INTEGRASI *HIERARCHY FINITE STATE MACHINE* DAN LOGIKA *FUZZY* UNTUK DESAIN STRATEGI *NPC GAME*

Yunifa Miftachul Arif [1)], **Mochamad Hariadi** [2)] , **Supeno Mardi S. N** [3)]
[1)] Pasca Sarjana Teknik Elektro ITS, Surabaya 60111
[2)] Jurusan Teknik Elektro, ITS Surabaya 60111
[3)] Jurusan Teknik Elektro, ITS Surabaya 60111
yunif4@gmail.com [1)]; mochar@ee.its.ac.id [2)] ; mardi@ee.its.ac.id [3)]

Abstrak - *Tujuan pengembangan kecerdasan buatan adalah untuk membuat aksi dan reaksi otonom agen atau NPC (Non-Player Character) dari game. Dua NPC bisa saling membantu dalam menjalankan strategi menyerang terhadap musuh. Penelitian ini menjelaskan tentang bagaimana membuat strategi menyerang yang dilakukan oleh NPC dengan menggunakan Hierarchy Finite State Machine untuk mendesain perilaku . Dua NPC yang dimaksud adalah NPC Scout yang bertugas memancing serangan musuh, dan NPC Sniper yang bertugas memberikan back up serangan dari jarak jauh.. Selanjutnya digunakan logika fuzzy untuk menentukan respon perilaku terhadap kondisi yang dihadapi. Perilaku yangdimaksud adalah menyerang brutal, menyerang, bertahan dan melarikan diri. Masing-masing perilaku diujicobakan dalam game First Person Shooter menggunakan Torque Game Engine. Dalam simulasi game terjadi respon perubahan perilaku masing-masing NPC terhadap kondisi yang dihadapi. Strategi menyerang dalam penelitian ini mempunyai tingkat kemenangan hingga 80% ketika diujicobakan dengan musuh yang mempunyai perilaku umum yaitu menyerang dan menghindari tembakan.*

Kata kunci: NPC, strategi menyerang, HFSM, logika *fuzzy*.

1. PENDAHULUAN

Penelitian tentang AI (*Artificial Intelligence*) pada NPC (*Non-Player Character*) dalam game, hingga saat ini masih terus di kembangkan. AI tersebut di kembangkan untuk merancang perilaku NPC [8]. Ketika kita mengatakan bahwa game sudah mempunyai AI yang baik, berarti bahwa karakter permainan menunjukkan perilaku yang konsisten dan realistis, bereaksi dengan tepat kepada tindakan pemain dan karakter lain. AI pada game FPS umumnya terdiri dari perencanaan *path*, mengambil *item*, menggunakan *item*, dan berperang. Khusus untuk berperang NPC juga diharapkan mempunyai strategi-strategi khusus seperti halnya manusia [9]. Strategi yang dimaksud bisa berupa strategi mengejar lawan, menyerang lawan maupun menghindari lawan.

Model strategi menyerang bisa bermacam-macam, misalnya strategi menyerang dengan memancing serangan musuh, kemudian pada kondisi tertentu perilaku berubah menjadi menghindar. Jika NPC bergerak menghindar ke arah tim, maka NPC akan mendapatkan bantuan serangan dari rekan satu tim. Bagaimanapun model strategi menyerang, umumnya mempunyai tujuan akhir mengalahkan musuh. Sehingga pada penelitian ini akan dibahas mengenai model strategi menyerang pada game FPS, dimana untuk mendesain perilaku NPC digunakan *Hierarchy Finite State Machine*, dan untuk menentukan respon perilaku yang dilakukan terhadap perubahan kondisi yang dihadapi NPC digunakan logika *fuzzy*.

Dengan HFSM *(Hierarchy Finite State Machine)*, state dapat lebih disempurnakan ke FSM lain. Dimana pada tingkat dasar,

hierarki menambahkan sesuatu pada model komputasinya, dengan tidak mengurangi jumlah *state*. Tetapi secara signifikan dapat mengurangi jumlah transisi dan membuat FSM lebih intuitif dan mudah dimengerti, sehingga lebih mudah diterjemahkan ke dalam bahasa pemrograman.

Implementasi dari penelitian ini adalah untuk game ber-*genre* FPS dimana NPC musuh akan berinteraksi langsung dengan karakter pemain. *Game engine* yang digunakan untuk menguji perilaku NPC adalah *Torque Game Engine 1.4*, yang merupakan *Game Engine 3D*. *Game Engine* ini dipasarkan oleh *Garege Games*, dengan ciri pemrograman setiap elemen game terdiri dari *class-class* [6].

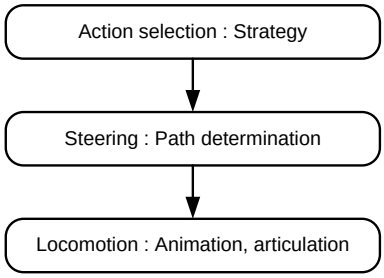
2. TEORI PENUNJANG

Penelitian yang dilakukan ini didasarkan pada beberapa penelitian lain yang telah dilakukan sebelumnya. Untuk memberikan gambaran secara umum, pada bab 2 ini akan dibahas secara singkat mengenai *game*, NPC, *artificial intelegence game*, *Finite State Machine*, logika *fuzzy* dan teori penunjang lain yang berkaitan dengan penelitian ini.

2.1 Non-Player Character

Autonomous character adalah jenis *otonomous* agent yang ditujukan untuk penggunaan komputer animasi dan media interaktif seperti games dan *virtual reality*. Agen ini mewakili tokoh dalam cerita atau permainan dan memiliki kemampuan untuk improvisasi tindakan mereka. Ini adalah kebalikan dari seorang tokoh dalam sebuah film animasi, yang tindakannya ditulis di muka, dan untuk "avatar" dalam sebuah permainan atau virtual reality, tindakan yang diarahkan secara real time oleh pemain. Dalam permainan, karakter otonom biasanya disebut NPC (*Non-Player Character)*.

Perilaku karakter yang otonom dapat lebih baik dipahami dengan membaginya menjadi beberapa lapisan. Lapisan ini dimaksudkan hanya untuk kejelasan dan kekhususan dalam diskusi yang akan mengikuti. Gambar 1 menunjukkan sebuah divisi gerak perilaku otonom hirarki karakter menjadi tiga lapisan: seleksi tindakan, *steering*, dan penggerak.

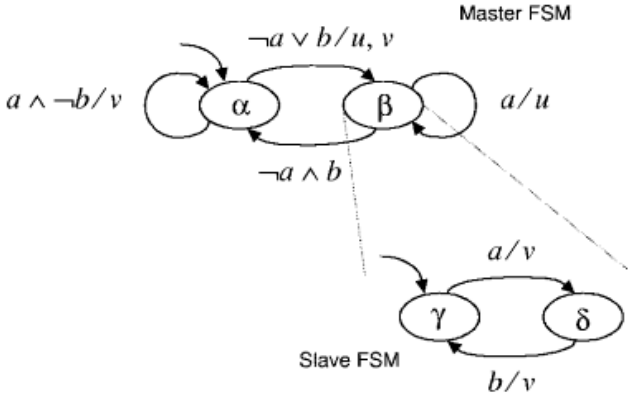


Gambar 1. Hirarki gerak perilaku

Tiga lapisan hirarki tersebut, adalah motivasi, tugas, dan motor. [5]. Pada penelitian ini lebih di fokuskan pada bagian *action selection* yang di dalamnya berisi strategi pergerakan NPC.

2.2 Hierarchy Finite State Machine

Pada *basic* FSM, masing-masing state tersusun lebih sederhana dan berurutan, tetapi memiliki banyak kelemahan karena sistem yang paling praktis memiliki jumlah *state* dan transisi yang banyak sehingga representasi dan analisis menjadi sulit. Salah satu solusi untuk masalah ini adalah hirarki. Dalam *Hierarchy* FSM, *state* dapat lebih disempurnakan ke dalam bentuk FSM lain. FSM lain yang dimaksud disebut FSM *slave* dan FSM di luar disebut *master* dalam suatu komposisi seperti diilustrasikan pada Gambar 2.



Gambar 2. *Hierarchy Finite State Machine*

Pada *level* dasar, *hierarchy* tidak menambahkan model komputasi, juga tidak mengurangi jumlah *state*. Tetapi dapat secara signifikan mengurangi jumlah transisi sehingga membuat FSM menjadi lebih intuitif dan mudah dimengerti. Transisi dari α dan β dalam gambar 1 adalah berupa *compact notation* yang sederhana untuk transisi dari γ ke α dan β ke δ. Ruang *state* pada *basic* FSM bernilai Q = {α,γ,δ}.

Alfabet input untuk *slave* FSM adalah himpunan bagian dari masukan alfabet dari FSM *master*. Sinyal input untuk *slave* FSM adalah *subset input* sinyal untuk *master*. Demikian pula, output sinyal dari *slave* FSM adalah *subset* dari sinyal output dari *master*.

Hirarki semantik menentukan bagaimana FSM *slave* bereaksi terhadap reaksi FSM *master*. Semantik mendefinisikan satu reaksi dari *hierarchy* FSM adalah sebagai berikut: jika *state* tersebut tidak didefinisikan lagi, *hierarchy* FSM akan berperilaku seperti pada *basic* FSM. Jika kondisi *state* adalah terdefinikan, maka yang pertama sesuai adalah *slave* FSM, dan kemudian FSM *master*. Jadi, ketika dua transisi dipicu, maka dua tindakan yang diambil, dimana keedua tindakan tersebut harus entah digabung menjadi satu.

Trace dari aksi hierarchy FSM yang mungkin terjadi dapat dilihat pada gambar 3. Dalam contoh gambar 3 dijelaskan juga bahwa dalam *state* β dan *substate* γ dan sinyal input a dalam kondisi *present*, aksi yang memicu *slave* FSM adalah "*v*" dan aksi yang memicu *master* FSM adalah "*u*". Output dari *hierarchy* FSM tersebut adalah *uv*, dimana output sinyal *uv* dalam kondisi *present* [2].

Pada gambar 2, *hierarchy* FSM hanya terdiri dari dua tingkatan. Namun, *slave* FSM dapat benar-benar menjadi *hierarchy* FSM yang lain. Dengan *hierarchy* FSM benar-benar mendorong setiap FSM yang ada di dalamnya. Modularitas yang merupakan ciri FSM tidak mengurangi kompleksitas dari desain, yang pada gilirannya mempermudah proses pembuatannya [1].

Current State	α	α	β,γ	β,δ	α	…
a	*present*	*absent*	*present*	*absent*	*absent*	…
b	*absent*	*absent*	*absent*	*present*	*absent*	…
Next State	α	β,γ	β,δ	α	β,γ	…
u	*absent*	*present*	*present*	*absent*	*present*	…
v	*present*	*present*	*present*	*present*	*present*	…

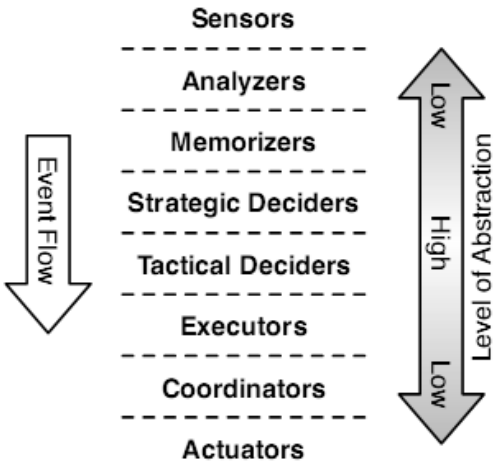
Gambar 3. *Trace* dari *Hierarchy* FSM

2.3 Modeling AI Game

Arsitektur model AI digambarkan dalam Gambar 4. Pada level pertama mengandung komponen yang mewakili sensor yang memungkinkan karakter untuk mengamati lingkungan serta state sendiri. *Sensors* menyaring informasi dan peristiwa serta mengirimnya ke tingkat berikutnya. Tingkat kedua berisi komponen *analyzers* yang menganalisis atau menghubungkan kejadian dari individu sensor, yang mungkin mengarah pada peristiwa generasi selanjutnya. Komponen memorizer bertugas menyimpan peristiwa yang telah terjadi.

Strategic deciders adalah komponen yang secara konseptual di tingkat tertinggi abstraksi. Mereka harus memutuskan strategi untuk karakter yang didasarkan pada kondisi saat ini dan memori. Pada tingkat berikutnya, yang *tactic deciders* merencanakan bagaimana membuat strategi yang dipakai sekarang dapat berjalan dengan baik. Executors atau pelaksana kemudian menerjemahkan keputusan dari *tactical deciders* untuk perintah tingkat rendah (*low-level commands*) sesuai dengan batasan yang digunakan oleh permainan atau simulasi. Komponen *coordinators* memahami hubungan antar-aktuator dan mungkin kembali memberikan perintah tingkat

rendah lebih lanjut. Akhirnya, *actuators* melakukan tindakan yang diinginkan [14]. Pada penelitian ini akan dibahas lebih banyak mengenai komponen *strategic deciders dan tactical deciders.*



Gambar 4. *AI model Architecture*

2.4 Logika *Fuzzy*

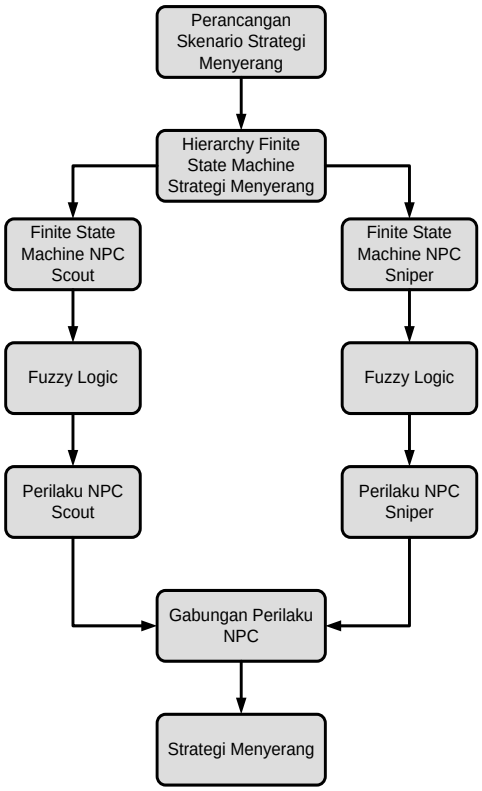
Logika *Fuzzy* adalah sebuah metode untuk menangani masalah ketidakpastian. Yang dimaksud dengan ketidakpastian yaitu suatu masalah yang mengandung keraguan, ketidaktepatan, kurang lengkapnya informasi, dan nilai kebenarannya bersifat sebagian. Ide tentang logika *Fuzzy* sebenarnya telah lama dipikirkan, yaitu semenjak jaman filsuf Yunani kuno. Dalam hal ini Plato adalah filsuf pertama yang meletakkan pondasi dasar dari logika *Fuzzy*. Plato menyatakan bahwa ada area ketiga selain benar dan salah. Terdapat banyak model aturan *Fuzzy* yang bisa digunakan dalam proses *inference* akan tetapi ada dua model aturan yang paling sering digunakan yaitu :

1. Model Mamdani
   Bentuk aturan yang digunakan pada model Mamdani adalah sebagai berikut:
   IF x_1 is A_1 AND … AND x_n is A_n THEN y is B ……(2.4)
   Dimana A_1,…, A_n, B adalah nilai-nilai linguistik, sedangkan " x_1 is A_1 " menyatakan bahwa nilai dari variabel x_1 adalah anggota himpunan fuzzy A.
2. Model Sugeno
   Model Sugeno merupakan varian dari model Mamdani dan memiliki bentuk aturan sebagai berikut :
   IF x_1 is A_1 AND … AND x_n is A_n THEN $y = f(x_1 , …. , x_n)$ ……(2.5)
   Dimana f bisa berupa sembarang fungsi dari variabel-variabel masukan yang nilainya berada dalam interval variabel keluaran.

Dari penjelasan tentang logika *Fuzzy* dapat diketahui bahawa suatu sistem yang menggunakan logika *Fuzzy* mampu menangani suatu masalah ketidakpastian dimana masukan yang diperoleh merupakan suatu nilai yang kebenarannya bersifat sebagian. Atas dasar itulah logika *Fuzzy* digunakan pada penelitian ini, dengan tujuan untuk mendapatkan respon perilaku NPC berdasarkan variabel input yang dimiliki.

3. METODE PENELITIAN

Metode yang digunakan untuk mendesain strategi menyerang pada penelitian ini adalah menggunakan *Hierarchy Finite State Machine,*sedangkan untuk menentukan perilaku NPC berdasarkan variable yang dimilikinya digunakan logika *fuzzy*. Gambar 5 menunjukkan tahapan-tahapan dalam penelitian yang meliputi perancangan skenario menyerang bertahan dan melarikan diri, perancangan *Hierarchy Finite State Machine, Finite State Machine* NPC *Scout*, *Finite State Machine* NPC *Sniper*, dan logika *fuzzy*.

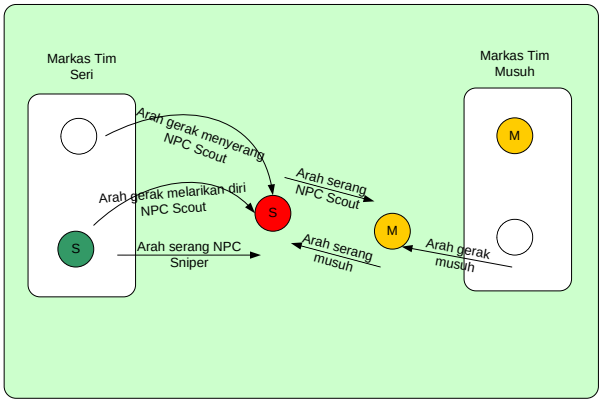


Gambar 5. Blok Diagram Metode Penelitian

3.1 Skenario Strategi Gerak Menghindar

Pada penelitian ini dibuat sebuah sekenario game perang menggunakan karakter tank. Ada dua tim yang terlibat peperangan dalam game ini, yaitu tim seri dan tim musuh. Tim seri yang dimaksud adalah NPC *Scout* dan NPC *Sniper*. Tim seri yang merupakan objek dalam penelitiaan ini mempunyai beberapa perilaku dalam melakukan strategi penyerangan. Perilaku yang dimiliki NPC *Scout* adalah menyerang brutal, menyerang, melarikan diri dan bertahan. NPC *Scout* cenderung lebih banyak menyerang, karena tugas utamanya adalah memancing serangan musuh. Sedangkan perilaku yang dimiliki NPC *Sniper* adalah menyerang, bertahan dan melarikan diri. Fungsi utama dari NPC *Sniper* adalah melakukan serangan jarak jauh sebagai salah satu bentuk *back up* serangan yang dilakukan oleh NPC *Scout*.

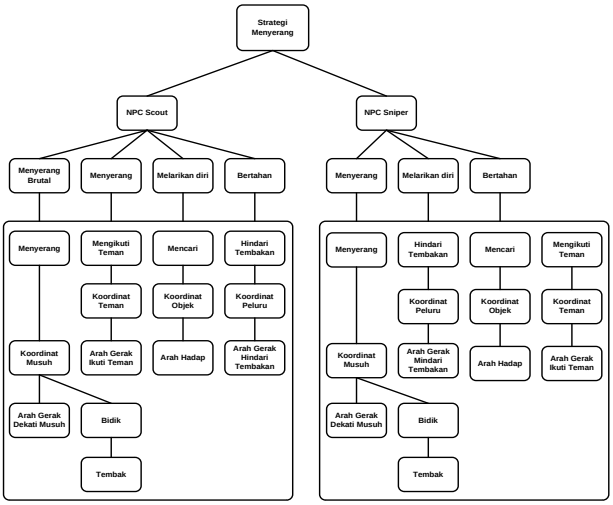
Gambaran *scenario* masing-masing NPC dapat dilihat pada Gambar 6.



Gambar 6. Skenario strategi penyerangan

3.1 *Hierarchy Finite Machine* Strategi Menyerang

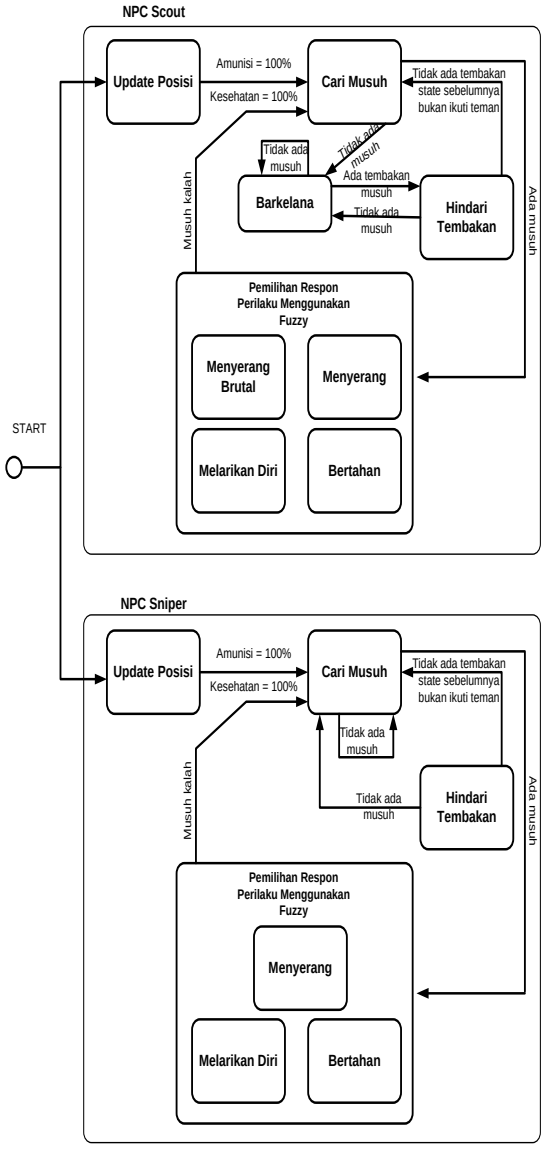
Hierarchy Finite State Machine untuk strategi menyerang dibuat berdasarkan skenario strategi. Secara garis besar hirarki yang dimaksud terbagi menjadi dua bagian, yaitu hirarki untuk NPC *Scout* dan hirarki untuk NPC *Sniper*, seperti dijelaskan pada Gambar 7.



Gambar 7. *Hierarchy Finite Machine* gerak menghindar

3.2 Top Level *Finite State Machine*

Top level finite state machine untuk gerak menghindar pada penelitian ini merupakan gabungan antara FSM NPC Angler dan FSM NPC Support, sebagaimana di jelaskan pada gambar 8.



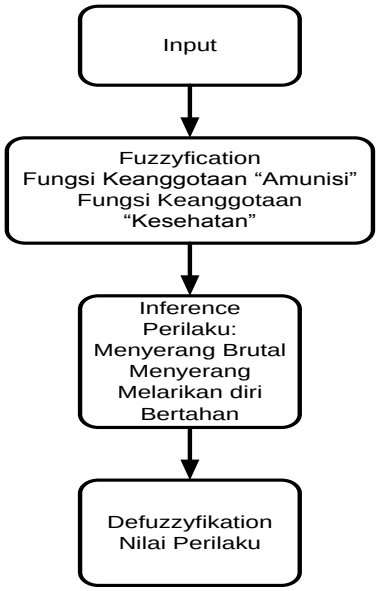
Gambar 8. *Top level finite state machine* strategi menyerang

3.3 Logika *Fuzzy*

Logika *fuzzy* dalam penelitian ini digunakan untuk menentukan variasi perilaku yang dilakukan oleh NPC, baik NPC *Scout* maupun NPC *Sniper*. Dengan adanya logika *fuzzy* tersebut masing-masing NPC dapat merespon perubahan variabel input menjadi perilaku yang sudah di desain menggunakan HFSM. Metode *fuzzy* yang digunakan adalah metode Sugeno, karena metode ini menghasilkan output yang berupa konstanta tegas, sehingga dapat mewakili nilai perilaku yang sudah didesain sebelumnya.

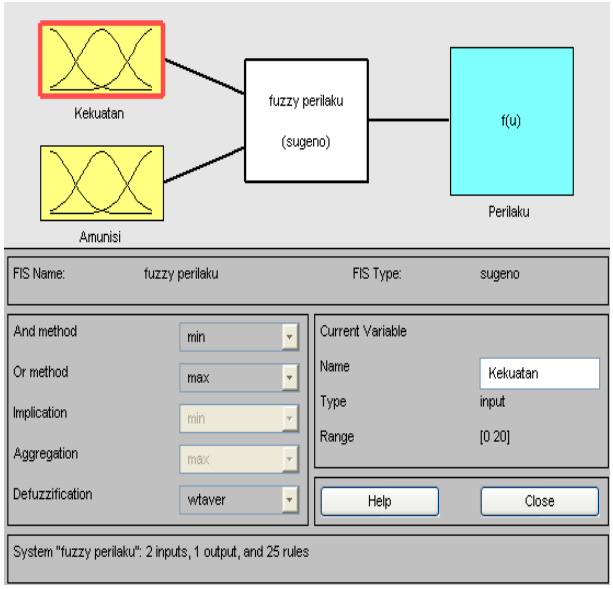
3.3.1 Desain *Fuzzy* NPC *Scout*

Untuk menghasilkan perilaku pada NPC *Scout* ada 2 variabel yang digunakan, yaitu, jumlah amunisi (sangat sedikit, sedikit, sedang, banyak, sangat banyak) dan kesehatan (sangat lemah, lemah, sedang, kuat, sangat kuat). Dengan menggunakan 2 variabel diharapkan dapat menentukan variasi perilaku yang akan dilakukan.Untuk menjadi otonom maka digunakan aturan sebab akibat antara perilaku dengan atribut variabel yang menempel pada NPC. Misalnya ketika amunisi sedikit dan kesehatan lemah maka NPC cenderung melakukan perilaku melarikan diri sesuai dengan hasil de*fuzzy*fikasi.



Gambar 9. Logika *fuzzy* untuk menghasilkan perilaku

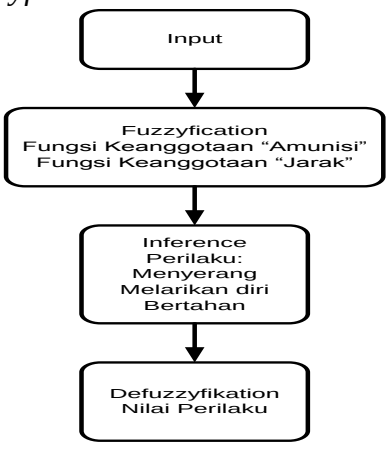
Gambar 9 menunjukkan bahwa dalam penelitian ini, dua atribut yang diberikan, yaitu amunisi dan kesehatan dalam logika *fuzzy* masing-masing menggunakan gabungan gabungan fungsi keanggotaan segitiga dan trapezium. Desain *fuzzy* untuk menghasilkan perilaku NPC *Scout* dapat dilihat pada Gambar 10.



Gambar 10. Desain *fuzzy* untuk menghasilkan perilaku NPC *Scout*

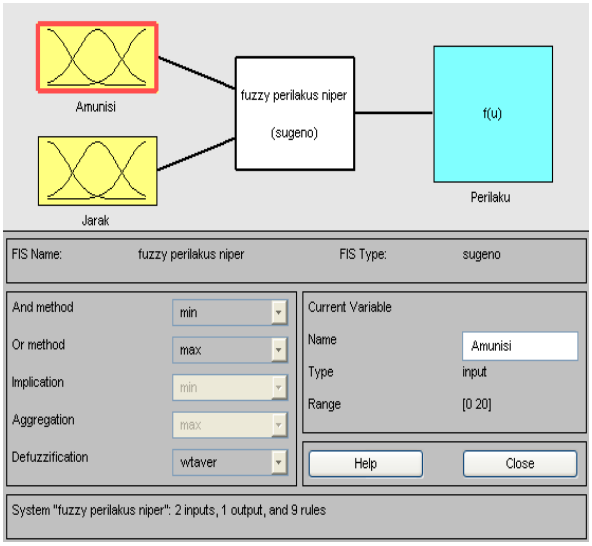
3.3.2 Desain *Fuzzy* NPC *Sniper*

Untuk menghasilkan perilaku pada NPC *Sniper* ada 2 variabel yang digunakan, yaitu, jumlah amunisi (sedikit, sedang, banyak) dan jarak musuh (dekat, sedang, jauh). Dengan menggunakan 2 variabel diharapkan dapat menentukan variasi perilaku yang akan dilakukan.Untuk menjadi otonom maka digunakan aturan sebab akibat antara perilaku dengan atribut variabel yang menempel pada NPC. Misalnya ketika amunisi banyak dan jarak musuh jauh maka NPC cenderung melakukan perilaku menyerang sesuai dengan hasil *defuzzyfikasi*.



Gambar 11. Logika *fuzzy* untuk menghasilkan perilaku

Gambar 11 menunjukkan bahwa dalam penelitian ini, dua atribut yang diberikan, yaitu amunisi dan jarak dimana dalam logika *fuzzy* masing-masing menggunakan gabungan fungsi keanggotaan segitiga dan trapezium. Desain *fuzzy* untuk menghasilkan perilaku NPC *Sniper* dapat dilihat pada Gambar 12.



Gambar 12. Desain *fuzzy* untuk menghasilkan perilaku NPC *Sniper*

4. HASIL DAN PEMBAHASAN

4.1 Pengujian Sistem

Pengujian diawali dengan pengujian logika *fuzzy* yang digunakan untuk menentukan variasi perilaku masing-masing NPC yang sudah dibangun menggunakan HFSM berdasarkan variable input yang dimiliki. Pengujian logika *fuzzy* yang pertama adalah pengujian *fuzzy* perilaku NPC *Scout*, dan yang kedua adalah perilaku NPC *Sniper*. Selanjutnya hasil system *fuzzy* yang diperoleh, diujicobakan pada game menggunakan Torque Game Engine.

4.2 Pengujian *Fuzzy* Perilaku NPC *Scout*

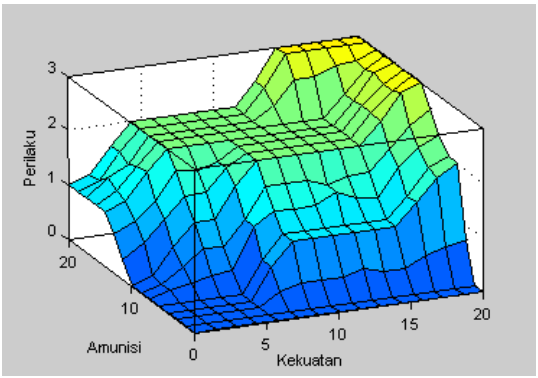
Sesuai variable yang diberikan untuk menghasilkan perubahan perilaku NPC *Scout*, maka beberapa parameter diujikan untuk mengetahui variasi perilaku yang

dihasilkan. Parameter yang diuji diantaranya adalah beberapa nilai variable amunisi mulai dari minimum hingga maksimum dan juga beberapa nilai variable kesehatan mulai dari minimum hingga maksimum.

Tabel 1. Hasil pengujian perilaku NPC *Scout* dengan menggunakan variable parameter masukan yang berbeda.

KEKUATAN \ AMUNISI	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0.25	0.5	0.75	1	1	1	1	1	1	1
1	0	0	0	0	0	0	0	0	0	0	0	0.25	0.5	0.75	1	1	1	1	1	1	1
2	0	0	0	0	0	0	0	0.17	0.17	0.22	0.33	0.72	0.9	1.06	1.33	1.33	1.4	1.5	1.33	1.33	1.33
3	0	0	0	0	0	0	0	0.17	0.17	0.44	0.67	0.94	1.1	1.28	1.67	1.67	1.6	1.5	1.67	1.67	1.67
4	0	0	0	0	0	0	0	0.25	0.25	0.75	1	1.25	1.5	1.75	2	2	2	2	2	2	2
5	0	0	0	0	0	0	0	0.25	0.25	0.75	1	1.25	1.5	1.75	2	2	2	2	2	2	2
6	0	0	0	0	0	0	0	0.33	0.5	0.67	1	1.33	1.5	1.67	2	2	2	2	2	2	2
7	0	0	0	0.25	0.5	0.5	0.5	1	1	1	1.5	1.75	1.75	1.75	2	2	2	2	2	2	2
8	0	0	0	0.43	1	1	1	1.33	1.5	1.67	2	2	2	2	2	2	2	2	2	2	2
9	0	0	0	0.43	1	1	1	1.25	1.5	1.75	2	2	2	2	2	2	2	2	2	2	2
10	0	0	0	0.43	1	1	1	1.25	1.5	1.75	2	2	2	2	2	2	2	2	2	2	2
11	0	0	0	0.43	1	1	1	1.25	1.5	1.75	2	2	2	2	2	2	2	2	2	2	2
12	0	0	0	0.43	1	1	1	1.33	1.5	1.67	2	2	2	2	2	2	2	2	2	2	2
13	0	0	0	0.5	1	1	1	1.5	1.5	1.5	2	2	2	2	2	2	2	2.25	2.5	2.5	2.5
14	0	0	0	0.43	1	1	1	1.33	1.5	1.67	2	2	2	2	2	2	2	2.57	3	3	3
15	0	0	0	0.43	1	1	1	1.25	1.5	1.75	2	2	2	2	2	2	2	2.57	3	3	3
16	0	0	0	0.43	1	1	1	1.25	1.5	1.75	2	2	2	2	2	2	2	2.57	3	3	3
17	0	0	0	0.64	1.4	1.33	1.33	1.72	1.9	2.06	2.33	2.39	2.4	2.39	2.33	2.33	2.4	2.79	3	3	3
18	0	0	0	0.64	1.6	1.67	1.94	1.94	2.1	2.28	2.67	2.61	2.6	2.61	2.67	2.67	2.6	2.79	3	3	3
19	0	0	0	0.86	2	2	2	2.25	2.5	2.75	3	3	3	3	3	3	3	3	3	3	3
20	0	0	0	0.86	2	2	2	2.25	2.5	2.75	3	3	3	3	3	3	3	3	3	3	3

Dari variasi variable masukan yang digambarkan pada Tabel 1 dapat diperoleh keluaran perilaku yang variatif, dimana selanjutnya dikelompokkan menjadi 4 model perilaku yaitu bertahan, melarikan diri, menyerang dan menyerang brutal. Respon keluaran *fuzzy* perilaku NPC *Scout* terhadap variasi masing – masing input direpresentasikan oleh grafik tiga dimensi pada Gambar 13.



Gambar 13. Respon *fuzzy* perilaku NPC *Scout* dalam grafik permukaan

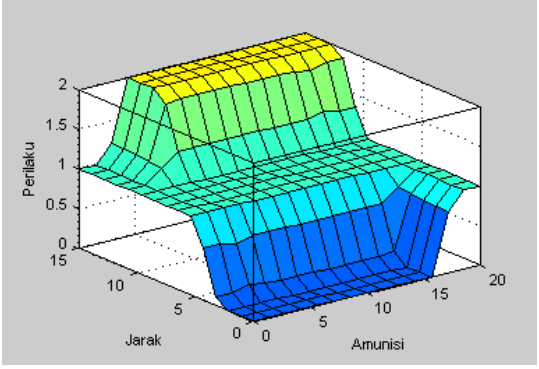
4.3 Pengujian *Fuzzy* Perilaku NPC *Sniper*

Pada pengujian *fuzzy* perilaku NPC *Sniper*, ada 2 variabel input yang diujikan, yaitu amunisi dan jarak musuh. Variabel amunisi diujikan mulai dari sedikit, sedang, hingga banyak. Sedangkan variable jarak musuh diujikan mulai dari dekat, sedang **hingga** jauh. Hasil pengujian *fuzzy* perilaku NPC *Sniper* dapat dilihat pada Tabel 4.2.

Tabel 2. Hasil pengujian *fuzzy* perilaku NPC *Sniper* dengan menggunakan variable masukan yang berbeda

JARAK MUSUH \ AMUNISI	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.73	1	1	1
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.73	1	1	1
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.73	1	1	1
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.73	1	1	1
4	0.53	0.53	0.53	0.53	0.53	0.53	0.53	0.53	0.53	0.53	0.53	0.53	0.53	0.53	0.53	0.53	0.53	0.81	1	1	1
5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
6	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
7	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
8	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
9	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
10	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
11	1	1	1	1.19	1.47	1.47	1.47	1.47	1.47	1.47	1.47	1.47	1.47	1.47	1.47	1.47	1.47	1.47	1.47	1.47	1.47
12	1	1	1	1.27	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
13	1	1	1	1.27	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
14	1	1	1	1.27	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
15	1	1	1	1.27	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2

Dari hasil pengujian Tabel 2 dapat diketahui bahwa terjadi perubahan repon perilaku terhadap perubahan kombinasi variable jarak musuh dan amunisi, dimana hasil respon fuzzy dapat dilihat pada Gambar 14.



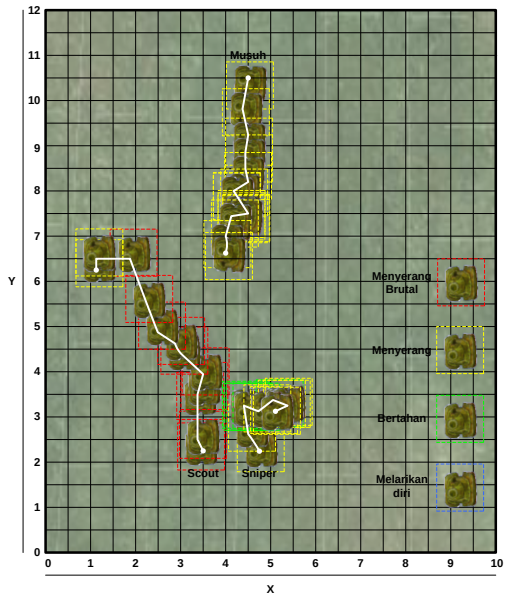
Gambar 14. Respon *fuzzy* perilaku NPC *Sniper* dalam grafik permukaan

4.4 Percobaan Perilaku NPC Sout dan *Sniper* Melawan Musuh

Percobaan ini dilakukan untuk melihat apakah perilaku masing-masing NPC sudah sesuai dengan perilaku yang sudah dirancang menggunakan HFSM dan respon perubahan perilaku berdasarkan variabel input menggunakan logika *fuzzy*.

4.4.1 Percobaan NPC Sout dan *Sniper* Melawan 1 NPC Musuh

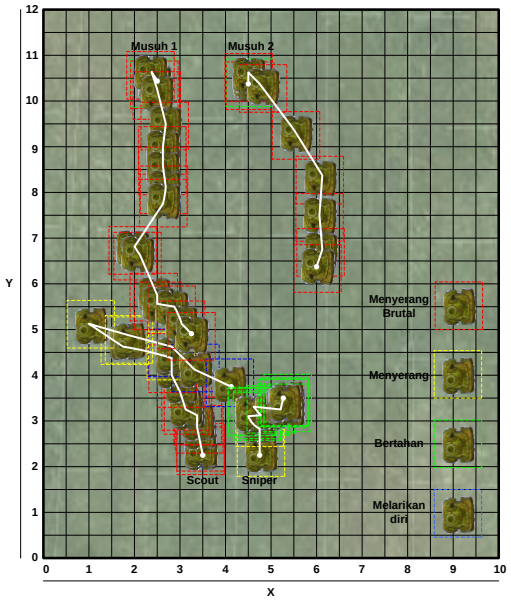
Dalam percobaan ini digunakan juga untuk menguji perubahan perilaku NPC berdasarkan variabel masukan yang dimiliki apabila hanya melawan 1 NPC Musuh saja. Simulasi pergerakan dan perilaku NPC dapat dilihat pada Gambar 15.



Gambar 15. Simulasi pergerakan NPC ketika melawan 1 NPC musuh

4.4.2 Percobaan NPC Sout dan *Sniper* Melawan 2 NPC Musuh

Percobaan ini dilakukan untuk menguji respon perubahan perilaku NPC berdasarkan variabel masukan yang dimiliki apabila melawan 2 NPC Musuh. Simulasi pergerakan dan perilaku NPC dapat dilihat pada Gambar 16.



Gambar 16. Simulasi pergerakan NPC ketika melawan 2 NPC musuh

Dari hasil 10 kali percobaan yang dilakukan untuk menguji tingkat kemenangan ketika melawan 2 NPC Musuh didapatkan hasil seperti pada Tabel 3. Dimana Tim *Scout* dan *Sniper* mendapatkan kemenangan sebanyak 8 kali sedangkan Tim Musuh mendapatkan 2 kali, sehingga tingkat kemenangan NPC *Scout* dan *Sniper* adalah 80%.

5. KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari hasil uji coba penelitian ini dapat diperoleh beberapa kesimpulan antara lain:

1. *Hierarchy Finite State Machine* digunakan untuk merancang perilaku masing-masing NPC dalam mendesain strategi menyerang.
2. Aturan *fuzzy* dapat diterapkan untuk menghasilkan perilaku NPC yang bervariasi sesuai dengan variable yang dimiliki.
3. Tingkat kemenangan strategi menyerang pada penelitian ini mencapai 80% jika melawan musuh yang mempunyai perilaku umum yaitu menyerang dan menghindar.

Tabel 3. Hasih Tim NPC *Scout* dan NPC *Sniper* melawan NPC Musuh

Percobaan	Tim NPC *Scout* dan NPC *Sniper*	Tim NPC Musuh 1 dan NPC Musuh 2
1	Menang	Kalah
2	Menang	Kalah
3	Kalah	Menang
4	Menang	Kalah
5	Menang	Kalah
6	Kalah	Menang
7	Menang	Kalah
8	Menang	Kalah
9	Menang	Kalah
10	Menang	Kalah

5.2. Saran

Untuk pengembangan strategi menghindar terhadap serangan musuh, di masa yang akan datang disarankan :

1. Agar strategi menyerang menjadi lebih menarik hendaknya ditambahkan juga variable experience dan jumlah musuh dalam aturan *fuzzy* untuk menentukan perilaku.
2. Meambahkan 2 atau 3 lagi NPC dalam sebuah tim, sehingga diharapkan strategi menjadi lebih menarik dengan memberikan perilaku yang berbeda-beda pada setiap NPC.

DAFTAR PUSTAKA

[1] Jørgen Havsberg Seland. *A Visual Programming Language for Hierarchical Finite State Machines in Game AI*. 2007.

[2] Alain Girault, Bilung Lee, and Edward A. Lee, Fellow. *Hierarchical Finite State Machines with Multiple Concurrency Models*. IEEE, 1999.

[3] Alex Mclean. *Hunting Down the Player in a Convincing Manner*. Pivotal Games, 2002.

[4] Bilung Lee and Edward A. Lee. *Interaction of Finite State* Machines *and Concurrency Models*. Proceeding of Thirty Second Annual Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, California, 1998.

[5] Craig W. Reynolds. *Steering Behaviors For Autonomous Characters*. Sony Computer Entertainment America.

[6] Edward F. maurina. *The Game Programmer's Guide to Torque*. A Garage Games Book 2006.

[7] David J. Sushil. *Torque Game Engine Primer*. 2007.

[8] JinHyuk Hong dan Sung-Bae Cho. *Evolving Reactive NPCs for the Real-Time Simulation Game*. CIG, 2005.

[9] Michelle McPartland and Marcus Gallagher. *Creating a Multi-Purpose First Person Shooter Bot with Reinforcement Learning*. IEEE, 2008.

[10] Lu´ıs Miguel. *Agents for Massive On-line Strategy Turn Based Games*. Landeiro Ribeiro, 2007.

[11] Schawab Brian. *AI Game Engine Programming,1st edition*. Charles River Media, INC, New York, 2004.

[12] ThomasWagner PeterWolstenholme Ferdinand Wagner, Ruedi Schmuki. *Modeling Software with Finite State Machines A Practical Approach*. Auerbach Publications, CRC Press, Taylor & Francis Group, 2006.

[13] Alex J. Champandard. *AI Game Development: Synthetic Creatures with Learning and Reactive Behaviors*. New Riders Publishing, 2003.

[14] Jörg Kienzle, Alexandre Denault, Hans Vangheluwe. *Model-based Design of Computer-Controlled Game Character Behavior*. McGill University, Montreal, QC H3A 2A7, Canada, 2007.