Pergantian Senjata *NPC* pada *Game FPS* Menggunakan *Fuzzy Sugeno*

Yunifa Miftachul Arif [1)], Ady Wicaksono [2)], Fachrul Kurniawan [3)]
[1,3)] Jurusan Teknik Informatika, Fakultas Saintek, UIN Maulana Malik Ibrahim Malang
[2)] Jurusan Multimedia, SMKN 3 Batu

Abstrak

Pengembangan kecerdasan buatan dalam game bertujuan untuk membuat aksi dan reaksi yang secara otomatis pada NPC (Non-Player Character). Penelitian ini membahas tentang bagaimana sistem perubahan senjata secara otomatis pada NPC berdasarkan perubahan kondisi lingkungan yang dihadapi. Metode yang digunakan untuk menentukan jenis senjata yang digunakan pada penelitian ini adalah fuzzy sugeno. Untuk menghasilkan output fuzzy yang bervariasi, maka digunakan variabel jarak musuh dan jumlah teman. Desain pergantian senjata yang dibuat menggunakan software Matlab selanjutnya diujicobakan dalam game First Person Shooter menggunakan Torque Game Engine. Dalam hasil ujicoba game terjadi respon pergantian senjata pada masing-masing NPC terhadap kondisi yang dihadapi sesuai dengan rule fuzzy yang sudah didesain sebelumnya.

Kata kunci: *NPC, pergantian senjata, otomatis, fuzzy sugeno.*

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 [1]. 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 [2]. Khusus untuk berperang NPC juga diharapkan mempunyai strategi-strategi khusus seperti halnya manusia [3].

Model strategi menyerang bisa bermacam-macam, misalnya strategi menyerang maupun strategi dalam hal penggunaan senjata [4]. Seperti halnya didunia nyata, setiap senjata tentu mempunyai karakteristik dan fungsi yang berbeda. Misalnya senapan laras pendek digunakan untuk menyerang musuh jarak dekat, sedangkan meriam digunakan untuk menyerang musuh dari jarak jauh. Apabila suatu NPC dapat menggunakan senjata sesuai dengan fungsinya, atau dengan kata lain orientasi penggunaan senjata dapat menyesuaikan dengan kondisi yang dihadapi oleh NPC, maka sebuah game akan Nampak lebih menarik dan lebih natural. Pada penelitian ini akan dibahas mengenai model strategi penggunaan senjata oleh NPC pada *game* FPS, dimana untuk menentukan respon jenis senjata yang digunakan terhadap perubahan kondisi yang dihadapi NPC digunakan logika *fuzzy*.

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* [5].

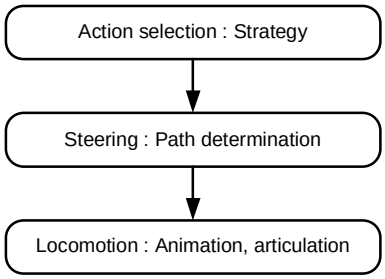
2. TEORI PENUNJANG

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.

Non-Player Character

Autonomous character adalah jenis *otonomous agent* yang ditujukan untuk penggunaan komputer animasi dan media interaktif seperti *game*s 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

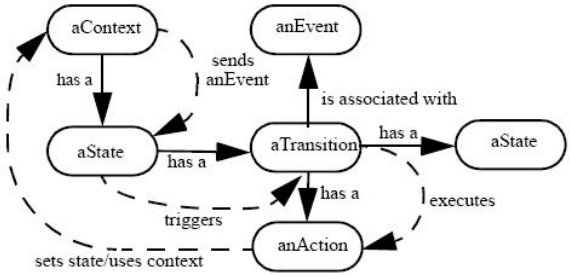
Tiga lapisan hirarki tersebut, adalah motivasi, tugas, dan motor. [6]. Pada penelitian ini lebih di fokuskan pada bagian *action selection* yang di dalamnya berisi strategi pergerakan NPC.

Finite State Machine (FSM)

Dalam perancangan *AI* untuk *game*, *state machine* merupakan teknik yang paling banyak dipergunakan untuk permasalahan *"decision making"* dan, sekaligus dengan *scripting*nya, dipergunakan secara luas untuk merancang sistem *decision making* dalam *game* . *State machine* dikenal secara luas sebagai teknik untuk pemodelan fenomena atau kondisi berbasis *event*, termasuk penguraiannya, serta desain *interface. Finite State Machine (FSM)* atau juga disebut sebagai *Finite State Automata*, dianggap sebagai teknik yang secara luas dipergunakan dalam merancang *AI* dalam *game*. Teknik ini merupakan metodologi perancangan sistem untuk memodelkan perilaku (*behavior*) dari sistem atau obyek yang kompleks dengan yang kondisi yang telah didefinisikan dalam set.

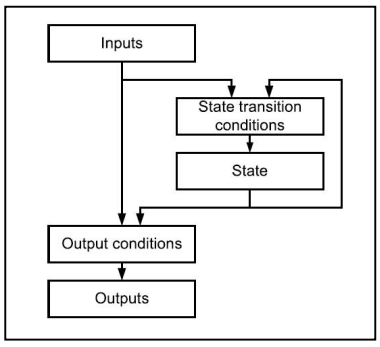
FSM merupakan model komputasi, dan sebagai sebagai teknik permodelan *AI*, *FSM* terdiri dari 4 komponen: *State*, merupakan kondisi yang mendefinisikan perilaku sekaligus memungkinkan terjadinya aksi; *State transition*, yang memicu perpindahan *state*; *Rules*, atau kondisi yang harus dipenuhi untuk dapat memicu transisi state; *Input events*, yang bisa dipicu secara internal ataupun eksternal yang memicu tercapainya *rule* dan mengarahkan terjadinya transisi *state*. Jika kemudian dikelompokkan dalam pembagian *set*, maka *FSM* terdiri dari: *state set*, *input set*, *output set*, dan *fungsi state transition* [7].

Dalam *FSM*, istilah *state* merupakan konsep yang sangat fundamental karena menyajikan informasi yang berkaitan dengan keadaan sistem saat sebelumnya. Dalam satu periode yang tetap, sistem berada dalam satu *state*, yang tiap *state*-nya mempunyai karakteristik perilaku dan aksi yang spesifik (yang sudah ditentukan). *State-state* dihubungkan melalui transisi antar *state*, selanjutnya masing-masing transisi mengarahkan ke *state* (kondisi) selanjutnya sebagai *target state*. Akan selalu ada *initial state* yang berfungsi sebagai *starting point*, lalu kondisi "saat ini' (*current state*) yang menyimpan informasi *state* sebelumnya. *Input event* (kejadian), yang bisa dipicu secara eksternal maupun internal sistem, berfungsi sebagai pemicu (*trigger*), yang mengarahkan pada proses evaluasi dari *rule* (aturan). Jika kondisi dan syaratnya terpenuhi, maka terjadi transisi dari *state* saat ini ke *state* selanjutnya sesuai dengan *rule* yang sudah ada. *Transitions* merupakan *class* dari obyek yang mengatur sistem [8]. *Transitions* mengontrol aliran dari eksekusi dengan melakukan *setting* pada *state* yang sedang aktif dari *state machine* melalui penggunaan dari kondisi. *Transitions* bisa berlaku *one-to-many*, dengan kata lain, bisa terjadi ada satu *state* terhubungkan ke sisi *input* dari *transitions*, dan beberapa *state* terhubung ke sisi *output* dari *transitions*, tergantung dari kondisi yang sedang berlangsung dalam *transition* tersebut. Prinsip dari komponen-komponen yang terintegrasi dalam *FSM* ditunjukkan dalam gambar 2.



Gambar 2 *Framework Finite State Machine*

State machine merupakan bagian logika yang bertanggung jawab pada perilaku (*behavior*) sistem. Sedangkan kata "*finite*" sendiri merujuk pada jumlah yang sudah dibatasi (ditentukan) untuk *state* yang ditangani oleh sistem. Sejarah dari perubahan *input* diperlukan untuk dapat menentukan secara jelas perilaku, dan komponen ini disimpan dalam variabel internal yang disebut sebagai *state*. Selanjutnya, kondisi transisi *state* dan *output* merupakan fungsi dari input dan *state*. Gambar 3 menjelaskan contoh dari pengertian dasar mengenai *state machines* [9].



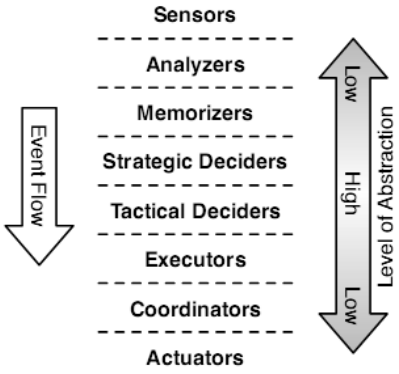
Gambar 3 Dasar pengertian *State Machine*

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 NPC yang didasarkan pada kondisi saat ini dan memori. Pada tingkat berikutnya, *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-*actuators* dan mungkin kembali memberikan perintah tingkat rendah lebih lanjut. Akhirnya, *actuators* melakukan tindakan yang diinginkan [10]. Bahasan penelitian ini lebih berada pada tingkat komponen *strategic deciders* dan *tactical deciders.*



Gambar 4 *AI model Architecture*

Logika *Fuzzy*

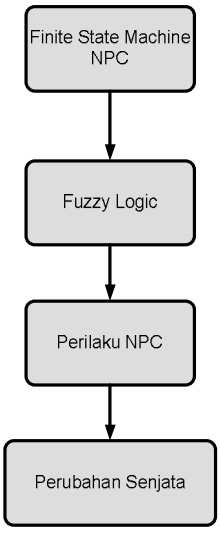
Logika *Fuzzy* adalah sebuah metode untuk menangani masalah ketidakpastian. Yang dimaksud dengan ketidakpastian yaitu suatu masalah yang mengandung keraguan, ketidaktepatan, kurang lengkapnya informasi, dan nilai kebenarannya bersifat sebagian. Ide tentang logika *Fuzzy* sebenarnya telah lama dipikirkan, 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 [11]. Atas dasar itulah pada penelitian ini, logika *fuzzy* digunakan dengan tujuan untuk mendapatkan respon perubahan senjata NPC berdasarkan variabel input yang dimiliki. Selanjutnya dalam menentukan perubahannya digunakan model *Fuzzy* Sugeno, dimana setiap output senjata NPC diwakili oleh konstanta tetap yang sudah ditentukan sebelumnya.

3. METODE PENELITIAN

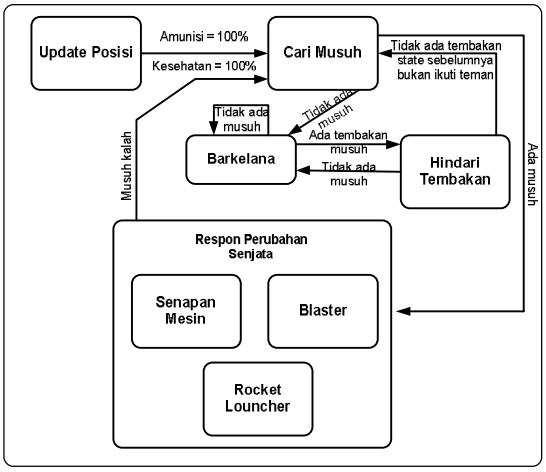
Penelitian ini menggunakan FSM sebagai metode yang digunakan dalam mendesain perilaku NPC, sedangkan untuk menentukan perilaku dalam melakukan perubahan senjata NPC berdasarkan variable yang dimilikinya digunakan logika *fuzzy*. Gambar 5 menunjukkan tahapan-tahapan dalam penelitian yang meliputi perancangan FSM, perancangan logika *fuzzy* hingga terjadi perubahan senjata NPC berdasarkan kondisi lingkungan yang dihadapi .



Gambar 5 Diagram blok penelitian

FSM Perilaku NPC

Secara garis besar desain FSM pada penelitian ini ditunjukkan pada Gambar 6. Ada beberapa state dasar yang dijelaskan pada gambar tersebut, dimana masing-masing state mewakili perilaku yang dimiliki oleh NPC. Diantaranya adalah update posisi pada awal permainan, mencari musuh, berkelana, menghindari tambakan jika terjadi serangan oleh musuh, dan respon perubahan senjata yang didesain dengan menggunakan fuzzy sugeno. Tiga output senjata yang digunakan antara lain adalah Senapan mesin, Blaster, dan juga Rocket Louncher.



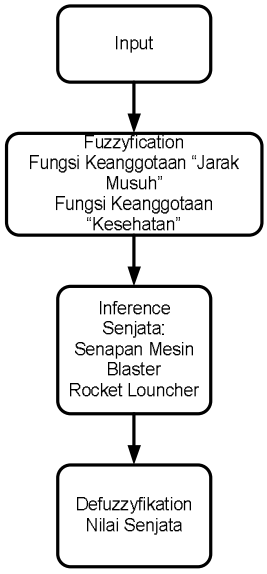
Gambar 6 FSM (*Finite state machine)* NPC

Logika *Fuzzy*

Variasi perubahan senjata yang digunakan oleh NPC, didesain dengan menggunakan logika *fuzzy*. Model *fuzzy* yang digunakan adalah Sugeno, dengan pertimbangan bahwa output yang dihasilkan model *fuzzy* Sugeno adalah berupa konstanta tegas, sehingga dapat mewakili nilai perilaku yang sudah didesain sebelumnya.

Desain *Fuzzy* NPC

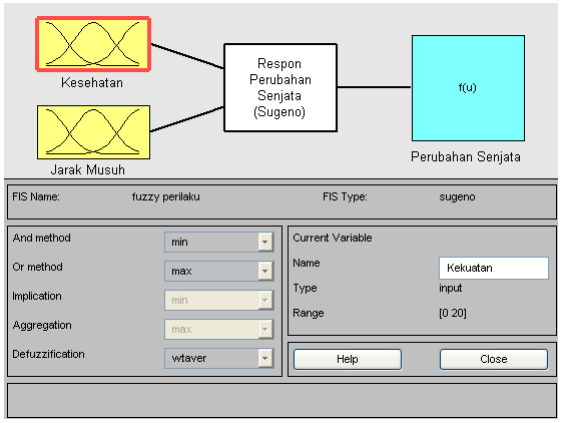
Dua variabel yang mempengaruhi perubahan senjata NPC, adalah tingkat kesehatan (sangat lemah, lemah, sedang, kuat, sangat kuat) dan jarak musuh (sangat dekat, dekat, sedang, jauh). Dengan adanya 2 variabel tersebut diharapkan perubahan perilaku NPC dalam hal perubahan senjata lebih bervariasi. Selanjutnya untuk menjadi otonom maka digunakan aturan sebab akibat antara perilaku dengan atribut variabel yang menempel pada NPC. Misalnya ketika kesehatan sangat lemah dan jarak musuh sangat dekat maka NPC cenderung menggunakan senjata laras pendek paling ringan diantara senjata yang lain (senapan mesin) sesuai dengan hasil de*fuzzy*fikasi.



Gambar 7 Logika *fuzzy* perilaku NPC *Scout*

Gambar 7 menunjukkan bahwa dalam penelitian ini, ada dua atribut yang diberikan terhadap NPC, yaitu Jarak musuh dan kesehatan. Dimana dalam logika *fuzzy* masing-masing digunakan gabungan gabungan fungsi keanggotaan segitiga dan trapezium.

Desain *fuzzy* untuk menghasilkan perilaku NPC *Scout* dapat dilihat pada Gambar 8.



Gambar 8 Desain *fuzzy* perubahan senjata NPC

4. HASIL DAN PEMBAHASAN

Pengujian Sistem

Pengujian system yang pertama dilakukan dalam penelitian ini adalah pengujian terhadap desain logika *fuzzy* yang dilakukan dengan menggunakan software Mathlab. Dari hasil system *fuzzy* yang diperoleh, selanjutnya diujicobakan pada FPS *game* menggunakan *Torque Game Engine*.

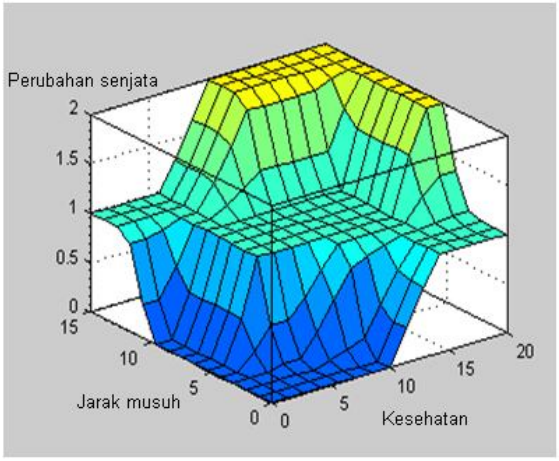
Dalam pengujian perubahan perilaku NPC terhadap senjata yang digunakan, beberapa parameter diujikan untuk mengetahui variasi perilaku yang

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

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

KESEHATAN \ JARAK MUSUH	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0.47	1	1	1	1
1	0	0	0	0	0	0	0	0	0	0	0	0.47	1	1	1	1
2	0	0	0	0	0	0	0	0	0	0	0	0.47	1	1	1	1
3	0	0	0	0	0.12	0.5	0.5	0.5	0.5	0.5	0.5	0.77	1	1	1	1
4	0	0	0	0	0.25	1	1	1	1	1	1	1	1	1	1	1
5	0	0	0	0	0.25	1	1	1	1	1	1	1	1	1	1	1
6	0	0	0	0	0.25	1	1	1	1	1	1	1	1	1	1	1
7	0	0	0	0.18	0.44	1	1	1	1	1	1	1.12	1.25	1.25	1.25	1.25
8	0	0	0	0.36	0.62	1	1	1	1	1	1	1.24	1.5	1.5	1.5	1.5
9	0	0	0	0.54	0.81	1	1	1	1	1	1	1.36	1.5	1.5	1.5	1.75
10	0	0	0	0.72	1	1	1	1	1	1	1	1.47	2	2	2	2
11	0.25	0.25	0.25	0.79	1	1	1	1	1	1	1	1.47	2	2	2	2
12	0.5	0.5	0.5	0.86	1	1	1	1	1	1	1	1.47	2	2	2	2
13	0.75	0.75	0.75	0.93	1	1	1	1	1	1	1	1.47	2	2	2	2
14	1	1	1	1	1	1	1	1	1	1	1	1.47	2	2	2	2
15	1	1	1	1	1	1	1	1	1	1	1	1.47	2	2	2	2
16	1	1	1	1	1	1	1	1	1	1	1	1.47	2	2	2	2
17	1	1	1	1	1.13	1.5	1.5	1.5	1.5	1.5	1.5	1.74	2	2	2	2
18	1	1	1	1	1.25	2	2	2	2	2	2	2	2	2	2	2
19	1	1	1	1	1.25	2	2	2	2	2	2	2	2	2	2	2
20	1	1	1	1	1.25	2	2	2	2	2	2	2	2	2	2	2

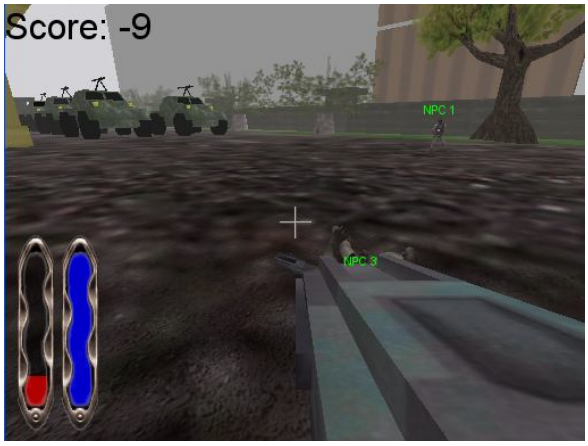
Dari variasi variable masukan yang digambarkan pada Tabel 1 dapat diperoleh keluaran perilaku yang variatif, dimana selanjutnya dikelompokkan menjadi 3 model perilaku NPC dalam hal perubahan senjata yaitu menggunakan senapan mesin, blaster dan rocket louncher. Nilai output untuk senapan mesin ditunjukkan pada blok warna hijau (nilai<=0.5), blaster ditunjukkan pada blok warna merah muda (nilai>0.5 dan nilai<=1.5), sedangkan untuk rocket launcher ditunjukkan pada blok warna biru (nilai>1.5). Selanjutnya respon keluaran *fuzzy* perilaku NPC terhadap senjata yang digunakan direpresentasikan oleh grafik tiga dimensi pada Gambar 9.



Gambar 9 Respon *fuzzy* perubahan senjata NPC dalam grafik permukaan

Game Simulasi

Dari hasil desain *fuzzy* yang sudah didapatkan, selanjutnya disimulasikan dalam game FPS yang dibangun dengan menggunakan Torque Game Engine. Hasil tampilan user interface game ditunjukkan pada gambar 10. Sedangkan hasil perubahan senjata yang digunakan oleh NPC ditunjukkan pada gambar 11.



Gambar 10a Tampilan user interface game

Gambar 10b Tampilan user interface game (kondisi berperang)

Gambar 11 Tampilan hasil perubahan senjata NPC

5. PENUTUP

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

1. Aturan *fuzzy* dapat diterapkan untuk menghasilkan perubahan senjata NPC yang bervariasi sesuai dengan nilai variable yang dimiliki.
2. Diperlukan tambahan variasi input untuk mendapatkan hasil output yang lebih natural, misalnya dengan menambahkan variable jumlah musuh atau jumlah amunisi.

Daftar Referensi

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

[2] Yunifa Miftachul Arif, Fachrul Kurniawan dan Fresy Nugroho. *Desain Perubahan Perilaku pada NPC Game Menggunakan Logika Fuzzy*. National Seminar on Electrical, Informatics, and Its Education, 2011.

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

[4] Yunifa Miftachul Arif. *Strategi Menyerang pada Game FPS menggunakan Hierarchichal Finite State Machine dan Logika Fuzzy*. Master Thesis, 2010.

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

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

[7] Arif, Yunifa Miftachul and Ririen Kusumawati. *Attack Strategy For NPC In FPS Game Using Fuzzy Sugeno*. Basic Science International Conference, 2012.

[8] 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.

[9] Alex Mclean. *Hunting Down the Player in a Convincing Manner*. Pivotal *Game*s, 2002.

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

[11] Sri Kusumadewi. *Artificial intelligence (teknik dan aplikasinya)*. Yogyakarta : Graha Ilmu, 2003.