PERILAKU *NON PLAYABLE CHARACTER* (NPC) MUSUH PADA *GAME* SEPEDA MENGGUNAKAN *FUZZY STATE MACHINE* (FuSM)

SKRIPSI

OLEH
FAUZADIN GANSALA
NIM. 09650096

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2016

PERILAKU *NON PLAYABLE CHARACTER* (NPC) MUSUH PADA *GAME* SEPEDA MENGGUNAKAN *FUZZY STATE MACHINE* (FuSM)

SKRIPSI

Diajukan Kepada
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang
untuk Memenuhi Salah Satu Persyaratan Dalam
Memperoleh Gelar Sarjana Komputer (S.Kom)

OLEH
FAUZADIN GANSALA
NIM. 10650006

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2016

PERILAKU *NON PLAYABLE CHARACTER* (NPC) MUSUH PADA *GAME* SEPEDA MENGGUNAKAN *FUZZY STATE MACHINE* (FuSM)

SKRIPSI

Oleh

Nama : Fauzadin Gansala
NIM : 09650096
Jurusan : Teknik Informatika
Fakultas : Sains dan Teknologi

Telah Diperiksa dan Disetujui Diuji
Tanggal, 9 Juni 2016

Pembimbing I,	**Pembimbing II,**
Fresy Nugroho, MT	**Fachrul Kurniawan, M.MT**
NIP. 19710722 201101 1 001	**NIP. 19771020 200912 1 001**

Mengetahui,
Ketua Jurusan Teknik Informatika

Dr. Cahyo Crysdian
NIP. 19740424 200901 1 008

PERILAKU *NON PLAYABLE CHARACTER* (NPC) MUSUH PADA *GAME* SEPEDA MENGGUNAKAN *FUZZY STATE MACHINE* (FuSM)

SKRIPSI

Oleh
FAUZADIN GANSALA
NIM. 09650096

Telah Dipertahankan di Depan Dewan Penguji Skripsi dan
Dinyatakan Diterima Sebagai Salah Satu Persyaratan Untuk
Memperoleh Gelar Sarjana Komunikasi (S.Kom)

Tanggal 27 Juni 2016

Susunan Dewan Penguji		Tanda Tangan
1. Penguji Utama	**: Yunifa Miftachul Arif, M.T** **NIP. 19830616 201101 1 004**	**(…….)**
2. Ketua	**: Hani Nurhayati, M.T** **NIP. 19780625 200801 2 006**	**(…….)**
3. Sekretaris	**: Fresy Nugroho, M.T** **NIP. 19710722 201101 1 001**	**(…….)**
4. Anggota	**: Fachrul Kurniawan, M.MT** **NIP. 19771020 200912 1 001**	**(…….)**

Mengesahkan,
Ketua Jurusan Teknik Informatika

Dr. Cahyo Crysdian
NIP. 19740424 200901 1 008

SURAT PERNYATAAN
ORISINALITAS PENELITIAN

Saya yang bertanda tangan di bawah ini:

Nama : Fauzadin Gansala

NIM : 09650096

Fakultas / Jurusan : Sains dan Teknologi / Teknik Informatika

Judul Penelitian : PERILAKU *NON PLAYABLE CHARACTER* (NPC) MUSUH PADA *GAME* SEPEDA MENGGUNAKAN *FUZZY STATE MACHINE* (FuSM)

Menyatakan dengan sebenarnya bahwa skripsi yang saya tulis ini benar-benar merupakan hasil karya saya sendiri, bukan merupakan pengambil alihan data, tulisan atau pikiran oarang lain yang saya akui sebagai hasil tulisan atau pikiran saya sendiri, kecuali dengan mencantumkan sumber cuplikan pada daftar pustaka. Apabila di kemudian hari terbukti atau dapat dibuktikan skripsi ini hasil jiplakan, maka saya bersedia menerima sanksi atas perbuatan tersebut

Malang, 9 Juni 2016
Yang Membuat Pernyataan,

Fauzadin Gansala
NIM. 09650096

MOTO

"Kegagalan terjadi bila kita menyerah"

{Lessing, Philosof German}

- Menyerah itu bukan menunjukkan besarnya hambatan, tetapi itu cuma menunjukkan besarnya alasan dan kemalasan. -

إِنَّ ٱللَّهَ لَا يُغَيِّرُ مَا بِقَوۡمٍ حَتَّىٰ يُغَيِّرُواْ مَا بِأَنفُسِهِمۡۗ وَإِذَآ أَرَادَ ٱللَّهُ بِقَوۡمٖ سُوٓءٗا فَلَا مَرَدَّ لَهُۥۚ وَمَا لَهُم مِّن دُونِهِۦ مِن وَالٖ ١١

"Sesungguhnya Allah tidak merubah keadaan sesuatu kaum sehingga mereka merubah keadaan yang ada pada diri mereka sendiri. Dan apabila Allah menghendaki keburukan terhadap sesuatu kaum, maka tak ada yang dapat menolaknya; dan sekali-kali tak ada pelindung bagi mereka selain Dia."

(Q.S. Ar-Rad: 11)

PERSEMBAHAN

Bismillaahirrahmaanirrahiim

Yang Utama Dari Segalanya...

Sembah sujud serta syukur kepada Allah SWT. Taburan cinta dan sayang-Mu telah memberikanku kekuatan, membekaliku dengan ilmu serta memperkenalkanku dengan cinta. Atas karunia serta kemudahan Engkau berikan akhirnya skripsi yang sederhana ini dapat terselesaikan. Sholawat dan selalu terlimpahkan kehariban Rasulullah Muhammad SAW.

Kupersembahkan karya sederhana ini kepada semua pihak yang telah membantu dalam menyelesaikannya

- Ayahanda A. Asrori Fatchullah dan Ibunda Mutimmah (Alm) yang senantiasa dengan penuh kesabaran dan kasih sayang membinmbingku dalam setiap langakah hidupku
- Kakak-kakakku tercinta Faiq Maulana, Aulia Nada R, Ninik Azizah, Zakka Maziyyati, Diana Sholihah, Hendra P yang telah banyak membantuku dan selalu memberiku motivasi. Serta adikku Intan Fakhrun Ni'am yang menjadi motivasiku untuk menjadi teladan yang baik
- Sahabat dan saudara kelas C Teknik Informatika 2009 yang selalu menjadi kekuatan dalam semangat
- Nazar Pesona Wildan sebagai rekan tim yang solid dalam pembuatan *game*
- Teman seperjuangan di Malang, Bang Jek (Zakki), Andang, Syamsul
- Serta rekan-rekan CV.Khasanah Konsultama: Mas hamdan, Heni, Pepenk, Mas Nasch, Bang Zakki, Bang Beben, Rizqi, Fevi, Doni, Afi dan Mas Ubet yang senantiasa tiada hentinya memberikan support.
- Kepada setiap orang yang telah membantu

Baarakallah

KATA PENGANTAR

Assalaamu'alaikum Warahmatullaahi Wabaarakaatuh

Segala puji bagi Allah SWT atas rahmat, taufik serta hidayah-Nya, sehingga penulis mampu menyelesaikan peyusunan skripsi ini sebagai salah satu syarat untuk memperoleh gelar sarjana dalam bidang teknik informatika di Fakultas Sains dan Teknologi, Universitas Islam Negeri Maulana Malik Ibrahim Malang.

Shalawat serta salam semoga senantiasa Allah limpahkan kepada Nabi Muhammad SAW, keluarga, sahabat dan ahlinya yang telah membimbing umat menuju kebahagiaan dunia dan akhirat.

Penulis menyadari adanya banyak keterbatasan yang penulis miliki dalam proses penyusunan skripsi ini, sehingga penulis banyak mendapat bimbingan dan arahan dari berbagai pihak. Untuk itu ucapan terima kasih yang sebesar-besarnya dan penghargaan setinggi-tingginya penulis sampaikan terutama kepada:

1. Prof. Dr. H. Mudjia Rahardjo, M.Si, selaku rektor Universitas Islam Negeri Maulana Malik Ibrahim Malang.
2. Dr. Hj. Bayyinatul Muchtaromah., drh. M.Si selaku Dekan Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang.
3. Fresy Nugroho, M.T dan Fachrul Kurniawan, M.MT selaku dosen pembimbing I dan II yang telah meluangkan waktu untuk membimbing,

memotivasi, mengarahkan dan memberi masukan dalam pengerjaan skripsi ini.

4. Segenap sivitas akademika Jurusan Teknik Informatika, terutama seluruh dosen, terima kasih atas segenap ilmu dan bimbingannya.
5. Bapak dan Ibuku tercinta, adikku dan seluruh keluarga besar yang senantiasa memberikan doa dan restunya kepada penulis dalam menuntut ilmu serta dalam menyelesaikan skripsi ini.
6. Semua pihak yang tidak mungkin penulis sebutkan satu-persatu, atas segala yang telah diberikan, penulis ucapkan terima kasih yang sebesar-besarnya.

Sebagai penutup, penulis menyadari dalam skripsi ini masih banyak kekurangan dan jauh dari sempurna, untuk itu peneulis selalu menerima segala kritik dan saran dari pembaca. Harapan penulis, semoga karya ini bermanfaat bagi kita semua.

Wasslaamu'alaikum Warahmatullahi Wabarakaatuh

Malang, 9 Juni 2016

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

ABSTRAK

Gansala, Fauzadin. 2016. **Perilaku *Non Playable Character* (NPC) Musuh Pada *Game* Sepeda Dengan Menggunakan *Fuzzy State Machine* (FuSM).** Skripsi. Jurusan Teknik Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang.

Pembimbing: (I) Fresy Nugroho, M.T (II) Fachrul Kurniawan, M.MT

Kata Kunci: *game sepeda, simulasi, AI, NPC musuh, fuzzy state machine, perilaku menyerang.*

Perkembangan dunia *game* dari masa ke masa terus mengalami perkembangan, hingga kini bermunculan beberapa *Genre game* dari *text based* hingga yang memiliki animasi 3D. Sebuah *game* tidak hanya berkembang pada *Interface* saja, kemampuan *game* untuk menjadi agen *learning* menjadikan sebuah *game* itu lebih hidup. *Agen learning* dalam *game* dikenal sebagai *Artifical intelegent (AI)*. AI tersebut di kembangkan untuk merancang perilaku otonom agen atau *Non Playable Character (NPC)* dari *game* atau simulasi yang seakan-akan NPC tersebut mempunyai kecerdasan dan pergerakan sealami mungkin.

Penelitian ini membahas perubahan perilaku menyerang NPC Musuh pada *game* sepeda yang di atur oleh *Fuzzy State Machine* (FuSM), sehingga ada interaksi antara karakter pemain dan NPC musuh. *Game* 3D yang dibuat ini bertujuan sebagai media simulasi. *Game* dibuat menggunakan *game* engine Unity 3D, dan desain FuSM perilaku NPC Musuh dibuat menggunakan software MATLAB. Hasil uji coba implementasi FuSM pada perilaku menyerang NPC Musuh d sesuai dengan desain output perilaku yang didesain sebelumnya, yaitu patrol, mengejar dan menyerang untuk NPC Musuh.

ABSTRACT

Gansala, Fauzadin. 2016. **Behavior of *Non Playable Character* (NPC) Enemy In *Game* Bike Using *Fuzzy State Machine* (FuSM)**. Thesis. Informatics Department of Faculty of Science and Technology. Maulana Malik Ibrahim State Islamic University, Malang.

Adviser: (I) Fresy Nugroho, M.T (II) Fachrul Kurniawan, M.MT

Keywords: *bike games, simulation, AI, NPC enemy, fuzzy state machine, attack behavior*.

The development of the *game* from time to time continue to experience growth, up to now emerge some of the text based *game* genre to which has a 3D animation. A *game* is not only growing in the Interface only, the *game*'s ability to be an agent of learning that makes a *game* more alive. Agent learning in the *game* known as Artifical intelligent (AI). The AI was developed to design autonomous behavior agent or Non Playable Character (NPC) from the *game* or simulation as if the NPC has the intelligence and movement as natural as possible.

This study discusses the change in behavior in the *game* attacking enemy NPC bikes set by *Fuzzy* State Machine (FuSM), so there is an interaction between the *player* character and enemy NPCs. 3D *game*s are created is intended as a media simulation. *Game*s created using the Unity 3D *game* engine and design FuSM Enemy NPC behavior created using MATLAB software. The trial results on the implementation FuSM NPC Enemies attack behavior d in accordance with the design output behavior designed previously, namely patrol, chasing and attacking the enemy's NPC.

الملخص

غنسالا, فؤازالدين, أعمال الشخصية غير قابلة (NPC) في لعبة الدراجة باستخدام آلة الدولة فوزي (FuSM). البحث الجامعي. قسم التقنية المعلوماتية كلية العلوم والتكنولوجيا الجامعة الإسلامية الحكومية مولانا مالك إبراهيم مالانج.

المشرف: (1) فريشي نوغروهو الماجيستر، (2) فخر الكورنيوان الماجيستر

كلمات البحث: ألعاب الدراجة، المحاكاة، (الشخصية غير قابلة) NPC عدو، *AI*, آلة الدولة فوزي، أعمال الهجوم.

تطوير اللعبة من وقت إلى وقت لا تزال تشهد نموا، حتى تظهر الآن بعض من نوع اللعبة من النص على أساس إلى الذي لديه الرسوم المتحركة 3D. اللعبة لا تنمو إلا في واجهة فقط، قدرة اللعبة لتكون وكيلة للتعلّم تجعل اللعبة أحياة. تعلم وكيلة للتعلّم في اللعبة معروفة باسم الاصطناعي الذكي (AI). وقد وضعت (AI) المذكور في تصيم لابتكار الأعمال المستقلة أو (الشخصية غير قابلة) NPC من اللعبة أو محاكاة كأن NPC المذكور لديه ذكاء والحركة الطبيعية قدر الإمكان.

تبحث هذه الدراسة بأن التغير في أعمال الهجوم NPC للعدو في لعبة الدراجات التي وضعتها آلة الدولة فوزي (FuSM)، حتى يكون هناك تفاعل بين حرف لاعب و NPC للعدو. تنشاء هذه الألعاب D3 بالقصد به أن تكون وسائل المحاكاة. الألعاب المنشؤة باستخدام محرك اللعبة الوحدة 3D و تصميم FuSM أأعمال NPC للعدو المنشؤ باستخدام برمجيات MATLAB. نتائج الاختبارة في تنفيذ FuSM على أعمال الهجوم NPC للعدو وفقا تصميم النتائج للأعمال المصممة سابقا، وهي دورية، ومطاردة ومهاجمة NPC للعدو.

BAB I

PENDAHULUAN

1.1 Latar Belakang

Perkembangan dunia *game* dari masa ke masa terus mengalami perkembangan, hingga kini bermunculan beberapa *genre game* dari *text based* hingga yang memiliki animasi 3D. Sebuah *game* tidak hanya berkembang pada *Interface* saja, kemampuan *game* untuk menjadi agen *learning* menjadikan sebuah *game* itu lebih hidup. *Agen learning* dalam *game* dikenal sebagai *Artifical intelegent (AI).*

AI tersebut di kembangkan untuk merancang perilaku otonom agen atau *Non Playable Character (NPC)* dari *game* atau simulasi yang seakan-akan NPC tersebut mempunyai kecerdasan dan pergerakan sealami mungkin. Keberadaan NPC sendiri dalam suatu *game* merupakan salah satu faktor dan komponen penting dalam komputer modern yang menentukan *game* itu menarik atau tidak. Konsep agen cerdas merupakan salah satu model yang digunakan dalam membuat NPC. Sifat otonom dari agen cerdas merupakan keunggulan dalam memodelkan suatu NPC *game*. Salah satu metode atau kecerdasan buatan yang dapat digunakan untuk menentukan perilaku NPC yaitu *Fuzzy State Machine* (FuSM).

Penelitian kali ini juga akan memakai *Fuzzy State Machine* sebagai kecerdasan *game*, akan tetapi dengan konten edukasi berbeda yaitu simulasi dalam *game* sepeda. *Fuzzy State Machine* tidak diterapkan pada peperangan yang nantinya cenderung adanya aksi kekerasan, akan tetapi

diterapkan pada perilaku NPC musuh. Perilaku NPC musuh lebih variatif didapatkan dari algoritma *Fuzzy* yang digunakan. *Fuzzy State Machine* sendiri adalah gabungan dari metode *Fuzzy* dan *Finite State Machine* (FSM). *Fuzzy* adalah algoritma yang mengidentifikasi nilai samar atau daerah ketidakpastian dan *FSM* adalah alur sistem state per state yang ada pada *game*. Kelebihan *Fuzzy State Machine* adalah dihasilkannya output perilaku pada NPC yang lebih variatif dari pada FSM biasa, jika FSM memiliki output ON dan OFF, maka *Fuzzy State Machine* memiliki state diantara ON dan OFF.

Melihat *statement* diatas maka dilakukan penelitian ini guna mengembangkan penelitian yang sebelumnya dilakukan oleh Haris Budi (Budi: 2014) yaitu tentang Navigasi *Player* Untuk Pencarian *Obstacle* Pada *Game* Sepeda, yang mana *game* tersebut masih terlihat monoton hanya ada karakter utama saja sehingga terlihat membosankan ketika dimainkan. Oleh sebab itu agar terlihat menarik nantinya akan dikembangkan dengan menambah sebuah "NPC musuh" dalam *game* tersebut, sehingga dapat berinteraksi dengan karakter utamanya.

1.2 Rumusan Masalah

Bagaimana implementasi *Fuzzy State Machine (FuSM)* sebagai kecerdasan NPC musuh sehingga berperilaku natural?

1.3 Batasan Masalah

Permasalahan yang akan dibahas kiranya perlu dilakukan pembatasan terhadap masalah tersebut, hal ini dilakukan agar dalam penelitian ini terfokus

pada permasalahan yang akan diteliti dan tidak melebar pada luar batas penelitian. Adapun batasan masalah dalam penelitian ini antara lain:

1. Strategi gerak yang di bahas dalam penelitian ini adalah gerak pada NPC musuh.
2. Disimulasikan menggunakan *game engine Unity 3D*

1.4 Tujuan Penelitian

Adalah mengimplementasikan *Fuzzy State Machine* (FuSM) sebagai kecerdasan NPC musuh sehingga berperilaku natural.

1.5 Manfaat Penelitian

Hasil dari penelitian ini diharapkan akan memberikan manfaat terhadap pengembangan *game* di Indonesia antara lain:

1. Mengembangkan teknologi virtual untuk simulasi di Indonesia
2. Membuat media belajar berupa *game* interaktif

1.6 Metode Penelitian

Berikut adalah langkah-langkah metode yang digunakan dalam penelitian, terdiri dari:

1. Analisis

   Pada tahap ini menganalisa setiap permasalahan yang akan muncul dalam pengembangan pembuatan *game* ini, diantaranya:

   1) Identifikasi masalah

Mengidentifikasi kelemahan dan kelebihan pada *game* yang sudah ada sebelumnya.

2) Analisis masalah

Setelah semua masalah teridentifikasi, kemudian di analisis untuk menentukan solusi dari masalah tersebut.

3) Analisis literatur

Dalam memecahkan masalah, kita akan mendapatkan solusi dari beberapa sumber referensi yang berkaitan dengan penelitian yang dilakukan, pada penelitian ini topik yang dikaji diantaranya: algorithma *Fuzzy State Machine* (FuSM), beserta materi pendukung dalam pembuatan *game*.

2. Desain

Pada tahap ini membahas tentang pengembangan desain sistem pada *game* yang sudah ada sebelumnya, meliputi:

1) Pengembangan pembuatan desain lapangan virtual yang digunakan.

Membuat perubahan desain dari *game* sudah ada sebelumnya.

2) Pembuatan desain *output*.

Output yang akan dihasilkan adalah respon NPC musuh terhadap *player.*

3) Pembuatan desain *input*.

Input pada *game* untuk di gunakan menggunakan *Fuzzy State Machine.*

4) Pembuatan desain proses.

Tahapan pada sistem untuk menghasilkan respon NPC musuh pada *game* menggunakan *Fuzzy State Machine.*

5) Pembuatan desain antarmuka pada *Game engine*

Rancangan desain *game* dan GUI akan di gambarkan di sini.

3. Implementasi

Pada tahap ini membahas tentang implementasi dari desain sistem pada tahapan sebelumnya.

1) Implementasi algoritma

Mengimplementasikan algoritma FuSM pada *game* untuk proses perubahan perilaku NPC musuh terhadap *player*

2) Perancangan dan pembuatan *game*

Merancang *game* menggunakan FuSM pada proses respon perilaku NPC musuh

3) *Debugging*

Melakukan pembenahan pada sistem yang mengandung *bug* agar tidak mengalami kesalahan ketika bermain.

4. Ujicoba

Ujicoba *game* pada *player* secara langsung.

5. Pembuatan laporan

Pembuatan laporan skripsi. Sebagai dokumentasi tugas akhir.

1.7 Sistematika Penulisan Skripsi

Sistematika dalam penulisan skripsi ini akan dibagi menjadi beberapa bab sebagai berikut:

BAB I Pendahuluan

Pada bab ini berisi latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian, dan sistematika penulisan tugas akhir ini.

BAB II Tinjauan Pustaka

Pada bab ini menjelaskan tentang dasar-dasar teori yang digunakan sebagai penunjang untuk penyusunan tugas akhir ini. Dalam dasar teori yang akan dibahas yaitu dasar teori yang berkaitan dengan permasalahan yang diambil.

BAB III Analisis, dan Perancangan Sistem

Bab ini menjelaskan mengenai analisa *game* yang sudah ada dan perancangan pengembangan *game* yang akan dibuat nantinya seperti apa.

BAB IV Hasil Dan Pembahasan

Bab ini membahas tentang implementasi dari algoritma pada NPC musuh terhadap pengembangan *game* sepeda yang sudah ada sebelumnya.

BAB V Penutup

Bab ini berisi kesimpulan dari keseluruhan dari laporan tugas akhir dan saran yang diharapkan dapat bermanfaat untuk pengembangan pembuatan *game* selanjutnya.

BAB II

TINJAUAN PUSTAKA

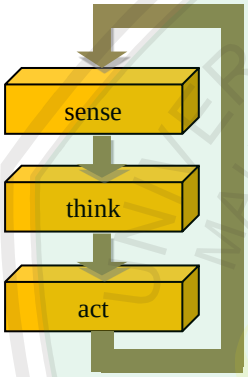
2.1 *Non Playable Character* (NPC)

Autonomous character adalah jenis *autonomous agent* yang ditujukan untuk penggunaan komputer animasi dan media interaktif seperti *games*, simulasi 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 permaian, karakter otonom biasanya disebut NPC (*Non Playable Character*) (Reynold).

Yunifa Mifatchul Arif (Arif, Kurniawan and Nugroho) meneliti pergantian senjata NPC pada *game* FPS menggunakan *fuzzy sugeno*. Penelitian lain juga dilakukan oleh Ady Wicaksono dkk (Wicaksono, Hariadi and Supeno) yang meneliti tentang pergantian senjata NPC pada *game* FPS menggunakan *fuzzy state machine*. Selain itu ada penelitian yang dilakukan oleh Alberto Alvarez (Alvarez) yang mana meneliti tentang pemodelan gaya atau cara berjalan seseorang manusia dengan menggunakan *fuzzy state machine* sebagai pembangkitnya. Nantinya juga dapat dikembangkan kembali dengan menggabungkan beberapa metode

untuk mendapat model gaya seorang manusia yang lebih variatif sesuai kondisi lingkungan sekitarnya.

Perilaku NPC pada dasarnya akan berperilaku dengan cara mengulangi tiga tahap yaitu *sense, think,* dan *act* (Kim: 2006). Seperti diperlihatkan pada Gambar 2.1 yang menunjukkan langkah dan aktivitas perilaku NPC.



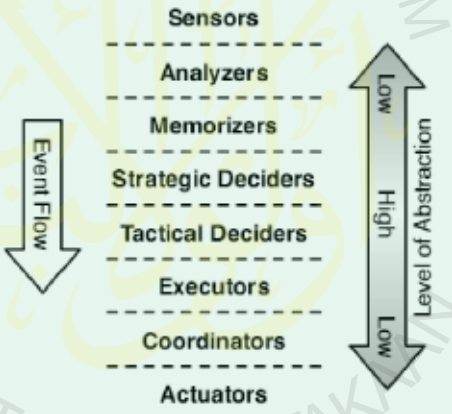
Tahap	Aktivitas
Sense	Memahami situasi melalui yang terjadi saat itu pada sistem.
Think	Menentukan aksi apa yang tepat dan sesuai dengan aturan dalam situasi tersebut.
Act	Memerintahkan sistem untuk melakukan perbuatan atau aksi yang telah *ditentukan.*

Gambar 2.1 Tahap perilaku NPC (Kim: 2006)

2.2 *Modelling AI Game*

Arsitektur model AI digambarkan dalam Gambar 2.2, pada *level* pertama mengandung komponen yang mewakili *sensor* yang memungkinkan karakter untuk mengamati lingkungan serta *state* sendiri. *Sensor* menyaring informasi dan peristiwa serta mengirimnya ke tingkat berikutnya. Tingkat kedua berisi komponen *analyzer* yang menganalisis atau menghubungkan kejadian dari individu sensor, yang mungkin mengarah pada peristiwa generasi selanjutnya. Komponen *memorizer* bertugas menyimpan peristiwa yang telah terjadi. *Strategic decider* 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, *tactical desciders* merencanakan bagaimana membuat strategi yang dipakai sekarang dapat berjalan dengan baik. *Executor* atau pelaksana kemudian menerjemahkan keputusan dari *tactical decider* untuk perintah tingkat rendah *(low-level command)* sesuai dengan batasan yang digunakan oleh permainan atau simulasi. Komponen *coordinator* memahami hubungan antar-*actuators* dan mungkin kembali memberikan perintah tingkat rendah lebih lanjut. Akhirnya, *actuator* melakukan tindakan yang dinginkan. Bahasan ini penelitian ini berada pada tingkat komponen *tactical decider* (Kienzle: 2007).



Gambar 2.2 *AI model architecture* (Kienzle: 2007)

2.3 Logika *Fuzzy*

Logika *fuzzy* merupakan pengembangan dari logika klasik, dimana nilai kebenarannya berada pada interval [0.1]. Logika *Fuzzy* adalah algoritma yang mengidentifikasi nilai samar atau daerah ketidakpastian. Logika ini diperkenalkan pertama kali pada tahun 1965 oleh Lotfi A. Zadeh yang merupakan seorang professor dari University of California di

Berkley melalui makalahnya yang berjudul "*Fuzzy Sets"*. Pada saat itu logika *fuzzy* diperkenalkan bukan sebagai metodologi untuk mengatur, tetapi merupakan suatu cara untuk memproses data dimana himpunan keanggotaan parsial diperbolehkan, sebagai pengganti himpunan keanggotaan atau bukan keanggotaan yang *crisp*.

Definisi formal logika *fuzzy*, logika *fuzzy* merupakan sebuah logika yang dipresentasikan oleh ekspresi *fuzzy* (rumus) yang memenuhi kriteria sebagai berikut:

i. Nilai kebenaran, 0 dan 1, dan vaiabel $x_i (\in [0,1], i = 1,2,..n)$ merupakan ekspresi *fuzzy*.

ii. Jika f merupakan ekspresi *fuzzy*, $\sim f$ juga merupakan ekspresi *fuzzy*.

iii. Jika f dan g merupakan ekspresi *fuzzy*, $f \wedge g$ dan $f \vee g$ juga merupakan ekspresi *fuzzy* (Lee, 2005:201).

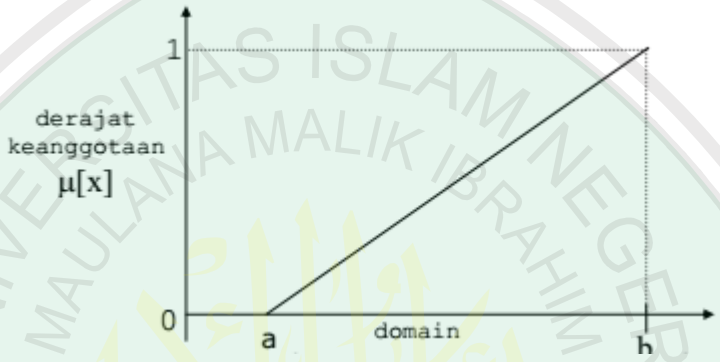
Fungsi Keanggotaan (membership function) adalah suatu kurva yang menunjukkan pemetaan titik-titik input data ke dalam nilai keanggotaannya (sering juga disebut dengan derajat keanggotaan) yang memiliki interval antara 0 sampai 1. Salah satu cara yang dapat digunakan untuk mendapatkan nilai keanggotaan adalah dengan melalui pendekatan fungsi. Ada beberapa fungsi yang bisa digunakan (Dewi, 2003).

a) Representasi Linear

Pada representasi linear, pemetaan input ke derajat keanggotannya digambarkan sebagai suatu garis lurus. Bentuk ini paling sederhana

dan menjadi pilihan yang baik untuk mendekati suatu konsep yang kurang jelas.

Ada 2 keadaan himpunan *fuzzy* yang linear. Pertama, kenaikan himpunan dimulai pada nilai domain yang memiliki derajat keanggotaan nol [0] bergerak ke kanan menuju ke nilai domain yang memiliki derajat keanggotaan lebih tinggi



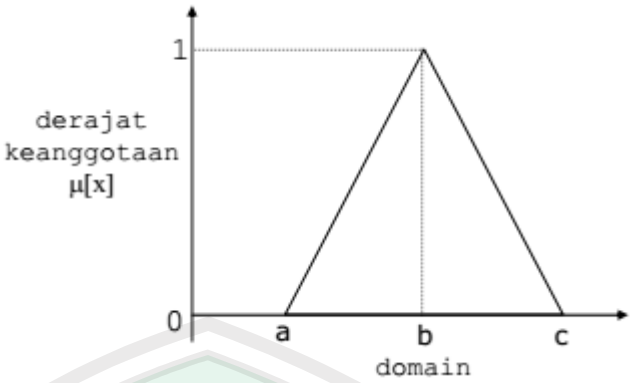
Gambar 2.3 Representasi Linear Naik (Dewi: 2003)

Fugsi Keanggotaa:

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

b) Represesntasi Kurva Segitiga

Kurva Segitiga pada dasarnya merupakan gabungan antara 2 garis (linear) seperti terlihat pada Gambar berikut:



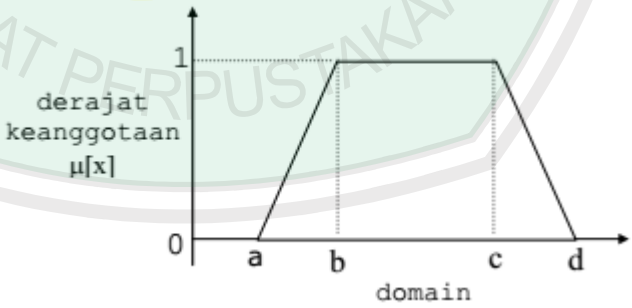
Gambar 2.4 Kurva Segitiga (Dewi: 2003)

Fungsi Keanggotaan:

$$\mu[x] = \begin{cases} 0; & x \le a \text{ atau } x \ge c \\ (x-a)/(b-a); & a \le x \le b \\ (b-x)/(c-b); & b \le x \le c \end{cases}$$

c) Representasi Kurva Trapesium

Kurva Segitiga pada dasarnya seperti bentuk segitiga, hanya saja ada beberapa titik yang memiliki nilai keanggotaan 1



Gambar 2.5 Kurva Trapesium (Dewi: 2003)

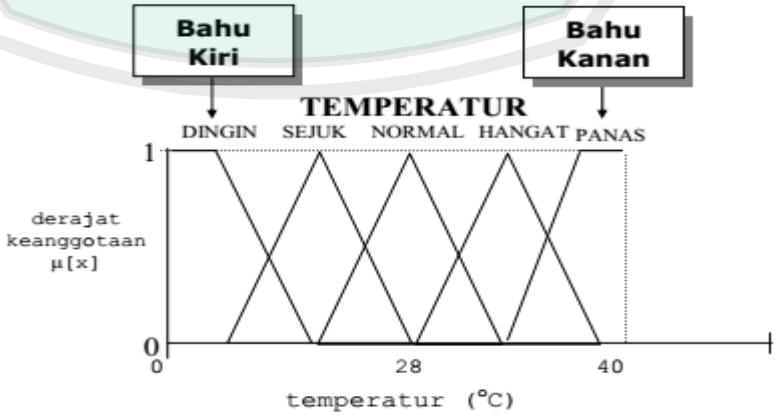
Fungsi Keanggotaan:

$$\mu[x] = \begin{cases} 0; & x \le a \text{ atau } x \ge d \\ (x - a)/(b - a); & a \le x \le b \\ 1; & b \le x \le c \\ (d - x)/(d - c); & x \ge d \end{cases}$$

d) Representasi Kurva Bentuk Bahu

Daerah yang terletak di tengah-tengah suatu variabel yang direpresentasikan dalam bentuk segitiga, pada sisi kanan dan kirinya akan naik dan turun (misalkan: DINGIN bergerak ke SEJUK bergerak ke HANGAT dan bergerak ke PANAS). Tetapi terkadang salah satu sisi dari variabel tersebut tidak mengalami perubahan. Sebagai contoh, apabila telah mencapai kondisi PANAS, kenaikan temperatur akan tetap berada pada kondisi PANAS. Himpunan *fuzzy* 'bahu', bukan segitiga, digunakan untuk mengakhiri variabel suatu daerah *fuzzy*. Bahu kiri bergerak dari benar ke salah, demikian juga bahu kanan bergerak dari salah ke benar.

Gambar berikut menunjukkan variabel TEMPERATUR dengan daerah bahunya.



Gambar 2.6 Daerah ''bahu'' pada variabel TEMPERATUR (Dewi: 2003)

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 .....*

Aturan tersebut menyatakan bahwa A_1,…, A_n, *B* adalah nilai-nilai linguistik, sedangkan *“* x_1 *is* A_1 *“* menyatakan bahwa nilai dari variabel *x1* adalah anggota *fuzzy* set A_1.

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* $y = f(x_1 , \ldots , x_n)$ .....

Dimana *f* bisa berupa sembarang fungsi dari variabel-variabel masukan yang nilainya berada dalam interval variabel keluaran. Dari penjelasan tentang logika *fuzzy* dapat diketahui bahwa suatu sistem yang menggunakan logika *fuzzy* mampu menangani suatu masalah ketidakpastian dimana masukan yang

diperoleh merupakan suatu nilai yang kebenarannya bersifat sebagian (Dewi, 2003).

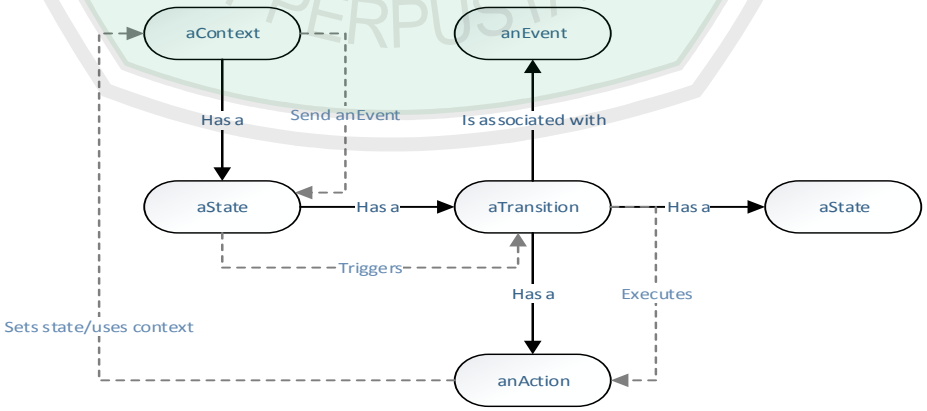
Atas dasar itulah pada penelitian ini, logika *fuzzy* digunakan dengan tujuan untuk mendapatkan respon perilaku NPC berdasarkan variabel *input* yang dimiliki. Selanjutnya dalam menentukan perubahannya digunakan model *Fuzzy* Sugeno, dimana setiap *output* perilaku NPC musuh diwakili oleh konstanta tetap yang sudah ditentukan nantinya.

2.4 *Finite State Machine* (FSM)

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 event*, 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* (Yunifa dkk: 2012).

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 (Lee: 1998). *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 berikut:



Gambar 2.7 Framework *Finite State Machine*

(Sumber: Yunifa dkk, 2012)

2.5 *Fuzzy State Machine* (FuSM)

Untuk mengurangi tingkat prediksi dalam perilaku *Artificial Intelligent* (AI) adalah dengan cara memungkinkan agen AI untuk menggabungkan beberapa perilaku pada waktu yang sama. Hal ini bisa dicapai melalui penggunaan *fuzzy logic* untuk mengimplementasikan *Fuzzy State Machine* (FuSM). FuSM menyatukan logika *fuzzy* dan FSM. FuSM memberikan derajat yang berbeda keanggotaan masing-masing *state*. Oleh karena itu, bukan hanya *state* on / off atau hitam / putih, FuSM dapat di *state* 'on' atau hampir 'off' (Sweetser, Penelope & Wiles, Janet: 2002). FuSM mirip dengan FSM biasa tetapi menggunakan logika *fuzzy* untuk menangani transisi antara *state* dan / atau tindakan di dalam *state-state*.

Generalisasi dari *finite state machine* menjadi *fuzzy state machine* adalah cukup mudah. Sebuah *state fuzzy* adalah sebuah *state* dengan derajat kemungkinan keanggotaan $0 \leq \mu \leq 1$, sebagai lawan dari *state* (*crisp*) dengan derajat implisit dari keanggotaan nol atau satu. Artinya, *fuzzy state machine* memungkinkan sistem untuk menjadi bagian dalam kondisi saat ini.

FuSM punya kelebihan dibanding dengan FSM biasa, yaitu *state*nya sangat modular, karena kita cukup fokus pada behaviour *state* tersebut, tidak perlu memikirkan *state* lain. Selain itu juga tidak akan tercipta *state oscillation* seperti pada FSM. Transisi diwakili oleh aturan

fuzzy. Transisi menerima derajat keanggotaan. FuSM bisa berada di lebih dari satu *state* pada saat yang sama waktu. (Adriano Cruz: 2008)

Dan beberapa poin penting yang harus diperhatikan sebelum masuk lebih dalam ke FuSM ini adalah:

1. FuSM sedikit mengambil istilah-istilah dari *Fuzzy* Logic, misalnya: istilah *fuzzy* itu sendiri & istilah *Degree of Membership* (DOM) dari *fuzzy* logic diganti menjadi *Degree of Activation* (DOA).
2. Terdapat *Degree of Activation* (DOA) yang akan menentukan apakah suatu *state* dapat diaktifkan atau tidak.

2.6 Unity 3D

Unity3D adalah salah satu *game engine cross-platform*. *Engine* ini mendukung tiga bahasa dengan *framework Mono open source*, C#, JavaScript dan Phyton. Pada Unity3D terdapat fasilitas yang memudahkan untuk penggolongan seperti, *asset*, animasi, tekstur, suara dan juga memungkinkan untuk mengimport model 3D dari aplikasi modeller lain, seperti Blender, untuk membuat *real-time* grafis menggunakan mesin *rendering* buatan sendiri dan juga dikombinasikan dengan *nVidia PhysX physics engine.*

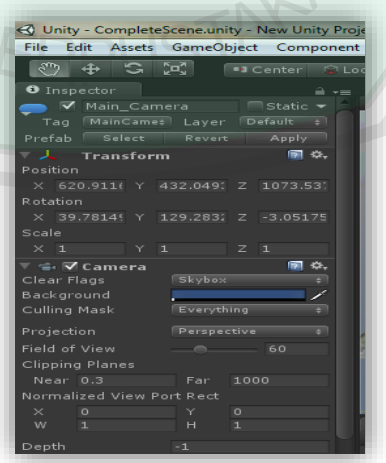
2.6.1 Antar Muka dan Unity 3D Control

Gambar 2.8 Antar muka keseluruhan Unity 3D

(Sumber: Tim Litbang Wahana Komputer, 2014.)

- *Inspector*

  Berisi tentang semua detail objek yang di sorot pada *scene*, di dalamnya kita bisa mengatur berbagai hal, seperti ukuran objek, letak koordinat objek, model kamera dan lain-lain. Control objek bisa menggunakan C# yang mana dapat mengendalikan dari *input*an yang telah di berikan.

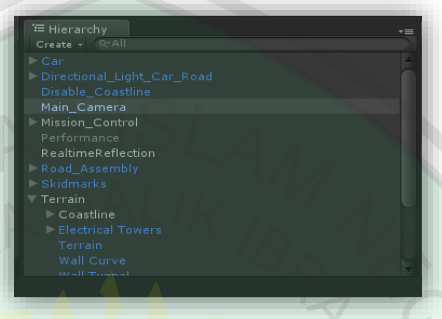


Gambar 2.9 Tampilan *inspector* pada Unity 3D

(Sumber: Tim Litbang Wahana Komputer, 2014)

- *Hierarchy*

  Hierarchy berisi tentang semua yang ada pada *game*, dan juga menggabungkan antara objek satu dengan yang lainnya, seperti halnya menggambarkan turunan dari suatu objek tertentu

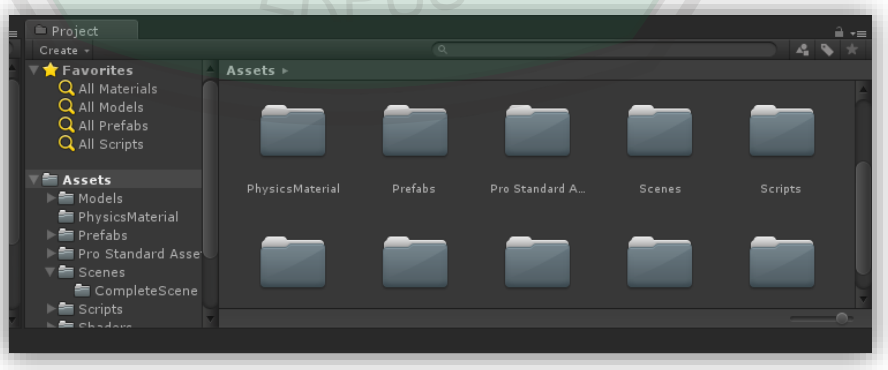


Gambar 2.10 Tampilan *hierarchy* pada Unity 3D

(Sumber: Tim Litbang Wahana Komputer, 2014)

- *Project*

  Project menampilkan semua file yang sudah kita buat pada proyek, pada gambar 4 dapat dilihat pada folder *asset* terdapat model, *texture*, *scene*, *script* dan lain lain.



Gambar 2.11 Tampilan project pada Unity 3D

(Sumber: Tim Litbang Wahana Komputer, 2014)

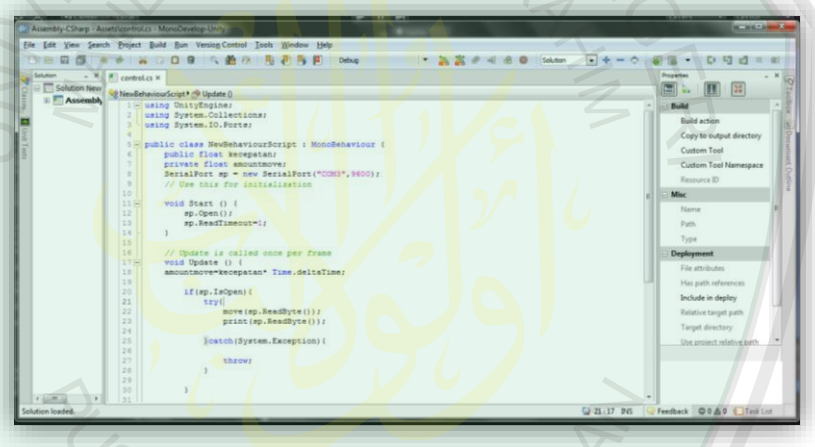
- *Scene*

  Scene menampilkan lembar kerja untuk memanipulasi objek yang ada pada tampilan, meliputi melakukan penggeseran objek, rotasi objek edit ukuran, perspektif objek.

- *Game*

  Untuk menampilkan objek hasil dari manipulasi *scene* secara *realtime.*

2.6.2 Antar Muka *Develop*



Gambar 2.12 Antar muka *develope* untuk kode program

(Sumber: Tim Litbang Wahana Komputer, 2014)

Pada *Game engine* untuk menggunakan Mono develop sebagai IDE untuk keperluan scripting pada *game* yang akan di buat, di dalam *game engine* tersedia 3 pilihan bahasa pemrograman diantaranya C# Script , JavaScript, dan Boo Script ,Fungsi Monodevelop adalah di gunakan untuk memudahkan kita dalam menulis sebuah program / script yang akan di masukkan pada *game*, kemudahan tersebut diantaranya adalah

mengkoreksi adanya kesalahan penulisan kode, dalam penulisan kode program biasanya tidak luput dalam penulisan huruf atau kekurangan suatu atribut, selain itu digunakan untuk debugging program, sehingga memudahkan proses penanganan error, dan mempermudah proses perbaikannya.

BAB III

DESAIN DAN IMPLEMENTASI

3.1 Desain Sistem

3.1.1 Keterangan Umum *Game*

Game yang dibangun adalah *game single player* yang berjenis *game* simulasi. Dalam permainan ini terdapat sebuah karakter (pengendara sepeda) sebagai pemain utama, dan terdapat beberapa karakter *Non Player Character* (NPC) musuh yang cerdas yang di mainkan oleh komputer. Objek penelitian dalam permainan ini adalah desain animasi pergerakan NPC musuh yang dikendalikan komputer.

3.1.2 *Storyline*

Game gowes ini merupakan sebuah *game* simulasi yang menggambarkan seseorang mengendarai sepeda mengelilingi lingkungan hutan. Dalam *game* ini pemain diharuskan berkeliling mengendarai sepeda untuk mencari *obstacle* yang mempunyai poin. Setiap *obstacle* memiliki poin 10 yang akan diakumulasikan di akhir permainan.

Pada sisi *game* yang akan di dibangun, dengan background lingkungan hutan atau pedesaan dengan jalan yang masih berupa tanah, dan juga dengan tambahan misi *game* untuk mengkoleksi *point*, dan misi tertentu, dengan model di buat mendekati dengan aslinya. Selain itu *game* ini akan dibangun dengan menambahkan NPC sebagai musuh yang

bertugas untuk mengganggu pemain dalam mencapai misi. NPC dalam *game* akan ditempakan di beberapa posisi yang ditentukan.

3.1.3 Deskripsi NPC Musuh

Dalam *game* ini NPC musuh yang dibuat berupa karakter robot yang membawa senjata. NPC musuh ini akan melakukan *patrol* pada saat *game* dimulai dan akan berubah megejar ketika didekati *player.* NPC musuh juga bisa menyerang apabila kondisi *player* sangat dekat dan tidak berhasil kabur. NPC musuh mempunyai 3 bentuk animasi yaitu pada saat patrol, menyerang dan mengejar.

1. Rancangan NPC musuh patrol

NPC musuh yang memiliki pergerakan patrol dimodelkan seperti orangyang sedang berjalan bolak-balik ke *point* yang sudah ditentukan. Model NPC musuh yang sedang patrol dapat dilihat pada gambar 3.1 berikut ini:

Gambar 3.1 Model NPC musuh sedang patrol

2. Rancangan NPC musuh mengejar

NPC musuh yang sedang mengejar *player* dimodelkan sebagai robot yang sedang berlari menuju *player*. NPC musuh berlari mendekati *player* tetapi tidak sampai menyentuhnya. Model NPC

musuh yang sedang mengejar *player* dapat dilihat pada gambar 3.2 berikut ini:

Gambar 3.2 Model NPC musuh mengejar *player*

3. Rancangan NPC musuh menyerang

NPC musuh yang sedang menyerang dimodelkan seperti robot yang berlari menuju *player*. Animasinya sama seperti NPC musuh pada saat mengejar, akan tetapi pada saat menyerang NPC musuh memiliki kecepatan yang lebih tinggi dan mendekati *player* sampai menyentuhnya. Model NPC musuh pada saat menyerang dapat dilihat pada gambar 3.3 berikut ini:

Gambar 3.3 Model NPC musuh menyerang *player*

3.2 Algortima *Game*

3.2.1 Skenario Perubahan Perilaku pada NPC Musuh

Untuk penelitian ini, dibuat sekenario untuk *game* yang dijadikan untuk simulasi atau uji coba. Karakter dibagi dalam dua bagian, yakni pemain (*player*) dan NPC musuh yang menjadi obyek penelitian ini. NPC musuh mempunyai perilaku menyerang sekaligus variabel yang mempengaruhi perubahan perilaku seperti pada tabel dibawah. Skenario ini tentang respon NPC musuh yang akan mengejar dan menyerangnya,

Tabel 3.1 Perubahan Perilaku NPC musuh

NPC	**Variabel *Input* Perilaku**	**Variabel *Output* Perilaku**
NPC musuh	Jarak terhadap *player*, kesehatan, kecepatan	Patrol, mengejar *player*, menyerang *player*

3.2.2 Rancangan FSM NPC Musuh

State utama yang tersusun dalam FSM dapat digambarkan sebagai berikut:

1. *Spawn / start*

   Merupakan *state* posisi awal NPC berada

2. *Walk / patrol state* berjalan / patroli

   NPC musuh bergerak menuju *state* yang sudah ditentukan sebelumnya

3. Menyerang

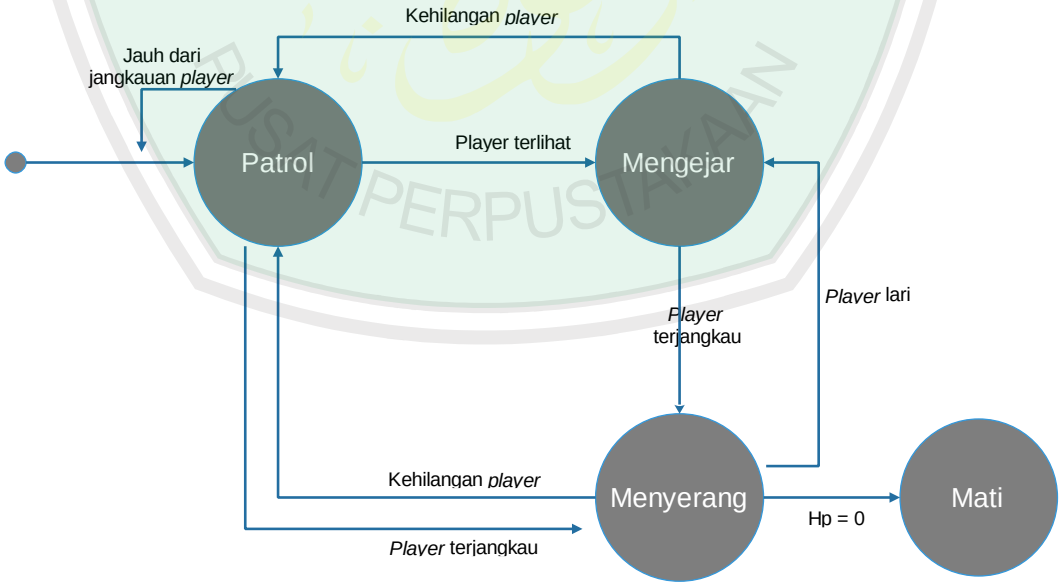
   State NPC musuh terlibat pertempuran dipicu jangkauan NPC musuh terhadapa *player*. Terdapat *state* menyerang dan mengejar dalam *state* ini.

4. Mati / *Game* Over

   Nilai kesehatan *player* = 0

Daftar *state transition* pada *FSM* dapat disusun dalam list sebagai berikut:

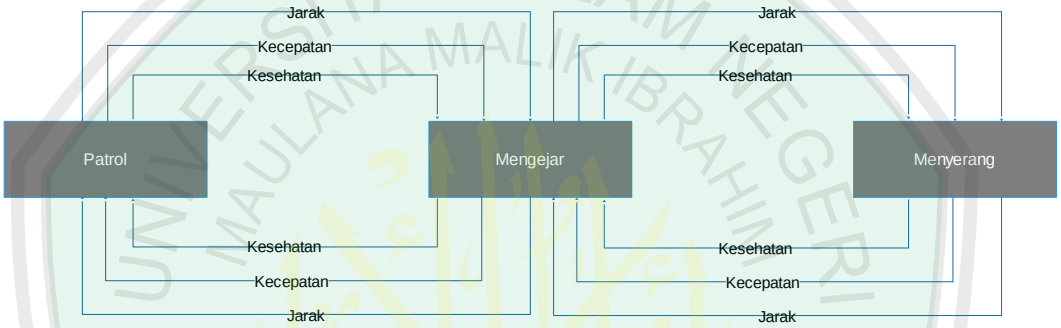
1. *Player* dalam jarak pandang
2. Kesehatan = 0
3. *Player* mati



Gambar 3.4 FSM NPC musuh

3.2.3 Rancangan FuSM NPC Musuh

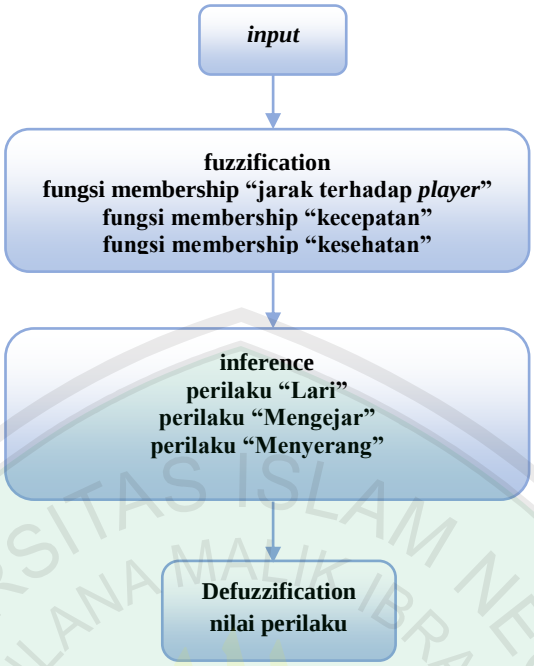
Dalam penelitian ini, logika *fuzzy* diaplikasikan pada perilaku NPC musuh pada saat strategi menyerang dilakukan. Perilaku ini muncul saat *state* menyerang aktif. Tahapan *fuzzy* untuk menentukan *output* sebagai keputusan perilaku NPC yaitu fuzzifikasi, implikasi dan komposisi aturan, serta defuzzifikasi atau penegasan. Rancang *state machine* untuk *state* menyerang dijabarkan dalam gambar 3.2.



Gambar 3.5 FuSM NPC musuh

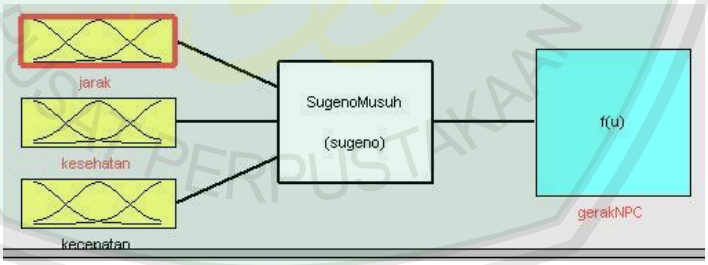
3.3 Desain *Fuzzy* NPC Musuh

Tiga variabel digunakan untuk merancang perilaku NPC musuh yaitu variabel “jarak terhadap *player*”, variabel “kecepatan” dan variabel “kesehatan”. Logika *fuzzy* untuk menghasilkan perilaku NPC musuh ditunjukkan dalam Gambar 3.6.



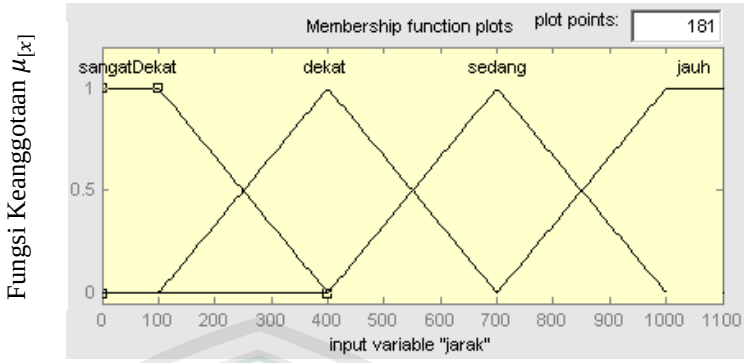
Gambar 3.6 Logika *fuzzy* untuk menghasilkan perilaku NPC musuh

Desain *fuzzy* untuk menghasilkan perilaku NPC musuh dapat dilihat Gambar 3.7 dibawah, atribut yang diberikan untuk NPC adalah jarak, kesehatan dan kecepatan terhadap *player*.



Gambar 3.7 Desain *Fuzzy* untuk menghasilkan perilaku NPC musuh

Untuk fungsi keanggotaan "jarak", variabel linguistik mempunyai nilai-nilai linguistik dalam interval numerik yang semantiknya didefinisi oleh fungsi keanggotaannya. Gambar 3.8 menunjukkan derajat keanggotaan (*membership degree*) untuk *input* jumlah anak panah yang mempunyai nilai dalam interval antara 0 sampai 1100.



Gambar 3.8 Derajat keanggotaan untuk *Input* jarak

Untuk fungsi keanggotaan "jarak terhadap *player*" mempunyai beberapa variabel linguistik serta notasinya dengan interval nilai beragam, dalam interval 0 sampai 1100 yang ditentukan seperti dalam Tabel 3.2

Tabel 3.2 Variabel *linguistik input* jarak terhadap *player*

***Input* Jarak**		
Variabel	**Notasi**	**Nilai**
SD	Sangat Dekat	0 – 400
D	Dekat	100 – 700
S	Sedang	400 – 1000
J	Jauh	700 - 1100

Perhitungan nilai *fuzzyfikasi* didapatkan dari beberapa fungsi, fungsi yang digunakan pada variabel poin ada 4 yaitu fungsi trapesium turun, segitiga, segitiga dan trapesium naik. Berikut perhitungan manual dari keempat fungsi tersebut:

Trapesium Turun : jSangat Dekat

$$\mu_{jsangatdekat[x]} = \begin{cases} 0; & x \geq 400 \\ 1; & x \leq 100 \\ \frac{400-x}{400-100}; & 100 < x \leq 400 \end{cases} \quad \ldots\ldots\ldots\ldots (1)$$

Segitiga : jDekat

$$\mu_{jdekat[x]} = \begin{cases} 0; \; x \leq 100 \; atau \; x \geq 700 \\ \frac{x-100}{400-100}; & 100 < x \leq 400 \\ \frac{700-x}{700-400}; & 400 \leq x < 700 \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots (2)$$

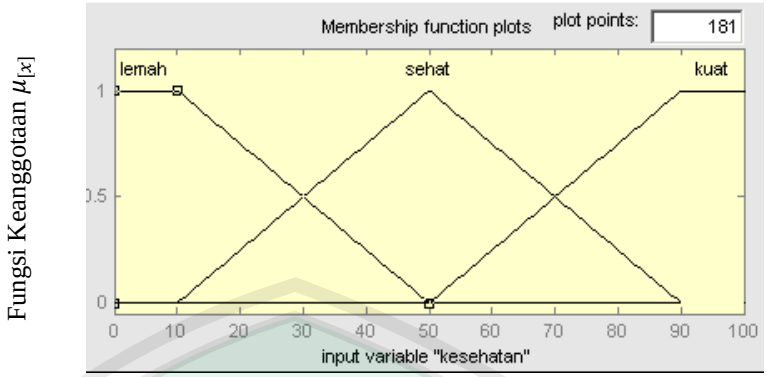
Segitiga : jSedang

$$\mu_{jsedang[x]} = \begin{cases} 0; x \; \leq 400 \; atau \; x \geq 1000 \\ \frac{x-400}{700-400}; \; 400 < x \leq 700 \\ \frac{1000-x}{1000-700}; 700 \leq x < 1000 \end{cases} \quad \ldots\ldots\ldots\ldots\ldots (3)$$

Trapesium Naik : jJauh

$$\mu_{jjauh[x]} = \begin{cases} 0; & x \leq 700 \\ \frac{x-700}{1000-700}; & 700 < x \leq 1000 \\ 1; & x \geq 1000 \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots (4)$$

Untuk fungsi keanggotaan "kesehatan", variabel linguistik mempunyai nilai-nilai linguistik dalam inrterval numerik yang semantiknya didefinisi oleh fungsi keanggotaannya. Gambar 3.9 menunjukkan derajat keanggotaan (*membership degree*) untuk *input* jumlah anak panah yang mempunyai nilai dalam interval antara 0 sampai 100.



Gambar 3.9 Derajat kenggotaan untuk *Input* Kesehatan terhadap *Player*

Untuk *input* ‘’kesehatan” mempunyai beberapa variabel linguistik serta notasinya dengan interval nilai beragam, dalam interval 0 sampai 100 seperti disusun dalam Tabel 3.3

Tabel 3.3 Variabel linguistik *input* kesehatan

***Input* Kesehatan**		
Variabel	**Notasi**	**Nilai**
L	Lemah	0 – 50
S	Sedang	10 – 90
K	Kuat	50 – 100

Perhitungan nilai *fuzzyfikasi* didapatkan dari beberapa fungsi, fungsi yang digunakan pada variabel poin ada 3 yaitu fungsi trapesium turun, segitiga dan trapesium naik. Berikut perhitungan manual dari ketiga fungsi tersebut:

Trapesium Turun : kLemah

$$\mu_{klemah[x]} = \begin{cases} 0; x \geq 50 \\ 1; x \leq 10 \\ \frac{50-x}{50-10}; 10 < x \leq 50 \end{cases} \quad \text{.....(5)}$$

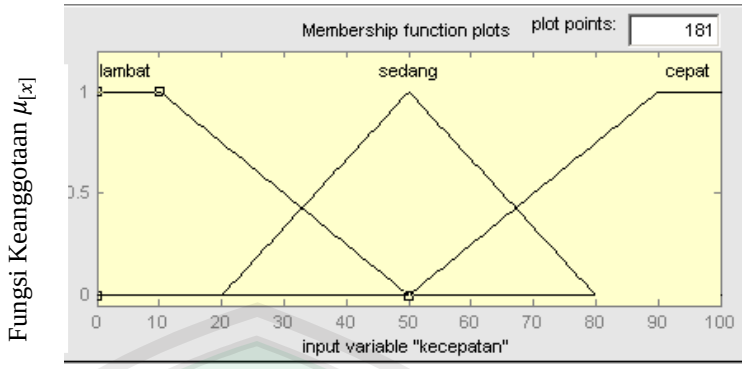
Segitiga : kSedang

$$\mu_{ksedang[x]} = \begin{cases} 0;\ x \leq 10\ atau\ x \geq 90 \\ \frac{x-10}{50-10}; 10 < x \leq 50 \\ \frac{90-x}{40}; 50 \leq x < 90 \end{cases} \quad \text{.....(6)}$$

Trapesium Naik : kKuat

$$\mu_{kkuat[x]} = \begin{cases} 0; x \leq 50 \\ \frac{x-50}{90-50}; 50 < x \leq 90 \\ 1; x \geq 90 \end{cases} \quad \text{.....(7)}$$

Sedangkan untuk fungsi keanggotaan "kecepatan", variabel linguistik mempunyai nilai-nilai linguistik dalam inrterval numerik yang semantiknya didefinisi oleh fungsi keanggotaannya. Gambar 3.10 menunjukkan derajat keanggotaan (*membership degree*) untuk *input* jumlah anak panah yang mempunyai nilai dalam interval antara 0 sampai 100.



Gambar 3.10 Derajat kenggotaan untuk *Input* Kecepatan terahadap *Player*

Untuk *input* ‘’kecepatan” mempunyai beberapa variabel linguistik serta notasinya dengan interval nilai beragam, yang ditentukan seperti disusun dalam Tabel 3.4

Tabel 3.4 Variabel linguistic *input* kecepatan

***Input* Kecepatan**		
Variabel	**Notasi**	**Nilai**
L	Lambat	0 – 50
S	Sedang	10 – 90
C	Cepat	50 – 100

Perhitungan nilai *fuzzyfikasi* didapatkan dari beberapa fungsi, fungsi yang digunakan pada variabel poin ada 3 yaitu fungsi trapesium turun, segitiga dan trapesium naik. Berikut perhitungan manual dari ketiga fungsi tersebut:

Trapesium Turun : kLambat

$$\mu_{klambat[x]} = \begin{cases} 0; x \geq 50 \\ 1; x \leq 10 \\ \frac{50-x}{50-10}; 10 < x \leq 50 \end{cases} \text{.....(8)}$$

Segitiga : kSedang

$$\mu_{ksedang[x]} = \begin{cases} 0;\ x \leq 10\ atau\ x \geq 90 \\ \frac{x-10}{50-10}; 10 < x \leq 50 \\ \frac{90-x}{90-50}; 50 \leq x < 90 \end{cases} \text{.....(9)}$$

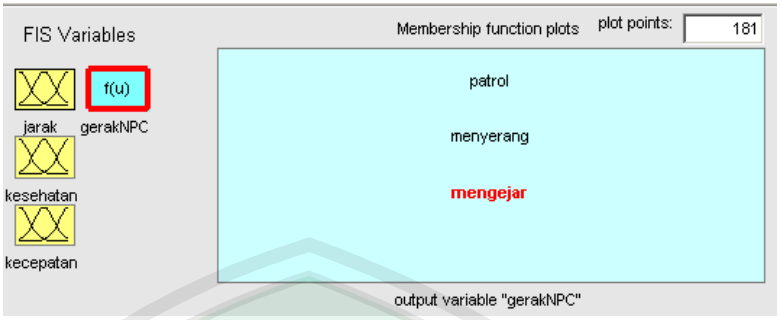
Trapesium Naik : kCepat

$$\mu_{kcepat[x]} = \begin{cases} 0; x \leq 50 \\ \frac{x-50}{90-50}; 50 < x \leq 90 \\ 1; x \geq 90 \end{cases} \text{.....(10)}$$

Selanjutnya Gambar 3.8 menunjukkan keanggotaan untuk *output* pada NPC musuh. Untuk *output* perilaku NPC musuh dijelaskan melalui Tabel 3.5, nilai linguistiknya dibagi menjadi 3, dengan notasi Patrol (P), Mengejar (MJ) dan Menyerang (MY).

Tabel 3.5 Variabel linguistik *output* NPC musuh

***Output* NPC musuh**	
Variabel	**Notasi**
P	Patrol
MJ	Mengejar
MY	Menyerang



Gambar 3.11 Keanggotaan *output* perilaku NPC musuh

Selanjutnya aturan *fuzzy (fuzzy rule)* yang disusun untuk menghasilkan perilaku NPC musuh dijelaskan pada Tabel 3.6. *Fuzzy rule* yang tersusun dalam matriks tabel tersebut merupakan formulasi dari 3 *input* yaitu jarak terhadap *player*, kesehatan dan kecepatan. Tabel 3.6 Aturan *fuzzy* untuk menghasilkan perilaku NPC musuh.

Tabel 3.6 Aturan *Fuzzy* perilaku NPC musuh

Input			***Output***
Jarak	**Kesehatan**	**Kecepatan**	**Gerak**
Sangat dekat	Lemah	Lambat	Menyerang
Sangat dekat	Lemah	Sedang	Menyerang
Sangat dekat	Lemah	Cepat	Mengejar
Sangat dekat	Sehat	Lambat	Menyerang
Sangat dekat	Sehat	Sedang	Menyerang
Sangat dekat	Sehat	Cepat	Mengejar
Sangat dekat	Kuat	Lambat	Menyerang

Sangat dekat	Kuat	Sedang	Menyerang
Sangat dekat	Kuat	Cepat	Mengejar
Dekat	Lemah	Lambat	Menyerang
Dekat	Lemah	Sedang	Mengejar
Dekat	Lemah	Cepat	Mengejar
Dekat	Sehat	Lambat	Menyerang
Dekat	Sehat	Sedang	Mengejar
Dekat	Sehat	Cepat	Mengejar
Dekat	Kuat	Lambat	Menyerang
Dekat	Kuat	Sedang	Mengejar
Dekat	Kuat	Cepat	Mengejar
Sedang	Lemah	Lambat	Mengejar
Sedang	Lemah	Sedang	Mengejar
Sedang	Lemah	Cepat	Patrol
Sedang	Sehat	Lambat	Mengejar
Sedang	Sehat	Sedang	Mengejar
Sedang	Sehat	Cepat	Patrol
Sedang	Kuat	Lambat	Mengejar
Sedang	Kuat	Sedang	Mengejar
Sedang	Kuat	Cepat	Patrol
Jauh	Lemah	Lambat	Patrol

Jauh	Lemah	Sedang	Patrol
Jauh	Lemah	Cepat	Patrol
Jauh	Sehat	Lambat	Patrol
Jauh	Sehat	Sedang	Patrol
Jauh	Sehat	Cepat	Patrol
Jauh	Kuat	Lambat	Patrol
Jauh	Kuat	Sedang	Patrol
Jauh	Kuat	Cepat	Patrol

Dari tabel *fuzzy rule* pada tabel 3.6, dapat diperoleh *fuzzy rule* ***IF/THEN*** yang menjelaskan hubungan antara *input* dan *output* variabel linguistik, yang tersusun sebagai berikut:

1) *If* (jarak *is* sangatDekat) *and* (kesehatan *is* lemah) *and* (kecepatan *is* lambat) *then* (gerakNPC *is* menyerang)
2) *If* (jarak *is* sangatDekat) *and* (kesehatan *is* lemah) *and* (kecepatan *is* sedang) *then* (gerakNPC *is* menyerang)
3) *If* (jarak *is* sangatDekat) *and* (kesehatan *is* lemah) *and* (kecepatan *is* cepat) *then* (gerakNPC *is* mengejar)
4) *If* (jarak *is* sangatDekat) *and* (kesehatan *is* sehat) *and* (kecepatan *is* lambat) *then* (gerakNPC *is* menyerang)
5) *If* (jarak *is* sangatDekat) *and* (kesehatan *is* sehat) *and* (kecepatan *is* sedang) *then* (gerakNPC *is* menyerang)
6) *If* (jarak *is* sangatDekat) *and* (kesehatan *is* sehat) *and* (kecepatan *is* cepat) *then* (gerakNPC *is* mengejar)
7) *If* (jarak *is* sangatDekat) *and* (kesehatan *is* kuat) *and* (kecepatan *is* lambat) *then* (gerakNPC *is* menyerang)

8) *If* (jarak *is* sangatDekat) *and* (kesehatan *is* kuat) *and* (kecepatan *is* sedang) *then* (gerakNPC *is* menyerang)
9) *If* (jarak *is* sangatDekat) *and* (kesehatan *is* kuat) *and* (kecepatan *is* cepat) *then* (gerakNPC *is* mengejar)
10) *If* (jarak *is* dekat) *and* (kesehatan *is* lemah) *and* (kecepatan *is* lambat) *then* (gerakNPC *is* menyerang)
11) *If* (jarak *is* dekat) *and* (kesehatan *is* lemah) *and* (kecepatan *is* sedang) *then* (gerakNPC *is* mengejar)
12) *If* (jarak *is* dekat) *and* (kesehatan *is* lemah) *and* (kecepatan *is* cepat) *then* (gerakNPC *is* mengejar)
13) *If* (jarak *is* dekat) *and* (kesehatan *is* sehat) *and* (kecepatan *is* lambat) *then* (gerakNPC *is* menyerang)
14) *If* (jarak *is* dekat) *and* (kesehatan *is* sehat) *and* (kecepatan *is* sedang) *then* (gerakNPC *is* mengejar)
15) *If* (jarak *is* dekat) *and* (kesehatan *is* sehat) *and* (kecepatan *is* cepat) *then* (gerakNPC *is* mengejar)
16) *If* (jarak *is* dekat) *and* (kesehatan *is* kuat) *and* (kecepatan *is* lambat) *then* (gerakNPC *is* menyerang)
17) *If* (jarak *is* dekat) *and* (kesehatan *is* kuat) *and* (kecepatan *is* sedang) *then* (gerakNPC *is* mengejar)
18) *If* (jarak *is* dekat) *and* (kesehatan *is* kuat) *and* (kecepatan *is* cepat) *then* (gerakNPC *is* mengejar)
19) *If* (jarak *is* sedang) *and* (kesehatan *is* lemah) *and* (kecepatan *is* lambat) *then* (gerakNPC *is* mengejar)
20) *If* (jarak *is* sedang) *and* (kesehatan *is* lemah) *and* (kecepatan *is* sedang) *then* (gerakNPC *is* mengejar)
21) *If* (jarak *is* sedang) *and* (kesehatan *is* lemah) *and* (kecepatan *is* cepat) *then* (gerakNPC *is* patrol)
22) *If* (jarak *is* sedang) *and* (kesehatan *is* sehat) *and* (kecepatan *is* lambat) *then* (gerakNPC *is* mengejar)
23) *If* (jarak *is* sedang) *and* (kesehatan *is* sehat) *and* (kecepatan *is* sedang) *then* (gerakNPC *is* mengejar)

24) *If* (jarak *is* sedang) *and* (kesehatan *is* sehat) *and* (kecepatan *is* cepat) *then* (gerakNPC *is* patrol)
25) *If* (jarak *is* sedang) *and* (kesehatan *is* kuat) *and* (kecepatan *is* lambat) *then* (gerakNPC *is* mengejar)
26) *If* (jarak *is* sedang) *and* (kesehatan *is* kuat) *and* (kecepatan *is* sedang) *then* (gerakNPC *is* mengejar)
27) *If* (jarak *is* sedang) *and* (kesehatan *is* kuat) *and* (kecepatan *is* cepat) *then* (gerakNPC *is* patrol)
28) *If* (jarak *is* jauh) *and* (kesehatan *is* lemah) *and* (kecepatan *is* lambat) *then* (gerakNPC *is* patrol)
29) *If* (jarak *is* jauh) *and* (kesehatan *is* lemah) *and* (kecepatan *is* sedang) *then* (gerakNPC *is* patrol)
30) *If* (jarak *is* jauh) *and* (kesehatan *is* lemah) *and* (kecepatan *is* cepat) *then* (gerakNPC *is* patrol)
31) *If* (jarak *is* jauh) *and* (kesehatan *is* sehat) *and* (kecepatan *is* lambat) *then* (gerakNPC *is* patrol)
32) *If* (jarak *is* jauh) *and* (kesehatan *is* sehat) *and* (kecepatan *is* sedang) *then* (gerakNPC *is* patrol)
33) *If* (jarak *is* jauh) *and* (kesehatan *is* sehat) *and* (kecepatan *is* cepat) *then* (gerakNPC *is* patrol)
34) *If* (jarak *is* jauh) *and* (kesehatan *is* kuat) *and* (kecepatan *is* lambat) *then* (gerakNPC *is* patrol)
35) *If* (jarak *is* jauh) *and* (kesehatan *is* kuat) *and* (kecepatan *is* sedang) *then* (gerakNPC *is* patrol)
36) *If* (jarak *is* jauh) *and* (kesehatan *is* kuat) *and* (kecepatan *is* cepat) *then* (gerakNPC *is* patrol)

Sehingga dalam menentukan *output* atau keputusan dapat dilakukan dengan menggunakan rumus sebagai berikut:

1. *Fuzzyfikasi*

   Menghitung nilai $\mu_{[x]}$ sebagai nilai derajat keanggotaan *fuzzy* untuk setiap variabel *input*, menggunakan rumus fungsi keanggotaan 1 – 10.

2. *Implikasi*

$$\alpha_n = \mu_{predikatRn} = \min(\mu_{[x]}; \mu_{[y]}) \ldots\ldots\ldots\ldots\ldots\ldots (11)$$

   $(\mu_{[x]}; \mu_{[y]})$ = variabel *input* yang saling berhubungan, lihat tabel 3.6

3. *Defuzzyfikasi*

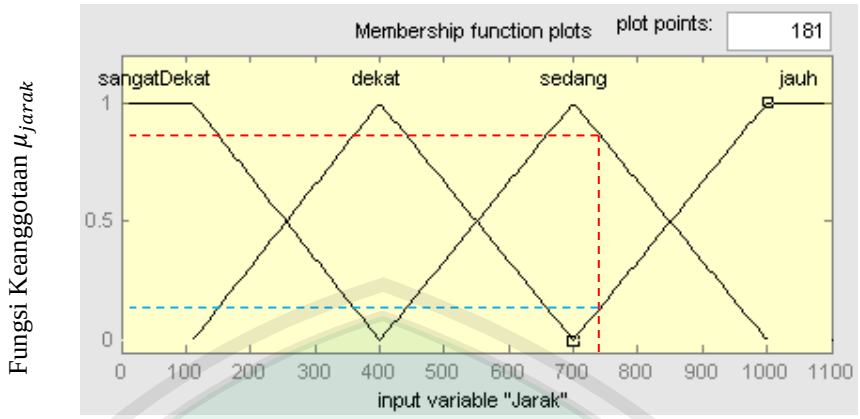
$$k = \max(\alpha_1, \alpha_{2,\ldots\ldots\ldots\ldots}, \alpha_n) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (12)$$

Berikut ini adalah studi kasus FuSM perilaku NPC musuh : Contoh kasus, misal pemain berada pada jarak 750 dari musuh dengan kecepatan 95 dan mempunyai kekuatan 80, maka untuk mengetahui perilaku pergerakan musuh sebagai berikut :

Diketahui:

- Jarak terhadap musuh = 750
- Kesehatan = 80
- Kecepatan = 95

1) Fuzzifikasi



Gambar 3.12 Fungsi Keanggotaan Nilai Jarak 750

Dengan fungsi keanggotaan sebagai berikut:

Trapesium Turun : jSangat Dekat

$$\mu_{jsangatdekat[x]} = \begin{cases} 0; & x \geq 400 \\ 1; & x \leq 100 \\ \dfrac{400 - x}{400 - 100}; & 100 < x \leq 400 \end{cases}$$

Segitiga : jDekat

$$\mu_{jdekat[x]} = \begin{cases} 0;\ x \leq 100\ atau\ x \geq 700 \\ \dfrac{x - 100}{400 - 100}; & 100 < x \leq 400 \\ \dfrac{700 - x}{700 - 400}; & 400 \leq x < 700 \end{cases}$$

Segitiga : jSedang

$$\mu_{jsedang[x]} = \begin{cases} 0; x\ \leq 400\ atau\ x \geq 1000 \\ \dfrac{x - 400}{700 - 400}; & 400 < x \leq 700 \\ \dfrac{1000 - x}{1000 - 700}; & 700 \leq x < 1000 \end{cases}$$

Trapesium Naik : jJauh

$$\mu_{jjauh[x]} = \begin{cases} 0; & x \leq 700 \\ \dfrac{x - 700}{1000 - 700}; & 700 < x \leq 1000 \\ 1; & x \geq 1000 \end{cases}$$

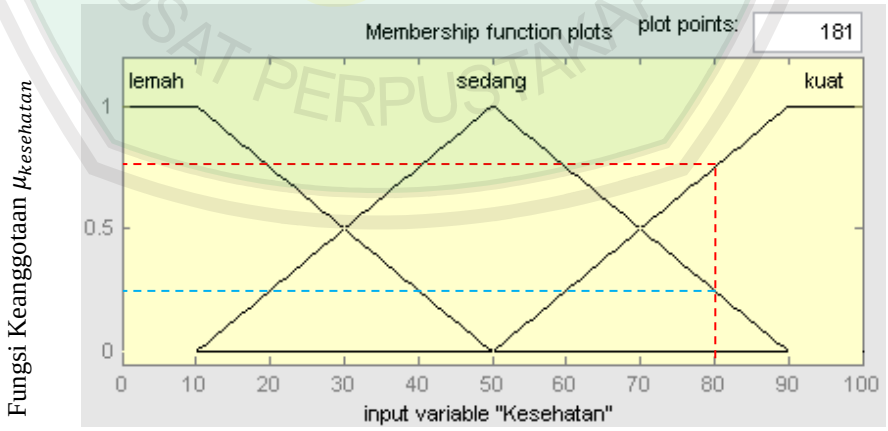
Nilai jarak 750 termasuk kedalam himpunan *fuzzy* sedang dan jauh dengan tingkat keanggotaan sesuai fungsi sebagai berikut:

$\mu_{jsangatdekat}$ (750) = 0; $Jarak \geq 400$

μ_{jdekat} (750) = 0; $jarak \geq 700$

$\mu_{jsedang}$ (750) = 0; $jarak \geq 1000$

μ_{jjauh} (750) = $\frac{750-700}{1000-700} = 0.17$; $700 \leq jarak \leq 1000$



Gambar 3.13 Fungsi Keanggotaan Nilai Kesehatan 80

Dengan fungsi keanggotaan sebagai berikut:

Trapesium Turun : kLemah

$$\mu_{klemah[x]} = \begin{cases} 0; x \geq 50 \\ 1; x \leq 10 \\ \dfrac{50 - x}{50 - 10}; 10 < x \leq 50 \end{cases}$$

Segitiga : kSedang

$$\mu_{ksedang[x]} = \begin{cases} 0;\ x \leq 10\ atau\ x \geq 90 \\ \dfrac{x - 10}{50 - 10}; 10 < x \leq 50 \\ \dfrac{90 - x}{40}; 50 \leq x < 90 \end{cases}$$

Trapesium Naik : kKuat

$$\mu_{kkuat[x]} = \begin{cases} 0; x \leq 50 \\ \dfrac{x - 50}{90 - 50}; 50 < x \leq 90 \\ 1; x \geq 90 \end{cases}$$

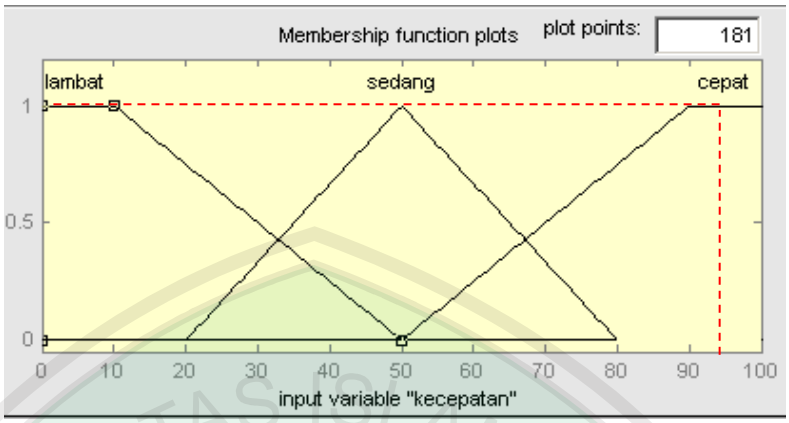
Nilai kesehatan 80 termasuk kedalam himpunan *fuzzy* sedang dan kuat dengan tingkat keanggotaan sesuai fungsi sebagai berikut:

μ_{klemah} (80) = 0; $kesehatan \geq 50$

$\mu_{ksedang}$ (80) = $\frac{90-80}{40} = 0.25; 50 \leq kesehatan \leq 90$

μ_{kkuat} (80) = $\frac{80-50}{90-50} = 0.75\ ; 50 \leq kesehatan \leq 90$

Fungsi Keanggotaan $\mu_{kecepatan}$



Dengan fungsi keanggotaan kecepatan sebagai berikut:

Trapesium Turun : kLambat

$$\mu_{klambat[x]} = \begin{cases} 0; x \geq 50 \\ 1; x \leq 10 \\ \dfrac{50 - x}{50 - 10}; 10 < x \leq 50 \end{cases}$$

Segitiga : kSedang

$$\mu_{ksedang[x]} = \begin{cases} 0;\ x \leq 10\ atau\ x \geq 90 \\ \dfrac{x - 10}{50 - 10}; 10 < x \leq 50 \\ \dfrac{90 - x}{90 - 50}; 50 \leq x < 90 \end{cases}$$

Trapesium Naik : kCepat

$$\mu_{kcepat[x]} = \begin{cases} 0; x \leq 50 \\ \dfrac{x - 50}{90 - 50}; 50 < x \leq 90 \\ 1; x \geq 90 \end{cases}$$

Nilai kecepatan 95 termasuk kedalam himpunan *fuzzy* sedang dan kuat dengan tingkat keanggotaan sesuai fungsi sebagai berikut:

$\mu_{klambat}$ (95) = 0; $kecepatan \geq 50$

$\mu_{ksedang}$ (95) = 0; $kecepatan \geq 90$

μ_{kcepat} (95) = 1; $kecepatan \geq 90$

2) Implikasi

Dari perhitungan pada fuzzifikasi di atas diperoleh:

$\mu_{jsangatdekat}$ (750) = 0, μ_{jdekat} (750) = 0, $\mu_{jsedang}$ (750) = 0 dan μ_{jjauh} (750) = 0,17 untuk fungsi jarak,

μ_{klemah} (80) = 0, $\mu_{ksedang}$ (80) = 0,25 dan μ_{kkuat} (80) = 0,75 untuk fungsi kesehatan, dan

$\mu_{klambat}$ (95) = 0, $\mu_{ksedang}$ (95) = 0 dan μ_{kcepat} (95) = 1 untuk fungsi kecepatan.

Berdasarkan aturan – aturan yang sesuai dengan kondisi tersebut, dengan menggunakan rumus (8) maka diperoleh:

1. $\alpha_1 = \mu_{predikatR1} = \min(0;\ 0;\ 0) = 0$
2. $\alpha_2 = \mu_{predikatR2} = \min(0;\ 0;\ 0) = 0$
3. $\alpha_3 = \mu_{predikatR3} = \min(0;\ 0;\ 1) = 0$
4. $\alpha_4 = \mu_{predikatR4} = \min(0;\ 0{,}25;\ 0;) = 0$
5. $\alpha_5 = \mu_{predikatR5} = \min(0;\ 0{,}25;\ 0) = 0$
6. $\alpha_6 = \mu_{predikatR6} = \min(0;\ 0{,}25;\ 1) = 0$
7. $\alpha_7 = \mu_{predikatR7} = \min(0;\ 0{,}75;\ 0) = 0$
8. $\alpha_8 = \mu_{predikatR8} = \min(0;\ 0{,}75;\ 0) = 0$
9. $\alpha_9 = \mu_{predikatR9} = \min(0;\ 0{,}75;\ 1) = 0$
10. $\alpha_{10} = \mu_{predikatR10} = \min(0;\ 0;\ 0) = 0$
11. $\alpha_{11} = \mu_{predikatR11} = \min(0;\ 0;\ 0) = 0$
12. $\alpha_{12} = \mu_{predikatR12} = \min(0;\ 0;\ 1) = 0$
13. $\alpha_{13} = \mu_{predikatR13} = \min(0;\ 0{,}25;\ 0;) = 0$
14. $\alpha_{14} = \mu_{predikatR14} = \min(0;\ 0{,}25;\ 0) = 0$
15. $\alpha_{15} = \mu_{predikatR15} = \min(0;\ 0{,}25;\ 1) = 0$
16. $\alpha_{16} = \mu_{predikatR16} = \min(0;\ 0{,}75;\ 0) = 0$
17. $\alpha_{17} = \mu_{predikatR17} = \min(0;\ 0{,}75;\ 0) = 0$
18. $\alpha_{18} = \mu_{predikatR18} = \min(0;\ 0{,}75;\ 1) = 0$
19. $\alpha_{19} = \mu_{predikatR19} = \min(0;\ 0;\ 0) = 0$
20. $\alpha_{20} = \mu_{predikatR20} = \min(0;\ 0;\ 0) = 0$
21. $\alpha_{21} = \mu_{predikatR21} = \min(0;\ 0;\ 1) = 0$

22. $\alpha_{22} = \mu_{predikatR22} = \min(0;\ 0{,}25;\ 0;) = 0$

23. $\alpha_{23} = \mu_{predikatR23} = \min(0;\ 0{,}25;\ 0) = 0$

24. $\alpha_{24} = \mu_{predikatR24} = \min(0;\ 0{,}25;\ 1) = 0$

25. $\alpha_{25} = \mu_{predikatR25} = \min(0;\ 0{,}75;\ 0) = 0$

26. $\alpha_{26} = \mu_{predikatR26} = \min(0;\ 0{,}75;\ 0) = 0$

27. $\alpha_{27} = \mu_{predikatR27} = \min(0;\ 0{,}75;\ 1) = 0$

28. $\alpha_{28} = \mu_{predikatR28} = \min(0{,}17;\ 0;\ 0) = 0$

29. $\alpha_{29} = \mu_{predikatR29} = \min(0{,}17;\ 0;\ 0) = 0$

30. $\alpha_{30} = \mu_{predikatR30} = \min(0{,}17;\ 0;\ 1) = 0$

31. $\alpha_{31} = \mu_{predikatR31} = \min(0{,}17;\ 0{,}25;\ 0;) = 0$

32. $\alpha_{32} = \mu_{predikatR32} = \min(0{,}17;\ 0{,}25;\ 0) = 0$

33. $\alpha_{33} = \mu_{predikatR33} = \min(0{,}17;\ 0{,}25;\ 1) = 0{,}17$

34. $\alpha_{34} = \mu_{predikatR34} = \min(0{,}17;\ 0{,}75;\ 0) = 0$

35. $\alpha_{35} = \mu_{predikatR35} = \min(0{,}17;\ 0{,}75;\ 0) = 0$

36. $\alpha_{36} = \mu_{predikatR36} = \min(0{,}17;\ 0{,}75;\ 1) = 0{,}17$

Tabel 3.7 Rule *evaluation fuzzy*

		Kesehatan		
	Variabel	Lemah = 0	Sehat = 0.25	Kuat = 0.75
Jarak	SD = 0	MY = 0	MY = 0	MY = 0
	D = 0	MY = 0	MY = 0	MY = 0
	S = 0	MJ = 0	MJ = 0	MJ = 0
	J = 0.17	P = 0	P = 0.17	P = 0.17

		Kecepatan		
		Lambat = 0	Sedang = 0	Cepat = 1
Jarak kesehatan	MY = 0	MY = 0	MY = 0	MJ = 0
	MY = 0	MY = 0	MJ = 0	MJ = 0
	MJ = 0	MJ = 0	MJ = 0	P = 0
	P = 0	P = 0	P = 0	P = 0

		Kecepatan		
		Lambat = 0	Sedang = 0	Cepat = 1
Jarak kesehatan	MY = 0	MY = 0	MY = 0	MJ = 0
	MY = 0	MY = 0	MJ = 0	MJ = 0
	MJ = 0	MJ = 0	MJ = 0	P = 0
	P = 0.17	P = 0	P = 0	P = 0.17

		Kecepatan		
		Lambat = 0	Sedang = 0	Cepat = 1
Jarak kesehatan	MY = 0	MY = 0	MY = 0	MJ = 0
	MY = 0	MY = 0	MJ = 0	MJ = 0
	MJ = 0	MJ = 0	MJ = 0	P = 0
	P = 0.17	P = 0	P = 0	P = 0.17

Diketahui ada 2 *rule* yang akan aktif, dengan proses defuzzifikasi maka akan didapatkan 1 *output* sebagai keputusan respon perilaku NPC musuh.

3) Defuzzifikasi

Dengan menggunakan rumus (9) diperoleh nilai *output*:

$$k = \max(\alpha_{1,}\alpha_{2,\dots\dots\dots\dots,}\alpha_n)$$

$$k = \max \begin{pmatrix} 0;\ 0;\ 0;\ 0;\ 0;\ 0;\ 0;\ 0;\ 0;\ 0;\ 0;\ 0;\ 0;\ 0;\ 0;\ 0;\ 0;\ 0;\ 0;\ 0;\ 0;\ 0;\ 0;\ 0; \\ 0;\ 0;\ 0;\ 0;\ 0;\ 0;\ 0;\ 0;\ 0{,}17;\ 0;\ 0;\ 0{,}17 \end{pmatrix}$$

$$k = 0{,}17$$

Diketahui index tertinggi adalah 0.17 dengan *output* perilaku P atau Patrol. Jadi, Patrol adalah respon perilaku NPC musuh pada saat jarak 750, kekuatan 80 dan kecepatan 95.

BAB IV

HASIL DAN PEMBAHASAN

4.1 *Game* Perspektif Islam

Dalam Islam *game* sepeda di identikkan dengan olahraga yang menyehatkan dan menyenangkan, dan salah satu olah raga yang di sukai oleh nabi adalah mengendarai (kuda). Hal ini terdapat pada firman Allah pada QS.An-Nahl ayat 8:

وَٱلۡخَيۡلَ وَٱلۡبِغَالَ وَٱلۡحَمِيرَ لِتَرۡكَبُوهَا وَزِينَةٗۚ وَيَخۡلُقُ مَا لَا تَعۡلَمُونَ ٨

"Dan (dia telah menciptakan) kuda, bagal (yaitu peranakan kuda dengan keledai) dan keledai, agar kamu menunggganginya dan (menjadikannya) perhiasan. Dan Allah menciptakan apa yang kamu tidak mengetahuinya." (QS. An-Nahl: 8)

- Tafsir QS. An-Nahl ayat 8:

وَالْخَيْلَ وَالْبِغَالَ وَالْحَمِيرَ لِتَرْكَبُوهَا وَزِينَةً siapa yang memandang kuda-kuda yang tangguh dan kuat, atau binatang lain, maka hatinya akan berdecak kagum karena keindahannya.

Dan bukan hanya itu sebagai alat transportasi dan hiasan, tetapi Dia yakni Allah SWT secara terus menerus وَيَخْلُقُ aneka ciptaan, baik alat

transportasi maupun perhiasan مَا لَا تَعْلَمُونَ sekarang tetapi kelak akan kamu ketahui dan gunakan jika kamu mau berpikir dan mengarahkan segala potensi yang ada, dan Allah menciptakan apa yang kamu tidak akan mengetahuinya sama sekali hingga ciptaan itu kamu lihat dan ketahui (Al-Misbah).

- Tafsir Al-Maraghi

وَالْخَيْلَ وَالْبِغَالَ وَالْحَمِيرَ لِتَرْكَبُوهَا وَزِينَةً

Dia juga menciptakan bagi kalian kuda, baghal, dan keledai untuk kalian tunggangi, serta menjadikannya sebagai perhiasan bagi kalian, disamping manfaat-manfaat lain bagi kalian yang terdapat didalamnya.

وَيَخْلُقُ مَا لَا تَعْلَمُونَ

Dan dia menciptakan apa yang tidak kalian ketahui selain binatang ternak ini, seperti apa yang dicapai oleh ilmu dan diproduksi oleh akal, berupa kereta darat dan laut, pesawat terbang yang mengangkut barang-barang dan kalian kendarai dari satu negeri ke negeri lain, dan dari belahan bumi ke belahan bumi lain, balon-balon udara yang berjalan di angkasa, kapal-kapal selam yang berjalan dibawah air, dan hal-hal lain yang menakjubkan kalian serta menggantikan kedudukan kuda, baghal dan keledai sebagai pengangkut dan perhiasan (Al-Maraghi: 1993).

Dan hadits yang di riwayatkan oleh ibnu majah (Al-Qazwiniy) :

عَنْ عُقْبَةَ بْنِ عَامِرٍ الْجُهَنِيِّ قَالَ قَالَ رَسُولُ اللَّهِ صَلَّى اللَّهُ عَلَيْهِ وَسَلَّمَ ...ارْمُوا وَارْكَبُوا وَأَنْ تَرْمُوا أَحَبُّ إِلَيَّ مِنْ أَنْ تَرْكَبُوا وَإِنَّ كُلَّ شَيْءٍ يَلْهُو بِهِ الرَّجُلُ إِلَّا بَاطِلٌ رَمْيَةَ الرَّجُلِ بِقَوْسِهِ وَتَأْدِيبَهُ وَمُلَاعَفَرَسَهُ عَبَتَهُ امْرَأَتَهُ . ...رواه ابن ماجه

Artinya: *"....Memanahlah dan kenderailah olehmu (kuda). Namun, memanah lebih saya sukai daripada berkuda. Sesungguhnya setiap hal yang menjadi permainan seseorang adalah batil kecuali yang memanah dengan busurnya, mendidik/ melatih kudanya dan bersenang-senang dengan istrinya..."*. (HR. Ibn Majah)

Semua ini sebagai dorongan Nabi terhadap masalah pacuan kuda. Sebab berpacu kuda sebagaimana dikatakan diatas, adalah permainan, olaharaga juga suatu latihan (Qardhawi: 1993).

Dari kutipan alqur'an dan hadits di atas, berkendara adalah salah olah raga yang di sukai oleh nabi, sama halnya dengan berkendara dengan sepeda. Pada *game* simulasi ini untuk memainkannya seakan-akan dengan menggunakan sepeda, dengan penambahan NPC musuh didalam permainannya agar terlihat lebih menantang dan menarik untuk dimainkan.

4.2 Implementasi Antarmuka

Pada sub bab implementasi antarmuka ini akan dijelaskan komponen-komponen yang ada pada *Game* Sepeda.

4.2.1 *Main Menu*

Adalah menu utama sebelum melakukan permainan, terdapat beberapa pilihan diantaranya sebagai berikut:

Gambar 4.1. Menu Utama

1. *Play*

   Tombol untuk memulai *game*.
2. *Option*

   Menu untuk beberapa pilihan yang telah disediakan, diantaranya:
   a. *Control*

      Tombol ini Berisi halaman Kontrol pada *game* yang mana *player* dapat mengetahui control yang ada pada *game*.
   b. *Credit*

      Tombol ini Berisi tentang hal hal seputar pembuat *game* ini.
   c. *Back*

      Tombol ini akan membawa Kembali ke menu utama.
3. *Exit*

   Tombol untuk keluar dari *game*

4.2.2 *Game*

Pada tampilan *game* ini menampilkan beberapa item diantaranya:

Gambar 4.2. Tampilan *Game* Sepeda

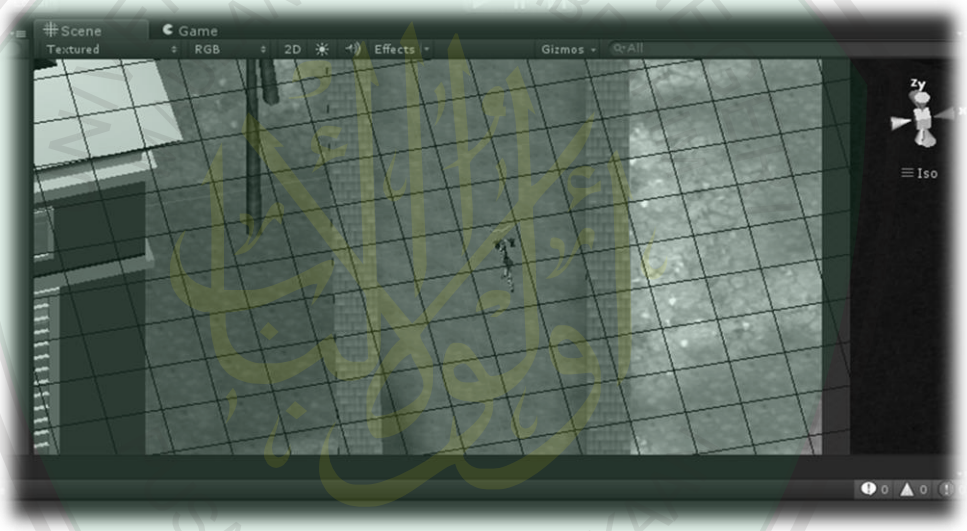
1. *Score* : Menampilkan *score* yang di dapat
2. *Timer* : Menampilkan waktu yang tersisa menjalankan suatu misi
3. Health : Menunjukkan kesehatan *player*
4. *Point pick*up : Berupa kubus yang apabila di koleksi / berbenturan pada sepeda , akan menambah *score*
5. *Minimaps* : Berisi map kecil, yang menampilkan jalan rute jalan.

4.3 Implementasi Sistem *Game*

Sistem *game* yang akan di implementasikan pada *game* sepeda ini menggunakan bahasa pemrograman C# dan javascript sebagai *core* dari *game*, dan menggunakan *editor monodevelop.* Sedangkan desain *visual* dan *interface* mengunakan *Game engine*.

4.3.1 Pengaturan *Grid*

Dalam suatu *game* simulasi dibutuhkan yang namanya Map, di dalam map kita dapat mengetahui koodrinat dari suatu objek,luas suatu map sangat berpengaruh terhadap kinerja algoritma *A**, jika semakin luas map maka semakin berat proses pencariannya, untuk menangani hal itu maka kita perlu membuat sebuah pengaturan yang di sebut *grid,* yaitu dengan mengkotak kotakkan suatu map yang bisa kita atur jumlah dan skalanya, sehingga dapat mempermudah jalanya proses algoritma *A**.



Gambar 4.3. *Grid* pada area *game* yang disesuaikan dengan skala

Pada gambar di atas (Gambar 4.3) terlihat grid pada waktu *debugging mode.* Besarnya skala pada *grid* akan mempengaruhi jumlah dari pada *matrix grid* yang ada pada *maps*, dan jumlah *grid* akan berpengaruh terhadap proses pencarian, karena semakin banyak grid akan semakun banyak proses pencarianya.

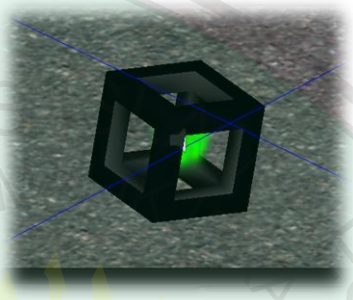
Berikut adalah potongan kode dari kelas *gridmanager.cs*

```
void OnDrawGizmos()
{
    //Draw Grid
    if (showGrid)
    {
         DebugDrawGrid(transform.position, numOfRows,
        numOfColumns, gridCellSize, Color.blue);
    }
    //Grid Start Position
    Gizmos.DrawSphere(transform.position, 0.5f);
    //Draw Obstacle obstruction
    if (showObstacleBlocks)
    {
  Vector3 cellSize = new Vector3(gridCellSize, 1.0f,
  gridCellSize);

     if (obstacleList != null && obstacleList.Length > 0)
        {
            foreach (GameObject data in obstacleList)
              {
              Gizmos.DrawCube(GetGridCellCenter(GetGridIndex(d
              ata.transform.position)), cellSize);
            }
        }
    }
}
```

Dari potongan *Source code* di atas kita bisa melihat *DebugDrawGrid* adalah suatu prosedur untuk menampilkan grid pada saat *debug*, dan juga void *OnDrawgizmos()* adalah *method* yang di sediakan *Game engine* untuk keperluan *debugging*, sehingga *grid* akan terlihat hanya pada saat kita pada debug mode, dan pada saat proses *running mode/building project* grid ini tidak tampak.

4.3.2 Pengaturan *Score*

Game pada umumnya terdapat *scoring*, begitu juga pada *game* ini, proses penambahan *score* ini dilakukan pada saat sepeda berbenturan dengan objek yang bernama *pickup*, berbentuk kubus (Gambar4.4).



Gambar 4.4. Pickup untuk point

Berikut adalah GUI dari *Score* yang ada pada *game* (Gambar 4.5)

Gambar 4.5. Gui *score*

Untuk implementasinya mengunakan beberapa kelas, diantaranya *pickupcontroller*.cs dan *cbpickup*.cs. Berikut adalah baris kodenya:

a) Kelas *Cbpickup.cs*

```
#pragma strict
function OnTriggerEnter (info : Collider){
     if (info.tag == "Sepeda"){
     update*Score*.current*score* += 10;
     //yield  WaitForSeconds(5);
     Destroy(*game*Object);
     }
}
```

Kelas di atas menangani proses *scoring* pada *game*, apabila objek yang menyentuh adalah sepeda maka poin bertambah.

b) Kelas *Pickupcontrolles.cs*

```
function Awake()
{
spawnPointList =
*game*Object.Find*Game*ObjectsWithTag("SpawnPoint");
     numberOfSpawnPoints = spawnPointList.length;

     if (numberOfPickups > numberOfSpawnPoints)
numberOfPickups = numberOfSpawnPoints;
     for (var i:int = 0;  i < numberOfSpawnPoints; i++)
     {
          spawnIndexAvailableList[i] = true;
     }
     for (var j:int = 0;  j < numberOfPickups; j++)
     {
          SpawnPickup();
     }
```

Kelas diatas menangani spawning dari kubus, sehingga kubus bisa muncul secara otomatis pada *game*.

4.3.3 Pengaturan *Timer*

Pada *game* ini suatu misi dibatasi oleh waktu, sehingga *player* bermain berdasarkan waktu yang ditentukan, waktu yang disediakan di hitung mundur dan ditampilkan pada layar seperti (gambar 4.6)

Gambar 4.6. *Timer*

Berikut adalah implementasi kedalam kode C dalam kelas *coundowntimer.cs:*

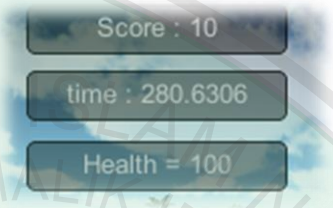
```
public float currentTime = 90;
public float offsety     = 40;
public float sizex       = 100;
public float sizey       = 40;
public string nextLevelToLoad   ;
// Use this for initialization

void FixedUpdate ()
      {
      if (currentTime <= 0) {

AutoFade.LoadLevel(nextLevelToLoad,1,1,Color.white);
            }
            currentTime -= Time.deltaTime;
      }
      void OnGUI(){
            GUI.Box(new Rect(Screen.width/2-
      sizex/2,offsety, sizex,sizey),"time : "
      +currentTime );
      }
}
```

Dari potongan kode di atas, sekilas kita mengetahui dapat mengatur watu yang ditentukan melalui variabel *currentTime.*

Apabila kita melebihi waktu yang ditentukan maka secara otomatis *game* akan permainan akan berakhir.

4.3.4 *Health*

Health adalah kesehatan yang diterapkan pada *player*. Berikut adalah tampilan *Health* pada layar *game*.



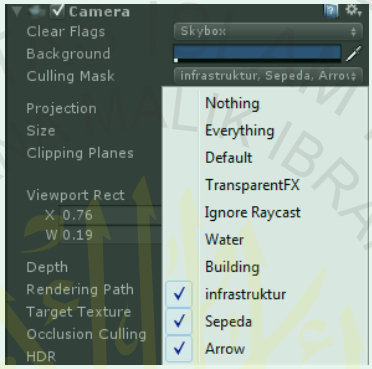
Gambar 4.7. Tampilan *health*

4.3.5 *Mini Map*

Mini map adalah Map kecil yang ditempatkan pada kanan atas layar yang berfunsi sebagai komponen pembatu navigasi, yang nantinya berisi garis garis *rute* yang akan dilalui *player*. *Mini map* ini akan mengikuti kemanapun arah pergerakan *player* dan juga memberikan arah setiap pergerakannya secara *realtim*e. Berikut adalah tampilan *mini map* pada layar *game* (Gambar 4.8)

Gambar 4.8. Tampilan mini map

Pembuatan Mini map menggunakan *Second Camera*, yang di tempatkan pada atas *player* sehingga dapat terlihat lingkungan sekitar *player*, untuk pengaturan objek apa saja yang hanya bisa di lihat oleh mini map adalah layer infrastruktur, sepeda, dan arrow yang dapat dilihat pada gambar berikut:



Gambar 4.9. Layer pada kamera

4.3.6 *Obstacle*

Obstacle adalah halangan atau rintangan, yang dalam *game* ini menjadi misi utama, yakni mencari halangan berupa kata berbahasa arab dengan mengurutkan angka arab mulai 1 – 7, *Obstacle* dibuat dengan warna mencolok dan berbeda dengan yang lainnya agar mudah dilihat ,seperti pada gambar di bawah ini.

Gambar 4.10. *Obstacle*

Gambar di atas adalah *obstacle* pertama yang beruliskan "*wahid*" yang berarti angka pertama(gambar di lingkari), *Player* akan di arahkan rute menuju *Obstacle* dari angka satu sampai tujuh ,dengan map kecil yang ada pada pojok kanan atas. *Obstacle* ini di susun secara berurutan, sehingga *player* tidak bisa mengambilnya secara acak, dan misi akan selesai jika *player* sudah mengumpulkan sebanyak 7 *Obstacle* dalam waktu yang sudah di tentukan.

4.4 Implementasi *Fuzzy State Machine* (FuSM) pada NPC Musuh

Proses Implementasi adalah proses pembangunan komponen-komponen pokok suatu sistem yang didasarkan pada desain dan rancangan yang telah dibuat sebelumnya. Implementasi perancangan *Artificial Intelligence* pada penelitian ini diterapkan pada pengaturan respon perilaku NPC dengan metode *Fuzzy State Machine* (FuSM).

Pada bagian ini membahas mengenai implementasi *Fuzzy State Machines* (FuSM) untuk pengaturan respon perilaku NPC musuh sehingga dihasilkan *output* perilaku yang lebih natural. FuSM bekerja pada saat pemain bertemu dengan NPC

musuh sehingga NPC musuh akan berperilaku natural seperti di dunia nyata yaitu menyerang. Begitu ketika pemain menjaih NPC musuh akan mengejar dan kembali patrol.

4.4.1. Perilaku Patrol NPC Musuh

Gambar 4.11. Perilaku Patrol NPC Musuh

Pada bagian ini membahas mengenai untuk pengaturan perilaku NPC musuh yaitu patrol. NPC musuh akan patrol ketika kondisi *player* masih berada pada nilai tertinggi. Pada saat *player* berada pada jarak di luar jangkauan NPC musuh dan masih memiliki kesehatan yang penuh. Berikut ini adalah *script* untuk perilaku patrol NPC musuh dalam kelas *Patrol.js*:

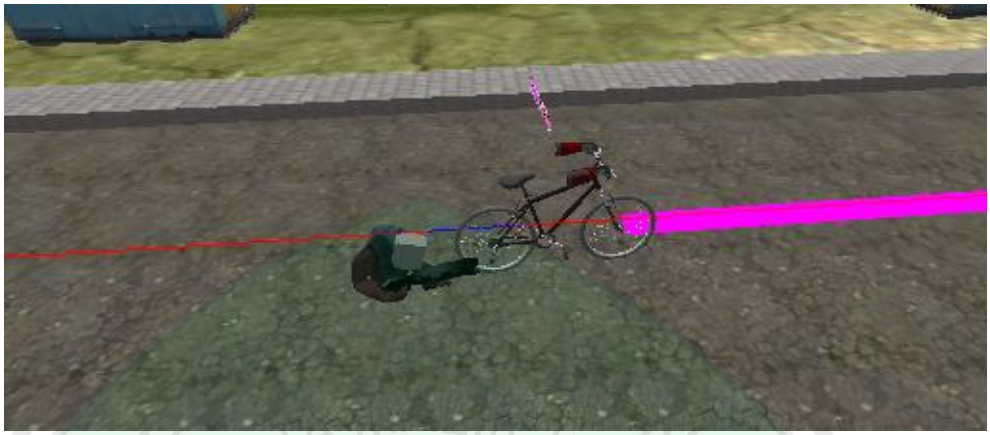
```
if(state == "patrol")

    {

        //if(event == "enter")
```

```
            if(prevState != state)
            {
graph.AStar(currentNode,goalLocation);
                currentWP = 0;
                this.animation.Play("run");
                this.animation["run"].wrapMode =
 WrapMode.Loop;
                event = "update";
                prevState = state;
            }
            else if (event == "update")
            {
            //if there is no path or we are at the
            //end of the path don't do anything
            if(graph.getPathLength() == 0 ||
            currentWP == graph.getPathLength())
            {
                state = "idle";
```

```
                event = "enter";

            return;

            }

//the node we are closest to at this moment

currentNode = graph.getPathPoint(currentWP);

            //if we are close enough to the current

            //waypoint start moving toward the next

if(Vector3.Distance(graph.getPathPoint(currentWP)

.transform.position,transform.position) <

accuracy)

            {

                currentWP++;

            }


            //if we are not at the end of the path

            if(currentWP < graph.getPathLength())

            {

                //keep on movin'
```

```
                direction =
graph.getPathPoint(currentWP).transform.position
- transform.position;

transform.rotation =
    Quaternion.Slerp(transform.rotation,
    Quaternion.LookRotation(direction),
    rotationSpeed *Time.deltaTime);
            transform.Translate(0, 0,
                Time.deltaTime * speed);
        }
        }
        else if (event == "exit")
        {
        }
    }
```

4.4.2. Perilaku Menyerang NPC musuh



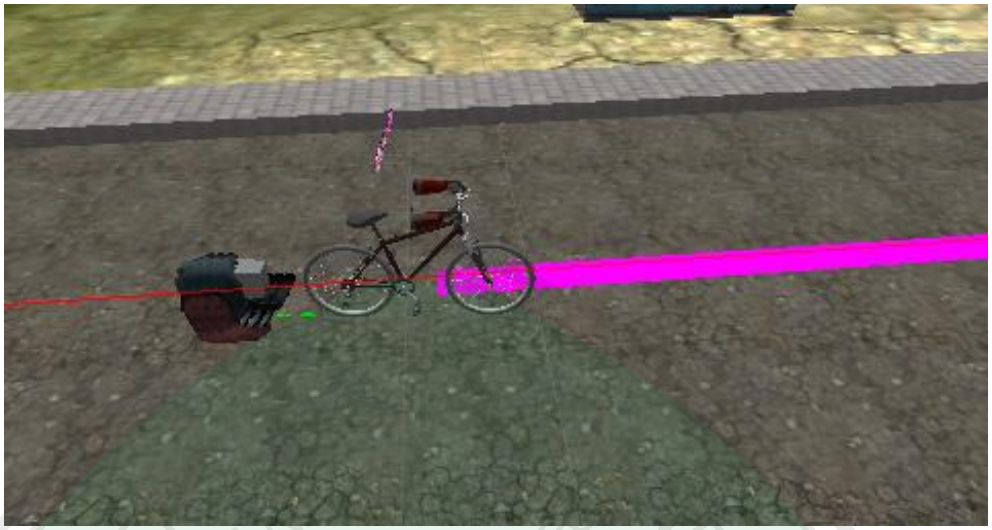
Gambar 4.12. Perilaku menyerang NPC musuh

Pada bagian ini membahas mengenai untuk mengatur perilaku NPC musuh menyerang *player*. Ketika NPC musuh mengejar *player* dan pada saat yang bersamaan *player* masih dalam jangkauan NPC musuh bahkan semakin mendekat, maka perilaku NPC musuh selanjutnya yaitu menyerang. Apabila NPC musuh berhasil menyentuh *player*, maka kesehatan *player* akan berkurang. Berikut *script* untuk mengatur berkurangnya kesehatan *player* dalam kelas *Patrol.js*:

```
else if(state == "attack")
    {
        if(prevState != state)
        {
            this.animation.CrossFade("shoot");
```

```
            this.animation["shoot"].wrapMode =
WrapMode.Loop;
        event = "update";
        }
        else if(event == "update")
        {
            position = target.position;
            direction = position -
transform.position;
            direction.y = 0;
            // Rotate towards the target
            transform.rotation =

   Quaternion.Slerp(transform.rotation,

   Quaternion.LookRotation(direction),
                rotationSpeed *
Time.deltaTime);
            transform.position.y =
target.position.y +
                heightOffset;
        }
   }
```

4.4.3. Perilaku Mengejar dan kembali Patrol NPC musuh



Gambar 4.13. Perilaku NPC mengejar

Pada bagian ini membahas mengenai untuk pengaturan perilaku NPC musuh yaitu mengejar. Berdasarkan perhitungan parameter pada skenario permainan, NPC musuh akan mengejar *player* apabila jarak *player* berada pada jangkauan NPC musuh. Sesuai dengan skenario tersebut maka dapat dibuat *script* untuk mengatur perilaku mengejar NPC musuh. Berikut *script* yang dibuat dalam kelas *Patrol.js*:

```
else if(state == "pursue")
{
if(prevState != state)
    {
```

```
this.animation.CrossFade("run");
this.animation["run"].wrapMode = WrapMode.Loop;
event = "update";
     }
     else if(event == "update")
     {
          position =
               target.gameObject.
GetComponent("breadcrumbs").breadcrumbs[0].
               transform.position;
if(Vector3.Distance(position,this.transform.positi
on)< 2)
{
target.gameObject.
GetComponent("breadcrumbs").RemoveBreadCrumb();
}
          //position = target.position;
direction = position - transform.position;
          direction.y = 0;
```

```
            // Rotate towards the target
transform.rotation =
            Quaternion.Slerp(transform.rotation,
                  Quaternion.LookRotation(direction),
                  rotationSpeed * Time.deltaTime);
            // Move the character
if(direction.magnitude > keepDistance)
{
transform.Translate(0,0,Time.deltaTime * speed);
}
transform.position.y = target.position.y +
                        heightOffset;
      }
}
else if(state == "idle")
{
this.animation.Play("idle");
event = "update";
```

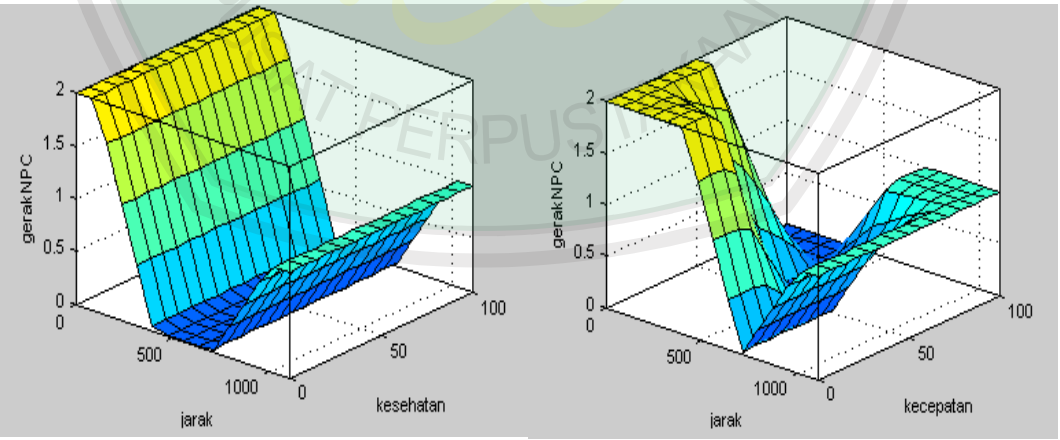
```
prevState = state;

if(event == "update")

     {

          //just remain idle most of the time

     if(Random.Range(0,500) < 1)

          {

               state = "patrol";

               event = "enter";

               currentNode = findClosestWP();

     if(currentNode ==

          GameObject.Find("Sphere17"))

     goalLocation =

          GameObject.Find("Sphere2");

     else

          goalLocation =

                    GameObject.Find("Sphere17");

          }

     }

}
```

4.5 Uji Coba

Untuk mengetahui sejauh mana implementasi algoritma terhadap NPC musuh, maka perlu dilakukan pengujian. Pengujian dilakukan beberapa kali dengan studi kasus yang berbeda dan di jalankan pada computer dengan spesifikasi sebagai berikut:

1. Prosesor intel Core2Duo T6570 @2.10 GHz
2. HardDisk 500 GB - 5600 RPM
3. RAM 4GB(2x2GB)
4. VGA IntelHD 45 series 1 Gb Shared memory
5. Keyboard
6. Monitor 14’’

4.6 Hasil Pengujian Perilaku NPC Musuh Terhadap *Player*



Gambar 4.14. Respon *fuzzy* perilaku NPC musuh dengan perhitungan Matlab

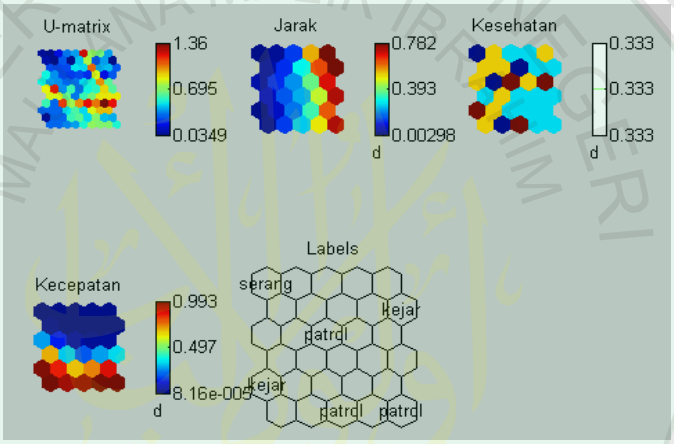
Dari variasi variable masukan yang digambarkan pada perhitungan sebelumnya dapat diperoleh keluaran yang variatif, dimana selanjutnya dikelompokkan menjadi 3 model perilaku yaitu menyerang, mengejar dan patrol. Respon keluaran *fuzzy* perilaku NPC Musuh terhadap variasi masing-masing input dipresentasikan oleh gambar *grafik* tiga dimensi diatas.

Tabel 4.1 Pengujian FuSM NPC musuh

	Nilai Input			Output
No	**Jarak**	**Kesehatan**	**Kecepatan**	**FuSM**
1	300	40	70	menyerang
2	900	70	45	patrol
3	700	95	85	mengejar
4	650	75	50	menyerang
5	650	70	90	patrol
6	800	80	50	mengejar
7	875	65	55	patrol
8	600	90	45	menyerang
9	950	50	85	patrol
10	350	30	45	menyerang
11	250	80	90	mengejar
12	400	35	85	mengejar
13	950	45	65	patrol
14	850	65	55	patrol
15	500	65	65	mengejar
16	200	70	95	mengejar
17	850	85	95	patrol
18	175	70	50	menyerang
19	225	35	70	menyerang
20	325	75	85	mengejar
21	235	45	45	menyerang
22	475	65	25	menyerang
23	325	60	35	menyerang
24	875	35	40	patrol
25	625	25	75	patrol

26	650	10	80	mengejar
27	500	70	65	mengejar
28	525	60	35	menyerang
29	375	90	85	mengejar
30	225	30	75	menyerang

Dari tabel 4.1 dapat dilihat bahwa semua *output* sudah sesuai dengan rule yang telah ditentukan. Perilaku yang dihasilkan dari *output* tersebut adalah patrol = 30.00%, mengejar = 33% dan menyerang = 36.67% dari 30 data yang ada.



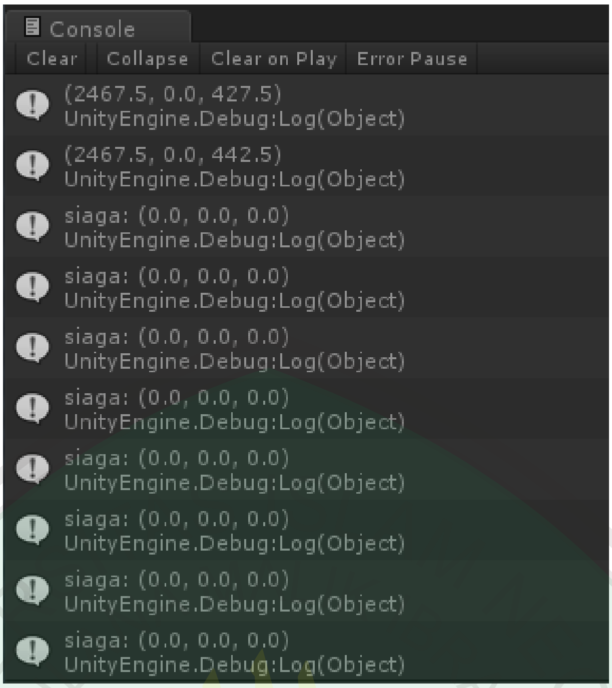
Gambar 4.15. *Grafik hexagonal* perilaku NPC musuh dalam perhitungan Matlab

Adapun juga dalam bentuk grafik *hexagonal* yang mana juga memetakan model perilaku NPC musuh. Dalam grafik dilambangkan dengan warna dengan penempatan angka yang terkecil berada pada pojok kiri atas sampai yang terbesar berada pada pojok kanan bawah. *U-matrix* yang berada pada kiri atas, komponen variabel input dan label map unit pada pojok kanan bawah, semuanya terhubung dengan posisi data yang diolah. Pewarnaan *hexagon* diumpamakan dengan panas yang artinya semakin tinggi nilai input maka warnanya akan semakin merah.

Dalam grafik warna biru berada di kiri atas karena area itu merupakan area unit yang memiliki nilai yang rendah. Sedangkan pada posisi kanan bawah berwarna merah yang menandakan posisi tersebut mempunyai nilai yang paling tinggi.

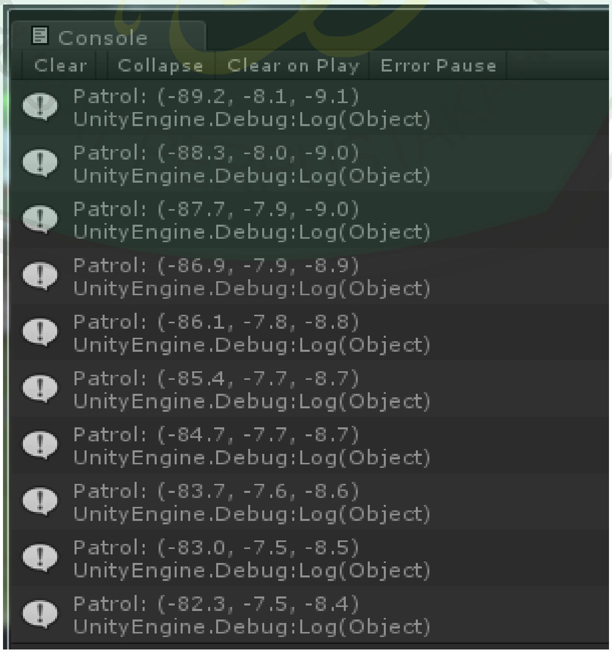
Pada masing- masing pola, *hexagon* pada beberapa posisi cocok dengan yang ada di map unit. Pada *U-matrix hexagon* tambahan berada diantara semua pasangan dari map unit yang berdekatan. Dalam *U-matrix* merupakan gambaran pola yang ada dalam label map unit. Sehingga penggambaran berdasarkan warna dilakukan dengan memunculkan warna dari *output* yang berdekatan nilainya. Sebagai contoh, pada label map unit kiri atas merupakan representasi nilai rendah dari jarak, kesehatan dan kecepatan. Posisi tersebut dalam map unit dituliskan sebagai "serang" dan pada *U-matrix* dapat dilihat bahwa posisi tersebut sangat dekat dengan unit lain yang memiliki nilai yang berdekatan.

Hasil pengujian perubahan perilaku NPC musuh terhadap *player* juga dilakukan dengan menjalankan *game,* dan melihat perilaku NPC musuh pada saat posisi *player* berada pada kondisi awal sampai kondisi *player* berhadapan atau mendekati NPC musuh dan mendapat serangan. Perubahan respon perilaku NPC musuh ditampilkan pada *console game engine* dalam *debug mode*. Berikut beberapa hasil respon perilaku NPC musuh:



Gambar 4.16. NPC musuh dalam kondisi siaga

Pada gambar 4.16 menunjukkan perilaku NPC musuh dalam kondisi siaga yang mana aksi apa yang akan diputuskan dengan melihat kondisi disekitar NPC musuh.

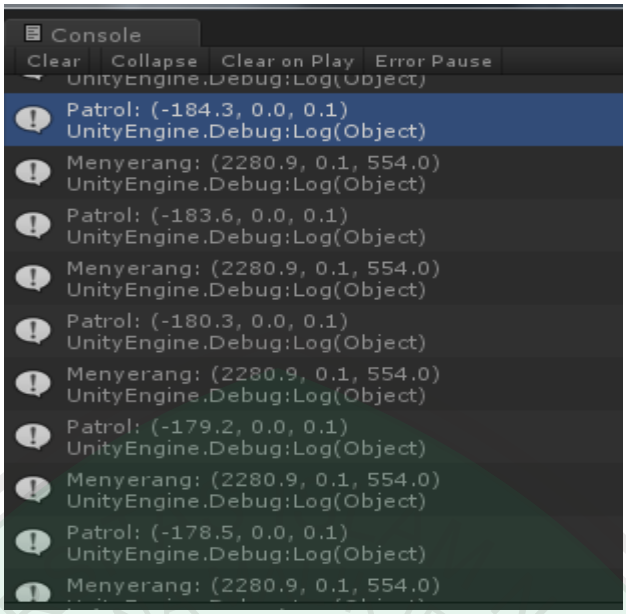


Gambar 4.17. NPC musuh kondisi patrol

Selanjutnya pada gambar 4.17 menunjukkan perilaku NPC musuh yaitu patrol dikarenakan kondisi kesehatan *player* masih penuh dan jarak NPC musuh terhadap *player* masih jauh.

Gambar 4.18. NPC musuh kondisi mengejar *player*

Pada gambar 4.18 menunjukkan salah satu perilaku NPC musuh mengejar *player*, dikarenakan posisi *player* berdekatan dengan NPC musuh. Sedangkan NPC musuh yang lain dalam kondisi patrol.



Gambar 4.19. NPC musuh kondisi menyerang *player*

Pada gambar 4.19 menunjukkan perilaku NPC musuh menyerang *player* dikarenakan jarak *player* terhadap NPC musuh sangat dekat dan dapat menjangkaunya.

BAB V

PENUTUP

5.1. Kesimpulan

Berdasarkan *game* yang telah di buat dan uji coba yang telah dilakukan, maka dapat ditarik kesimpulan yaitu Algoritma *Fuzzy State Machine* (FuSM) dapat dimplementasikan pada NPC musuh untuk menentukan perilaku NPC musuh terhadap *player*. Berdasarkan dari uji coba dari 30 data yang berbeda, perilaku NPC musuh terhadap *player* dapat dipresentasikan yaitu adalah patrol = 30.00%, mengejar = 33% dan menyerang = 36.67%.

5.2. Saran

Berdasarkan kesimpulan diatas, terdapat beberapa saran untuk pengembangan *game* ini selanjutnya:

1. *Game* yang dibangun memiliki tantangan yang kompleks dan menantang, sehingga dapat menarik.
2. Penambahan algoritma yang lebih bagus agar *game* berjalan lebih baik.
3. Penambahan beberapa NPC musuh dengan masing-masing mempunyai *variabel* yang berbeda, sehingga memiliki *output* perilaku yang berbeda juga.

DAFTAR PUSTAKA

Peters, Clifford, Aung Sithu Kyaw and Thet Naing Swe. *Unity 4.x Game AI Programming*. BIRMINGHAM - MUMBAI: Packt Publishing, 2013.

Al-Qazwiniy. *Sunan Ibn Majah.Dar al-Ihya al-Kutub al-'Arabiyah*. n.d.

Alvarez, Alberto. *"Human Gait Modelling Using a Generic Fuzzy Finite State Machine."* IEEE Journal (2012): 18 pages.

Arif, Yunifa Miftachul, Fachrul Kurniawan and Fresy Nugroho. *"Desain Perubahan Perilaku pada NPC Game Menggunakan Logika Fuzzy."* Seminar On Electrical, Informatics, And ITS Education (2011): 8.

Craig, W., Reynolds. *"Steering Behaviors For Autonomous Characters."* Boulevard, California , n.d.

Cruz, Adriano. *"Fuzzy State Machine. NCE/UFRJ." NCE/UFRJ* (2008).

Dadlos, Elmer P and Soo Ho Park. *"Real Time Robot Soccer Game Event Detection Using Finite State Machines with Multipke Fuzzy Logic Probability Evaluators."* Hindawi Publishing Corporation - International Journal of Computer *Game* Technology 10 (2009): 12 pages.

Erwanto, Haris Budi. *"Implementasi Metode Pathfinding A* Pada Player Untuk Pencarian Obstacle Dalam Game Sepeda."* (2014).

Galochkin, Igor. *Implementation of a cross-platform strategy multiplayer game based on Unity3D*. MÜNCHEN , 2013.

Kim, Chong Han, et al. *"Verification of FSM using Attributes Definition of NPCs Models."* IJCSNS International Journal of Computer Science and Network Security Vol 6 No 7A (2006).

Kusumadewi, Sri. *Analisis dan Desain Sistem Fuzzy Menggunakan Tool Box Matlab*. Yogyakarta: Graha Ilmu, 2002.

Kyaw, Aung Sithu, Clifford Peters and Thet Naing Swe. *Unity 4.x Game AI Progamming*. Birmingham: Packt Publishing Ltd, 2013.

Lee, K.H. *"First Course on Fuzzy Theory and Aplication (J. Kacprzk, Penyunt.)."* (J. Kacprzyk, Peyunt.) Taejon, Republik of South Korea : Springer (2005).

Nugroho, Supeno Mardi Susiki dkk. *"Perilaku Taktis Untuk Non-Player Characters Di Game Peperangan Meniru Strategi Manusia Menggunakan Fuzzy Logic dan Hierarchal Finite State Machine."* 6 (Januari 2011).

Putra, Fahrul Pradhana, Ahmad Zainul Fanani and Moch Hariadi. *"Perilaku Otonomi dan Adaptif Non Player Character Musuh pada Game 3 Dimensi Menggunakan Fuzzy State Machine dan Rule Based System."* Seminar Nasional Teknologi dan Komunikasi Terapan (SEMANTIK). Semarang, 2014.

Safari, Heri Ahmad, Agung Harsoyo and Kuspriyanto. *"Design and Implementation of Zoopedia : Behaviour of Non Playable Character (NPC) of Tiger Hunting the Prey."* 3rd International Conference on e-Learning (2011): 1-6.

Tim Litbang, Wahana Komputer. *Mudah Membuat Game 3 Dimensi Menggunakan Unity 3D*. Yogyakarta: Penerbit ANDI, 2014.

Wicaksono, Ady, Mochamad Hariadi and Supeno Mardi. *"Strategi Menyerang NPC Game FPS Menggunakan Fuzzy Finite State Machine."* Seminar Nasional Teknologi Informasi dan Multimedia 2013. STIMIK AMIKOM Yogyakarta, 2013.

Xiao Cui. *"Direction Oriented Pathfinding In Video Games."* International Journal of Artificial Intelligence & Applications (IJAIA) (2011): No.4.