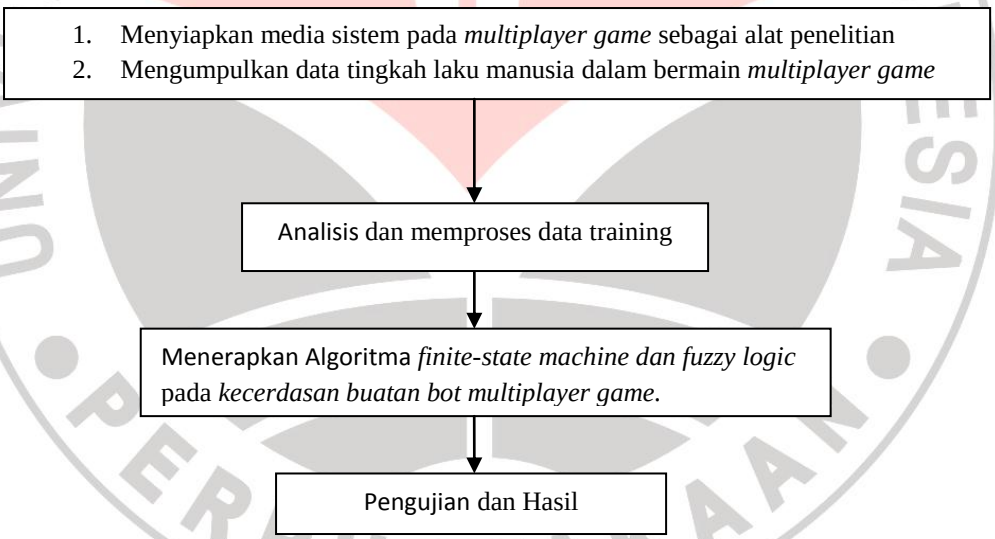
BAB III

DESIGN SISTEM

3.1 Desain Penelitian

Desain penelitian dibagi kedalam 4 tahap yang mencakup tahap pengumpulan data, tahap analisis dan memproses data, tahap implementasi algoritma serta tahap pengujian, seperti pada Gambar 3.1 berikut:



Gambar 3.1 Desain Penelitian

3.1.1 Tahap Pengumpulan Data

Sebelum melakukan pengumpulan data maka dilakukan penyiapan media sistem pada *multiplayer game* dengan membuat *map* bermain yang disesuaikan dengan kebutuhan sistem.

Setelah media sistem tersedia maka pengumpulan data tingkah laku manusia dalam bermain *multiplayer game* dilakukan dengan cara perekaman *input* yang dilakukan pada beberapa orang yang sedang bermain *multiplayer game.*

3.1.2 Tahap Analisis dan memproses data

Analisis dan proses data dilakukan dengan mengolah data tingkah laku manusia dalam bermain *multiplayer game* dan menerapkan algoritma *fuzzy logic* juga algoritma *finite-state machine* sehingga menghasilkan suatu tingkah laku manusia dalam bermain *multiplayer game* seperti pada tabel berikut:

Tabel 3.1 Tabel Tingkah Laku Tingkah Laku Manusia

Jarak / Kesehatan	Sangat Dekat	Dekat	Sedang	Jauh	Sangat Jauh
Penuh	-Menjauhi Musuh -Attack	-Attack	-Attack	-Mendekati Musuh	-Mendekati Musuh
Sangat Kuat	-Mendekati Musuh -Dark	-Attack -Dark	-Mendekati Musuh	-Mendekati Musuh	-Auto Dark Arrow

	Arrow -Menjauhi Musuh	Arrow	-Attack		-Silence
Kuat	-Menjauhi Musuh -Dark Arrow -Life Drain	-Mendekati Musuh -Attack	-Mendekati Musuh	-Mendekati Musuh	-Auto Dark Arrow
Sedang	-Menjauhi Musuh -Dark Arrow -Life Drain	-Mendekati Musuh -Attack	-Mendekati Musuh	-Silent -Stop	-Silence
Lemah	-Menjauhi Musuh -Dark Arrow -Life Drain	-Auto Dark Arrow -Dark Arrow	-Mendekati Musuh	-Stop	
Sangat Lemah		-Mendekati Musuh -Attack	-Dark Arrow -Life drain	-Stop	-Silence
Kritis		-Attack	-Mendekati Musuh -Dark arrow -life drain	-Stop	

Sample data training dapat dilihat pada “Lampiran I Data training”.

3.1.3 Tahap Implementasi algoritma

Penerapan algoritma *fuzzy logic* dan *finite-state machine* dilakukan dengan penggabungan *fuzzy logic* pada *finite-state machine* yang menghasilkan aturan-aturan yang digunakan pada finite-state machine dan menerapkan tingkah laku manusia sebagai aksi dari *finite-state machine* yang kemudian secara keseluruhan diterapkan pada *bot multiplayer game.*

3.1.4 Tahap Pengujian

Pengujian dilakukan pada beberapa skenario pertempuran dengan cara merekam *input* yang dihasilkan oleh sistem.

Hasil pengujian akan mencocokan tingkat kemiripan tingkah laku manusia dan tingkah laku sistem *bot* saat bermain *multiplayer game*.

3.2 Fokus Penelitian

Fokus penelitian pada skripsi ini adalah:

1. Tahapan-tahapan dalam pembuatan kecerdasan buatan *bot* pada *multiplayer game.*
2. Penerapan algoritma *finite-state machine dan fuzzy logic* dalam kecerdasan buatan *bot* pada *multiplayer game*.

3.3 Alat dan Bahan

Pada penelitian ini digunakan alat penelitian berupa perangkat keras dan perangkat lunak sebagai berikut:

1. Perangkat Keras:
    a. Processor Intel(R) Celeron(R) CPU 847 @ 1.1GHz 1.10 GHz
    b. RAM 4 GB
    c. *Hard Disk* 320 GB
    d. Intel(R) HD *Graphics* 1548 MB
    e. *Current display* 1366x768
    f. LAN kabel
2. Perangkat Lunak:
    a. Sistem Operasi Windows 7 Ultimate 32-bit
    b. Warcraft III Frozen Throne Ver 1.24.3.6384
    c. Warcraft *display* 1024x768
    d. AutoIt ver 3.3.8.1
3. Koneksi internet bila diperlukan

Bahan penelitian yang digunakan adalah data training tingkah laku manusia dan *bot* pada *multiplayer game.*

3.4 Metode Penelitan

3.4.1 Proses Pengumpulan Data

Pada penelitian ini pengumpulan data dan informasi akurat yang dapat menunjang proses penelitian. Berikut ini merupakan metode pengumpulan data yaitu

a. Eksplorasi dan Studi Literatur

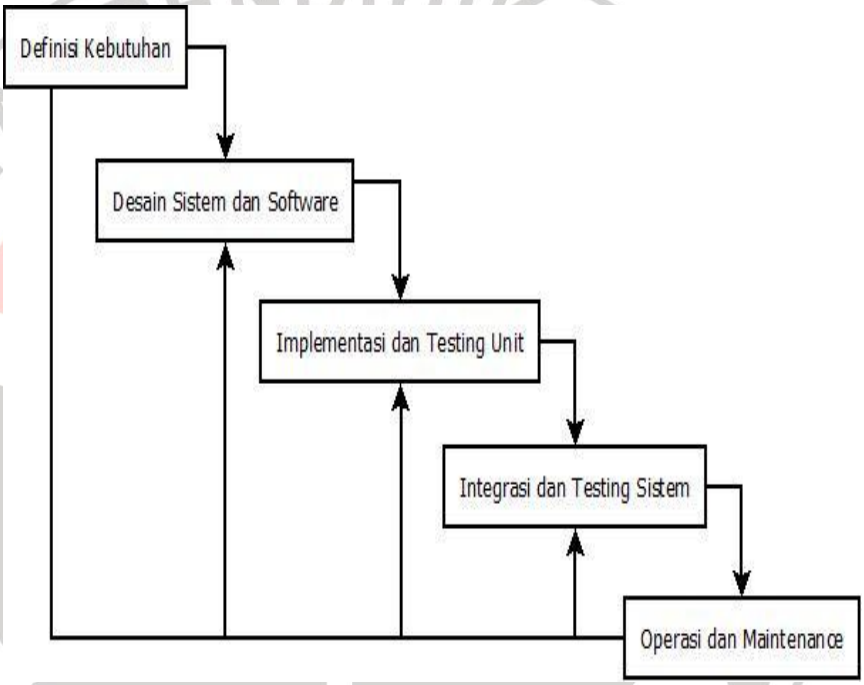
Eksplorasi dan studi literatur dilakukan dengan mempelajari kecerdasan buatan *bot*, algoritma *finite-state machine dan fuzzy logic*, bahasa pemograman au3, jurnal, karya ilmiah, paper dan sumber ilmiah lainnya yang didapat dari *internet*.

b. Observasi

Dengan melakukan pengamatan pada kemampuan yang dimiliki oleh *bot* pada *multiplayer game*.

3.4.2 Proses Rekayasa Sistem

Proses rekayasa sistem menggunakan model *waterfall/classical*. Tahap-tahapannya adalah analisis, desain, konstruksi dan implementasi serta operasional dan perawatan (Eri, 2011), seperti pada Gambar 3.2 berikut:



Gambar 3.2 Model *Waterfall*

a. Definisi kebutuhan

Dalam tahap ini, dilakukan pengumpulan data serta informasi yang kemudian dianalisis sehingga mendapatkan gambaran sistem yang tepat.

b. Desain Sistem dan *Software*

Dalam tahap ini, dirancang *automation bot* pada *multiplayer game* yang tepat untuk menerapkan algoritma *finite-state machine dan fuzzy logic*.

c. Implementasi dan testing *unit*

Dalam tahap ini, dibuat *automation bot* pada *multiplayer game* yang mampu menerapkan algoritma *finite-state machine dan fuzzy logic*.

d. Integrasi dan testing sistem

Dalam tahap ini dilakukan suatu pengujian terhadap kecerdasan *automation bot* yang telah dibuat pada *multiplayer game*.

e. Operasi dan *Maintenance*

Dalam tahap ini, dilakukan penyempurnaan kecerdasan buatan *automation bot* pada *multiplayer game* tersebut. Apabila ada yang kurang akan dilakukan perbaikan.

3.5 Perancangan Sistem

Perancangan sistem dibutuhkan untuk membantu proses pengembangan sistem. Pada perancangan sistem ini, akan diuraikan mengenai perancangan media tempat sistem diujikan dan elemen-elemen pembangun sistem.

3.5.1 Media Uji Sistem

Pada penelitian ini sistem yang dibuat diujikan pada sebuah *map* yang di *custom* dari *map* ryoko *hero tavern* yang merupakan bagian dari sebuah *multiplayer game* bernama Warcraft III Frozen Throne. Langkah-langkah yang dilakukan untuk membuat map *custom* tersebut adalah sebagai berikut:

3.5.1.1 *Terrain* Editor

Terrain yang digunakan adalah *terrain* jenis *sunken ruin* dimana lv0 *terrain* berada dibagian luar dengan *texture* berupa *shalow water* dan lv2 berada dibagian dalam dengan *texture ruined walls* pada bagian sisi dalam, *texture* grass pada area arena dan sebagian *texture dark grass* yang digunakan untuk membentuk tulisan “ILKOM” seperti yang terlihat pada gambar berikut:

Gambar 3.3 *Terrain Map*

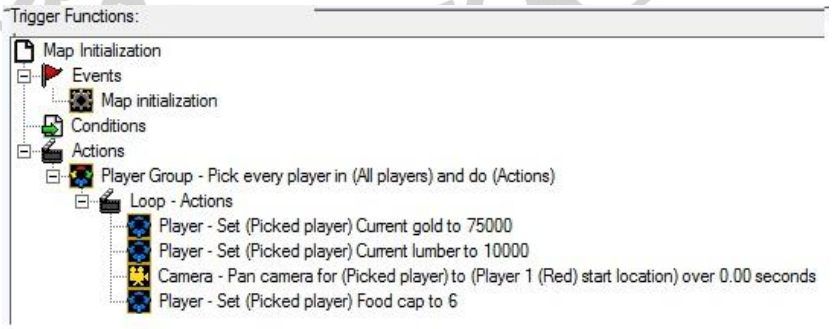
3.5.1.2 *Units* Editor

Pada pembuatan map ini *unit* yang digunakan adalah *unit* building dan *hero unit*, dimana *hero unit* akan diletakan secara langsung pada *map*. Untuk *unit building* atau bangunan yang digunakan adalah *unit zone indikator* yang berfungsi sebagai pengindikasi dari *area arena* dan *hero* dan yang terakhir digunakan adalah *unit start location* merah sebagain tempat munculnya *hero* merah dan *start location* biru sebagai tempat munculnya *hero* biru. Setelah dimasukannya *unit* yang diperlukan maka tampilan map akan seperti berikut:

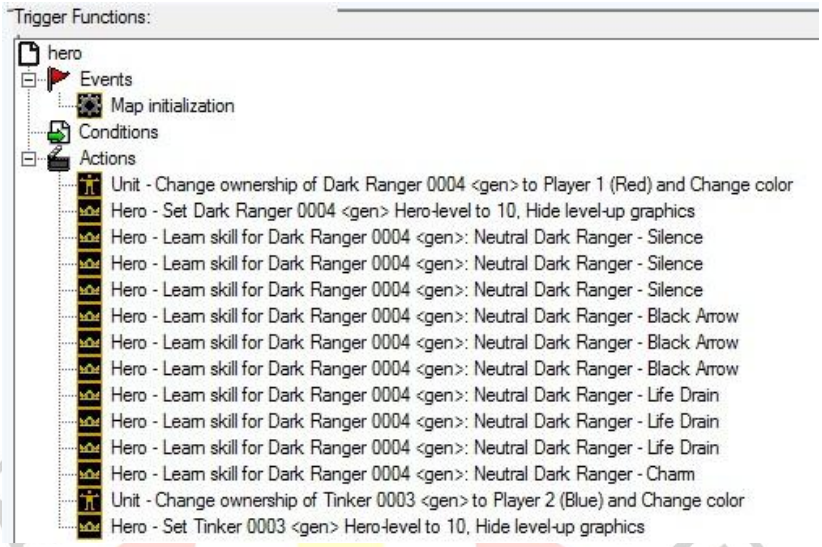
Gambar 3.4 *Unit* Pada *Map*

3.5.1.3 Triggering

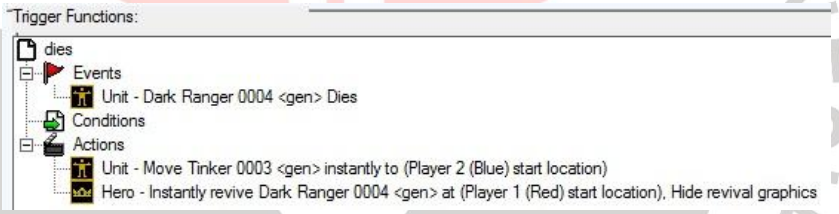
Trigerring yang diperlukan pada *map* antara lain adalah *trigger map initialization* yang berisi *trigger-trigger* yang berkaitan dengan inisiasi *map*, *trigger hero* yang berisi *trigger-trigger* yang berkaitan dengan inisiasi awal dari *hero*, *trigger hero dies* yang berisi *trigger* saat *hero* mati dan *trigger enemy dies* yang berisi *trigger* saat musuh mati, seperti yang tampak pada gambar-gambar berikut:



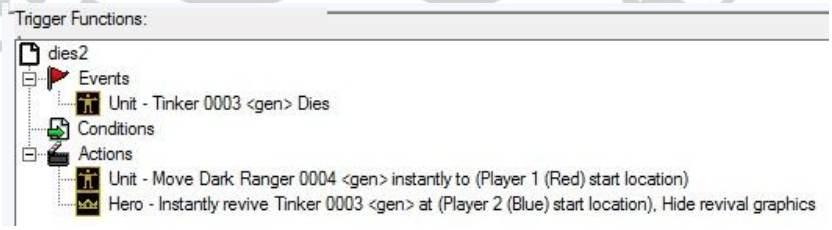
Gambar 3.5 *Trigger Map Initialization*



Gambar 3.6 *Trigger Hero*



Gambar 3.7 *Trigger Hero Dies*



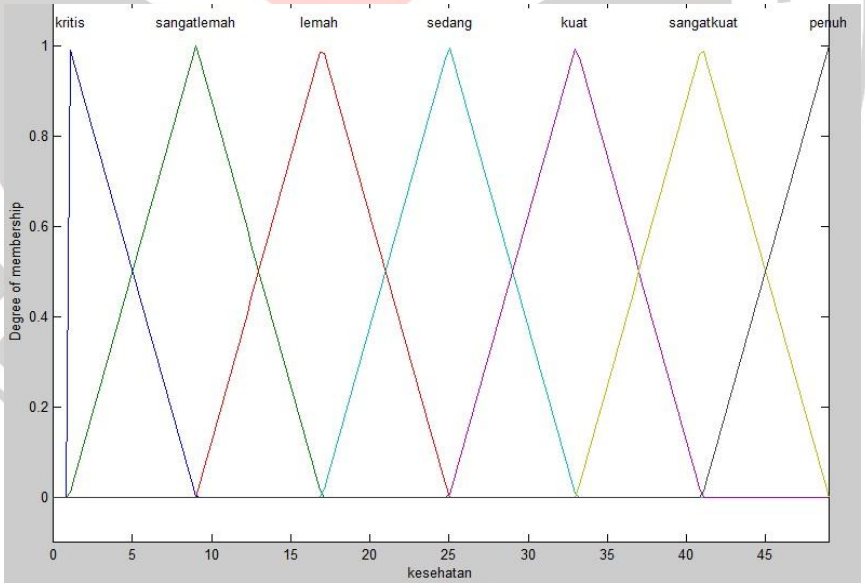
Gambar 3.8 *Trigger Enemy Dies*

3.5.2 Variabel *Fuzzy*

Variabel yang akan diproses menggunakan logika *fuzzy* terdiri dari 2 variabel, diantaranya adalah variabel kesehatan dan variabel jarak.

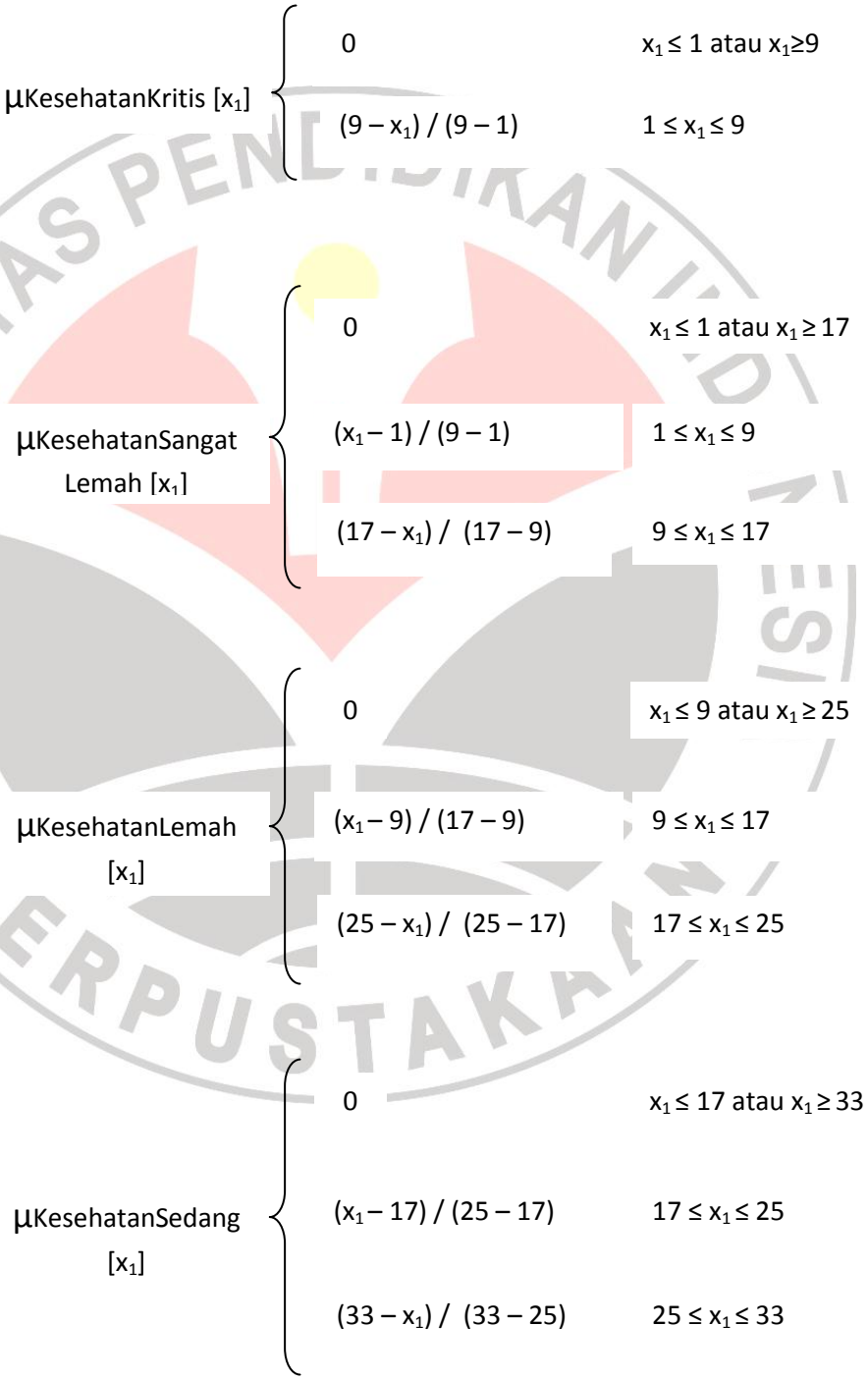
3.5.2.1 Variabel Kesehatan

Variabel kesehatan dibagi menjadi tujuh himpunan *fuzzy*, yaitu kritis, sangatlemah, lemah, sedang, kuat, sangatkuat dan penuh. Himpunan kritis, sangatlemah, lemah, sedang, kuat, sangatkuat dan penuh menggunakan pendekatan fungsi keanggotaan yang berbentuk segitiga, seperti tampak pada gambar berikut:



Gambar 3.9 Variabel Kesehatan

Fungsi keanggotaan pada variabel kesehatan dirumuskan sebagai berikut:



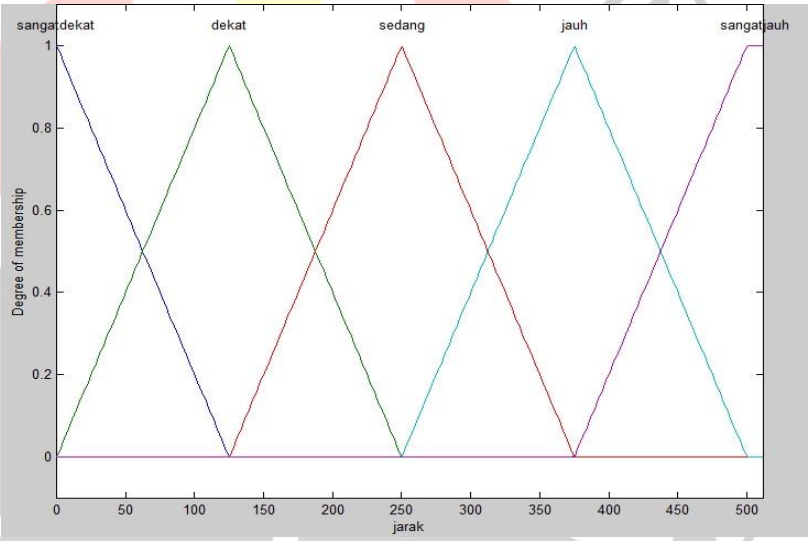
$$\mu KesehatanKuat[x_1] = \begin{cases} 0 & x_1 \le 25 \text{ atau } x_1 \ge 41 \\ (x_1 - 25) / (33 - 25) & 25 \le x_1 \le 33 \\ (41 - x_1) / (41 - 33) & 33 \le x_1 \le 41 \end{cases}$$

$$\mu KesehatanSangat\ Kuat[x_1] = \begin{cases} 0 & x_1 \le 33 \text{ atau } x_1 \ge 49 \\ (x_1 - 33) / (41 - 33) & 33 \le x_1 \le 41 \\ (49 - x_1) / (49 - 41) & 41 \le x_1 \le 49 \end{cases}$$

$$\mu KesehatanPenuh[x_1] = \begin{cases} 0 & x_1 \le 41 \text{ atau } x_1 \ge 49 \\ (x_1 - 41) / (49 - 41) & 41 \le x_1 \le 49 \end{cases}$$

3.5.2.2 Variabel Jarak

Variabel jarak dibagi menjadi lima himpunan *fuzzy*, yaitu sangatdekat, dekat, sedang, jauh dan sangatjauh. Himpunan sangatdekat, dekat, sedang dan jauh menggunakan pendekatan fungsi keanggotaan yang berbentuk segitiga sedangkan himpunan sangatjauh menggunakan pendekatan keanggotaan yang berbentuk bahu, seperti tampak pada gambar berikut:



Gambar 3.10 Variabel Jarak

Fungsi keanggotaan pada variabel jarak dirumuskan sebagai berikut:

$$\mu\ \text{JarakSangatDekat}[x_2] \begin{cases} 0 & X_2 \le 0 \text{ atau } x_2 \ge 125 \\ (125 - x_2) / (125 - 0) & 0 \le x_2 \le 125 \end{cases}$$

μ JarakDekat $[x_2]$

0	$x_2 \leq 0$ atau $x_1 \geq 250$
$(x_2 - 0) / (125 - 0)$	$0 \leq x_2 \leq 125$
$(250 - x_2)/(250 - 125)$	$125 \leq x_2 \leq 250$

μ JarakSedang $[x_2]$

0	$x_2 \leq 125$ atau $x_1 \geq 375$
$(x_2 - 125) / (250 - 125)$	$125 \leq x_2 \leq 250$
$(375 - x_2)/(375 - 250)$	$250 \leq x_2 \leq 375$

μ JarakJauh $[x_2]$

0	$x_2 \leq 250$ atau $x_1 \geq 500$
$(x_2 - 250) / (375 - 250)$	$250 \leq x_2 \leq 375$
$(500 - x_2)/(500 - 375)$	$375 \leq x_2 \leq 500$

μ JarakSangatJauh $[x_2]$

0	$x_2 \leq 375$ atau $x_1 \geq 512$
$(x_2 - 375) / (500 - 375)$	$375 \leq x_2 \leq 500$
1	$500 \leq x_2 \leq 512$

3.5.3 Automation and Scripting Language (AutoIt)

AutoIt v3 adalah *freeware* seperti bahasa *scripting BASIC* yang dirancang untuk mengotomatisasi *GUI Windows* dan *scripting* umumnya. Ini menggunakan kombinasi simulasi dari penekanan tombol, gerakan *mouse* dan *Windows*/kontrol manipulasi untuk mengotomatisasi tugas-tugas dengan cara yang tidak mungkin atau dapat diandalkan dengan bahasa lain (misalnya VBScript dan SendKeys). AutoIt juga sangat kecil, mandiri, dan akan berjalan pada semua versi *Windows out-of-the-box* tanpa memerlukan "*runtimes*" yang menjengkelkan.

AutoIt pada awalnya dirancang untuk *PC "roll out*" situasi yang mengandalkan pada otomatisasi dan mengkonfigurasi ribuan *PC*. Seiring waktu itu telah menjadi bahasa kuat yang mendukung ekspresi kompleks, fungsi pengguna, *loop* dan segala sesuatu yang *scripters* veteran harapkan.

Pada penelitian ini digunakan beberapa *command* dari autoit, diantaranya sebagai berikut:

- *Hotkeyset*

  Perintah yang digunakan untuk menentukan *hotkey* yang dapat digunakan ketika sistem berjalan.

- *Pixelgetcolor*

  Perintah yang digunakan untuk mendapatkan nilai warna pada koordinat *pixel* yang telah ditentukan.

- *Mousemove*

  Perintah yang digunakan untuk menggerakan *mouse* menuju koordinat *pixel* yang telah ditentukan.

- *Mouseclick*

  Perintah yang digunakan untuk melakukan klik pada *mouse* baik itu klik kanan maupun klik kiri

- *Sleep*

  Perintah yang digunakan untuk melakukan istirahat sistem selama waktu yang ditentukan dengan format 1 detik sama dengan 1000 nilai *sleep*.

- *Pixelsearch*

  Perintah yang digunakan untuk melalukan pencarian nilai warna sesuai dengan warna yang telah ditentukan dalam area koordinat *pixel* yang telah ditentukan pula.

- *Controlsend*

  Perintah yang digunakan untuk memberikan masukan yang telah ditentukan sesuai dengan masukan yang ada pada *keyboard*.

- *Tooltip*

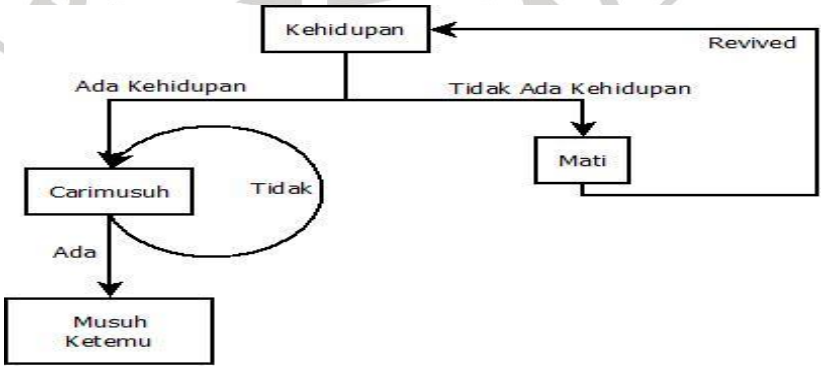
  Perintah yang digunakan untuk menampilkan status yang dialami sistem, misalnya dalam kondisi *pause* dan sebagainya.

3.5.4 FSM State

Pada penelitian ini digunakan beberapa *state FSM* yang akan dipilih secara otomatis sesuai dengan hasil yang telah diproses melalui logika *fuzzy*, seperti pada uraian berikut:

Berikut adalah FSM yang digunakan pada sistem:

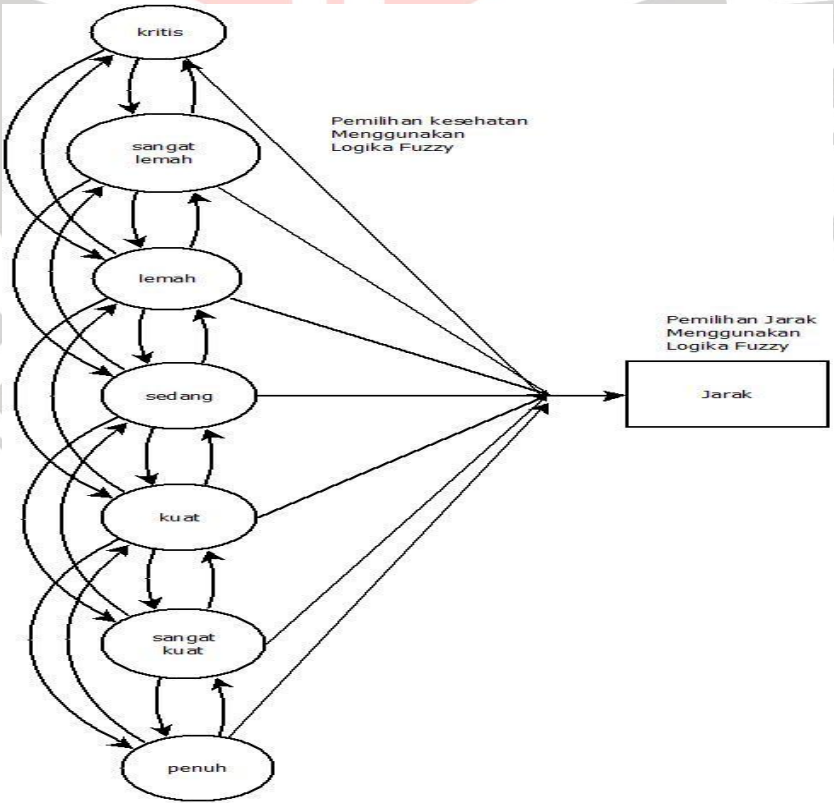
- FSM Kehidupan dan Cari Musuh



Gambar 3.11 State kehidupan dan cari musuh

State kehidupan berfungsi untuk memeriksa apakah *hero* memiliki kehidupan atau tidak, jika tidak memiliki kehidupan maka akan menuju *state* mati dimana *hero* akan dihidupkan kembali dan menuju *state* kehidupan. Namun jika *hero* memiliki kehidupan maka akan menuju *state* berikutnya yaitu *state* cari musuh, apabila pencarian berhasil menemukan musuh maka *state* akan berlanjut pada *state* musuh ketemu namun apabila tidak ditemukan *state* akan berulang mencari musuh sampai ditemukan.

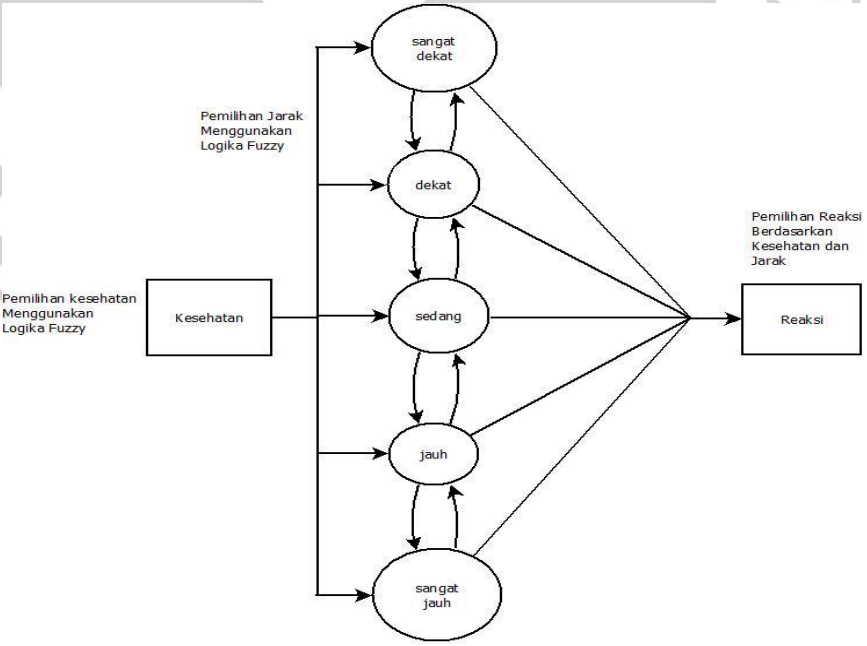
- FSM Kesehatan



Gambar 3.12 State Kesehatan

State kesehatan penuh merupakan kondisi dimana kesehatan *hero* berada pada tingkat penuh, jika kesehatan *hero* berada pada tingkat kesehatan sangat kuat maka *state* akan menuju sangat kuat, jika kesehatan *hero* berada pada tingkat kesehatan kuat maka *state* akan menuju kuat, jika kesehatan *hero* berada pada tingkat kesehatan sedang maka *state* akan menuju sedang, jika kesehatan *hero* berada pada tingkat lemah maka *state* akan menuju lemah, jika kesehatan *hero* berada pada tingkat sangat lemah maka *state* akan menuju sangat lemah dan jika kesehatan *hero* ada pada tingkat kritis maka *state* akan menuju kritis. Kemudian *state* kesehatan akan menuju *state* jaarak.

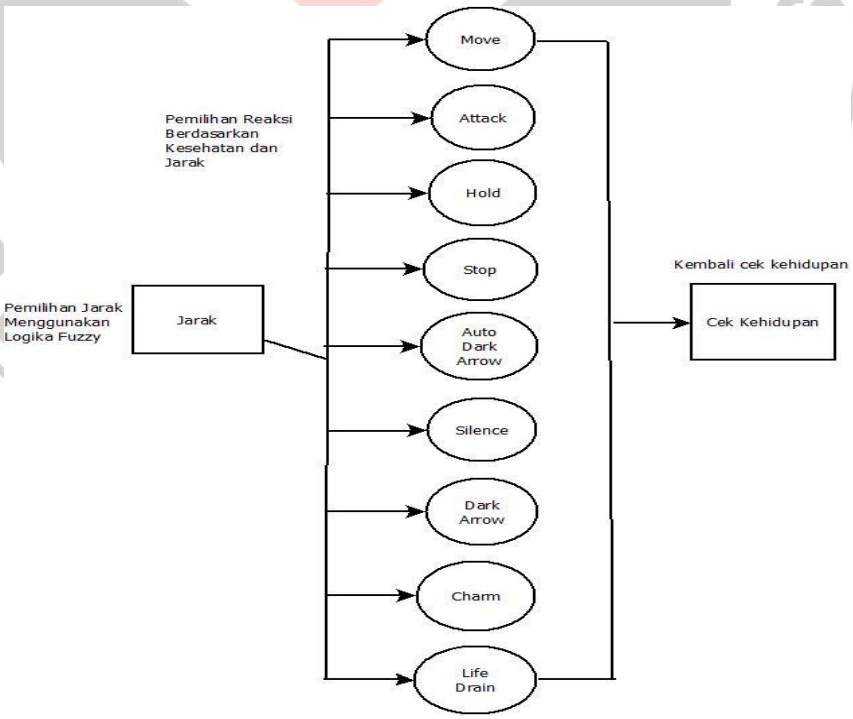
- FSM Jarak



Gambar 3.13 State Jarak

Setelah *state* kesehatan maka *state* akan menuju *state* jarak, dimana apabila jarak *hero* dengan musuh berada pada tingkat sangat jauh maka *state* akan berada pada *state* sangat jauh, jika jarak *hero* dengan musuh berada pada tingkat jauh maka *state* akan berada pada *state* jauh, jika jarak *hero* dan musuh berada pada tingkat sedang maka *state* akan berada pada *state* sedang, jika jarak *hero* dan musuh berada pada tingkat dekat maka *state* akan berada pada *state* dekat dan jika *state* berada pada tingkat sangat dekat maka *state* akan berada pada *state* sangat dekat. Kemudian *state* akan menuju *state* reaksi.

- FSM Reaksi



Gambar 3.14 State Reaksi

Setelah *state* jarak terlewati maka *state* akan menuju *state* reaksi, dimana *state attack* akan membuat *hero* melakukan *attack*, *state move* akan membuat *hero* melakukan pergerakan, *state hold* akan membuat *hero* menahan posisi, *state stop* akan membuat *hero* berhenti dari aktifitas, *state auto dark arrow* akan membuat *hero* mengaktifkan otomatisasi *dark arrow*, *state dark arrow* akan membuat *hero* menggunakan *skill dark arrow*, *state silence* akan membuat *hero* menggunakan *skill silence*, *state life drain* akan membuat *hero* menggunakan *skill life drain* dan *state charm* akan membuat *hero* menggukanan *skill charm*.