BAB 2
LANDASAN TEORI

2.1 Pertempuran Lima Hari Di Semarang

Pertempuran lima hari di Semarang adalah serangkaian pertempuran antara rakyat Indonesia di Semarang melawan Tentara Jepang. Pertempuran ini adalah perlawanan terhebat rakyat Indonesia terhadap Jepang pada masa transisi. Pertempuran dimulai pada tanggal 15 Oktober 1945 dan berakhir tanggal 20 Oktober 1945.

Transitnya 675 orang pasukan Jepang, yang mereka dalam perjalanan dari Irian ke Jakarta, tapi karena persoalan logistik, pasukan ini singgah di Semarang. Pasukan ini merupakan pasukan tempur yang mempunyai pengalaman di medan perang Irian. Keadaan kontras sekali, karena para pemuda pejuang kita harus menghadapi pasukan Jepang yang berpengalaman tempur dan lebih lengkap persenjataannya, sementara kelompok pasukan pemuda belum pernah bertempur, dan hampir-hampir tidak bersenjata.

Pertempuran lima hari di Semarang ini diawali dengan berontakan 400 tentara Jepang yang bertugas membangun pabrik senjata di Cepiring dekat Semarang. Pertempuran antara pemberontak Jepang melawan Pemuda ini berkobar sejak dari Cepiring hingga Jatingaleh yang terletak di bagian atas kota. Di Jatingaleh ini pasukan Jepang yang dipukul mundur menggabungkan diri dengan pasukan Kidobutai lainnya.

Suasana kota Semarang menjadi panas. Terdengar bahwa pasukan Kidobutai Jatingaleh akan segera mengadakan serangan balasan terhadap para Pemuda Indonesia. Meluasnya desas-desus bahwa cadangan air minum di Candi (Siranda) telah diracuni oleh Pihak Jepang dan melucuti 8 orang polisi Indonesia yang menjaga tempat tersebut untuk menghindarkan peracunan cadangan air minum itu. Dr Karyadi, Kepala Laboratorium Pusat Rumah Sakit Rakyat (Purasara) ketika mendengar berita ini langsung meluncur ke Siranda untuk mengecek kebenarannya. Tetapi Dr. Karyadi tidak sampai tujuan karna di bunuh oleh tentara Jepang di jalan Pandanaran Semarang.

Pasukan Kidobutai melancarkan serangannya ke tengah-tengah kota Semarang. Pertempuran sengit antara pasukan Jepang melawan para Pemuda yang bertahan di bekas Gedung NIS (Lawang Sewu) dan di Gubernuran (Wisma Perdamaian). Diperkirakan 2.000 pasukan Jepang terlibat dalam pertempuran besar-besaran melawan pemuda Indonesia. Pertempuran yang paling banyak menelan korban terjadi di Simpang Lima, untuk mengenang pertempuran di semarang maka didirikan monument perjuangan tugu muda [4].

2.2 *Game*

Game merupakan salah satu media hiburan yang paling popular untuk semua kalangan usia. Sejak pertama kali ditemukan sampai saat sekarang, teknologi *game* telah mengalami kemajuan yang terbilang pesat. Hal ini ditandai dengan berkembangnya jenis *game*, produk, alat dan jenis interaksi *game* dengan penggunaan yang semakin beragam bentuknya[5].

2.2.1 Pengertian *Game*

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

Menurut Andang Ismail terdapat dua pengertian *game* (permainan). Pertama, *game* (permainan) adalah sebuah aktifitas bermain yang murni mencari kesenangan tanpa mencari menang atau kalah. Kedua, permainan diartikan sebagai aktifitas bermain yang dilakukan dalam rangka mencari kesenangan dan kepuasan, namun ditandai pencarian menang – kalah.

Berdasarkan representasinya, *game* dapat dibedakan menjadi 2 jenis yaitu *game* 2 dimensi (2D) dan 3 dimensi (3D). *Game* 2D adalah *game* yang secara matamatis hanya melibatkan 2 elemen koordinat kartesius yaitu x dan y, sehingga konsep kamera pada *game* 2D hanya menentukan gambar pada *game* yang dapat dilihat oleh pemain. Sedangkan *game* 3D adalah *game* yang selain melibatkan elemen x dan y juga melibatkan elemen z pada perhitungannya sehingga konsep

kamera pada *game* 3D benar-benar menyerupai konsep kamera pada kehidupan nyata [5].

2.2.2 Sejarah *Game*

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

Game pertama kali dibuat pada tahun 1966 oleh Ralph Baer bersama dengan timnya yang berjumlah 500 orang yang terdiri dari para insinyur dan teknisi. Setelah sebulan bekerja keras, Baer berhasil menampilkan dua titik putih yang bekejar-kejaran dilayar. Pada tahun 1972 muncul nama baru dalam dunia *game*. Nolan Bushel mendirikan perusahaan Atari dan membuat *game Arcade* Pong. Pada era baru dalam perkembagan dunia *game* terjadi pada tahun 1988 yang didominasi oleh perusahaan Jepang. Nintendo yang awalnya hanya memproduksi mesin *fotocopy,* dengan *video game* sistemnya telah mencapai omset miliaran dolar. Setalah masa awal perkembagan *game* ini, dunia *game* telah melalui beberapa *fase* yaitu *fase game* PC, fase *game console* dan fase *game online*. Sekarang ketiga fase *game* tersebut sudah semakin banyak variannya dan masing-masing memiliki kelompok penggemarnya sendiri [5].

2.2.3 *Genre Game*

Berikut klasifikasi *game* berdasarkan genre permainannya [6] :

1. Aksi *Shooting*

   Video game jenis ini sangat memerlukan kecepatan refleks, koordinasi mata dan tangan, juga timing, inti dari *game* jenis ini adalah tembak, tembak dan tembak. Termasuk didalamnya tedapat :

   a. *First person shooter (FPS)* seperti *Counter Strike* dan *Call of Duty.*

      Drive n shoot, menggunakan unsur simulasi kendaraan tetapi tetap dengan tujuan utama menembak dan menghancurkan lawan, contoh : *Spy Hunter, Rock and Roll Racing, Road Rash.*

   b. *Shoot em up,* seperti Raiden, 1942, dan gradius.

   c. Beat em up (tonjok hajar) seperti *Double Dragon* dan *Final Fight,* lalu *hack and slash* (tusuk tebas) seperti *Shinobi* dan *Legend of Kage.*

d. *Light gun shooting*, yang menggunakan alat yang umumnya berbentuk seperti senjata, seperti *Virtua Cop* dan *Time Crisis.*
e. *Fighting* (pertarungan) Ada yang mengelompokan *video game fighting* di bagian Aksi, jenis ini memang memerlukan kecepatan refleks dan koordinasi mata-tangan, tetapi inti dari *game* ini adalah penguasaan jurus (hafal caranya dan lancar mengeksekusinya).

2. Petualangan.

   Jenis *video game* aksi-petualangan memerlukan kelihaian pemain dalam bergerak, *refleks*, berlari, melompat hingga memecut atau menembak tidak diperlukan di sini. *Video Game* murni petualangan lebih menekankan pada jalan cerita dan kemampuan berpikir pemain dalam menganalisa tempat secara *visual*, memecahkan teka-teki maupun menyimpulkan rangkaian peristiwa dan percakapan karakter hingga penggunaan benda-benda tepat pada tempat yang tepat.

3. Simulasi, Konstruksi dan manajemen.

   Video Game jenis ini seringkali menggambarkan dunia di dalamnya sedekat mungkin dengan dunia nyata dan memperhatikan dengan detil berbagai faktor. *Video Game* jenis ini membuat pemain harus berpikir untuk mendirikan, membangun dan mengatasi masalah dengan menggunakan dana yang terbatas. Contoh: *Sim City, The Sims, Tamagotchi.*

4. *Role Playing.*

   Video game jenis ini sesuai dengan terjemahannya, bermain peran, memiliki penekanan pada tokoh/peran perwakilan pemain di dalam permainan, yang biasanya adalah tokoh utamanya, dimana siring kita memainkannya, karakter tersebut dapat berubah dan berkembang ke arah yang diinginkan pemain (biasanya menjadi semakin hebat, semakin kuat, semakin berpengaruh, dll) dalam berbagai parameter yang biasanya ditentukan dengan naiknya *level*, baik dari status kepintaran, kecepatan dan kekuatan karakter. seperti *Final Fantasy, Dragon Quest* dan *Xenogears. Never Winter Nights, baldurs gate, Elder Scroll,* dan *Fallout.*

5. *Puzzle.*

   Video game jenis ini sesuai namanya berintikan mengenai pemecahan teka-teki, baik itu menyusun balok, menyamakan warna bola, memecahkan perhitungan matematika, melewati labirin, sampai mendorong-dorong kota masuk ke tempat yang seharusnya, itu semua termasuk dalam jenis ini. Sering pula permainan jenis ini adalah juga unsur permainan dalam video *game* petualangan maupun *game* edukasi. *Tetris, Minesweeper, Bejeweled, Sokoban* dan *Bomberman.*

6. Simulasi kendaraan.

   Video *Game* jenis ini memberikan pengalaman atau interaktifitas sedekat mungkin dengan kendaraan yang aslinya, muskipun terkadang kendaraan tersebut masih eksperimen atau bahkan fiktif, tapi ada penekanan khusus pada detil dan pengalaman realistik menggunakan kendaraan tersebut.

2.2.4 *First person shooter*

First Person Shooter atau disingkat FPS merupakan *genre* sebuah *game* yang mengambil sudut pandang orang pertama. Dimana pemain seolah-olah menjadi karakter utama dalam *game* yang berpusat pada permainan tersebut. *First person shooter* sama seperti *third person shooter game* yaitu mengutamakan *shooting* dan *combat* dari perspektif karakter yang bertujuan untuk memberikan pemandangan yang lebih luas dan gerakan yang lebih banyak. Salah satu genre atau jenis *game FPS (First Person Shooter)* adalah jenis *game* yang mengutamakan kecepatan dan gerakan pandangan pemain untuk melihat tampilan dilayar, seperti yang terlihat dari pandangan mata pemain itu sendiri.

Seorang penembak orang pertama adalah permainan terstruktur sekitar penembakan, dan dimana pemain tidak dapat melihat avatar atau karakter tetapi hanya bisa melihat pandangan yang ada di layar monitor dalam pandangan sudut orang pertama. *Genre* ini menarik untuk dimainkan karena *gameplaynya* yang *simple* membuat orang-orang mudah untuk memainkannya. Beberapa contoh *game* dengan *genre first Person Shooter* adalah *game Counter Strike*, *Ghost Recon, Rainbow Six, Far Cry*, dan *Doom*.

Sound atau audio dalam *game FPS (First Person Shooter)* merupakan salah satu yang berperan di dalam permainan. Hampir semua *game* FPS *(First Person Shooter)* memanfaatkan audio sebagai objek di dalam permainan, dengan menambahkan audio yang bertujuan untuk menambah warna di dalam suasana saat-saat tertentu. Audio merupakan salah satu unsur terbaik dalam sebuah *game*, karena dengan adanya audio dapat membuat *game* menjadi terasa lebih menegangkan atau lebih segar serta terlihat lebih nyata. Selain audio, animasi merupakan salah satu faktor pendukung di *game FPS (First Person Shooter)*. Karena dengan adanya animasi pergerakan karakter utama akan lebih variatif. Seperti pergerakan memukul musuh, menembak musuh, mereload senjata dan mengganti senjata terlihat lebih natural dan nyata[2].

2.2.5 Unsur *Game*

Dalam sebuah *game* terdapat unsur-unsur yang melengkapinya, diantaranya adalah interaksi antarmuka pengguna melalui gambar yang dihasilkan oleh interaksi berupa video dan juga gambar yang menarik sehingga pemain bisa memainkan sebuah *game*. Berikut beberapa unsur dalam suatu *game* yaitu [6]:

1. Warna

   Warna adalah unsur penting dalam sebuah desain. Dengan penggunaan warna yang tepat dan sesuai memungkinkan sebuah pesan visual dapat mempengaruhi emosi serta pemahaman orang yang melihat pesan visual tersebut. Mata manusia tertarik oleh warna pada suatu *level* karena warna dari objek diterima sebelum detail-detail dipisahkan oleh bentuk-bentuk dan garisnya. Terdapat dua macam warna diantaranya adalah Warna *primer* dan *sekunder,* warna *primer* adalah warna-warna dasar sedangkan warna *sekunder* adalah warna yang dihasilkan dari campuran dua warna *primer* dalam sebuah ruang warna contah warna *primer* dan *sekunder* dapat dilihat pada gambar 2.1.



Gambar 2.1 Warna *Primer* dan *Sekunder* [6]

2. Komposisi

Komposisi adalah pengaturan segala elemen didalam sebuah karya desain yang dapat memenuhi kebutuhan dan tujuan desain yang mudah dipahami. Beberapa *layout* seperti perencanaan, penempatan semua unsur mulai dari tulisan, gambar, ilustrasi, teks, nama dan sebagainya adalah komposisi dengan penggunaan visual 2D dan 3D penerapan warna-warna dalam seluruh aspek desain. Dengan komposisi, desain akan tampak lebih menarik dan enak dipandang dengan pengaturan letak dan perbandaingan objek-objek yang mendukung dalam suatu desain. Dengan demikian perlu menata sedemikian rupa agar tujuan dapat tercapai dan menghasilkan desain yang menarik.

3. Bentuk dasar

Bentuk dasar adalah bentuk-bentuk yang mudah ditemui dalam kehidupan sehari-hari seperti kotak, lingkaran, segitiga dan lain sebagainya. Bentuk-bentuk ini merupakan bentuk yang dapat menghasilkan sebuah karya desain yang menarik karena dengan penggabungan dari beberapa bentuk dasar ini dapat mengahasilkan sebuah karya desain yang diinginkan. Contohnya seperti bentuk rumah, gedung, jembatan dan sebagainya. Bentuk-bentuk ini merupakan yang dapat membedakan didalam sebuah desain diantaranya adalah didalam sebuah lingkungan *game*. Dibawah ini merupakan contoh bentuk-bentuk dasar yang dapat dilihat pada gambar 2.2.



Gambar 2.2 Bentuk Dasar [6]

4. Tipografi

Tipografi adalah perpaduan antara seni dan teknik mengatur tulisan, agar maksud serta arti tulisan dapat tersampaikan dengan baik secara visual kepada pembaca. Pengolahan tipografi tidak hanya terbatas lewat pemilihan jenis huruf, ukuran huruf, dekorasi, kesesuaian dengan tema, tetapi juga meliputi tata letak vertikal atau horizontal tulisan pada sebuah bidang desain.

5. Audio

Audio adalah sinyal elektrik yang digunakan untuk membawa suara dalam batas pendengaran manusia. Audio merupakan komponen sistem yang sudah termasuk didalamnya atau dapat ditambahkan pada komputer. Audio merupakan salah satu yang berperan didalam sebuah *game*, karna dengan menggunakan audio di dalam sebuah *game* akan lebih menarik dan nyata.

2.2.6 Game 3D

Pada saat kemampuan proses pada komputer semakin cepat, para developer juga senantiasa menciptakan game yang semakin canggih sehingga muncul engine dengan grafik 3D (3 dimensi). Maka dari itu, game 3D dengan hitungan polygon yang sangat besar dan pencahayaan yang sudah canggih, juga tekstur maping mulai diproduksi. Game 3D merepresentasikan objek dalam bentuk 3 dimensi sehingga objek akan terlihat lebih nyata seperti dalam kehidupan nyata.

Game bertipe 3 dimensi merupakan game dengan grafis yang baik dalam penggambaran secara realita, kebanyakan game-game ini memiliki perpindahan kamera (angle) hingga 360 derajat sehingga dapat melihat secara keseluruhan dunia game tersebut [7].

2.2.7 3D Grafis

Grafik komputer 3 dimensi biasa disebut 3D atau adalah bentuk dari benda yang memiliki panjang, lebar, dan tinggi. Grafik 3 Dimensi merupakan teknik penggambaran yang berpatokan pada titik koordinat sumbu x(datar), sumbu y(tegak), dan sumbu z(miring). Representasi dari data geometrik 3 dimensi sebagai hasil dari pemrosesan dan pemberian efek cahaya terhadap grafika komputer 2D.

Tiga dimensi, biasanya digunakan dalam penanganan grafis. 3D secara umum merujuk pada kemampuan dari sebuah *video card*. Saat ini video *card* menggunakan variasi dari instruksi-instruksi yang ditanamkan dalam *video card* itu sendiri (bukan berasal dari *software*) untuk mencapai hasil grafik yang lebih realistis dalam memainkan *game* komputer.

Grafik 3D merupakan perkembangan dari grafik 2D. Didalam grafika komputer, 3D merupakan bentuk grafik yang menggunakan representasi data geometri tiga dimensi. Suatu objek rangka 3D apabila disinari dari arah tertentu akan membentuk bayangan pada permukaan gambar. Proses pembuatan grafik komputer 3D dapat dibagi ke dalam tiga fase, yaitu 3D modeling yang mendeskripsikan bentuk dari sebuah objek, layout dan *animation* yang mendeskripsikan gerakan dan tata letak sebuah objek, dan 3D *rendering* yang memproduksi image dari objek tersebut. Istilah atau pengertian grafik 3D adalah sebuah gambar, garis, lengkungan dan sebagainya yang memiliki titik-titik yang menghubungkan menjadi sebuah bentuk 3D di dalam dunia *game*, 3D secara umum merujuk pada kemampuan dari sebuah *video card*. Saat ini *video card* menggunakan variasi dari instruksi-instruksi yang ditanamkan dalam *video card* itu sendiri (bukan berasal dari *software*) untuk mencapai hasil grafik yang lebih realistis dalam memainkan *game* komputer [8].

2.3 Kecerdasan Buatan

Salah satu unsur yang berperan penting dalam sebuah *game* adalah kecerdasan buatan. Dengan kecerdasan buatan, elemen-elemen dalam *game* seperti musuh biasa berinteraksi dengan *player* dalam berbagai hal, mulai dari bertarung hingga berjalan. khusus proses berjalan, algoritma pathfinding adalah algoritma yang dapat dimanfaatkan membantu musuh menemukan alur jalannya.

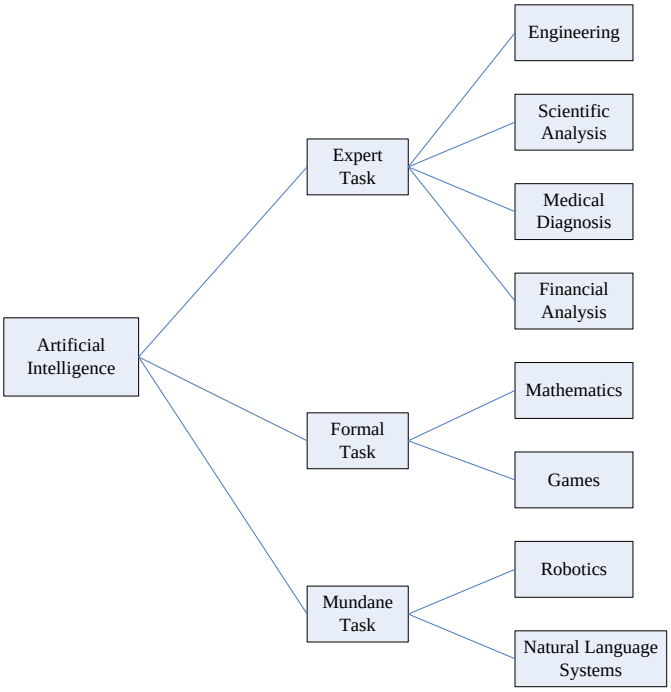
Kecerdasan buatan atau *Artificial Intelligence* (AI) merupakan cabang dari ilmu komputer yang berhubungan dengan pengotomatisan tingkah laku cerdas. Kecerdasan buatan didasarkan pada teori suara (*sound theoretical*) dan prinsip-prinsip aplikasi dari bidangnya. Prinsip-prinsip ini meliputi struktur data yang digunakan dalam representasi pengetahuan algoritama yang diperlukan untuk

mengaplikasikan pengetahuan tersebut serta bahasa dan teknik pemrograman yang digunakan dalam mengimplementasikannya.

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 dan *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 penyelesaian suatu masalah dan dengan aplikasi bidang AI yang berbeda. Pada gambar 2.3 dapat dilihat bidang-bidang tugas dari AI.



Gambar 2.3 Bidang-Bidang Tugas (*task domains*) dari AI [9]

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 melalukan 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 berdasarikan pengalaman.

2.4 Algoritma Pencarian

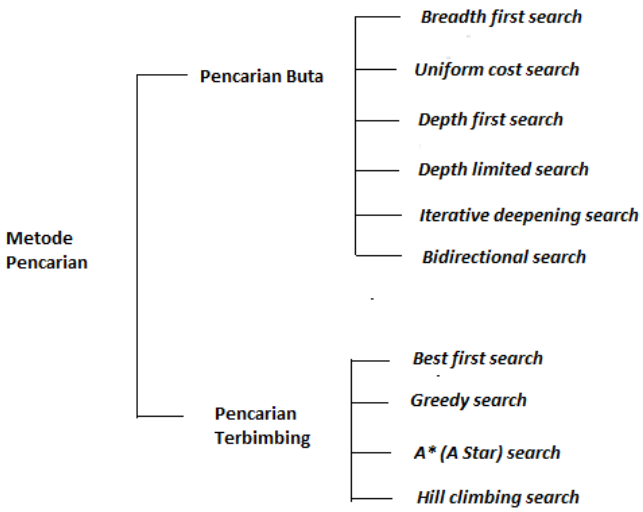
Algoritma Pencarian adalah salah satu bagian dari kecerdasan buatan. Algoritma Pencarian merupakan salah satu algoritma yang sering digunakan dalam beberapa penyelesaian masalah dan untuk memberikan kecerdasan buatan dalam permainan. Algoritma ini akan mencari solusi berdasarkan ruang solusi yang ada secara sistematis tetapi tidak semua ruang solusi akan diperiksa, hanya pencarian yang mengarah kepada solusi yang akan diproses.

Permasalahan pencarian dapat diselesaikan dengan beberapa metode yaitu [9]:

1. Metode pencarian yang pertama adalah metode sederhana yang hanya 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 2.4 Bagan Metode Pencarian [9]

Pada gambar 2.4 adalah metode penulusuran dibagi menjadi dua golongan, yakni pencarian buta *(blind search)* dan pencarian terbimbing *(heuristic search)*.

Beberapa contoh algoritma pencarian yang menggunakan metode *heuristic search* adalah : *Best First Search, Greedy Search,* A* *(A Star) Search,* dan *Hill Climbing Search.*

2.4.1 Pencarian Buta

Pencarian buta (*Blind Search*) adalah pencarian solusi tanpa adanya informasi yang dapat mengarahkan pencarian untuk mencapai *goal state* dari *current state*. 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 banyanknya *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* [9].

2.4.2 Pencarian Terbimbing

Kata heuristik berasal dari kata kerja bahasa Yunani, heuriskein, yang berarti mencari atau menemukan. Dalam dunia pemograman, 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 [9].

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 state* digunakan fungsi *heuristic*. Fungsi *heuristic* berbeda dari pada algoritma dimana *heuristic* lebih merupakan perkiraan untuk membantu algoritma dan tidak harus valid setiap waktu. Beberapa contoh algoritma pencarian yang mengguanakan metode *heuristic search* adalah *Best First Search, Greedy Search*, A* (A Star) dan *Hill Climbing Search* [9].

2.5 Algoritma A* *(A star)*

Algoritma A* *(A Star)* merupakan perbaikan dari metode BFS *(Best First Search)* dengan memodifikasi fungsi *heuristic*nya. A* *(A Star)* akan meminimumkan total biaya lintasan. Pada kondisi yang tepat, A* akan memberikan solusi yang terbaik dalam waktu yang optimal. Pencarian menggunakan algoritma A* mempunyai prinsip yang sama dengan algoritma BFS hanya saja dengan dua faktor tambahan yaitu :

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

2. *Cost* dari setiap simpul ke simpul tujuan bisa diperkirakan. Ini membantu pencarian sehingga lebih kecil kemungkinan dapat mencari ke arah yang salah.

Cost untuk setiap simpul tidak harus berupa jarak. *Cost* bisa saja berupa waktu bila ingin mencari jalan dengan waktu tercepat untuk dilalui. Algoritma A* bekerja dengan prinsip yang hampir sama 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. *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 menuju simpul tersebut ditemukan.

Cost antara simpul adalah jaraknya dan perkiraan *cost* dari suatu simpul ke simpul tujuan adalah penjumlahan jarak dari simpul tersebut ke simpul tujuan. Untuk lebih mudah dimengerti, berikut rumusnya.

$$F(n) = G(n) + H(n) \quad (2.1)$$

keterangan :

F(n) = jumlah nilai dari fungsi G(n) dan H(n). Dengan nilai terkecil F(n) adalah jalur terpendek menuju tujuan akhir.

G(n) = nilai pada pergerakan simpul awal menuju simpul berikutnya.

H(n) = perkiraan nilai pergerakan simpul awal menuju tujuan akhir simpul atau fungsi heuristik.

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. Keoptimalan dari A* cukup langsung dinilai optimal jika H(n) adalah *admissible heuristic* yaitu nilai H(n) tidak akan memberikan penilaian lebih pada *cost* untuk mencapai tujuan. Salah satu contoh dari *admissible heuristic* adalah jarak dengan menarik garis lurus karena jarak terdekat dari dua titik adalah dengan menarik garis lurus [9].

Beberapa terminologi dasar yang terdapat pada algoritma ini adalah *starting point*, *current node*, simpul, *neighbor node*, *open set*, *closed set*, *came from*, harga (*cost*), *walkability*, *target point*.

1. Simpul awal adalah sebuah terminologi untuk posisi awal sebuah benda.
2. *Current node* adalah simpul yang sedang dijalankan dalam algoritma pencarian jalan terpendek.
3. Simpul adalah petak-petak kecil sebagai representasi dari area *pathfinding*. Bentuknya dapat berupa persegi, lingkaran, maupun segitiga.
4. *Neighbor node* adalah simpul-simpul yang bertetangga dengan *current node*.
5. *Open List* adalah tempat menyimpan data simpul yang mungkin diakses dari *starting point* maupun simpul yang sedang dijalankan.
6. *Closed List* adalah tempat menyimpan data simpul sebelum *current node* yang juga merupakan bagian dari jalur terpendek yang telah berhasil didapatkan.
7. *Parent* adalah tempat menyimpan data ketetanggaan dari suatu simpul, misalnya y *parent* x artinya *neighbor node* y dari *current node* x.
8. Harga (F) adalah nilai yang diperoleh dari penjumlahan nilai G, jumlah nilai tiap simpul dalam jalur terpendek dari *starting point* ke *current node*, dan H, jumlah nilai perkiraan dari sebuah simpul ke *target point*.
9. *Target point* yaitu simpul yang dituju.
10. *Walkability* adalah sebuah atribut yang menyatakan apakah sebuah simpul dapat atau tidak dapat dilalui oleh *current node*.

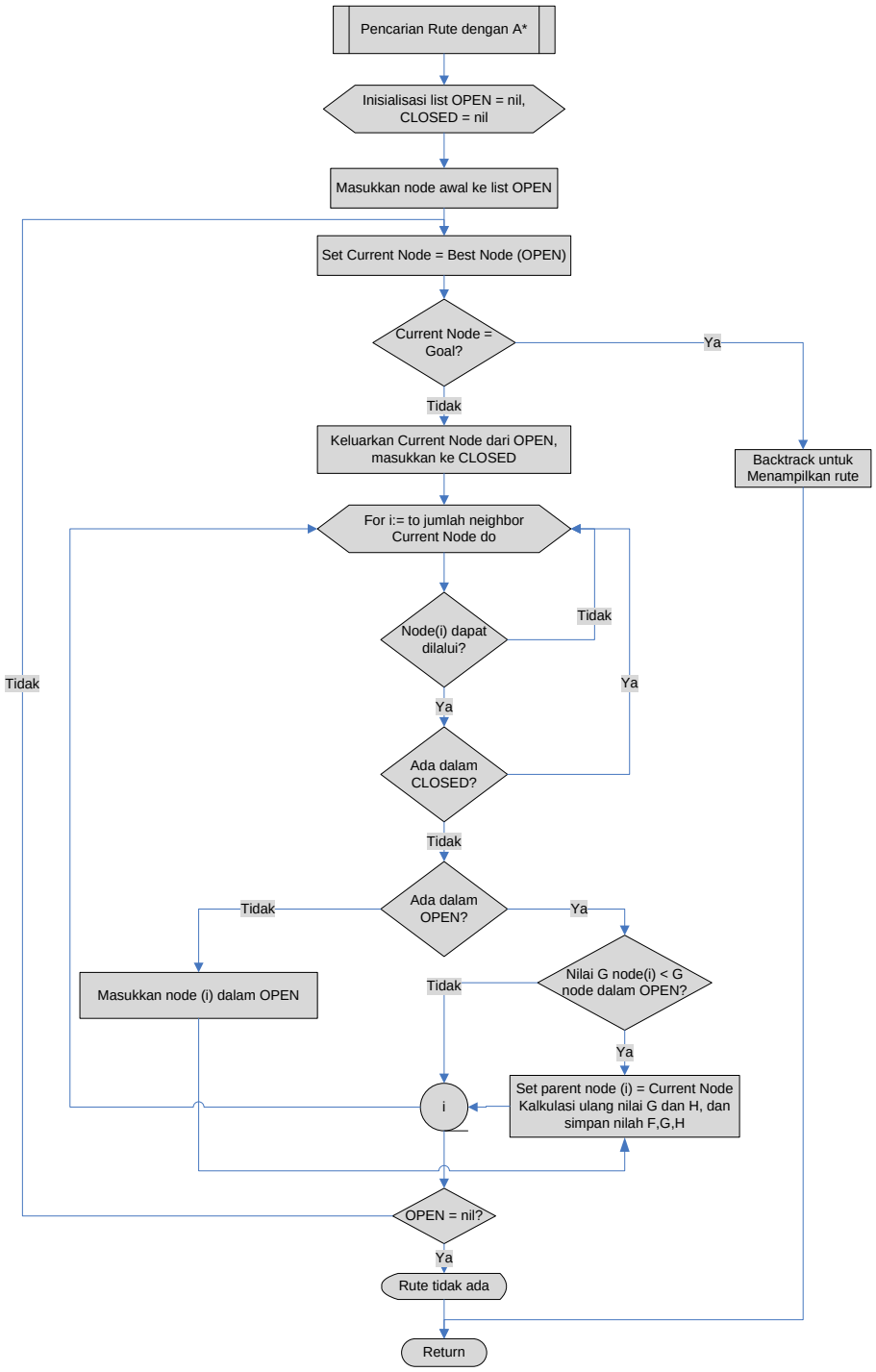
Algoritma A* secara ringkas langkah demi langkahnya adalah sebagai berikut:

1. Tambahkan simpul awal ke dalam *open list*.
2. Ulangi langkah berikut sampai pencarian berakhir:
    a. Carilah simpul n dengan biaya F(n) paling rendah, dalam *open* list. Simpul dengan biaya F terendah kemudian disebut *current node*.

b. Keluarkan *current node* dari *open list* dan masukkan ke dalam *closed list*.
c. Untuk setiap 8 simpul (*neighbor node*) dari *current node* lakukan langkah berikut:
    1. Jika sudah terdapat dalam *closed list* atau tidak *walkable*, maka abaikan, jika tidak lanjutkan.
    2. Jika belum ada pada *open list*, tambahkan ke *open list*. Simpan *current node* sebagai *parent* dari *neighbor node* ini. Simpan harga F masing-masing simpul.
    3. Jika sudah ada dalam *open list*, periksa apakah ini jalan dari simpul ini ke *current node* yang lebih baik dengan menggunakan biaya G sebagai ukurannya. Simpul dengan biaya G yang lebih rendah berarti bahwa ini adalah jalan yang lebih baik. Jika demikian, buatlah simpul ini (*neighbor node*) sebagai *parent* dari *current node*, dan menghitung ulang nilai G dan H dari simpul ini.
d. Berhenti ketika:
    1. Menambahkan *target point* ke dalam *closed list*, dalam hal ini jalan telah ditemukan, atau,
    2. Gagal untuk menemukan *target point*, dan *open list* kosong. Dalam kasus ini, tidak ada jalan.
e. Walaupun telah mencapai *target point*, jika masih ada *neighbor node* yang memiliki nilai yang lebih kecil, maka simpul tersebut akan terus dipilih sampai bobotnya jauh lebih besar atau mencapai *target point* dengan bobot yang lebih kecil dibanding dengan simpul sebelumnya yang telah mencapai *target point*.
f. Pada saat pemilihan simpul berikutnya, nilai F(n) akan dievaluasi, dan jika terdapat nilai F(n) yang sama maka akan dipilih berdasarkan nilai G(n) terbesar yang terakhir dicari.

3. Simpan jalan. Bekerja mundur dari *target point*, pergi dari masing-masing simpul ke simpul *parent* sampai mencapai *starting point*.

Secara umum dapat digambarkan dengan *flowchart* pada gambar 2.5.



Gambar 2.5 *Flowchart* Algoritma A* [10]

2.6 *Fuzzy Logic*

Pada *game* ini *fuzzy logic* digunakan untuk menentukan variasi prilaku yang dilakukan oleh NPC. Dengan adanya *fuzzy logic* masing-masing NPC dapat merubah perilaku berbasis perubahan variabel masukan. Metode *fuzzy* yang digunakan adalah metode sugeno karena metode ini menghasilkan keluaran yang berupa konstan tegas, sehingga dapat mewakili nilai perilaku yang telah dirancang.

Untuk menghasilkan prilaku pada NPC ada dua variabel yang digunakan, yaitu Jarak (dekat, sedang dan jauh) dan Kekuatan (lemah dan kuat). Dengan menggunakan dua variabel diharapkan NPC mampu menentukan perilaku yang akan dilakukan seperti menyerang, bertahan dan melarikan diri.

Konsep tentang *fuzzy logic* atau logika *fuzzy* diperkenalkan oleh Prof. Lotfi Astor Zadeh pada 1962. Logika *fuzzy* adalah metodologi sistem kontrol pemecahan masalah, yang cocok diimplementasikan pada sistem, mulai dari sistem yang sederahana, sistem kecil, *embedded system*, jaringan PC, *multi-channel* atau *workstation* berbasis akurasi data, dan sistem kontrol. Metodologi ini dapat diterapkan pada perangkat keras, perangkat lunak, atau kombinasi keduanya.

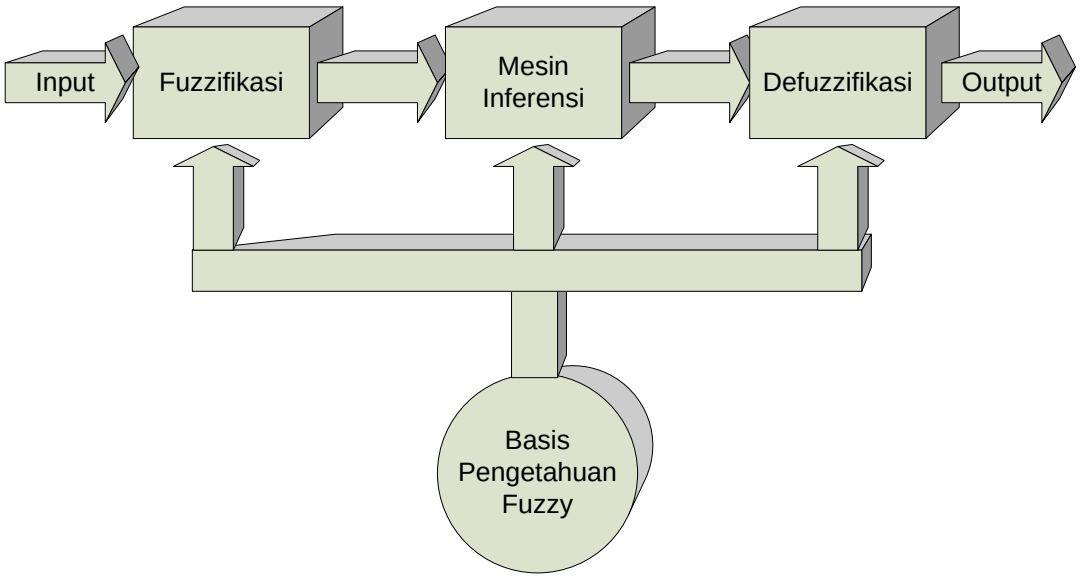
Dalam logika klasik dinyatakan bahwa segala sesuatu bersifat biner, yang artinya adalah hanya mempunyai dua kemungkinan, “Ya atau Tidak”, “Benar atau Salah”, “Baik atau Buruk” dan lain-lain. Oleh karena itu, semua ini dapat mempunyai nilai keanggotaan 0 dan 1. Akan tetapi, dalam logika *fuzzy* memungkinakan nilai kenaggotaan berada diantara 0 dan 1. Artinya, bisa saja suatu keadaan mempunyai dua nilai “Ya dan Tidak”, “Benar dan Salah”, “Baik dan Buruk” secara bersamaan, namun besar nilainya tergantung pada bobot keanggotaan yang dimilikinya[11].

Logika *fuzzy* dapat digunakan diberbagai bidang, seperti pada sistem diagnosis penyakit (dalam bidang kedokteran), pemodelan sistem pemasaran, riset operasi (dalam bidang ekonomi), kendali kualitas air, prediksi adanya gempa bumi, klasifikasi dan pencocokan pola (dalam bidang teknik).

Konsep himpunan *fuzzy* memiliki 2 atribut, yaitu :

1. Linguistik, yaitu nama suatu kelompok yang mewakili suatu keadaan tertentu dengan menggunakan bahasa alami, misalnya dingin, sejuk, panas mewakili *variable* temperatur.
2. Numeris, yaitu suatu nilai yang menunjukan ukuran dari suatu variabel, misalnya 10, 35, 40 dan sebagainya.

Pada gambar 2.6 adalah cara kerja logika fuzzy sebagai struktur elemen dasar sistem inferensi *fuzzy*.



Gambar 2.6 Struktur Sistem Inferensi Fuzzy

Keterangan:

1. Basis Pengetahuan *Fuzzy* merupakan kumpulan *rule-rule fuzzy* dalam bentuk pernyataan IF…THEN.
2. *Fuzzyfikasi* adalah proses untuk mengubah input sistem yang mempunyai nilai tegas menjadi variabel linguistic menggunakan fungsi keanggotaan yang disimpan dalam basis pengetahuan *fuzzy.*
3. Mesin Inferensi merupakan proses untuk mengubah input *fuzzy* dengan cara mengikuti aturan-aturan (*IF-THEN Rules*) yang telah ditetapkan pada basis pengetahuan *fuzzy.*

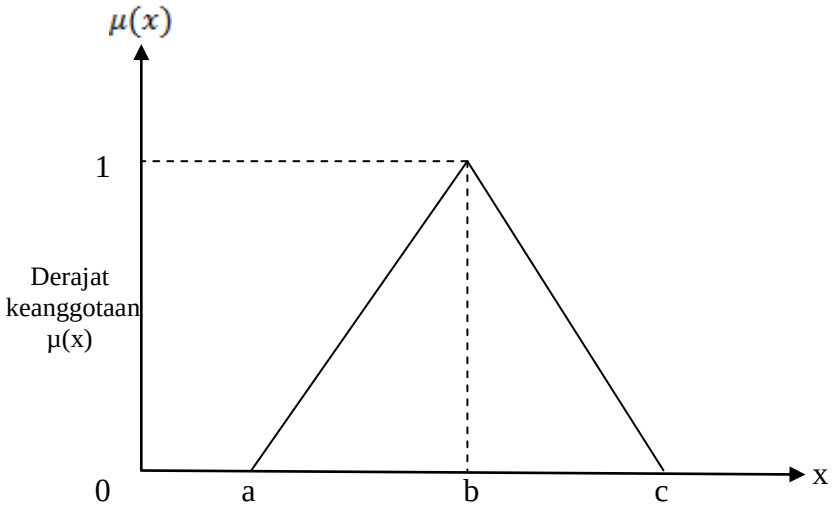
4. *DeFuzzyfikasi* merupakan proses mengubah output *fuzzy* yang diperoleh dari mesin inferensi menjadi nilai tegas menggunakan fungsi keanggotaan yang sesuai dengan saat dilakukan *fuzzyfikasi*.

2.6.1 Fungsi Keanggotaan

Pada *fuzzy system*, fungsi keanggotaan memainkan peran yang sangat penting untuk merepresentasikan masalah dan menghasilkan keputusan yang akurat. Terdapat banyak sekali fungsi keanggotaan yang dapat digunakan, diantaranya adalah [11] :

1. Fungsi Segitiga

Fungsi segitiga hanya terdapat satu nilai x yang memiliki derajat keanggotaan sama dengan 1, yaitu ketika x = b. Tetapi, nilai-nilai disekitar b memiliki derajat keanggotaan yang turun cukup tajam (menjauhi 1). Grafik dan notasi matematika dari fungsi segitiga dapat dilihat pada gambar 2.7.



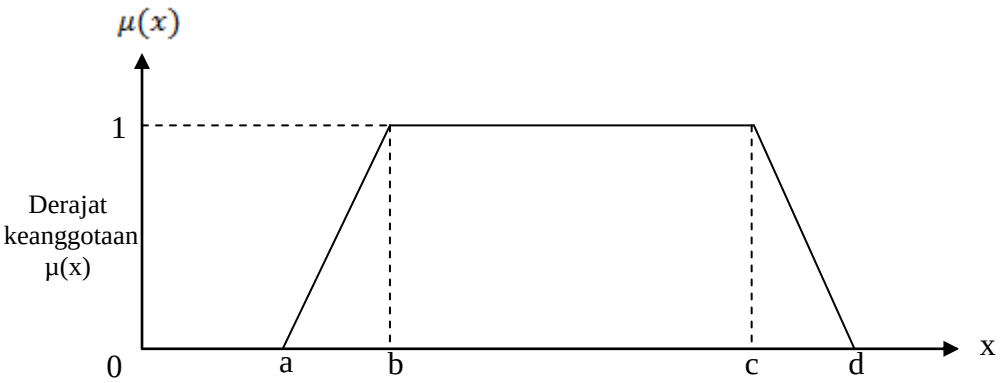
Gambar 2.7 Grafik Fungsi Segitiga [11]

Fungsi keanggotaan :

$$\mu(x) = \begin{cases} 0;\ x \leq a\ atau\ x\ \geq c \\ (x-a)/(b-a)\,;\, a \leq x \leq b \\ (c-x)/(c-b)\,;\, b \leq x \leq c \end{cases} \quad (2.2)$$

2. Fungsi Trapesium

Berbeda dengan fungsi segitiga, pada fungsi ini terdapat beberapa nilai x yang memiliki derajat keanggotaan sama dengan 1, yaitu ketika $b \leq x \leq c$. Tetapi, derajat keanggotaan untuk $c \leq x \leq d$ memiliki karakteristik yang sama dengan fungsi segitiga. Grafik dan notasi matematika dari fungsi Trapesium dapat dilihat pada gambar 2.8.



Gambar 2.8 Grafik Fungsi Trapesium [11]

Fungsi Keanggotaan :

$$\mu(x) = \begin{cases} 0; & x \leq a \text{ atau } x \geq d \\ (x-a)/(b-a); & a \leq x \leq b \\ 1; & b \leq x \leq c \\ (d-x)/(d-c); & c \leq x \leq d \end{cases} \quad (2.3)$$

2.6.2 Metode Sugeno

Fuzzy metode sugeno merupakan metode inferensi fuzzy untuk aturan yang direpresentasikan dalam bentuk IF – THEN, dimana output (konsekuen) sistem tidak berupa himpunan fuzzy, melainkan berupa konstanta atau persamaan linear. Metode ini diperkenalkan oleh Takagi-Sugeno Kang pada tahun 1985. Model Sugeno menggunakan fungsi keanggotaan Singleton yaitu fungsi keanggotaan yang memiliki derajat keanggotaan 1 pada suatu nilai crisp tunggal dan 0 pada nilai crisp yang lain. Penalaran dengan metode Sugeno hampir sama dengan penalaran Mamdani, hanya saja output (konsekuen) sistem tidak berupa

himpunan fuzzy, melainkan berupa konstanta atau persamaan linear. Michio Sugeno mengusulkan penggunaan singleton[11].

Tahapan-tahapan dalam metode Sugeno yaitu sebagai berikut:

1. Pembentukan himpunan Fuzzy

   Pada tahapan ini variabel input (crisp) dari sistem fuzzy ditransfer ke dalam himpunan fuzzy untuk dapat digunakan dalam perhitungan nilai kebenaran dari premis pada setiap aturan dalam basis pengetahuan. Dengan demikian tahap ini mengambil nilai-nilai crisp dan menentukan derajat di mana nilai-nilai tersebut menjadi anggota dari setiap himpunan fuzzy yang sesuai.

2. Aplikasi fungsi implikasi

   Tiap-tiap aturan (proposisi) pada basis pengetahuan fuzzy akan berhubungan dengan suatu relasi fuzzy. Bentuk umum dari aturan yang digunakan dalam fungsi implikasi adalah sebagai berikut:

   IF x is A AND y is B THEN z is f(x,y) (2.6)

   Dengan x dan y adalah skalar, dan A dan B adalah himpunan fuzzy. Proposisi yang mengikuti IF disebut sebagai antesenden sedangkan proposisi yang mengikuti THEN disebut konsekuen. Proposisi ini dapat diperluas dengan menggunakan operator fuzzy seperti,

   IF(x1 is A1) o (x2 is A2) o (x3 is A3) o…o (xN is AN) THEN y is B (2.7)

   dengan o adalah operator (misal: OR atau AND).

   Fungsi ini akan menskala output himpunan fuzzy. Pada metode Sugeno ini , fungsi implikasi yang digunakan adalah fungsi min.

3. Defuzzifikasi (Defuzzification)

   Input dari proses defuzzifikasi adalah himpunan fuzzy yang dihasilkan dari proses komposisi dan output adalah sebuah nilai (crisp).

   Defuzzifikasi pada metode Sugeno lebih sederhana, karena hanya menghitung *center of single-ton* atau titik pusat nilai *crisp* dengan metode rata-rata (*average*)

$$z^* = \frac{\Sigma \mu_i \, . \, z_i}{\Sigma \mu_i} \quad . \tag{2.8}$$

dengan z^* = titik pusat nilai *crisp*

μ_i = derajat keanggotaan masing-masing variabel

z_i = nilai domain pada variabel linguistik z

2.7 OOP (*Object Oriented Programming*)

OOP (*Object Oriented Programming*) adalah pemodelan atau alat untuk mendesain sebuah *game.* Desainnya bisa dari objek karakter ataupun dari programnya.

Objek adalah kesatuan entitas yang memiliki sifat dan tingkah laku. Dalam kehidupan sehari-hari, objek adalah benda, baik benda berwujud nyata seperti manusia, hewan, mobil, komputer, handphone, pena, ataupun benda yang tidak nyata atau konsep, seperti halnya tabungan bank, sistem antrian, sistem internet banking, dan sebagainya. Jadi pengertian OOP adalah konsep yang membagi program menjadi objek-objek yang saling berinteraksi satu sama lain. Objek adalah benda, baik benda yang berwujud nyata maupun benda yang tidak nyata (konsep). Jika menggunakan OOP maka akan ada enam keuntungan yang dapat diperoleh, yaitu [12]:

1. Alami (*Natural*).
2. Dapat diandalkan (*Reliable*).
3. Dapat digunakan kembali (*Reusable*).
4. Mudah untuk dalam perawatan (*Maintainable*).
5. Dapat diperluas (Extendable).
6. Efisiensi waktu.

Berikut ini beberapa bahasa pemrograman yang sudah menggunakan konsep OOP, adalah :

1. *C++*.
2. *Visual C++*.

3. *Visual Basic*.

4. *Java*.

2.8 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 mendefinisikan diagram-diagram sebagai berikut[12]:

1. *use case diagram*
2. *class diagram*
3. *statechart diagram*
4. *activity diagram*
5. *sequence diagram*
6. *collaboration diagram*
7. *component diagram*
8. *deployment diagram*

2.8.1 *Use Case Diagram*

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”. Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya *login* ke sistem, meng-*create* sebuah daftar belanja, dan sebagainya. Seorang atau sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu.

Use case diagram dapat sangat membantu menyusun *requirement* sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua *feature* yang ada pada sistem.

Sebuah *use case* dapat meng-*include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di-*include* akan dipanggil setiap kali *use case* yang meng-*include* dieksekusi secara normal.

Sebuah *use case* dapat di-*include* oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common*.

Sebuah *use case* juga dapat meng-*extend use case* lain dengan *behaviour*-nya sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain [13].

2.8.2 *Class Diagram*

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi).

Class diagram menggambarkan struktur dan deskripsi *class*, *package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain [13].

2.8.3 *Statechart Diagram*

Statechart diagram menggambarkan transisi dan perubahan keadaan (dari satu *state* ke *state* lainnya) suatu objek pada sistem sebagai akibat dari *stimuli* yang diterima. Pada umumnya *statechart diagram* menggambarkan *class* tertentu (satu *class* dapat memiliki lebih dari satu *statechart diagram*).

Dalam UML, *state* digambarkan berbentuk segiempat dengan sudut membulat dan memiliki nama sesuai kondisinya saat itu. Transisi antar *state* umumnya memiliki kondisi *guard* yang merupakan syarat terjadinya transisi yang

bersangkutan, dituliskan dalam kurung siku. *Action* yang dilakukan sebagai akibat dari *event* tertentu dituliskan dengan diawali garis miring [13].

2.8.4 *Activity Diagram*

Activity diagram menggambarkan berbagai alur aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alur berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

Activity diagram merupakan *state diagram* khusus, dimana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu *activity diagram* tidak menggambarkan *behaviour* internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum [13].

2.8.5 *Sequence Diagram*

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait).

Sequence diagram biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah *event* untuk menghasilkan *output* tertentu. Diawali dari apa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan [13].

2.8.6 *Collaboration Diagram*

Collaboration diagram juga menggambarkan interaksi antar objek seperti *sequence diagram*, tetapi lebih menekankan pada peran masing-masing objek dan bukan pada waktu penyampaian *message*. Setiap *message* memiliki *sequence number*, di mana *message* dari level tertinggi memiliki nomor 1. *Messages* dari level yang sama memiliki *prefiks* yang sama [13].

2.8.7 *Component Diagram*

Component diagram menggambarkan struktur dan hubungan antar komponen piranti lunak, termasuk ketergantungan (*dependency*) di antaranya. Komponen piranti lunak adalah modul berisi *code*, baik berisi *source code* maupun *binary code*, baik *library* maupun *executable*, baik yang muncul pada *compile time*, *link time*, maupun *run time*.

Umumnya komponen terbentuk dari beberapa *class* dan/atau *package*, tapi dapat juga dari komponen-komponen yang lebih kecil. Komponen dapat juga berupa *interface*, yaitu kumpulan layanan yang disediakan sebuah komponen untuk komponen lain [13].

2.8.8 *Deployment Diagram*

Deployment/physical diagram menggambarkan detail bagaimana komponen di-*deploy* dalam infrastruktur sistem, di mana komponen akan terletak (pada mesin, server atau piranti keras apa), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi server, dan hal-hal lain yang bersifat fisikal

Sebuah *node* adalah server, *workstation*, atau piranti keras lain yang digunakan untuk men-*deploy* komponen dalam lingkungan sebenarnya. Hubungan antar *node* (misalnya TCP/IP) dan *requirement* dapat juga didefinisikan dalam diagram ini [13].

2.9 C#

C# merupakan sebuah bahasa pemrograman yang berorientasi objek yang dikembangkan oleh Microsoft sebagai bagian dari inisiatif kerangka .NET *Framework* [13]. C# adalah Java versi Microsoft, sebuah bahasa multi flatform yang didesain untuk bisa berjalan di berbagai mesin. C# adalah pemrograman berorientasi Object (OOP). C# memiliki kekuatan bahasa C++ dan portabilitas seperti *Java*. Fitur-fitur yang diambilnya dari bahasa C++ dan *Java* adalah desain berorientasi objek, seperti *garbage collection, reflection*, akar kelas (*root class*), dan juga penyederhanaan terhadap pewarisan jamak (*multiple inheritance*).

Bahasa pemrograman C# dibuat sebagai bahasa pemrograman yang bersifat *general-purpose* (untuk tujuan jamak), berorientasi objek, modern, dan

sederhana. C# ditujukan agar cocok digunakan untuk menulis program aplikasi baik dalam sistem *clien-server (hosted system)* maupun sistem *embedded (embedded system),* mulai dari program aplikasi yang sangat besar yang menggunakan sistem operasi yang canggih hingga kepada program aplikasi yang sangat kecil.

Meskipun aplikasi C# ditujukan agar bersifat 'ekonomis' dalam hal kebutuhan pemrosesan dan memori komputer, bahasa C# tidak ditujukan untuk bersaing secara langsung dengan kinerja dan ukuran program aplikasi yang dibuat dengan menggunakan bahasa pemrograman C [14].

2.10 *Unity Game Engine*

Unity Game Engine adalah *software* atau *game engine* yang digunakan untuk membuat *video game* berbasis dua atau tiga dimensi dan dapat digunakan secara gratis. Selain untuk membuat *game*, *unity* juga dapat digunakan untuk membuat konten yang interaktif lainnya seperti, visual arsitektur dan *real-time* 3D animasi.

Unity adalah sebuah sebuah *tool* yang terintegrasi untuk membuat *game*, arsitektur bangunan dan simulasi. *Unity* bisa digunakan untuk *game*s PC dan *games online*. Untuk *games online* diperlukan sebuah plugin, yaitu *Unity Web Player*, yang sama halnya dengan *flash player* pada *browser*. Bahasa pemrograman yang digunakan bermacam-macam, mulai dari *javascript*, C#, dan *boo*[15].

Unity tidak bisa melakukan desain atau *modelling*, dikarenakan *unity* bukan merupakan *tools* untuk mendesain. Banyak hal yang bisa di lakukan di unity, ada fitur *audio reverb zone , particle effect , sky box* untuk menambahkan langit, dan masih banyak lagi, dan juga bisa langsung *edit texture* dari *editor* seperti *photoshop* dll.

Features (Scripting) di dalam *unity* adalah sebagai berikut:

1. Mendukung 3 bahasa pemrograman, *JavaScript*, C#, dan *Boo*.
2. *Flexible and EasyMoving, rotating,* dan *scaling objects* hanya perlu sebaris kode. Begitu juga dengan *duplicating, removing,* dan *changing properties.*

3. *Multi Platform Game* bisa di deploy di PC, Mac, Wii, iPhone, iPad dan *browser*, *android*.
4. *Visual Properties Variables* yang di definisikan dengan *scripts* ditampilkan pada editor. Bisa digeser, di *drag* and *drop*, bisa memilih warna dengan *color picker*.
5. Berbasis .NET, penjalanan program dilakukan dengan *Open Source* .NET *platform, Mono.*

2.11 3D *Studio Max*

3ds Max atau *3D Studio Max* adalah salah *satu software* atau perangkat lunak yang sering digunakan oleh perancang produk untuk membuat animasi atau pemodelan dalam bentuk 3 dimensi. Aplikasi canggih ini dirilis oleh salah satu perusahaan *Autodesk Media & Entertainment* yang pada mulanya dikenal sebagai *Discreet and Kinetix*. *3D Max* merupakan salah satu dari sekian banyak aplikasi *modeling* untuk membuat model 3D dan paling banyak digunakan oleh perancang yang tersebar di seluruh dunia.

Sejalan dengan berkembangnya teknologi termasuk juga dalam bidang komputerisasi. 3D *Max* pun mengalami perubahan-perubahan untuk menyesuaikan dengan kemampuan komputer yang semakin tinggi dalam hal grafis. *3D Max* dikembangkan dari aplikasi sebelumnya yang bernama *3D Studio for Dos*, tetapi aplikasi ini hanya diperuntukan untuk *platform Win32*.

Dengan semakin canggihnya kemampuan *software* ini, maka tidak aneh *3D Max* menjadi program animasi komputer 3D dengan penjualan terbesar di dunia. *Software* ini memiliki kemampuan *modeling* yang kuat dan merupakan *plugin architecture* yang fleksibel dan bekerja dengan *platform Microsoft Windows*. *3D Studio Max* banyak digunakan oleh para pembuat dan perancang *video game, visual architecture, d esign product* dan juga studio TV untuk pembuatan animasi [16].