Real-Time Action Role-Playing Game dengan menggunakan *Fuzzy State Automata* untuk Mengontrol Pergerakan *Enemy*

Gary Nathanael M.[1], Gregorius Satiabudhi[2], Rolly Intan[3]
Program Studi Teknik Informatika Fakultas Teknologi Industri Universitas Kristen Petra
Jl. Siwalankerto 121 – 131 Surabaya 60236
Telp. (031) – 2983455, Fax. (031) – 8417658
E-Mail: echizengary@gmail.com[1], greg@petra.ac.id[2], rintan@petra.ac.id[3]

ABSTRAK

Video Game merupakan cara yang baik bagi seseorang untuk menghilangkan stres dan menikmati waktu luang. Sebagian besar dari *video game* yang beredar saat ini dapat dimainkan dengan seorang diri. Tetapi, seringkali ketika bermain *video game* seorang diri, orang cepat merasa bosan karena musuh dalam permainan bergerak dengan sangat konstan. Untuk mengatasi hal tersebut, dalam skripsi ini dibuat game dimana musuh dalam permainan tersebut bergerak secara yang tidak mudah untuk diprediksi.

Genre dari permainan yang dibuat adalah *Action Role-Playing Game* (ARPG). *Artificial Intelligence* (AI) yang dibuat akan menggunakan metode *Fuzzy State Automata*. Dengan adanya *Fuzzy State Automata* tersebut, musuh dalam permainan dapat memilih gerakan apa yang akan dilakukan yang sesuai dengan *fuzzy set input* yang ada. Durasi dari gerakan pada *fuzzy state automata* tersebut juga bersifat tidak konstan, sehingga menambah kesulitan untuk melakukan prediksi terhadap gerakan musuh.

Melalui hasil pengujian yang dilakukan, *Fuzzy State Automata* tersebut berhasil melakukan gerakan yang sesuai dengan situasinya. *Fuzzy State Automata* tersebut juga sukses menggunakan kelebihannya dalam durasi *state* yang tidak konstan untuk membuat gerakan *enemy* lebih susah diprediksi.

Kata Kunci: *Video games, Fuzzy State Automata, Action Role-Playing Game.*

ABSTRACT

Video Games are one of the most popular ways for someone to relieve stress or spend their free time. Most of the video game that is available right now can be played in single player. Unfortunately, most of the times when playing video games in single player, people usually get bored quite easily due to the enemy having static patterns as their actions. To overcome this problem, this thesis will be developing a game with an enemy that is not easily predictable.

The genre of the game is Action Role-playing Game (ARPG). The Artificial Intelligence involved is made using the Fuzzy State Automata method. Using the Fuzzy State Automata method, the enemy will have to decide which state that it with enter using the input fuzzy sets and act accordingly. The states of the fuzzy state automata in here also have no fixed durations, therefore adding another unpredictability element for the player to deal with.

Based on the testing of the Fuzzy State Automata, the artificial intelligence have the ability to react accordingly to its inputs. The Fuzzy State Automata also manages to successfully exhibit the capability of its states to have no fixed durations to make the enemy actions harder to predict.

Keywords: *Video games, Fuzzy State Automata, Action Role-Playing Game*

1. PENDAHULUAN

Video Game merupakan permainan dengan menggunakan interaksi pemain dengan gambar yang dihasilkan oleh piranti video. Video Game saat ini telah menjadi sesuatu yang sudah umum untuk dimainkan oleh berbagai kalangan, dari anak-anak, remaja, maupun dewasa. Video Game dibagi menjadi beberapa genre, salah satunya adalah Action Role-Playing Game.

Action Role-playing Game (ARPG) adalah sebuah genre permainan yang merupakan perpaduan dari dua genre permainan lainnya, yaitu Role-playing Game, dan Action Game. Role-playing Game adalah genre permainan yang mempunyai fokus pada setting dan penceritaannya. Di sisi lain, Action Game adalah genre permainan yang mempunyai aksentuasi pada tantangan fisik, seperti koordinasi mata dengan tangan atau reflek dari pemain. Action Role-Playing Game menggabungkan fokus dari kedua genre tersebut menjadi sebuah genre baru yang mempunyai penceritaan dan setting yang mencerminkan Role-Playing Game dengan aksentuasi pada pergerakan dan tantangan yang bersifat real-time seperti pada Action Game.

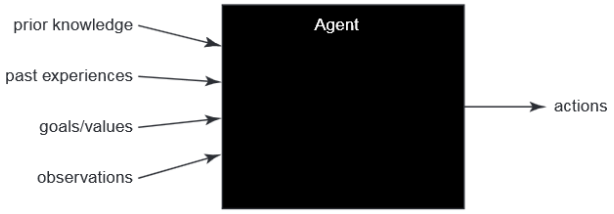
Pergerakan enemy dalam mayoritas game yang mempunyai genre ARPG secara umumnya menggunakan metode yang bersifat rule-based untuk membuat sebuah Finite *State* Automata. Dengan metode itu, gerakan yang dilakukan oleh enemy dalam permainan dipetakan kedalam berbagai macam *state* pada sebuah Finite *State* Automata, dimana apabila sebuah rule yang telah ditentukan oleh pembuat permainan dipenuhi, maka akan terjadi transisi antar *state*.

Metode Finite *State* Automata tersebut mempunyai kelemahan dimana kondisi atau rule transisi antara *state* bersifat pasti, sehingga menimbulkan kesan bahwa pergerakan enemy cenderung mudah untuk diprediksi oleh pemain. Dalam kasus ini, penggunaan Fuzzy *State* Automata (FFSA) bisa digunakan untuk membuat improvisasi terhadap sistem. Dengan menggunakan Fuzzy *State* Automata, kondisi atau rules dapat didesain secara non-linier sehingga mengakibatkan transisi non-linier yang mengakibatkan pergerakan dari enemy menjadi lebih susah untuk diprediksi oleh pemain.

2. LANDASAN TEORI

2.1 Artificial Intelligence

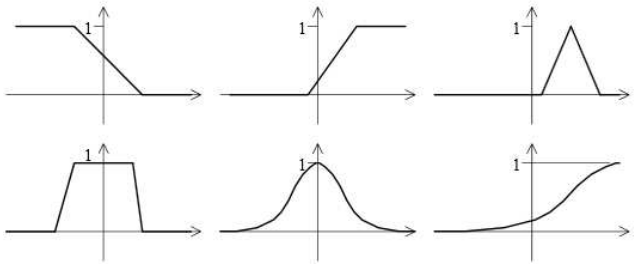
Kecerdasan Buatan atau *Artificial Intelligence* adalah studi dari desain sebuah agen intelijen. Agen disini adalah sesuatu yang berinteraksi dalam sebuah lingkungan. Agen mencakup mahkluk hidup seperti hewan sampai manusia, benda mati seperti pesawat, atau bahkan sebuah organisasi. Agen Intelijen sendiri adalah sistem yang berperilaku dengan cerdas: yaitu melakukan hal yang sesuai dengan tujuannya, fleksibel terhadap perubahan lingkungan dan perubahan tujuan, dan menentukan pilihan berdasarkan limitasi persepsi dan komputasi terbatas.[4]
Pada setiap waktu, agen dapat menerima beberapa macam input, seperti pengetahuan dasar mengenai dunianya, pengalaman tentang apa yang telah terjadi, tujuan yang ingin dicapai atau value mengenai apa yang penting, dan observasi mengenai lingkungan saat ini dan dirinya sendiri. Dari berbagai macam kombinasi input tersebut, agen dapat melakukan sesuatu. Untuk tiap agen yang berbeda, input dan apa yang akan dilakukan perlu dispesifikasikan sendiri. Desain dari agen ini dapat dilihat pada Gambar 1, dimana agen digambarkan menjadi sebuah *black box*. Agen direpresentasikan sebagai sebuah *black box*, dimana tujuan dari desainer dari agen adalah untuk mengisi apa yang ada dalam *black box* tersebut, agar tindakan yang dilakukan oleh agen tersebut masuk akal berdasarkan input yang diberikan.



Gambar 1. Agen sebagai *black box*[4]

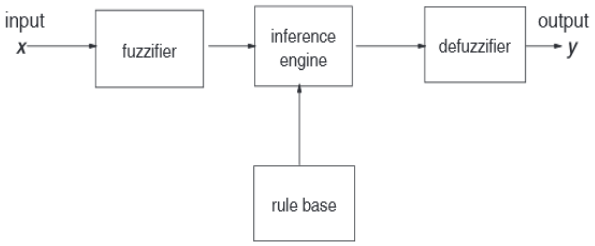
2.2 *Fuzzy Logic*

Fuzzy Logic adalah logika yang bertujuan untuk memberikan pondasi kepada pemikiran yang bersifat mengira-ngira. *Fuzzy Logic* menggunakan proposisi yang tidak pasti berdasarkan dari teori *Fuzzy Set* [1]. *Fuzzy Logic* merupakan salah satu bentuk dari *Many-valued logic* dimana *truth values* dari variablenya dapat berupa bilangan *real* dari 0 sampai 1, dibandingkan dengan *boolean logic* yang hanya menerima *input* 0 atau 1.
Fuzzy Set bisa diasumsikan sebagai *set* yang mempunyai *membership function* untuk mengetahui seberapa dalam sebuah objek berada dalam sebuah *set*. Gambar 2 menunjukkan beberapa macam bentuk *membership function* yang sering digunakan.



Gambar 2. Contoh Bentuk Membership Function[1]

Membership degree suatu objek selalu ada di interval [0,1]. Apabila objek berada di luar sebuat *set*, maka *membership degree* dari objek tersebut adalah 0. Sebaliknya, apabila suatu objek berada tepat di dalam *set,* maka *membership degree* dari objek tersebut adalah 1.
Untuk mengaplikasikan *Fuzzy Logic* kedalam suatu sistem, dibutuhkan sebuah sistem yang dinamakan dengan *Fuzzy Inference System. Fuzzy Inference System* mendefinisikan *mapping input* yang bersifat nonlinier menjadi sebuah input skalar dengan menggunakan *Fuzzy Rules*. *Mapping* tersebut meliputi *input/output membership function, Fuzzy Logic operator, Fuzzy if-then rules,* agregasi dari *set output*, dan *defuzzification*. Skema dari *Fuzzy Inference System* dapat dilihat pada Gambar 3.

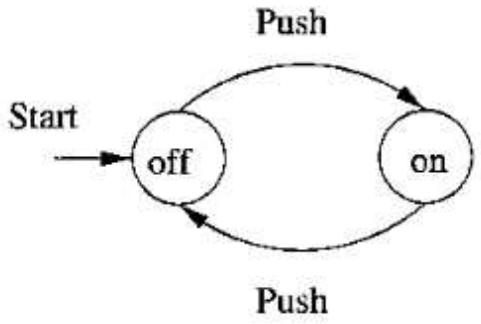


Gambar 3. Skema Fuzzy Inference System

Fuzzy Inference System mempunyai empat komponen utama, yaitu *Fuzzifier, Inference Engine, Rule base,* dan *Defuzzifier. Rule base* dari *Fuzzy Inference System* tersebut meliputi *linguistic rule* yang telah disiapkan oleh spesialis dari bidang yang akan ditelusuri dalam sistem. *Rule* juga mungkin didapatkan dari data yang bersifat numerik. Setelah *rule* selesai didefinisikan, *Fuzzy Inference System* dapat dilihat sebagai sistem yang menerima sebuah *input vector* dan mengolahnya menjadi sebuah *output vector. Fuzzifier* mengolah input yang berupa angka kedalam *fuzzy set* yang sesuai dan mengatur seberapa dalam *Membership Degree* dari *input* tersebut terhadap *Membership Function* yang ada. *Inference Engine* kemudian mengolah *input* tersebut menjadi sebuah *output fuzzy set* dengan cara membandingkan *input* tersebut terhadap *rules* yang telah didefnisikan pada *rule base*. *Output set* yang berupa *fuzzy set* tersebut kemudian diproses oleh *defuzzifier* untuk mendapatkan sebuah *crisp number* yang bisa diolah lebih lanjut.

2.3 *Finite State Automata*

Finite State Automata adalah automata dimana jumlah dari state yang berjumlah *finite*, atau terbatas. *State* sendiri digunakan untuk mengingat bagian yang relevan dari keadaan sebelumnya dari sistem. Karena jumlah *state* yang ada terbatas, seluruh keadaan sebelumnya dari sistem seringkali tidak dapat diingat, karena itu sistem harus didesain sedemikian rupa untuk bisa mengingat apa yang penting dan melupakan apa yang tidak. Keuntungan dari jumlah *state* yang terbatas adalah sistem dapat diimplementasikan dengan kebutuhan yang jumlahnya pasti, seperti dalam perangkat keras sebagai *circuit*. [2]
Bentuk yang paling ringan dari sebuah *finite automaton* adalah sebuah *switch on/off*. Perangkat mengingat apakah kondisi sekarang berada pada *"on" switch* atau *"off" switch* dan ketika pengguna menekan tombol tersebut, efek yang akan dihasilkan akan berbeda.

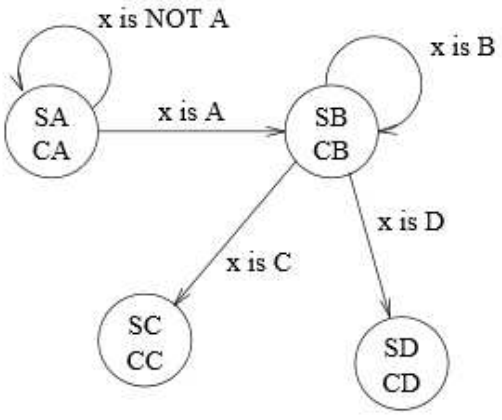


Gambar 4. Finite Automata on/off Switch[2]

Gambar 4 menggambarkan *Finite Automata* dari sebuah *on/off switch*, dimana pada contoh ini terdapat dua *state*, yang dinamakan *state on* dan *off*. Salah satu dari *state* dilabelkan sebagai *start state*, yang menandai *state* awal dari system. Pada kasus ini *state off* adalah *start state*. Panah pada contoh dilabelkan dengan input, yang merepresentasikan pengaruh eksternal terhadap sistem. Pada contoh ini, kedua panah tersebut dilabelkan dengan input *Push*, yang merepresentasikan pengguna menekan tombol tersebut. Maksud dari kedua panah tersebut adalah dimanapun *state* sistem saat ini, ketika tombol ditekan maka sistem akan berpindah ke *state* yang lain. Terkadang juga dibutuhkan untuk menandai sebuah ataupun lebih *state* sebagai *final* atau *accepting state*, dimana setelah barisan dari input dilakukan, apabila *final* atau *accepting state* dimasuki, maka *input* dinyatakan baik.

2.4 *Fuzzy State Automata*

Fuzzy State Automata adalah sebuah *automata* dimana transisi dari tiap *state* terjadi bukan dari *crisp events*, tetapi berdasarkan *fuzzy variables* dan transisi antar *state* juga bersifat *fuzzy.* Karena sifat itu, pada saat apapun sistem dapat berada di lebih dari satu *state* sekaligus[3], tetapi bisa berada di beberapa *state* pada saat yang sama. Tiap *state* mempunyai *membership value* masing-masing. [7] *Fuzzy State Automata* digunakan untuk memberikan *behavior* yang lebih susah untuk diprediksi dibandingkan dengan *Finite State Automata* biasa.[5]



Gambar 5. Fuzzy *State* Automata[6]

Fuzzy State Automata bersifat mirip dengan *Finite State Automata* tradisional, dimana *automata* merupakan sekumpulan dari *state* S_j, yang dihubungkan oleh *fuzzy transitions*. Seperti pada *Finite State Automata*, sebuah *state* merepresentasikan kondisi untuk melakukan sebuah tindakan. Tetapi, perbedaan yang membedakan *Finite State Automata* dengan *Fuzzy State Automata* adalah sebuah sistem tidak harus berada di sebuah *state* pada saat tertentu. Tiap *state* S_j dapat dihubungkan dengan sebuah *fuzzy state activity* yang berkisar antara 0 sampai dengan 1 yang merepresentasikan seberapa banyak sistem ada di *state* tersebut.

2.5 *Unity*

Unity merupakan sebuah *game engine* yang dikembangkan oleh *Unity Technologies* yang bisa digunakan untuk membuah sebuah *game* dalam berbagai platform sekaligus. Unity 1.0 pertama kali diluncurkan pada tanggal 8 Juni 2005 dan terus mengalami perkembangan hingga saat ini, dengan 5 versi yang telah dirilis dalam kurun waktu 2005-2015.[8]

3. DESAIN SISTEM

3.1 Desain Game

Judul *game* yang dibuat adalah *Chrono Fantasia*. *Game* tersebut menggunakan *Fuzzy Automata* untuk mengatur pergerakan *enemy* sehingga pemain lebih sulit menebak pergerakan *enemy*. *Fuzzy State Automata* dipilih karena sifatnya yang *non-deterministic*. Sifat *non-deterministic* tersebut menyebabkan *enemy* dapat bergerak lebih fleksibel karena *Fuzzy Automata* tidak dibatasi oleh beberapa restriksi yang mempengaruhi *Finite State Machine* biasa.

Game ini mengadaptasi jenis permainan *real-time Action Role-Playing Game* dimana pemain menggerakkan sebuah karakter untuk menyelesaikan berbagai macam *mission*. *Input* dilakukan dengan menggunakan *keyboard* dan *mouse*. Pemain diberikan beberapa *stage* dimana pemain harus menyelesaikan misi-misi di dalamnya untuk melanjutkan permainan ke stage berikutnya. Progresi misi bersifat *linear*, dan pemain tidak dapat memainkan *stage* yang berikutnya tanpa menyelesaikan *stage* yang sebelumnya terlebih dahulu.

Musuh digerakkan berdasarkan *Fuzzy State Automata* yang telah dibuat bagi masing-masing jenis *enemy*. Setiap jenis *enemy* mempunyai *rules* masing-masing yang berbeda. *Input* yang berpengaruh terhadap *rules* tersebut adalah *Health, Range,* dan *Tension. Input* tersebut kemudian dievaluasi oleh berbagai *rules* yang membentuk *Fuzzy Automata* untuk menghasilkan *states* yang dilakukan oleh *enemy*.

3.2 Desain Update Process

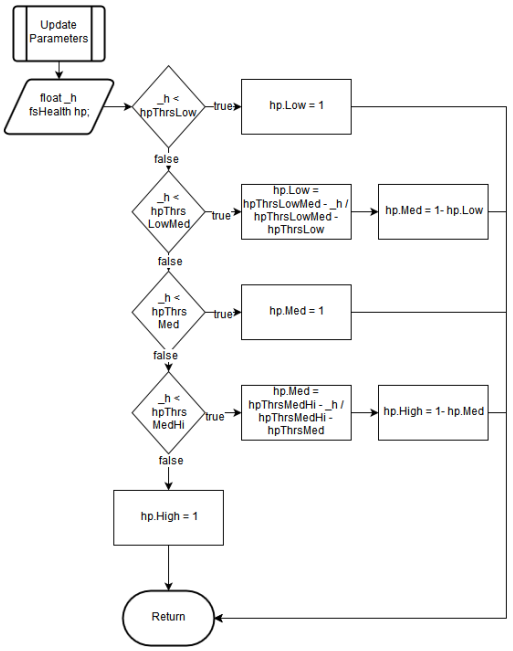
Update dapat disimpulkan kedalam tiga *subprocess* penting, yaitu *update parameters, assess states,* dan *output state*. *Update* dilakukan secara terus menerus selama permainan sedang berjalan. *Update* dilakukan dengan frekuensi yang konstan, dan pada *default* dilakukan sejumlah tiga puluh kali dalam satu menit.

3.3 Desain Update Parameters Process

Update Parameters adalah *proses* yang dilakukan untuk mengambil *input* yang berupa *crisp value* dan mengubahnya menjadi *fuzzy sets* berdasarkan dari *membership function* yang telah didesain. Fungsi ini menerima tiga buah nilai yang merepresentasikan *health, range,* dan *tension* dan mengolahnya menjadi *fuzzy sets* dalam *subprocess* masing-masing.

3.5 Desain Subprocess Update Health

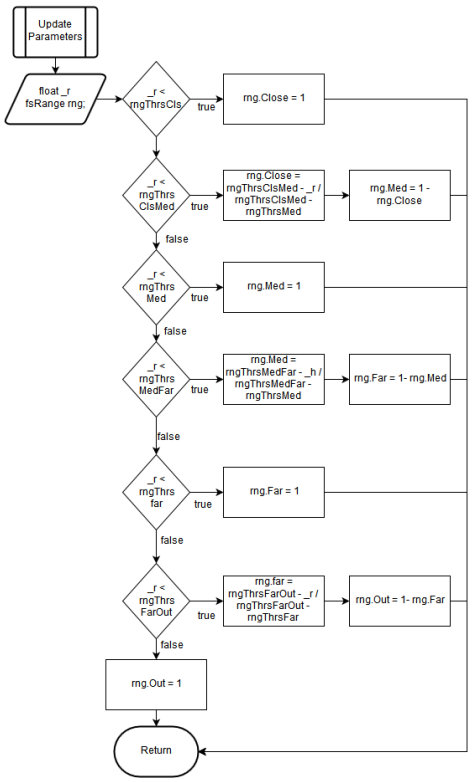
Update Health adalah subproses yang digunakan untuk mengubah health menjadi sebuah fuzzy set health. Subproses dilakukan dengan mengambil nilai health yang berupa crisp value dan mengolah angka tersebut dengan membership function health yang tersedia.



Gambar 6. Flowchart subprocess update health

3.6 Desain Subprocess Update Range

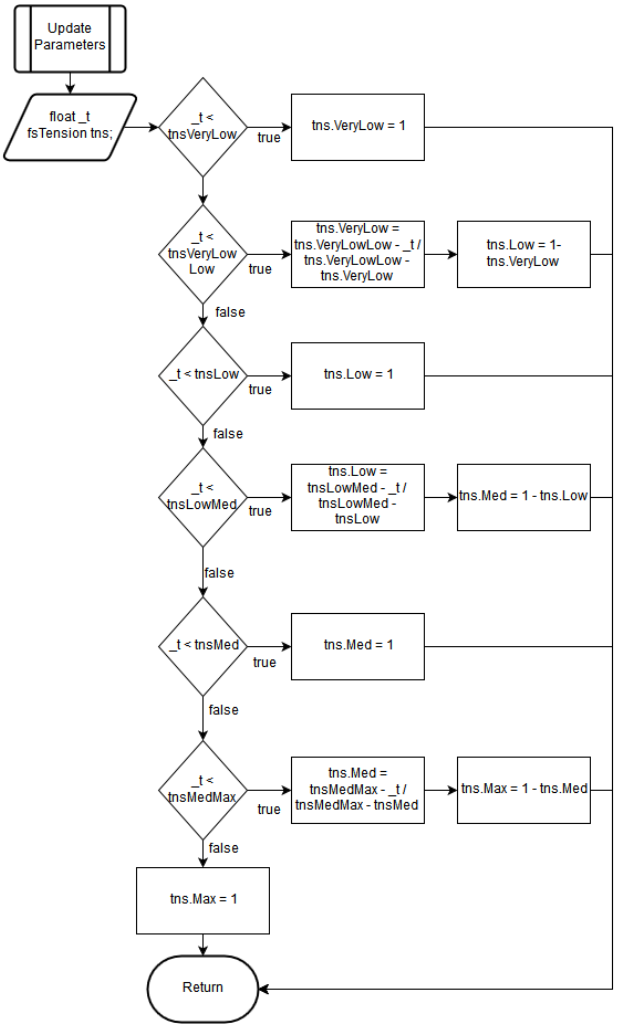
Update Range adalah subproses yang digunakan untuk mengubah range menjadi sebuah fuzzy set range. Subproses dilakukan dengan mengambil nilai range yang berupa crisp value dan mengolah angka tersebut dengan membership function range yang tersedia.



Gambar 7. Flowchart subprocess update range

3.7 Desain Subprocess Update Tension

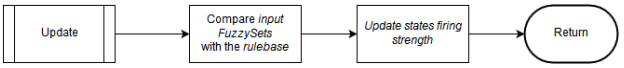
Update Tension adalah subproses yang digunakan untuk mengubah range menjadi sebuah fuzzy set tension. Subproses dilakukan dengan mengambil nilai tension yang berupa crisp value dan mengolah angka tersebut dengan membership function tension yang tersedia.



Gambar 8. Flowchart subprocess update tension

3.8 Desain Proses Assess *State*

Proses *Assess States* dilakukan dengan cara membandingkan *input fuzzy sets* dengan *rulebases* yang telah dibuat. *Fuzzy sets input* yang dipakai adalah *fuzzy set health, range, dan tension*. *Degree of Memberships* dari *input sets* digunakan untuk menentukan *firing strength* dari *state* yang sesuai dengan parameternya. Angka dari *firing strength* sendiri diambil dari *minimum* dari *degree of membership parameter* yang mempengaruhi *states* tersebut.

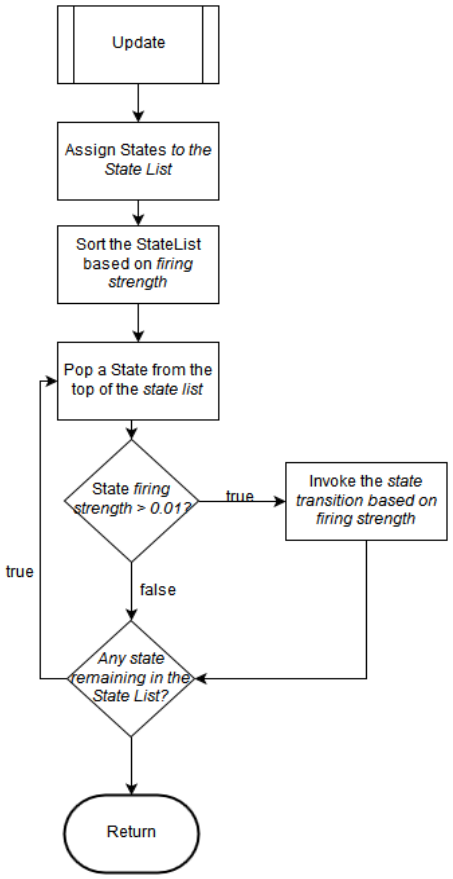


Gambar 9. Flowchart process Assess *State*

3.9 Desain Proses Output *State*

Proses Output *State* dilakukan dengan cara mengambil semua *state*s kedalam *state* list terlebih dahulu. Setelah itu firing strength dari tiap *state*s yang telah dihasilkan oleh fungsi assess *state*s digunakan untuk menentukan urutan *state*s dalam *state* list. *State* disorting dengan pembanding firing strength secara descending, dengan *state* yang mempunyai firing strength tertinggi berada dalam urutan teratas. *State* kemudian diambil dari top list satu persatu. Apabila *state* yang diambil tersebut mempunyai firing strength lebih dari 0.01, *state* dimasukkan

kedalam active *state*s list. *State*s yang berada di active *state*s list akan dilakukan secara berurutan dalam sebuah update period.



Gambar 9. Flowchart proses Output *State*

4. PENGUJIAN SISTEM

4.1 Pengujian Fuzzy *State* Automata

Pengujian dilakukan dengan memberikan nilai input yang berbeda-beda untuk melihat respon gerakan yang diberikan oleh enemy. Input yang digunakan tersebut adalah health, range, dan tension.

Melalui dua puluh kasus yang diamati pada update period 2 detik, perpindahan *state* yang dilakukan oleh fuzzy *state* automata sudah sesuai dengan rulebase yang ada. Parameter yang sudah dimasukkan kedalam membership function juga menghasilkan hasil yang sesuai.

Tabel 1. *States* dari *update period 2s* dimana lama *state <0.8*

No	*State* Change Timing	*State*	Lama *State*
3	1.70s	Idle	0.3s
10	1.44s	Attack	0.56s
12	1.31s	Attack	0.69s
16	1.67s	Alert	0.23s

Adapun *state* hanya dimasuki sekilas saja, yaitu dibawah 0.8 detik sehingga efek dari gerakan yang dihasilkan oleh *state-state* ¬tersebut tidak terlihat. Angka 0.8 detik sebagai takaran tersebut diambil karena semua animasi milik enemy mempunyai durasi sejumlah 0.8 detik. Gambar dari *state-state* tersebut juga menunjukkan bahwa enemy bahkan tidak sempat menyelesaikan animasi dari gerakan sebelumnya.

Selain itu, beberapa *state*s mengalami deviasi dari *state* change timing yang seharusnya. Deviasi ini terjadi diakibatkan sistem update unity game engine yang bersifat bergantung kepada framerate dari computer yang menjalankan. Pada kedua puluh kasus yang dijalankan pada framerate default tersebut, deviasi yang terjadi bisa dilihat pada Tabel 2

Tabel 2. Deviasi *state* pada *update period* 2s

No	Expected *State* Change Timing	*State* Change Timing	Deviation
6	1.18s	1.17s	0.01s
11	1.21s	1.19s	0.02s
12	1.32s	1.31s	0.01s
16	1.68s	1.67s	0.01s

Deviasi dari kasus-kasus yang diamati berkisar antara 0.01s sampai 0.02s, dimana perbedaan tersebut dinyatakan masih layak, dan pemain tidak akan merasakan perbedaan tersebut karena perbedaanya yang terlalu cepat.

Melalui dua puluh kasus yang diamati pada update period 3 detik, perpindahan *state* yang dilakukan oleh fuzzy *state* automata sudah sesuai dengan rulebase yang ada. Parameter yang sudah dimasukkan kedalam membership function juga menghasilkan hasil yang sesuai.

Tabel 3. *States* dari *update period 3s* dimana lama *state <0.8*

No	State Change Timing	State	Lama State	Gambar
4	2.709s	Follow	0.291s	5.24
11	2.326s	Alert	0.674s	5.31

Adapun state hanya dimasuki sekilas saja seperti pada update period 2s, sehingga efek dari gerakan yang dihasilkan oleh state-state tersebut tidak terlihat. Jumlah state yang mempunyai durasi di bawah 0.8 detik yang dipakai sebagai threshold animasi lebih rendah daripada update period 2s.

Selain itu, beberapa states yang mengalami deviasi dari state change timing akibat sistem update unity game engine yang bersifat bergantung kepada bisa dilihat pada Tabel 4

Tabel 4. Deviasi *state* pada update period 3s

No	Expected State Change Timing	State Change Timing	Deviation
3	1.8s	1.772s	0.028s
11	2.34s	2.326s	0.014s
12	2.16s	2.146s	0.014s
20	1.53s	1.508s	0.022s

Deviasi dari kasus-kasus yang diamati berkisar antara 0.01s sampai 0.028s, dimana perbedaan tersebut dinyatakan masih layak, dan pemain tidak akan merasakan perbedaan tersebut karena perbedaanya yang terlalu cepat.

Melalui dua puluh kasus yang diamati pada update period 4 detik, perpindahan *state* yang dilakukan oleh *fuzzy state automata* sudah sesuai dengan *rulebase* yang ada. Parameter yang sudah dimasukkan kedalam *membership function* juga menghasilkan hasil yaitu *output states* yang sesuai.

Pada *update period 4s*, tidak ada state yang mempunyai durasi kurang dari 0.8s, sehingga semua animasi dapat berjalan dengan lancar dan semua gerakan berjalan secara sepenuhnya.

Beberapa *states yang* mengalami deviasi dari *state change timing* akibat sistem *update unity game engine* yang bersifat bergantung kepada bisa dilihat pada Tabel 5.

Tabel 5. Deviasi *state* pada update period 4s

No	Expected State Change Timing	State Change Timing	Deviation
3	2.52s	2.51s	0.01s
5	2.2s	2.187s	0.013s
11	2.04s	2.012s	0.028s
13	2.36s	2.336s	0.024s
15	2.2s	2.171s	0.029s
20	2.4s	2.374s	0.026s

Deviasi dari kasus-kasus yang diamati berkisar antara 0.01s sampai 0.029s, dimana perbedaan tersebut dinyatakan masih layak, dan pemain tidak akan merasakan perbedaan tersebut karena perbedaanya yang terlalu cepat. Akan tetapi, frekuensi terjadinya deviasi dan rata-rata dari deviasi tersebut semakin meningkat semakin tinggi angka *update period*.

Tetapi, pada *update period 4s* beberapa *state* cenderung berjalan terlalu lama. Durasi berjalannya *state* yang lama tersebut mengakibatkan pergerakan *enemy* dan frekuensi *update* terasa terhambat dan tidak lancar.

5. KESIMPULAN DAN SARAN

Berdasarkan hasil pengujian yang telah dilakukan, dapat diambil kesimpulan sebagai berikut:

- Semakin rendah update period yang dipakai untuk menjalankan fuzzy automata, enemy akan berpindah *state* dengan lebih cepat. Kecepatan perpindahan *state* tersebut berarti enemy cenderung bersifat lebih responsif terhadap perubahan parameter. Sisi negatif dari update period yang terlalu rendah adalah perpindahan *state* tidak dapat dilihat jelas secara visual.
- Semakin tinggi update period yang dipakai untuk menjalankan fuzzy automata, enemy akan berpindah *state* dengan lebih lambat. Kecepatan perpindahan *state* yang lambat tersebut berarti gerakan enemy dapat dilihat dengan jelas. Sisi negatif dari update period yang terlalu tinggi adalah enemy bersifat kurang responsif dan cenderung terkesan lambat.
- Update period yang cenderung ideal berada pada range mendekati tiga detik, Enemy masih mengubah *state* dengan cukup cepat untuk terkesan responsif, dan enemy tidak terlalu cepat dalam mengubah *state*nya sehingga gerakan masih bisa diobservasi oleh pemain.

Setelah mengevaluasi keseluruhan dari skripsi ini, beberapa saran yang telah diberikan:

- Game dapat dikembangkan untuk mempunyai kapabilitas multiplatform seperti android sehingga game dapat dimainkan secara luas.
- Fuzzy Automata dalam game dapat dikembangkan dengan menambah jumlah parameter input dan *state*s yang bisa dilakukan.
- Penggabungan dengan metode *neural network* untuk membentuk sebuah model *neuro-fuzzy* untuk melakukan improvisasi gerakan dari *enemy*.

6. DAFTAR PUSTAKA

[1] Chen, Guanrong; Pham, Trung Tat. 2000. *Fuzzy Sets, Fuzzy Logic, and Fuzzy Control Systems.*

[2] Hopcroft, John E.; Motwani, Rajeev; Ullman, Jeffrey D. 2013. *Introduction to Automata Theory, Languages, and Computation (3rd ed.)*. Pearson.

[3] Millington, Ian; Funge, John. 2016. *Artificial Intelligence for Games Second Edition.* CRC Press.

[4] Poole, Mackworth & Goebel. 1998, *Computational Intelligence: A Logical Approach.*

[5] Pirovano, M. 2012. *The use of Fuzzy Logic for Artificial Intelligence in Games.*

[6] Reyneri, L.M. *2005. An Introduction to Fuzzy State Automata.*

[7] Stamenković, Ćirić & Ignjatović. 2012. *Different models of automata with fuzzy states.*

[8] Unity Technologies. 2015. *Unity - Game Engine*. Retrieved November 22, 2015, from http://unity3d.com/