BAB 2
LANDASAN TEORI

2.1. Game

2.1.1. Pengertian Game

Dalam kamus bahasa Indonesia "Game"diartikan sebagai permainan. Permainan adalah sesuatu yang dapat dimainkan dengan aturan tertentu sehingga ada yang menang dan ada yang kalah. Permainan dalam hal ini merujuk pada pengertian "kelincahan intelektual" (*intellectual playability*). Game juga bisa diartikan sebagai arena keputusan dan aksi pemainnya. Ada target-target yang ingin dicapai pemainnya.

Berikut adalah Pengertian *Game* menurut para ahli :

1. Menurut David Parlett, *Game* adalah sesuatu yang memiliki "akhir dan cara mencapainya": artinya ada tujuan, hasil dan serangkaian peraturan untuk mencapai keduanya
2. Menurut Roger Caillois seorang sosiolog Perancis, dalam bukunya yang berjudul *Les jeux et les hommes* menyatakan *game* adalah aktivitas yang mencakup karakteristik berikut:
   a. Fun (bebas bermain adalah pilihan bukan kewajiban)
   b. Separate (terpisah).
   c. Uncertain, non-productive, governed by rules (ada aturan).
   d. Fictitious (pura-pura).
3. Menurut Clark C.Abt, *Game* adalah kegiatan yang melibatkan keputusan pemain, berupaya mencapai tujuan dengan;dibatasi oleh konteks tertentu (misalnya, dibatasi oleh peraturan).
4. Menurut Bernard Suits, *Game* adalah upaya sukarela untuk mengatasi rintangan yang tidak perlu.
5. Menurut Greg Costikyan, *Game* adalah sebentuk karya seni di mana peserta, yang disebut Pemain, membuat keputusan untuk mengelola sumber daya yang dimilikinya melalui benda di dalam *game* demi mencapai tujuan.
6. Dari buku *Rules of Play, Game* tempat melakukan konflik bohongan [2].

Dalam beberapa kutipan di atas *game* dapat mempunyai banyak arti tetapi mempunyai inti yang sama yaitu memecahkan masalah atau konflik yang di bangun dan mencapai goal tertentu .

2.2. Tower Defense

Tower Defense adalah sebuah *game real time strategy* yang merupakan MOD (modifikasi) dari *game* strategi popular Warcraft 3 yang diporoduksi oleh Blizzard. Blizzard merupakan sebuah perusahaan pembuat *game* ternama yang telah berhasil meluncurkan berbagai *game* yang sering kali merebutkan hati para *gamer* di dunia.

Inti dari permainan *Tower Defence* ini adalah *gamer* ditugaskan untuk melindungi *goal point* (basis pertahanan) dari serbuan musuh yang dikendalikan oleh AI (*artificial intelligence).* Musuh ini akan datang secara bergelombang dan dalam jumlah yang banyak. *Gamer* melindungi target-tower atau menara-menara pertahanan yang akan menembaki setiap musuh yang mendekati target.

Permainan ini dibagi dalam tingkatan-tingkatan yang dimulai dari tingkatan rendah, menegah , hingga tingkatan tinggi sesuai dengan tingkat keberhasilan *gamer* dalam mempertahankan *goal point* setiap tingkatannya. Dan semakin tinggi tingkatan yang dicapai, musuh yang menyerbu semakin kuat dan banyak. Serta uang yang didapatkan dari membunuh musuh-musuh ini pun semakin meningkat untuk digunakan mendirikan menara-menara pertahan yang lebih canggih dan kuat.

2.3. *Artificial Intelligence* (Kecerdasan Buatan)

Banyak yang menerjemahkan *Artificial Intelligence* sebagai kecerdasan buatan, kecerdasan artifisial, intelijensia artifisial, atau intelejensia buatan. Pada bukunya [6], Suyanto sengaja tidak menerjemahkan istilah *Artificial Intelligence* ke bahasa Indonesia karena istilah tersebut sudah sangat akrab bagi orang Indonesia dan banyak digunakan.

AI diartikan secara berbeda-beda tergantung pada sudut pandang para ahli masing-masing. Ada yang hanya focus pada logika berpikir manusia saja dan ada

juga yang mendefinisikan AI lebih luas pada tingkah laku manusia. Stuart Russel dan Peter Norvig mengelompokkan definisi AI, yang diperoleh dari beberapa *textbook* berbeda, ke dalam empat kategori [7], yaitu :

A. Thinking humanly : *the cognitive modeling approach*

Pendekatan ini dilakukan dengan dua cara sebagai berikut :

1. Melalui introspeksi : mencoba menangkap pemikiran-pemikiran kita sendiri pada saat kita berpikir. Tetapi, seorang psikolog Barat mengatakan *"how do you know that you understand?"* Bagaimana anda tahu bahwa anda mengerti? Karena pada saat anda menyadari pemikiran anda. Sehingga definisi ini terkesan mengada-ngada dan tidak mungkin dilakukan.
2. Melalui eksperimen-eksperimen psikologi.

B. Acting humanly : *the Turing test approach*

Pada tahun 1950, Alan Turing merancang suatu ujian bagi komputer berintelijensia untuk menguji apakah komputer tersebut mampu mengelabui seorang manusia yang menginterogasinya melalui *teletype* (komunikasi berbasis teks jarak jauh). Jika *interrogator* tidak dapat membedakan yang d2nterogasi adalah manusia atau komputer, maka komputer berintelijensia tersebut lolos dari *Turning test*. Komputer tersebut perlu memiliki kemampuan : *Natural Language Processing, Knowledge Representation, Automated Reasoning, Machine Learning, Computer Vision, Robotics*. *Turing test* sengaja menghindari interaksi fisik antara *interrogator* dan komputer karena simulasi fisik manusia tidak memerlukan intelijensia.

C. Thinking rationally : *the laws of thought approach*

Terdapat dua masalah dalam pendekatan ini, yaitu :

1. 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%.
2. Terdapat perbedaan besar antara dapat memecahkan masalah "dalam prinsip" dan memecahkannya "dalam dunia nyata".

D. 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 aksi 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 reflex dan intuitif (berhubungan dengan perasaan), belum dapat ditirukan sepenuhnya oleh komputer. Dengan demikian, kedua definisi ini dirasa kurang tepat untuk saat ini..

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.

2.3.1. Algoritma Genetika

2.3.1.1 Pengertian

Algoritma genetik adalah algoritma yang berusaha menerapkan pemahaman mengenai evolusi alamiah pada tugas-tugas pemecahan-masalah (*problem solving*). Pendekatan yang diambil oleh algoritma ini adalah dengan menggabungkan secara acak berbagai pilihan solusi terbaik di dalam suatu kumpulan untuk mendapatkan generasi solusi terbaik berikutnya yaitu pada suatu kondisi yang memaksimalkan kecocokannya atau lazim disebut *fitness*. Generasi ini akan merepresentasikan perbaikan-perbaikan pada populasi awalnya. Dengan melakukan proses ini secara berulang, algoritma ini diharapkan dapat mensimulasikan proses evolusioner.

Pada akhirnya, akan didapatkan solusi-solusi yang paling tepat bagi permasalahan yang dihadapi. Untuk menggunakan algoritma genetik , solusi

masalah direpresentasikan sebagai kromosom. Tiga aspek yang penting untuk penggunaan algoritma genetik:

1. Defenisi fungsi *fitness*
2. Defenisi dan implementasi representasi genetik
3. Defenisi dan implementasi operasi genetik

Jika ketiga aspek di atas telah didefinisikan, algoritma genetik akan bekerja dengan baik. Tentu saja, algoritma genetik bukanlah solusi terbaik untuk memecahkan segala masalah. Sebagai contoh, metode tradisional telah diatur untuk untuk mencari penyelesaian dari fungsi analitis *convex* yang "berperilaku baik" yang variabelnya sedikit. Pada kasus-kasus ini, metode berbasis kalkulus lebih unggul dari algoritma genetik karena metode ini dengan cepat menemukan solusi minimum ketika algoritma genetik masih menganalisa bobot dari populasi awal.

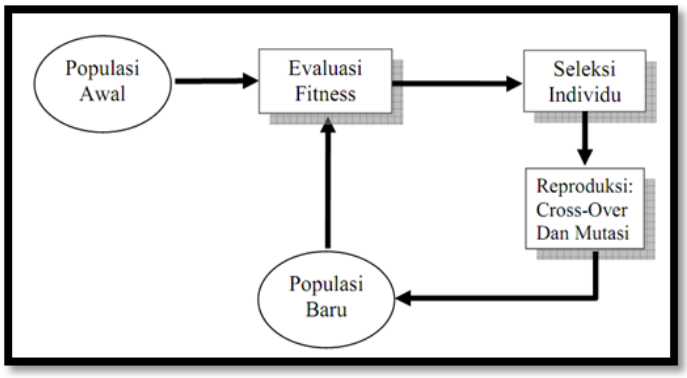
2.3.1.2 Teori Genetika

Teori tentang genetika sudah berkembang sejak lama. Teori yang menyatakan bahwa makhluk hidup menurunkan sifat dari induknya telah dikembangkan sejak zaman pra sejarah. Dimana teori tersebut digunakan dalam proses penyatuan jenis tumbuhan maupun hewan untuk didapatkan sifat yang sesuai dengan keinginan. Walaupun ilmu genetika modern yang menelusuri proses penurunan sifat itu dimulai sejak pertengahan abad ke-19 yang dilakukan oleh Gregor Mendel. Walaupun dia tidak tahu basis utama dari kesamaan sifat yang diturunkan itu, Mendel mengobservasi bahwa makhluk hidup menurunkan sifat melalui sebuah unit penurunan yang bukan lain adalah gen. Sejak ditemukan penurunan sifat melalui gen inilah proses perkembangan teori genetika ini berkembang dengan pesat. Mendel menyatakan gen terbagi dalam berbagi bagian dalam sel makhluk hidup yang adalah DNA, sebuah molekul yang dibangun dari himpunan 4 *nucleotide* yang berbeda. 4 *nucleotide* ini terhimpun secara selang seling dimana akan terbentuk himpunan DNA yang berbeda – beda. Himpunan yang berbeda inilah yang akhirnya akan menghasilakn sifat yang berbeda – beda pula. Himpunan *nucleotide* tersebut akan di terjemahkan oleh sel membentuk

asam amino yang nantinya akan menghasilkan protein, hubungan antara himpunan *nucleotide* tersebut dengan asam amino itulah yang disebut kode genetik. Untuk itu sampai saat ini banyak penelitian yang terus bereksperimen dengan mengganti-ganti himpunan susunan dari DNA agar dapat menghasilkan bentuk sifat yang diinginkan dari makhluk hidup.

2.3.1.3 Struktur Umum

Algoritma genetik memberikan suatu pilihan bagi penentuan nilai parameter dengan meniru cara reproduksi genetik, pembentukan kromosom baru serta seleksi alami seperti yang terjadi pada makhluk hidup. Inisialisasi populasi awal dilakukan untuk menghasilkan solusi awal dari suatu permasalahan algoritma genetika. Inisialisasi ini dilakukan secara acak sebanyak jumlah kromosom/populasi yang diinginkan. Selanjutnya dihitung nilai fitness dan seterusnya dilakukan seleksi dengan menggunakan metode *roulette – wheel* ,. Kemudian dilakukan perkawinan silang (*crossover*) dan mutasi. Setelah melalui beberapa generasi maka algoritma ini akan berhenti sebanyak generasi yang diinginkan. Algoritma Genetika secara umum dapat diilustrasikan dalam gambar 2.1 :



Gambar 2.1 Alur Proses Algoritma Genetika

Pada gambar 2.1 dapat dilihat alur proses algoritma genetika secara umum. Pertama – tama dibangkitkan populasi random awal sesuai dengan yang diinginkan. Tidak ada aturan khusus dalam pembangkitan populasi awal dalam

proses inisialisasi. Pembangkit populasi secara random ini menghasilkan calon solusi awal yang akan melalui tahap – tahap dalam algoritma genetika. Lalu kemudian dilakukan evaluasi terhadap nilai *fitness* kemudian dilakukan seleksi terhadap kormosom yang dihasilkan lalu dilakukan proses reproduksi *crossover* dan mutasi yang kemudian akan menghasilkan populasi baru.

Golberg (1989) mengemukakan bahwa algoritma genetik mempunyai karakteristik-karakteristik yang perlu diketahui sehingga dapat terbedakan dari prosedur pencarian atau optimasi yang lain, yaitu:

1. Algoritma genetika dengan pengkodean dari himpunan solusi permasalahan berdasarkan parameter yang telah ditetapkan bukan dengan parameter itu sendiri.
2. Algoritma genetika pencarian pada sebuah solusi dari sejumlah individu-individu yang merupakan solusi permasalahan bukan hanya dari sebuah individu.
3. Algoritma genetika informasi fungsi objektif (*fitness*), sebagai cara untuk mengevaluasi individu yang mempunyai solusi terbaik, bukan turunan dari suatu fungsi.

4. Algoritma genetik menggunakan aturan-aturan transisi peluang, bukan aturan-aturan deterministik.

Keempat karakteristik yang ada ini merupakan pembeda algoritma genetika dari prosedur pencarian atau optimasi yang lain.

2.3.1.4 Operator Genetika

Algoritma genetika merupakan proses pencarian yang heuristik dan acak sehingga penekanan pemilihan operator yang digunakan sangat menentukan keberhasilan algoritma genetika dalam menemukan solusi optimum suatu masalah yang diberikan. Setelah dilakukan pembangkitan populasi awal , hal lain yang harus diperhatikan adalah bagaimana menghindari terjadinya konvergensi *premature*, yaitu mencapai solusi optimum yang belum waktunya, dalam arti

bahwa solusi yang diperoleh adalah hasil optimum lokal. Operator genetika yang digunakan setelah proses evaluasi tahap pertama membentuk populasi baru dari generasi sekarang. Operator-operator tersebut adalah operator seleksi, *crossover* dan mutasi.

1. Seleksi

Seleksi bertujuan memberikan kesempatan reproduksi yang lebih besar bagi anggota populasi yang paling *fit*. Langkah pertama dalam seleksi ini adalah pencarian nilai *fitness*. Masing-masing kromosom dalam suatu wadah seleksi akan menerima probabilitas reproduksi yang tergantung pada nilai objektif dirinya sendiri terhadap nilai objektif dari semua kromosom dalam wadah seleksi tersebut. Nilai *fitness* inilah yang nantinya akan digunakan pada tahap seleksi berikutnya [1].

Ada beberapa metode untuk memilih kromosom yang sering digunakan antara lain adalah seleksi roda rolet (*roulette wheel selection*), seleksi ranking (*rank selection*) dam seleksi turnamen (*tournament selection*).

Dalam penelitian ini, metode yang digunakan adalah seleksi roda rolet (*roulette wheel selection*). Pada seleksi ini, orang tua dipilih berdasarkan fitness mereka. Lebih kecil nilai suatu kromosom, lebih besar peluangnya untuk terpilih. Probabilitas suatu individu terpilih untuk *crossover* sebanding dengan *fitness*-nya. Cara penyeleksian ini merupakan peniruan dari permainan roda rolet.

Metode seleksi dengan *roullete - wheel* ini merupakan metode yang paling sederhana dan sering dikenal dengan nama *stochastic sampling with replacement*. Cara kerja metode ini adalah sebagai berikut :

a. Dihitung nilai inverse fitness (*Q)* dari masing-masing kromosom (*fi*, dimana I adalah individu ke-1 s/d ke - i) dengan persamaan

$$Q[i] = \frac{1}{fitness} \qquad .. (2.1)$$

b. Dihitung nilai total dengan menjumlahkan semua nilai inverse fitness yang

telah di cari.

c. Dihitung probabilitas (P) dari masing-masing individu

$$\mathrm{P}[i] = \frac{Q[i]}{total} \qquad ..(2.2)$$

d. Dari probabilitas tersebut, akan mulai terlihat kromosom yang mempunyai fitness paling kecil mempunyai probabilitas untuk terpilih pada generasi selanjutny lebih besar dari individu yang lainnya.

e. Sebelum dilakukanny proses seleksi menggunakan *roulette – wheel* terlebih dahulu di cari nilai kumulatif probabilitas (C)

$$\mathrm{C}[i] = C[n] + P\ [n+1] \quad ..(2.3)$$

f. Setelah didapatkan nilai kumulatif probabilitas (C) maka proses seleksi dengan menggunakan *roulette – wheel* dapat dilakukan. Prosesnya adalah dengan membangkitkan bilangan acak R dalam range 0 – 1. Putar *roullete – wheel* sebanyak jumlah populasi yang di bangkitkan .

2. *Crossover*

Kawin silang (*crossover*) adalah operator dari algoritma genetika yang melibatkan dua induk untuk membentuk kromosom baru. Pindah silang menghasilakn titik baru dalam ruang pencarian yang siap untuk diuji. Operasi ini tidak selalu dilakukan pada semua indiidu yang ada. Individu dipilih secara acak untuk dilakukan crossing . Jika pindah silang tidak dilakukan , maka nilai dari induk akan diturunkan kepada keturunan.

Prinsip dari pindah silang ini adalah dengan melakukan operasi (pertukaran , aritmatika) pada gen – gen yang bersesuaian dari dua induk untuk menghasilkan individu baru. Proses *crossover* dilakukan pada setiap individu dengan probabilitas *crossover* yang di tentukan.

Operator *crossover* ini tergantuk pada representasi kromosom yang dilakukan. Berbagai model *crossover* sesuai dengan representasi kromosom. Ada pun beberapa contoh operator *crossover* antara lain *crossover* satu titik ,

crossover dua titik , *crossover* banyak titik dan *crossover* aritmatika biasanya terdapat pada *crossover* bilangan *biner* .

Dalam kasus ini crossover yang akan digunakan adalah crossover untuk representasi permutasi. Sejak pertengahan 80'an, beberapa metode operator pindah silang diciptakan untuk representasi permutasi, seperti *partial-mapped crossover, order crossover, cycle crossover, position-based crossover, order-based crossover,heuristic crossover,* dll.

Metode representasi permutasi yang akan digunakan adalah *Partial-Mapped Crossover (PMX).* PMX diciptakan oleh Goldberg dan Lingle. PMX merupakan rumusan modifikasi dari pindah silang dua-poin. Hal yang penting dari PMX adalah pindah silang 2-poin ditambah dengan beberapa prosedur tambahan. PMX mempunyai langkah kerja sebagai berikut :

Prosedur PMX

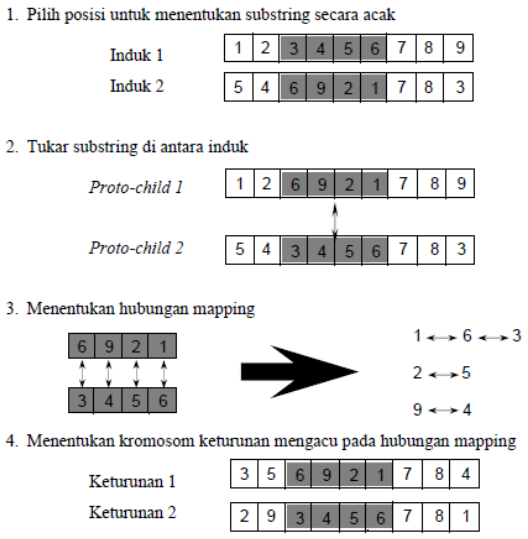
Langkah 1 : Tentukan dua posisi pada kromosom dengan aturan acak. Substring yang berada dalam dua posisi ini dinamakan daerah pemetaan.

Langkah 2 : Tukar dua substring antar induk untuk menghasilkan *proto-child.*

Langkah 3 : Tentukan hubungan pemetaan di antara dua daerah pemetaan.

Langkah 4 : Tentukan kromosom keturunan mengacu pada hubungan pemetaan.

Prosedur ini dapat dilihat ilustrasinya pada Gambar 2.2.



Gambar 2.2 Ilustrasi dari PMX operator

Pada ilustrasi *crossover* dengan PMX operator kromosom yang dijadikan induk dipilih secara acak dan jumlah kromosom yang ditukarkan secara acak. Kemudian dilakukan penentuan posisi tempat terjadinya pemotongan substring secara acak . Setelah itu menentukan hubungan mapping dengan menukarkan substring yang telah dipilih dari induk 1 ke induk 2 yang kemudian menghasilkan keturunan baru yang mengacu pada hubungan mapping .

2. Mutasi

Mutasi merupakan proses mengubah nilai dari satu atau beberapa gen dalam suatu kromosom. Operasi *crossover* yang dilakukan pada kromosom dengan tujuan untuk memperoleh kromosom-kromosom baru sebagai kandidat solusi pada generasi mendatang denga *fitness* yang lebih baik, dan lama-kelamaan menuju solusi optimum yang diinginkan. Akan tetapi, untuk mencapai hal ini, penekanan selektif juga memegang peranan yang penting. Jika dalam proses pemilihan kromosom-kromosom cenderung pada kromosom yang memiliki *fitness* yang tinggi saja, konvergensi *premature*, yaitu mencapai solusi yang optimal lokal sangat mudah terjadi.

Untuk menghindari konvergensi *premature* tersebut dan tetap menjaga perbedaan (*diversity*) kromosom-kromosom dalam populasi, selain melakukan penekanan selektif yang lebih efisien, operator mutasi juga dapat digunakan. Proses mutasi dalam system biologi berlangsung dengan mengubah isi *allele* gen pada suatu *locus* dengan *allele* yang lain. Proses mutasi ini bersifat acak sehingga tidak selalu menjamin bahwa setelah proses mutasi akan diperoleh kromosom dengan *fitness* yang lebih baik.

Prosedur mutasi sangatlah sederhana. Untuk semua gen yang ada, jika bilangan random yang dibangkitkan kurang dari probabilitas *Pmutasi* yang ditentukan maka ubah gen tersebut menjadi nilai kebalikannya (dalam binary encoding, 0 diubah 1, dan 1 diubah 0). Dalam kasus ini mutasi yang akan digunakan adalah *swapping mutation* . Dalam *swapping mutation* dilakukan pertukaran tempat node . Biasanya *Pmutasi* diset sebagai 1/*n* , dimana *n* adalah jumlah gen dalam kromosom . Dengan *Pmutasi* sebesar ini berarti mutasi terjadi

pada sekitar satu gen saja.

Contoh :

(1 **2** 3 4 5 6 **7** 9 8) = > (1 **7** 3 4 5 6 **2** 9 8)

2.3.2. *Fuzzy Logic* (Logika Fuzzy)

2.3.2.1.Pengertian

Fuzzy secara bahasa diartikan sebagai kabur atau samar-samar. Suatu nilai dapat bernilai besar atau salah secara bersamaan. Dalam *fuzzy* dikenal derajat keanggotaan yang memiliki rentang nilai 0 (nol) hingga 1(satu). Berbeda dengan himpunan tegas yang memiliki nilai 1 atau 0 (ya atau tidak).

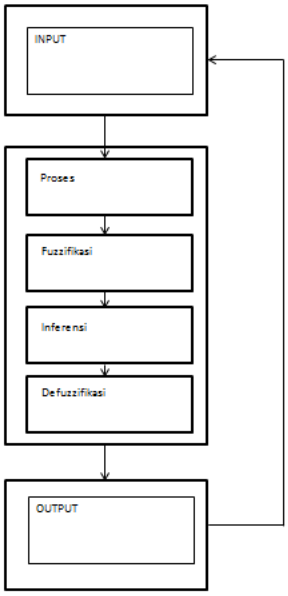
Algoritma *Fuzzy* merupakan suatu logika yang memiliki nilai kekaburan atau kesamaran (*fuzzyness*) antara benar atau salah. Dalam teori Algoritma *Fuzzy* satu nilai bias bernilai benar atau salah secara bersama. Namun berapa besar keberadaan dan kesalahan suatu tergantung pada bobot keanggotaan yang dimilikinya. Algoritma *Fuzzy* memiliki derajat keanggotaan dalam rentang 0 hingga 1. Berbeda dengan logika digital yang hanya memiliki dua nilai 1 atau 0. Algoritma *Fuzzy* digunakan untuk menterjemahkan suatu besaran yang diekspresikan menggunakan bahasa (*linguistic*), misalkan besaran kecepatan laju kendaraan yang diekspresikan dengan pelan, agak cepat, cepat, dan sangat cepat. Dan Algoritma *Fuzzy* menunjukan sejauh mana suatu nilai itu benar dan sejauh mana suatu nilai itu salah. Tidak seperti logika klasik (*crisp*) tegas, suatu nilai hanya mempunyai 2 kemungkinan yaitu merupakan suatu anggota himpunan atau tidak. Derajat keanggotaan 0 (nol) artinya nilai bukan merupakan anggota himpunan dan 1 (satu) berarti nilai tersebut adalah anggota himpunan.

Algoritma *Fuzzy* adalah suatu cara yang tepat untuk memetakan suatu ruang *input* kedalam suatu ruang *output*, mempunyai nilai kontinyu. *Fuzzy* dinyatakan dalam derajat dari suatu keanggotaan dan derajat dari kebenaran. Oleh sebab itu sesuatu dapat dikatakan sebagian benar dan sebagian salah pada waktu yang sama.

Algoritma *Fuzzy* memungkinkan nilai keanggotaan antara 0 dan 1, tingkat keabuan dan juga hitam dan putih, dan dalam bentuk *linguistik*, konsep tidak pasti

seperti "sedikit", "sedang" dan "banyak" [1].

Adapun alur yang terjadi dalam *fuzzy* secara garis besar adalah dapat dilihat pada gambar 2.3 .



Gambar 2.3 Alur Proses *Fuzzy*

Pertama dimasukan inputan yang kemudian akan diproses untuk mencari nilai bobot . Kemudian pada proses *fuzzifikasi* mencari aturan / rule yang berlaku setelah itu mengkomposisikan bobot yang didapatkan dari aturan . Lalu yang terakhir menghitung nilai keluarang dengan menggunakan metode yang berlaku yang kemudian akan menghasilkan output .

2.3.2.2.Konsep Dasar Logika Fuzzy

Teori himpunan *fuzzy* diperkenalkan pertama kali oleh Lotfi A. Zadeh pada tahun 1965. Ada beberapa alasan mengapa orang menggunakan logika *fuzzy*, antara lain :

1. Konsep logika *fuzzy* mudah dimengerti. Konsep matematis yang mendasari penalaran *fuzzy* sangat sederhana dan mudah dimengerti.
2. Logika *fuzzy* sangat *fleksibel*.
3. Logika *fuzzy* memiliki toleransi terhadap data-data yang tidak tepat.
4. Logika *fuzzy* mampu memodelkan fungsi-fungsi *nonlinear* yang kompleks.
5. Logika *fuzzy* dapat membangun dan mengaplikasikan pengalaman-pengalaman para pakar secara langsung tanpa harus melalui proses pelatihan.
6. Logika *fuzzy* dapat bekerjasama dengan teknik-teknik kendali secara konvensional.
7. Logika *fuzzy* didasarkan pada bahasa alami.

Dalam logika *fuzzy* dikenal berhingga keadaan dari nilai "0" sampai ke nilai "1". Logika *fuzzy* tidak hanya mengenal dua keadaan tetapi juga mengenal sejumlah keadaan yang berkisar dari keadaan salah sampai keadaan benar. Ada beberapa hal yang perlu diketahui dalam memahami sistem *fuzzy* ini, yaitu :

1. Variabel *Fuzzy*

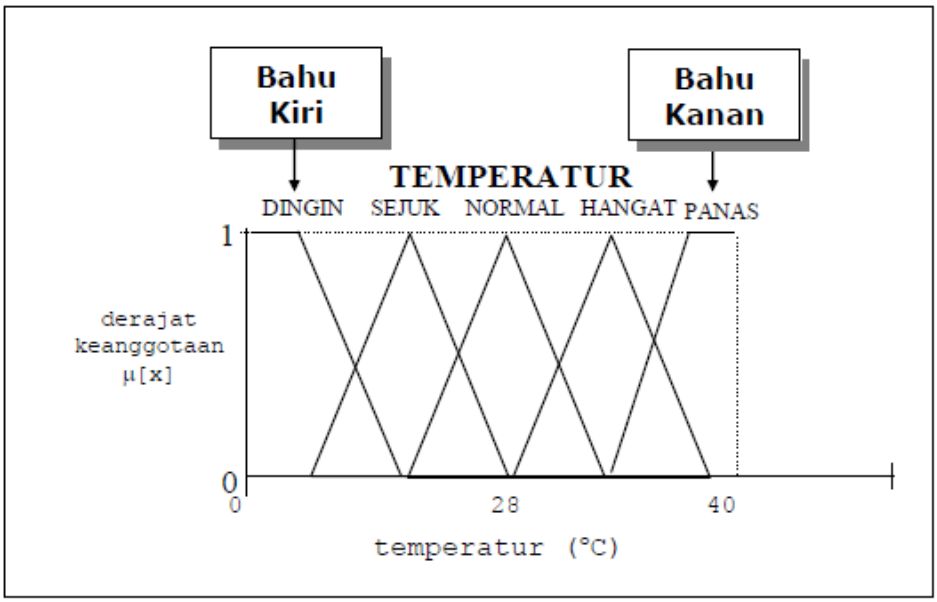
   Variabel *Fuzzy* merupakan variabel yang hendak dibahas dalam suatu sistem *Fuzzy*.
2. Himpunan *Fuzzy*

   Himpunan *Fuzzy* adalah himpunan yang tiap elemennya mempunyai derajat keanggotaan tertentu terhadap himpunannya.

   Himpunan *Fuzzy* memiliki dua atribut, yaitu :

   a. *Linguistik*, yaitu penamaan suatu grup yang mewakili suatu keadaan atau kondisi tertentu dengan menggunakan bahasa alami.
   b. *Numeris*, yaitu suatu nilai (angka) yang menunjukkan ukuran dari suatu variabel.

Contoh Himpunan *Fuzzy* : Misalkan Variabel temperatur, terbagi menjadi 5 himpunan *fuzzy*, yaitu: DINGIN, SEJUK, NORMAL, HANGAT, dan PANAS. Dapat dilihat pada gambar 2.4:



Gambar 2.4 Variable Temprature [1]

Pada himpunan tegas (*crisp*), nilai keanggotaan suatu item x dalam satu himpunan A, yang sering ditulis dengan μA[x], memiliki 2 kemungkinan, yaitu:

a. Satu (1), yang berarti bahwa suatu item menjadi anggota dalam suatu himpunan, atau
b. Nol (0), yang berarti bahwa suatu item tidak menjadi anggota dalam suatu himpunan.

Contoh :

Jika diketahui:

S = { 1, 2, 3, 4, 5, 6} adalah semesta pembicaraan.

A = {1, 2, 3}

B = {3, 4, 5}

bisa dikatakan bahwa:

Nilai keanggotaan 2 pada himpunan A, μA[2]= 1, karena 2 anggota .

Nilai keanggotaan 3 pada himpunan A, μA[3]= 1, karena 3 anggota A.

Nilai keanggotaan 4 pada himpunan A, μA[4]= 0, karena 4 bukan anggota A.

Nilai keanggotaan 2 pada himpunan B, μB[2]= 0, karena 2 anggota B.

Nilai keanggotaan 3 pada himpunan B, μB[3]= 1, karena 3 anggota B.

3. Semesta Pembicaraan

   Semesta pembicaraan adalah suatu keseluruhan nilai yang diperbolehkan untuk dioperasikan dalam suatu variabel *Fuzzy*. Nilai semesta pembicaraan dapat berupa bilangan positif atau bilangan negatif.

4. *Domain*

   Domain himpunan *Fuzzy* adalah keseluruhan nilai yang diijinkan dalam semesta pembicaraan dan boleh dioperasikan dalam suatu himpunan *Fuzzy*. Nilai *domain* dapat berupa bilangan positif maupun bilangan negatif.

2.3.2.3.Karakteristik Logika Fuzzy

Logika *Fuzzy* memiliki beberapa karakteristik yaitu himpunan *Fuzzy* dan fungsi keanggotaan [8].

1. Himpunan *Fuzzy*

Pada logika *boolean*, sebuah individu dipastikan sebagai anggota dari salah satu himpunan saja, sedangkan pada himpunan *fuzzy* sebuah individu dapat masuk pada dua himpunan yang berbeda. Seberapa besar eksistensinya dalam himpunan tersebut dapat dilihat pada nilai keanggotaannya.

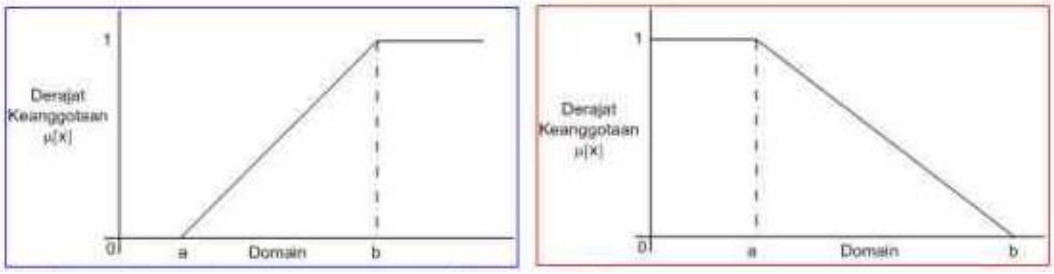
2. Fungsi Keanggotaan (*Membership Function*)

Fungsi keanggotaan adalah suatu kurva yang menunjukkan pemetaan titik-titik *input* data ke dalam nilai keanggotaannya (disebut juga dengan derajat keanggotaan) yang memiliki interval antara 0 sampai 1. Salah satu cara yang dapat digunakan untuk mendapatkan nilai keanggotaan adalah dengan melalui pendekatan fungsi. Derajat keanggotaan dalam himpunan (*degree of membership*) dilambangkan dengan μ .

Dalam sistem *fuzzy* banyak dikenal bermacam-macam fungsi keanggotaan (*membership function*) antara lain adalah fungsi keanggotaan linear , fungsi keanggotaan segitiga dan fungsi keanggotaan trapesium . Dalam game *tower defense* “Benteng Bertahan” fungsi keanggotaan yang akan digunakan adalah fungsi keanggotaan trapesium dan fungsi keanggotaan segitiga.

Representasi Kurva Trapesium

Kurva trapesium dinamakan demikian karena berbentuk trapesium. Dapat dilihat pada gambar 2.5 :



Gambar 2.5 Fungsi Keanggotan Kurva Trapesium

Fungsi keanggotaan trapesium :

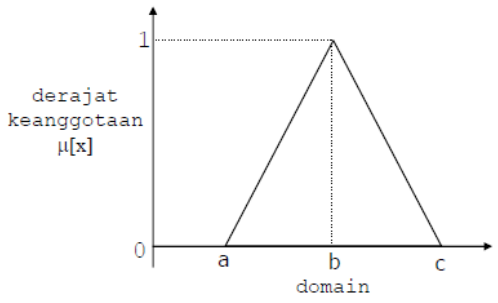
.. (2.4)

$$\mu[x] \begin{cases} 0; & x \le a \\ \frac{(x-a)}{(b-a)}; & a \le x \le b \\ 1 & x \ge b \end{cases}$$

$$\mu[x] \begin{cases} 1; & x \le a \\ \frac{(b-x)}{(b-a)}; & a \le x \le b \\ 0 & x \ge b \end{cases}$$

Representasi Kurva Segitiga

Kurva segitiga pada dasarnya merupakan gabungan antara dua garis dapat dilihat pada gambar 2.6 :



Gambar 2.6 Fungsi Keanggotan Kurva Segitiga

Fungsi keanggotaan Segitiga:

$$\mu[x] \begin{cases} 0; & x \leq a \; Or \; x \geq c \\ \frac{x-a}{b-a}; & a \leq x \leq b \\ \frac{c-x}{c-b}; & b \leq x \leq c \end{cases} \quad .. (2.5)$$

2.3.2.4.Operator Dasar Zadeh

Seperti halnya konvensional , ada beberapa operasi yang didefinisikan secara khusus untuk mengkombinasi dan memodifikasi himpunan *fuzzy*. Nilai keanggotaan sebagai hasil dari operasi dua himpunan sering dikenal dengan nama*fire strength* atau α-predikat. Ada tiga operator dasar yang diciptakan oleh zadeh , yaitu :

1. Operator *AND*

   Operator ini berhubungan dengan operasi interseksi pada himpunan. *Fire strength* sebagai hasil operasi dengan operator *AND* diperoleh dengan mengambil nilai keanggotaan terkecil antar elemen pada himpunan-himpunan yang bersangkutan.

$$\mu\,(A \cap B\,) = \min(\mu A\,[x], \mu B\,[y]) \quad .. (2.6)$$

2. Operator *OR*

   Operator ini berhubungan dengan operasi *union* pada himpunan. *Fire strength* sebagai hasil operasi dengan operator OR diperoleh dengan mengambil nilai keanggotaan terbesar antara elemen pada himpunan-himpunan yang bersangkutan.

$$\mu\,(A \cup B\,) = \max(\mu A\,[x], \mu B\,[y]) \quad .. (2.7)$$

3. Operator *NOT*

   Operator ini berhubungan dengan operasi komplemen pada himpunan.

Fire strength sebagai hasil operasi dengan operator *NOT* diperoleh dengan mengurangkan nilai keanggotaan elemen pada himpunan yang bersangkutan dari 1.

$$\mu\,[A] = 1 - \mu A\,[x] \qquad .. (2.8)$$

Operator Dasar zadeh yang akan digunakan adalah operator AND . Setelah setiap parameter dihitung dan kemudian mendapatkan nilai bobot , setiap nilai tersebut akan dibandingkan dan akan dicari nilai terkecil sesuai dengan aturan *fuzzy* tsukamoto

2.3.2.5.Fuzzyfikasi

Fuzzifikasi adalah pengubahan seluruh variabel *input/output* ke bentuk himpunan *fuzzy*. Rentang nilai variabel *input* dikelompokkan menjadi beberapa himpunan *fuzzy* dan tiap himpunan mempunyai derajat keanggotaan tertentu.

Bentuk *fuzzifikasi* yang dipakai pada sistem ini adalah bentuk segitiga dan bentuk bahu. Bentuk *fuzzifikasi* menentukan derajat keanggotaan suatu nilai rentang *input/output*. Derajat keanggotaan himpunan *fuzzy* dihitung dengan menggunakan rumus fungsi keanggotaan dari segitiga *fuzzifikasi* [8].

2.3.2.6.Inferensi Fuzzy

Setelah fungsi keanggotaan untuk variabel masukan dan keluarannya ditentukan, basis aturan pengendalian dapat dikembangkan untuk menghubungkan aksi keluaran pengendali terhadap kondisi masukannya. Tahap ini disebut sebagai tahap *inferensi*, yakni bagian penentuan aturan dari sistem logika *fuzzy*. Sejumlah aturan dapat dibuat untuk menentukan aksi pengendali *fuzzy*[8].

Pada basis aturan, aturan *If-Then* tersebut dapat menghubungkan banyak variabel masukan dan keluaran. Masukan *x* dipetakan menjadi keluaran *y*. Aturan *if-then* diinterpretasikan sebagai implikasi *fuzzy*. Terdapat banyak sekali model interpretasi implikasi yang telah dikembangkan. Pada aplikasi *Game* ini, metode yang digunakan adalah Metode Tsukamoto

Metode Tsukamoto pertama kali dperkenalkan oleh tsukamoto . Setiap konsekuen (kesimpulan) pada setiap aturan IF-THEN harus direpresentasikan dengan suatu himpunan fuzzy dengan fungsi keanggotaan monoton. Output hasil inferensi dari setiap aturan deberikan secara tegas (*crip*) berdasarkan α – predikat , kemudian menghitung rata-rata terbobot (*weight average*) yang kemudian dalam kasus ini pada tiap nilai akan dibandingkan dan kemudian akan mendapatkan nilai terkecil yang kemudian menjadi output bagi tower berupa perioritas serangan yang dilakukan oleh tower.

2.4. *Object Oriented Programing*

Object Oriented Programming (OOP) merupakan paradigm pemrograman yang berorientasikan kepada objek. Semua data dan fungsi di dalam paradigma ini dibungkus ke dalam kelas-kelas atau objek-objek. Kelebihan penggunaan model data berorientasi objek adalah memberikan fleksibilitas.

Kemudahan mengubah program, dan digunakan luas dalam teknik piranti lunak skala besar. Dengan menggunakan OOP maka dalam melakukan pemecahan suatu masalah tidak melihat bagaimana cara menyelesaikan suatu masalah tersebut (terstruktur) tetapi objek–objek apa yang dapat melakukan pemecahan masalah tersebut. Pemrograman berorientasi objek menekankan konsep berikut:

1. Kelas (*Class*) - kumpulan atas definisi data dan fungsi-fungsi dalam suatu unit untuk suatu tujuan tertentu. Sebagai contoh "*class of dog*" adalah suatu unit yang terdiri atas definisi-definisi data dan fungsi-fungsi yang menunjuk pada berbagai macam *behavior* atau perilaku dari anjing. 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.

2. 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.
3. 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.
4. 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.
5. Polimorfisme (*Polimorphism*) - 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 pesan tersebut dikirim. Contohnya, bila sebuah burung menerima pesan "gerak cepat", dia akan menggerakan sayapnya dan terbang. Bila seekor serigala 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.

6. 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) *behavior* mereka tanpa harus mengimplementasi ulang behavior tersebut (bahasa berbasis-objek tidak selalu memiliki inheritas) [10].

2.5. UML (*Unified Modeling Language*)

UML (*Unified Modeling Language*) adalah bahasa spesifikasi standar untuk mendokumentasikan, menspesifikasikan, dan membangun system (Flowler,2006). *Unified Modeling Language* (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 adalah suatu bahasa yang digunakan untuk menentukan, memvisualisasikan, membangun, dan mendokumentasikan suatu sistem informasi. UML dikembangkan sebagai suatu alat untuk analisis dan desain berorientasi objek oleh Grady Booch, Jim Rumbaugh, dan Ivar Jacobson. Namun demikian UML dapat digunakan untuk memahami dan mendokumentasikan setiap sistem informasi. Penggunaan UML dalam industri terus meningkat. Ini merupakan standar terbuka yang menjadikannya sebagai bahasa pemodelan yang umum dalam industri peranti lunak dan pengembangan System [9] .

UML menyediakan 10 macam diagram untuk memodelkan aplikasi berorientasi objek, yaitu :

1. *Use Case Diagram* untuk memodelkan proses bisnis.
2. *Conceptual Diagram* untuk memodelkan 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 Cases* dan *objects* dalam Sistem.
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. Tools yang digunakan

2.6.1. Adobe Flash Pro CS5.5

Adobe Flash Profesional CS 5.5 merupakan salah satu program animasi 2D vektor yang sangat digemari oleh para animator. Dalam perkembangannya, program adobe flash pro melakukan banyak penyempurnaan pada setiap versinya. Program ini dapat menampilkan beberapa fitur baru yang membuat adobe flash semakin canggih untuk urusan animasi 2D berbasis vektor.

Berbagai aplikasi animasi 2D mulai dari animasi kartun, animasi interaktif, game, company profile, presentasi, video clip, animasi web dan aplikasi animasi lainnya sesuai kebutuhan dapat diolah. Dan kini flash didukung dengan kemampuannya dalam embuat animasi transformasi 3D, serta panel yang khusus untuk membuat animasi dengan ActionScript yang lebih mudah pemakaiannya.

2.6.2. Adobe Illustrator CS5.1

Adobe Illustrator merupakan aplikasi pengolah vektor yang handal dalam menciptakan bentuk-bentuk geometri dasar, seperti titik, garis, kurva dan juga polygon dapat di kreasikan secara menarik. Fungsi utama dari aplikasi ini adalah sebagai pengolah *vektor* namun tidak hanya itu, aplikasi ini juga dapat menggambar kartun dan juga membuat efek-efek untuk sebuah gambar.

2.6.3. Adobe Photoshop

Photoshop adalah sebuah software photo editing dan desain grafis yang dibuat oleh Adobe yang sudah diakui kehandalannya, karena fasilitas yang diberikan didalamnya tergolong lengkap dari mulai fitur mengedit foto, memanipulasi foto, serta menciptakan kreasi – kreasi grafis yang menarik .