan antara titik A dan titik B, pencarian mendapat sedikit hambatan sebagai contoh pada gambar 2.1 jika AI hanya bergerak lurus kea rah pemain maka musuh akan berakhir menabrak dinding penghalang dan terjebak disana.



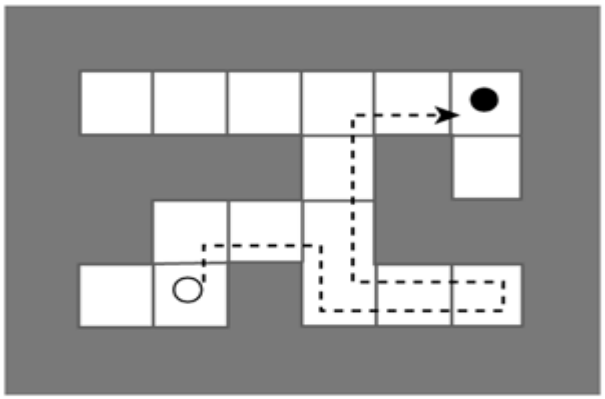
Gambar 2.2. Bukan solusi terbaik

Perilaku ini mungkin baik untuk beberapa permainan karena tidak benar-benar peduli apa bot lakukan, itu penting hanya apa yang mereka muncul untuk melakukan. Dalam dunia 3D dari sudut pandang pemain, mungkin terlihat seperti bot hanya menunggu di belakang dinding untuk pemain bukannya terjebak. Sebuah perspektif luas-mata adalah cerita yang berbeda. Temuan jalan yang hampir diperlukan untuk permainan dengan pemandangan luas-mata perspektif, seperti real-time strategi permainan atau game seperti The Sims. Anda tidak akan ingin memberitahu tim pejuang untuk menyerang sesuatu tetapi memiliki tim tidak dapat melakukannya karena tujuan berada di sisi lain dari dinding! Ayolah, gunakan pintu. Juga, membuat bot cukup pintar untuk menemukan jalan di sekitar rintangan membuatnya lebih menarik bagi pemain dan sulit untuk "trik" bot untuk melakukan sesuatu atau bergerak dengan cara tertentu Beberapa teknik dapat digunakan untuk berkeliling hambatan. Sebuah algoritma acak menemukan solusi akhirnya, jika ada satu. Teknik ini membuat bot berjalan-jalan secara acak, seperti pada Gambar 2.2, sampai menemukan apa yang cari.



Gambar 2.3. Bot ditemukan.

Tentu saja, algoritma acak tidak memberikan hasil terbaik tampan atau jalur terpendek. Siapa yang ingin menonton bergerak bot sekitar secara acak seperti itu hilang sirkuit atau dua? Ini algoritma acak dapat dimodifikasi untuk bergerak secara acak hanya sesekali dan pada waktu lain bergerak langsung ke arah pemain, yang akan membuat gerakan bot terlihatsedikitlebihcerdas.Solusi lain dapat ditemukan ketika lingkungan adalah labirin sederhana tanpa loop, seperti pada Gambar 2.3. Di sini, algoritma jalur-temuan adalah "tangan kanan di dinding" algoritma (atau "tangan kiri di dinding" algoritma, jika Anda ke hal semacam itu). Hanya menjaga tangan kanan Anda di dinding saat Anda bergerak melalui labirin, dan akhirnya Anda akan menemukan tujuan.



Gambar 2.4. Bot menemukan pemain [11]

2.1.8 Algoritma A* (*A Star*)

Algorima A*, dapat juga disebut sebagai Algoritma *A Star*, merupakan salah satu contoh algoritma pencarian yang cukup popular di dunia. Beberapa terminologi dasar yang terdapat pada algoritma ini adalah simpul (*node*), simpul asal/mulai (*source node*), simpul akhir/tujuan (*destination node*), simpul sekarang (*current node*), *openlist*, *closedlist*, *bestpath*, biaya (*cost*), halangan (*unwalkable/obstacle*). Simpul (*node*) adalah petak-petak kecil sebagai representasi dari area pencarian (*pathfinding*). Bentuknya dapat berupa persegi, lingkaran, maupun segitiga. Simpul asal/mulai (*source node*) adalah sebuah terminologi untuk posisi awal sebuah benda. Simpul akhir/tujuan (*destination node*) adalah tempat tujuan yang ingin dicapai pada pencarian. Simpul sekarang (*current node*) adalah simpul terbaik sebelumnya yang dipilih dan menjadi titik acuan untuk membangkitkan simpul tetangganya (*adjacent*). *Open list* adalah tempat menyimpan data simpul yang diakses dari simpul asal/mulai (*source node*) atau dari tetangga simpul sekarang (*current node*) yang belum pernah berada di *open list* maupun *closed list*. *Closed list* adalah tempat menyimpan simpul yang pernah menjadi simpul sekarang (*current node*). Biaya biasanya disimbolkan dengan huruf F (*total cost*) adalah nilai yang diperoleh dari penjumlahan nilai G (*actual cost*) dengan nilai H (*heuristic cost*). Halangan/penghalang adalah simpul yang tidak dapat dilalui. Awal dari algoritma ini bekerja adalah dengan menentukan terlebih dahulu simpul asal (*source node*) dan simpul tujuan (*destination node*).

Algoritma A* tanpa fungsi *heuristic* yang baik akan memperlambat pencarian dan dapat menghasilkan rute yang tidak tepat. Untuk menghasilkan rute yang benar – benar tepat, maka fungsi *heuristic*-nya harus *underestimate* terhadap biaya dari suatu *node* ke final *node*. Fungsi *heuristic* yang sempurna akan membuat algoritma A* langsung menuju final *node* tanpa harus mencari ke arah-arah lain. Sehingga jika fungsi *heuristic*-nya terlalu *underestimate* akan menyebabkan algoritma ini beranggapan bahwa ada rute yang lebih baik dari rute yang ada. Untuk fungsi *heuristic* yang *underestimate*, bila nilainya terlalu rendah akan menyebabkan algoritma ini seperti algoritma Dijkstra yang mencari ke segala arah yang mungkin. Hal ini dikarenakan tidak cukupnya informasi mengenai masalah yang dihadapi, sehingga menyebabkan algoritma A* melakukan pencarian lebih banyak dan lebih lama. Jika

algoritma ini diharapkan melakukan pencarian dengan lebih cepat tanpa memperoleh rute terpendek, maka fungsi *heuristic*-nya suatu saat dapat *overestimate*. Beberapa fungsi *heuristic* yang umum digunakan pada algoritma *pathfinding* A* adalah sebagai berikut [12] :

1. Manhattan Distance

Manhattan Distance adalah fungsi heuristic standar untuk algoritma A*. Digunakan pada aplikasi yang memiliki 4 arah gerakan (tidak dapat bergerak diagonal).

h(n) = d * (abs(Xn - Xgoal) + abs (Yn - Ygoal)) ……1

Dimana :

a. d adalah nilai biaya. Dimana nilai d didapat dari nilai minimum cost perpindahan antara node.
b. Xn adalah koordinat X dari node pertama pada grid.
c. Xgoal adalah koordinat X dari final node.
d. Yn adalah koordinat dari node pertama pada grid.
e. Ygoal adalah koordinat Y dari final node.

2. Straight Line Distance

Straight Line Distance adalah fungsi heuristic yang digunakan pada aplikasi yang dapat bergerak ke segala arah / sudut.

h(n) = sqrt((Xn – Xgoal)2 + (Yn – Ygoal)2) ……..2

Dimana :

a. Xn adalah koordinat X dari node pertama pada grid
b. Xgoal adalah koordinat X dari final node
c. Yn adalah koordinat dari node pertama pada grid
d. Ygoal adalah koordinat Y dari final node.

3. Diagonal Distance

Diagonal Distance adalah fungsi heuristic yang digunakan pada aplikasi yang memiliki delapan arah gerakan (dapat bergerak diagonal).

h(n) = d * max abs (Xn – Xgoal), abs (Yn – Ygoal))……3

Dimana :

a. d adalah nilai biaya. Dimana nilai d didapat dari nilai minimum cost

perpindahan antar node.

b. Xn adalah koordinat X dari node pertama pada grid.
c. Xgoal adalah koordinat X dari final node.
d. Yn adalah koordinat dari node pertama pada grid.
e. Ygoal adalah koordinat Y dari final node.

2.1.9 Pemrograman berorientasi objek

Pemrograman berorientasi objek (PBO) merupakan paradigma baru dalam rekayasa perangkat lunak yang memandang sistem sebagai kumpulan - kumpulan objek diskrit yang saling berinteraksi. Yang dimaksud berorientasi objek adalah bahwa mengorganisasikan perangkat lunak sebagai kumpulan objek - objek diskrit yang bekerja sama antara informasi atau struktur data dan perilaku (*behavior*) yang mengaturnya.

Objek adalah segala sesuatu yang ada di sekitar kita, dimana objek – objek-lah yang menyusun dunia ini. Misalkan mobil, bis, truck, motor, becak, sepeda, dan lain - lain adalah contoh-contoh objek kendaran. Setiap obyek mempunyai informasi-informasi atau atribut - attribut dan perilaku sebagai suatu operasi pengaturnya. [13]

2.1.10 *Unified Modeling Language* (UML)

Menurut Martin Fowler UML adalah bahasa pemodelan, bukan suatu metode. UML tidak memiliki notasi atas proses yang merupakan bagian penting dari metode, UML terdiri atas sembilan diagram. Kesembilan diagram tersebut, yaitu: [14]

A. *Use-Case* Model Diagram

Use-Case Diagram menggambarkan interaksi antara sistem dengan luar sistem dan sistem dengan *user*. Dengan kata lain, *Use-Case Diagram* secara grafis menggambarkan siapa yang akan menggunakan sistem dan dalam cara apa *user* ingin berinteraksi dengan sistem.

B. *Class Diagram*

Melukiskan struktur sistem dalam bentuk objek. Disni digambarkan objek *class* yang membangun sistem beserta hubungan antar *class.*

C. *Object Diagram*

Serupa dengan *class diagram,* tetapu disamping menggambarkan *object class*, digambarkan juga objek *instance* yang menampilkan nilai attribute dari *instance*. Diagram ini dapat digunakan untuk membantu tim pengembangan memahami struktur sistem lebih baik.

D. *Squence Diagram*

Secara grafis menggambarkan bagaimana objek saling berinteraksi melalui pesan dalam melakukan suatu operasi atau melakukan *use case.*

E. *Collaboration Diagram*

Serupa dengan *sequence diagram,* tetapi yang difokuskan disini bukanlah urutan *(Sequence)* melainkan interaksi antar objek dalam format jaringan.

F. *Statechart Diagram*

Digunakan untuk memodelkan aspek dinamis daru suatu objek. Pada *Statechart Diagram* diilustrasikan daur hidup objek, berbagai keadaan objek dan peristiwa yang menyebabkan transisi dari keadaan yang satu ke keadaan yang lain .

G. *Activity Diagram*

Digunakan untuk menggambarkan urutan aktivitas secara berurutan dari proses bisnis atau *Use Case.*

H. *Component Diagram*

Digunakan untuk menggambarkan organisasi dari sistem dan ketergantungan dari komponen software dalam sistem. *Component Diagram* dapat juga digunakan untuk menunjukan bagaimana kode program dibagi menjadi modul-modul (komponen).

I. *Deployment Diagram*

Deployment Diagram mendeskripsikan arsitektur fisik dalam *“node”* untuk *hardware* dan *software* dalam sistem. Di sini digambarkan konfigurasu dari komponen *software, processor* dan peralatan lain yang membangun arsitektur sistem secara *run-time.*

2.1.11 JAVA

Bahasa pemrograman Java ™ adalah tujuan umum, konkuren, *classbased*, bahasa berorientasi objek. Hal ini dirancang untuk menjadi cukup sederhana bahwa banyak programmer dapat mencapai kelancaran dalam bahasa. Bahasa pemrograman Java terkait dengan C dan C + + tetapi diselenggarakan agak berbeda, dengan sejumlah aspek C dan C + + dihilangkan dan beberapa ide dari bahasa lain disertakan. Hal ini dimaksudkan menjadi bahasa produksi, bukan bahasa penelitian, dan sebagainya, seperti CAR Hoare disarankan dalam makalah klasiknya pada desain bahasa, desain telah menghindari termasuk fitur baru dan belum teruji.

Bahasa pemrograman Java sangat dan statis diketik. spesifikasi ini jelas membedakan antara waktu kompilasi kesalahan yang dapat dan harus dideteksi pada waktu kompilasi, dan yang terjadi pada saat run-time. Kompilasi waktu biasanya terdiri menerjemahkan program menjadi representasi kode mesin-independen byte. Run-waktu kegiatan termasuk loading dan menghubungkan dari kelas yang dibutuhkan untuk melaksanakan sebuah program, mesin opsional kode generasi dan optimasi dinamis dari program, dan pelaksanaan program yang sebenarnya.

Bahasa pemrograman Java adalah bahasa yang relatif tinggi tingkat, dalam rincian dari representasi mesin tidak tersedia melalui bahasa. Ini mencakup manajemen penyimpanan otomatis, biasanya menggunakan kolektor sampah, untuk menghindari masalah keamanan dealokasi eksplisit (seperti dalam bebas C atau C++). Implementasi kinerja tinggi sampah-dikumpulkan dapat telah dibatasi jeda untuk dukungan sistem pemrograman dan aplikasi real-time. Bahasa tidak mencakup konstruksi yang tidak aman, seperti akses array tanpa memeriksa indeks, karena konstruksi yang tidak aman seperti itu akan menyebabkan program untuk berperilaku dengan cara yang tidak ditentukan

Kenapa harus java, dengan java anada dapat membuat *hardware accelerated l*ebih cepat, mendapatkan bahasa berorientasi objek yang komperhensif, *multithreading* sederhana juga melimpahnya *tools* yang tersedia dan banyak *open source liblary* yang fantastis [15]