PENERAPAN ALGORITME LOGIKA FUZZY UNTUK DYNAMIC DIFFICULTY SCALING PADA GAME LABIRIN

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Ilham Akbar Ahmadi
NIM: 145150200111061

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

LEMBAR PENGESAHAN

PENERAPAN ALGORITME LOGIKA FUZZY UNTUK DYNAMIC DIFFICULTY SCALING
PADA GAME LABIRIN
SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Ilham Akbar Ahmadi
NIM: 145150200111061

Skripsi ini telah diuji dan dinyatakan lulus pada
16 Januari 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Eriq Muh. Adams Jonemaro, S.T, M.Kom	Muhammad Aminul Akbar, S.Kom., M.T
NIP: 19850410 201212 1 001	NIK. 20160789 1013 001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP: 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 1 Januari 2018

Ilham Akbar Ahmadi

NIM: 145150200111061

KATA PENGANTAR

Segala puji dan syukur kehadirat Allah SWT atas berkah, rahamat dan hidyah-Nya yang senantiasa dilimpahakan kepada penulis, sehingga bisa menyelasaikan skripsi dengan judul "PENERAPAN ALGORITME LOGIKA FUZZY UNTUK DYNAMIC DIFFICULTY SCALING PADA GAME LABIRIN" sebagai syarat untuk menyelesaiakan Program Sarjana (S1) pada Program Sarjana Fakultas Ilmu Komputer Jurusan Informatika.

Dalam penyusunan skripsi ini banyak hambatan serta rintangan yang penulis hadapi namun pada akhirnya dapat melaluinya berkat adanya bimbingan dan bantuan dari berbgai pihak baik secara moral maupu spiritual. Untuk itu pada kesempatan ini penulis menyampaikan ucapan terimakasih kepada:

1. Tri Astoto Kurniawan, S.T, M.T, Ph.D Selaku Ketua Jurusan Informatika Fakultas Ilmu Komputer Universitas Brawijaya.
2. Eriq Muhammad Adams Jonemaro, S.T, M.Kom dan Muhammad Aminul Akbar , S.Kom., M.T selaku Dosen Pembimbing yang telah bersedia meluangkan waktu untuk memberikan arahan selama penyusunan skripsi.
3. Seluruh jajaran Dosen dan Staf Fakultas Ilmu Komputer Universitas Brawijaya.
4. Seluruh responden yang telah bersedia membantu dan meluangkan waktu dalam membantu pengujian.
5. Kedua Orang tua beserta kakak dan adik yang telah memberikan doa dan dukungan selama proses pembuatan skripsi.
6. Teman-teman Informatika 2014 yang telah memberikan semangat untuk menyelesaikan skripsi pada semester ini.
7. Semua pihak yang tidak dapat disebutkan satu persatu yang telah membantu memberikan dukungan.

Penulis menyadari bahwa skripsi ini masih jauh dari sempurna dikarenakan terbatasnya pengalaman dan pengetahuan yang dimiliki penulis. Oleh karena itu, penulis mengharapkan segala bentuk saran serta masukan bahkan kritik yang membangun dari berbagai pihak. Semoga skripsi ini dapat bermanfaat bagi para pembaca dan semua pihak khususnya dalam bidang industri permainan.

Malang, 1 Januari 2018

Penulis

Ilhamakbarahmadi10@gmail.com

ABSTRAK

Mayoritas pemain baru dalam sebuah game cenderung asal dalam memilih level. Hal ini menyebabkan terjadinya imbalance pada permainan, yang membuat game menjadi terlalu mudah atau bahkan terlalu susah. Untuk mengetahui tingkat kemampuan dan level, maka diperlukan suatu *input* dari pemain sebagai parameter yang digunakan untuk proses menentukan tingkat kesulitan pada game. Dalam menangani masalah tersebut, maka dalam penelitian ini akan dilakukan penerapan implementasi dengan metode *fuzzy* dalam membuat *Dynamic Difficulty Scaling* untuk memberikan *balance level* dalam *game* labirin. Metode pengujian yang dipakai meliputi pengujian validasi *behavior* dan performa dari permainan. Pengujian validasi *behavior* dilakukan dengan cara mencari 5 orang relawan dalam menguji permainan untuk melihat hasil keluaran *next level* labirin yang muncul serta menganalisis hasil dari *balance level* yang terjadi. Hasil dari analisis yang dilakukan yaitu metode *fuzzy* ini berhasil untuk menyetarakan *softskill* yang dimiliki pemain dengan *level* yang dimainkan. Sedangkan pengujian performa dilakukan dengan cara menguji besarnya *frames per second* (fps) yang dihasilkan selama permainan. Hasil menunjukkan bahwa performa terbaik terdapat pada map dengan ukuran 5x5 sebesar 107,22 FPS, serta performa terburuk terjadi pada labirin dengan ukuran 50x50 yang memiliki 87,65 FPS. Dari hasil pengujian performa tersebut dapat diketahui bahwa permainan ini layak untuk dimainkan mengingat batas minimum kelayakan FPS untuk permainan dikategorikan layak untuk dimainkan adalah 30 FPS.

Kata kunci: fuzzy, labirin, *dynamic difficulty scaling*

ABSTRACT

A majority of new players in a game tend to pick levels at random. This causes an imbalance in the game, which makes the game too easy or too hard. To find out their abilities and levels, input is required from players as a parameter used in the process of determining the difficulty level of a game. In order to address this issue, in this research an implementation of the fuzzy method was utilized for Dynamic Difficulty Scaling to provide a level balance in a labyrinth game. The utilized testing methods were testing of behavior validation and gameplay performance. Behavior validation testing was conducted by requesting 5 volunteers to test the game to see the next level labyrinth output and to analyze the level balance that occurred. The analysis results show that the fuzzy method succeeded in equalizing the soft skills possessed by the players with the level being played. Meanwhile, performance testing was conducted by calculating the number of frames per second (FPS) that are outputted during gameplay. The results show that the best performance was obtained for a map of 5x5 size at 107.22 FPS, while the worst performance was for a labyrinth of 50x50 size at 87.65 FPS. From the results of this performance testing, it can be concluded that the game can be reasonably played, considering that the lower limit for the FPS of a game that can be reasonably played is 30 FPS.

Keywords: fuzzy, maze, dynamic difficulty scaling

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

PENERAPAN ALGORITME LOGIKA FUZZY UNTUK DYNAMIC DIFFICULTY SCALING PADA GAME LABIRIN

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Ilham Akbar Ahmadi
NIM: 145150200111061

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

LEMBAR PENGESAHAN

PENERAPAN ALGORITME LOGIKA FUZZY UNTUK DYNAMIC DIFFICULTY SCALING
PADA GAME LABIRIN
SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Ilham Akbar Ahmadi
NIM: 145150200111061

Skripsi ini telah diuji dan dinyatakan lulus pada
16 Januari 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Eriq Muh. Adams Jonemaro, S.T, M.Kom	Muhammad Aminul Akbar, S.Kom., M.T
NIP: 19850410 201212 1 001	NIK. 20160789 1013 001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP: 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 1 Januari 2018

Ilham Akbar Ahmadi

NIM: 145150200111061

KATA PENGANTAR

Segala puji dan syukur kehadirat Allah SWT atas berkah, rahamat dan hidyah-Nya yang senantiasa dilimpahakan kepada penulis, sehingga bisa menyelasaikan skripsi dengan judul "PENERAPAN ALGORITME LOGIKA FUZZY UNTUK DYNAMIC DIFFICULTY SCALING PADA GAME LABIRIN" sebagai syarat untuk menyelesaiakan Program Sarjana (S1) pada Program Sarjana Fakultas Ilmu Komputer Jurusan Informatika.

Dalam penyusunan skripsi ini banyak hambatan serta rintangan yang penulis hadapi namun pada akhirnya dapat melaluinya berkat adanya bimbingan dan bantuan dari berbgai pihak baik secara moral maupu spiritual. Untuk itu pada kesempatan ini penulis menyampaikan ucapan terimakasih kepada:

1. Tri Astoto Kurniawan, S.T, M.T, Ph.D Selaku Ketua Jurusan Informatika Fakultas Ilmu Komputer Universitas Brawijaya.
2. Eriq Muhammad Adams Jonemaro, S.T, M.Kom dan Muhammad Aminul Akbar , S.Kom., M.T selaku Dosen Pembimbing yang telah bersedia meluangkan waktu untuk memberikan arahan selama penyusunan skripsi.
3. Seluruh jajaran Dosen dan Staf Fakultas Ilmu Komputer Universitas Brawijaya.
4. Seluruh responden yang telah bersedia membantu dan meluangkan waktu dalam membantu pengujian.
5. Kedua Orang tua beserta kakak dan adik yang telah memberikan doa dan dukungan selama proses pembuatan skripsi.
6. Teman-teman Informatika 2014 yang telah memberikan semangat untuk menyelesaikan skripsi pada semester ini.
7. Semua pihak yang tidak dapat disebutkan satu persatu yang telah membantu memberikan dukungan.

Penulis menyadari bahwa skripsi ini masih jauh dari sempurna dikarenakan terbatasnya pengalaman dan pengetahuan yang dimiliki penulis. Oleh karena itu, penulis mengharapkan segala bentuk saran serta masukan bahkan kritik yang membangun dari berbagai pihak. Semoga skripsi ini dapat bermanfaat bagi para pembaca dan semua pihak khususnya dalam bidang industri permainan.

Malang, 1 Januari 2018

Penulis

Ilhamakbarahmadi10@gmail.com

ABSTRAK

Mayoritas pemain baru dalam sebuah game cenderung asal dalam memilih level. Hal ini menyebabkan terjadinya imbalance pada permainan, yang membuat game menjadi terlalu mudah atau bahkan terlalu susah. Untuk mengetahui tingkat kemampuan dan level, maka diperlukan suatu *input* dari pemain sebagai parameter yang digunakan untuk proses menentukan tingkat kesulitan pada game. Dalam menangani masalah tersebut, maka dalam penelitian ini akan dilakukan penerapan implementasi dengan metode *fuzzy* dalam membuat *Dynamic Difficulty Scaling* untuk memberikan *balance level* dalam *game* labirin. Metode pengujian yang dipakai meliputi pengujian validasi *behavior* dan performa dari permainan. Pengujian validasi *behavior* dilakukan dengan cara mencari 5 orang relawan dalam menguji permainan untuk melihat hasil keluaran *next level* labirin yang muncul serta menganalisis hasil dari *balance level* yang terjadi. Hasil dari analisis yang dilakukan yaitu metode *fuzzy* ini berhasil untuk menyetarakan *softskill* yang dimiliki pemain dengan *level* yang dimainkan. Sedangkan pengujian performa dilakukan dengan cara menguji besarnya *frames per second* (fps) yang dihasilkan selama permainan. Hasil menunjukkan bahwa performa terbaik terdapat pada map dengan ukuran 5x5 sebesar 107,22 FPS, serta performa terburuk terjadi pada labirin dengan ukuran 50x50 yang memiliki 87,65 FPS. Dari hasil pengujian performa tersebut dapat diketahui bahwa permainan ini layak untuk dimainkan mengingat batas minimum kelayakan FPS untuk permainan dikategorikan layak untuk dimainkan adalah 30 FPS.

Kata kunci: fuzzy, labirin, *dynamic difficulty scaling*

ABSTRACT

A majority of new players in a game tend to pick levels at random. This causes an imbalance in the game, which makes the game too easy or too hard. To find out their abilities and levels, input is required from players as a parameter used in the process of determining the difficulty level of a game. In order to address this issue, in this research an implementation of the fuzzy method was utilized for Dynamic Difficulty Scaling to provide a level balance in a labyrinth game. The utilized testing methods were testing of behavior validation and gameplay performance. Behavior validation testing was conducted by requesting 5 volunteers to test the game to see the next level labyrinth output and to analyze the level balance that occurred. The analysis results show that the fuzzy method succeeded in equalizing the soft skills possessed by the players with the level being played. Meanwhile, performance testing was conducted by calculating the number of frames per second (FPS) that are outputted during gameplay. The results show that the best performance was obtained for a map of 5x5 size at 107.22 FPS, while the worst performance was for a labyrinth of 50x50 size at 87.65 FPS. From the results of this performance testing, it can be concluded that the game can be reasonably played, considering that the lower limit for the FPS of a game that can be reasonably played is 30 FPS.

Keywords: fuzzy, maze, dynamic difficulty scaling

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1 PENDAHULUAN

1.1 Latar belakang

Game adalah suatu program *virtual* yang dimainkan dengan menyelesaikan konflik buatan sebagai aturan permainan (Dogan, 2014). *Game* yang kita kenal saat ini pada umumnya memiliki beberapa tingkat kesulitan, dimulai dari kesulitan tingkat mudah (*easy*), sedang (*medium*), hingga sulit (*hard*). Namun tak sedikit pula game yang memberi kategori lebih untuk tingkat kesulitan, seperti pemain baru (*beginner*), sangat sulit (*very hard*), dan ahli (*master*). Perubahan tingkat kesulitan atau yang biasa disebut dengan istilah level dalam game memiliki fungsi untuk merubah tingkat kesulitan dalam permainan, semakin tinggi level yang diambil, maka tingkat kesulitan untuk menyelesaikan permainan juga akan semakin sulit.

Mayoritas pemain baru dalam sebuah game cenderung asal dalam memilih *level*. Hal ini menyebabkan terjadinya imbalance pada permainan, yang membuat game menjadi terlalu mudah atau bahkan terlalu susah. Kasus sering terjadi dikarenakan pemain tidak atau belum mengetahui kemampuan atau *softskill* mereka dalam bermain *game* yang baru dimainkan. *Softskill* merupakan istilah sosiologis yang berkaitan dengan kecerdasan emosional, sifat kepribadian, keterampilan, komunikasi, Bahasa, kebiasaan, keramahan, dan optimiste yang mencerminkan kemampuan seseorang (Sri Yuliani, 2015). Hal itu akan berdampak pada pemain, dimana pemain kurang mendapatkan *feedback* yang diharapkan saat pertama kali memainkan *game* yang pertama kali dimainkan.

Penentuan *level* permainan sebenarnya sudah banyak diterapkan dalam pembuatan game, namun dalam penerapannya masih cenderung hanya memiliki satu arah. Contohnya *game* "Piano Tiles" dimana tempo rythme dari permainan akan semakin cepat seiring lamanya permainan berlangsung, dan *game* "Neo fm" dimana setiap pemain melakukan "x" kali kesalahan, maka *level* permainan akan diturunkan ke 1 tingkat dibawah *current level*. Hal ini dilakukan demi menciptakan *balance* pada *game* untuk membuat pemain tertarik untuk menyelesaikan permainan. *Game Balancing* telah diakui oleh komunitas pengembang *game* sebagai salah satu faktor kunci dari kesuksesan permainan komputer (Falstein 2004). Maka dari itu diambillah sebuah metode untuk memberikan sebuah desain sistem yang dapat membaca kemampuan pemain serta menjadikannya sebagai parameter penentu pada level permainan yang dimainkan, yaitu dengan membuat *Dynamic Difficulty Scaling* yang difokuskan dalam pengkategorian kemampuan pemain dalam *scale* khusus untuk pengkategorian kemampuan pemain dan tingkat kesulitan sistem.

Untuk mengetahui tingkat kemampuan dan *level*, maka diperlukan suatu input dari pemain yang kemudian dijadikan sebagai parameter yang

kemudian diproses untuk menentukan tingkat kesulitan pada *game*. Pada tahap ini algoritme *Fuzzy* merupakan salah satu algoritme yang sesuai untuk membuat pengkategorian *level* berdasarkan tingkat kemampuan dari pemain. Logika *Fuzzy* sendiri merupakan suatu metode pendekatan sederhana yang dilakukan secara satuan untuk memindahkan aspek linguistic kedalam model matematis yang kemudian dapat digunakan untuk memberifikasi validasi dari penjelasan verbal. Logika *Fuzzy* merupakan salah satu komponen pembentuk *Soft Computing* dimana *membership function* menjadi ciri utama dari penalaran logika *fuzzy* (Kusumadewi S, Purnomo J, 2010). Algoritme *Fuzzy* dikatakan cocok untuk penerapan *level* karena keunggulannya dalam pemodelan aspek kualitatif dari pengetahuan dan kemampuan pemain, serta keputusan yang dibuat oleh pemain dengan menerapkan aturan dasar.

Tujuan dari penelitian ini adalah mengimplementasi algoritme *Fuzzy* yang digunakan sebagai pembuatan model untuk menentukan tingkat kesulitan otomatis berdasarkan kemampuan pemain saat dalam permainan, dimana tingkat kesulitan bisa bertambah, bisa berkurang, dan bisa bertahan pada tingkat kesulitan tertentu jika tingkat kesulitan itu dikategorikan sesuai dengan kemampuan pemain saat ini.

1.2 Identifikasi Masalah

Adapun identifikasi masalah yang diperoleh dari latar belakang adalah :

1. Kesulitan pemain dalam menentukan *level* permainan yang sesuai dengan kemampuannya.
2. Kurangnya minat pemain dalam menyelesaikan *game* yang memiliki tingkat kesulitan yang tidak sesuai.
3. Penerapan algoritme *Fuzzy* dalam *Dynamic Difficulty Scaling* untuk mengatur tingkat kesulitan permainan.

1.3 Rumusan masalah

Adapun rumusan masalah yang diangkat pada penelitian ini adalah :

1. Bagaimana cara menerapkan algoritme *fuzzy* dalam *Dynamic Difficulty Scaling* yang berperan sebagai penentu *balance level* untuk menentukan level dari *game* labirin?
2. Bagaimana hasil pengujian kinerja dari game labirin setelah dilakukan pengimplementasian logika *fuzzy*?

1.4 Tujuan Penelitian

Adapun tujuan yang ingin dicapai dari penelitian ini adalah

1. Menerapkan algoritme *fuzzy* dalam *Dynamic Difficulty Scaling* untuk memberikan fitur balance level pada game labirin.

2. Melakukan pengujian kinerja dari *game* labirin setelah pengimplementasian algoritme *fuzzy* pada *game.*

1.5 Manfaat Penelitian

Penerapan algoritme *fuzzy* dalam *Dynamic Difficulty Scaling* dapat meningkatkan *balance* agar dapat memberikan *feedback* tingkat kesulitan yang sesuai kepada pemain.

1.6 Batasan masalah

Adapun Batasan masalah pada penelitian ini adalah :

1. Penerapan algoritme *fuzzy* ditujukan kepana development game dalam membuat *scaling level* permainan.
2. Diterapkan dalam mode *singleplayer.*
3. algoritme *fuzzy* akan diterapkan pada game *puzzle labirin* sederhana yang dibuat oleh peneliti.
4. Aplikasi bekerja secara offline.

1.7 Sistematika pembahasan

Sistematika laporan skripsi dijelaskan sebagai berikut.

BAB I. Pendahuluan

Dalam bab pendahuluan, akan dijelaskan mengenai latar belakang, indentifikasi masalah, batasan masalah, tujuan penelitian, manfaat penelitian, batasan penelitian, dan sistematika pembahasan.

BAB II. Tinjauan Pustaka

Berisi landasan teori mengenai segala sesuatu yang diperlukan penulis untuk mendukung pembuatan skripsi berjudul "PENERAPAN ALGORITME LOGIKA FUZZY UNTUK DYNAMIC DIFFICULTY SCALING PADA GAME LABIRIN", yang meliputi *game, fuzzy,* kecerdasan buatan dalam *game, dynamic difficulty scaling,* dan Unity

BAB III. Metode Penelitian

Berisi metode yang digunakan dalam melakukan penelitian yang diuraikan dalam kajian pustaka, pengumpulan dan pengolahan data, desain *system fuzzy*, implementasi, serta pengujian dan analisis.

BAB IV. Perancangan

Berisi tentang perancangan metode *fuzzy* yang nantinya akan diterapkan pada *game* labirin.

BAB V. Implementasi

Berisi tentang spesifikasi computer yang digunakan, isi dari *source code* dalam implementasi *fuzzy,* pembahasan dari *source code* tersebut, serta gambaran singkat tentang *game* labirin.

BAB VI. Pengujian dan Analisis

Berisi tentang hasil pengujian yang dilakukan, serta hasil analisis dari pengujian.

BAB VII. Pengujian dan Analisis

Berisi tentang hasil pengujian yang dilakukan, serta hasil analisis dari pengujian.

BAB VIII. Penutup

Berisi tentang hasil Kesimpulan dan saran dari penelitan.

BAB 2 KAJIAN PUSTAKA

2.1 Game

Dalam kamus besar Bahasa Indonesia *game* diartikan sebagai permainan. Dalam hal ini pengertian permainan merujuk pada kelincahan intelektual *(Intellectual Playability Game)* yang dapat diartikan sebagai arena keputusan dan aksi pemainnya. Pada awalnya *game* merupakan istilah untuk suata kegiatan yang dapat menyenangkan hati anak-anak.

Manusia belajar melalui pengalaman, Pengalaman dalam bermain yang menyenangkan dengan media yang benar serta dukungan dari orang lain dapat membantu anak berkembang secara optimal (Mutiah, 2010). Dengan media yang tepat, sebuah game dapat berubah menjadi sarana yang cocok bagi pembelajaran siswa, melalui kesenangan, proses berpikir, dan pengalaman yang didapat dari bermain game, suatu materi dapat tersampaikan dengan baik.

Kasifikasi *game* berdasarkan tujuan penggunaannya terbagi menjadi 3, yaitu:

1. *Game as Game, Game* yang bertujuan untuk kesenangan semata.
2. *Game as Media, Game* yang bertujuan sebagai alat penyampai pesan yang dimuat dalam bentuk permainan.
3. *Game Beyond Game, Game beyond game* atau biasa disebut *gamification* merupakan konsep atau cara berpikir pengimplementasian *game design* untuk umum atau non *game*.

Game as media merupakan jenis *game* yang berpengaruh baik bagi proses belajar. *Game* edukatif adalah permaian yang dirancang atau dibuat untuk membantu merangsang daya pikir termasuk meningkatkan konsentrasi dan memecahkan masalah (Handriyantini, 2009). Jenis *game* ini memungkinkan pemain dalam bermain sekaligus meningkatkan konsentrasi pemain dalam memecahkan teka-teki untuk menyelesaikan *game*.

Terdapat dua jenis game, yaitu *game online* dan *game offline.* Pada dasarnya pengertian dari *game online* dan *game offline* hampir sama, namun pada *game online*, Pengguna memerlukan koneksi internet yang memungkinkan pemain terhubung ke network untuk dapat bermain bersama pemain lain, baik secara langsung maupun tidak langsung, *game online* juga cenderung menggunakan waktu yang sama untuk menyelaraskan waktu bermain.

Adapun elemen-elemen penting dalam sebuah game, diantaranya bermain (*play*), berpura-pura (*pretend*), tujuan (*goal*), dan peraturan (*rules*) (Ernest Adams, 2010:6-9). Definisi dari elemen tersebut adalah sebagai berikut :

1. Bermain (*Play)*

   Bermain merupakan sebuah sarana hiburan untuk menghibur diri. Ketika bermain sebuah permainan, pemain dapat membuat keputusan yang mempengaruhi jalannya suatu permainan yang menjadikan permainan itu terasa lebih hidup dan menghibur.

2. Berpura-pura (*Pretending*)

   Berpura-pura merupakan suatu kegiatan yang memberikan sensasi realitas dari khayalan yang ada dalam fikiran manusia. Dalam penerapannya pada *game* mengacu pada batas yang memisahkan dunia *game* dan dunia nyata. Dengan kata lain batas antara dunia nyata dan khayalan.

3. Tujuan (*Goal*)

   Sebuah *game* harus memiliki tujuan. Tujuan dari *game* terbentuk dari peraturan dan bentuk *game* itu sendiri sesuai dengan keinginan pembuat atau pemain itu sendiri.

4. Peraturan (*Rules*)

   Peraturan merupakan suatu Batasan yang tidak boleh dilanggar oleh pemain saat memainkan sebuah permainan. Peraturan memungkinkan pemain untuk mengetahui aktifitas apa yang diperbolehkan serta mengevaluasi setiap tindakan yang membuat pemain dapat mencapat *goal* dari *game* tersebut.

2.2 Fuzzy Logic

Logika *fuzzy* adalah salah satu cara untuk menentukan suatu ruang *input* kedalam suatu ruang *output.* Logika *fuzzy* pertama kali dibuat dan dikenalkan oleh Lofti A Zadeh pada tahun 1965, dimana dalam paper yang dibuatnya dia memperkenalkan teori yang memiliki obyek dari himpunan fuzzy yang dinyatakan dalam derajat (*degree*). Konsepini dikenal dengan istilah *Fuzziness* dan teorinya dinamakan *Fuzzy Set Theory.*

Logika *fuzzy* memungkinan nilai keanggotaanya antara salah (0) dan benar (1), dimana tingkat kebenarannya keabuan dan bisa juga hitam dan putih, dalam bentuk linguistic. Konsep tidak pasti seperti “sedikit”, “jarang”, “lumayan”, dan “sering” (Zadeh, L, 1965). Operasi yang digunakan dalam logika dan himpunan *fuzzy* sama dengan dalam logika dan himpunan biasa, namun memiliki definisi yang agak berbeda.

1. Gabungan

Merupakan gabungan antara himpunan A dengan himpunan B.

2. Irisan

   Merupakan potongan himpunan yang merupakan milik himpunan A dan himpunan B.

3. Komplemen

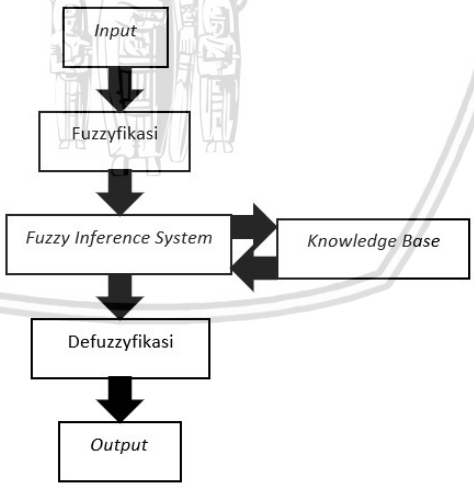
   Merupakan himpunan yang tidak merupakan bagian dari himpunan A.

Konsep dasar yang merupakan bagian dari logika *fuzzy* disebutkan sebagai berikut

1. Derajat keanggotaan
2. Label
3. Fungsi keanggotaan
4. Masukan crisp
5. Lingkup / domain
6. Daerah Batasan crisp

2.2.1 Struktur Dasar Logika *Fuzzy*

Struktur dasar dari logika *fuzzy* ditunjukkan pada gambar 2.1.



Gambar 2.1 Struktur Dasar Logika *Fuzzy*

Pada gambar 2.1 ditunjukkan bahwa logika *fuzzy* memiliki struktur dasar dalam proses perhitungannya. Proses yang dilakukan pertama adalah memberikan *input* berupa variabel. Selanjutnya dari *input* variabel tersebut dilakukan proses fuzzyfikasi yaitu proses untuk menghitung derajat keanggotaan dari setiap variabel. Selanjutnya

dilakukan proses *fuzzy inference system* yaitu proses untuk mencari nilai dari setiap derajat keanggotaan berdasarkan pengetahuan yang ada pada *knowledge base*. Selanjutnya adalah proses defuzzyfikasi yaitu proses untuk merubah nilai *output* keluaran fuzzy menjadi nilai crisp yang menentukan hasil dari output.

2.2.2 Himpunan *Fuzzy*

Himpunan *fuzzy* adalah konsep yang mendasari terbentuknya logika *fuzzy*, yang merupakan suatu himpunan dengan anggota yang memiliki derajat keanggotaan tertentu yang telah ditentukan oleh fungsi keanggotaan (*membership function*) tertentu atau disebut juga fungsi karakteristik (*characteristic function*). Himpunan *fuzzy* memungkinkan untuk suatu data masuk kedalam 2 himpunan yang berbeda.

2.2.3 Himpunan *Crisp*

Sebelum masuk pada proses *fuzzyfikasi*, pada himpunan tegas (*crisp*), nilai keanggotaan suatu item x dalam suatu himpunan A, yang sering ditulis dengan μA[x], memiliki 2 kemungkinan (Kusumadewi S, Purnomo H, 2010) yaitu:

1. Satu (1), yang berarti bahwa suatu item menjadi anggota dalam suatu himpunan, atau
2. Nol (0), yang berarti bahwa suatu item tidak menjadi anggota dalam suatu himpunan.

2.2.4 Fuzzyfikasi

Fuzzyfikasi merupakan langkah awal dalam proses *fuzzy logic* yang mengandung transformasi *domain*. Untuk melakukan transformasi himpunan crisp kedalam *input fuzzy*. Adapun langkah-langkah yang perlu dilakukan, diantaranya adalah menetukan himpunan *fuzzy* dan menentukan fungsi keanggotaan.

2.2.5 Fuzzy Inference System

Fuzzy Inference System merupakan proses untuk mencari predikat dari setiap *rule base* yang ada pada *knowledge base* yang telah diberikan.

2.2.6 Knowledge Base

Knowledge base merupakan dasar dari pengetahuan yang digunakan dalam knowledge management. *Knowledge base* bisa didapat dari data hasil riset atau bisa juga merupakan data yang

diberikan oleh si pembuat. Knowledge base berisi aturan aturan yang sering ditulis dengan perintah ***IF… THEN…***

2.2.7 Defuzzyfikasi

Defuzzyfikasi merupakan langkah terakhir dalam system logika fuzzy. Berisi skema atau proses yang dilakukan untuk mengkonveksi setiap hasil dari output *fuzzy* menjadi nilai *crisp*.

2.3 Kecerdasan Buatan Dalam *Game*

Kecerdasan buatan atau yang biasa disebut dengan *Artificial Intelligence* (AI) merupakan suatu system yang memiliki kemampuan untuk berfikir serta mengambil tindakan berdasarkan pola tertentu yang diperoleh dari interaksi yang dilakukan, baik dari user, maupun yang ditanamkan oleh pembuatnya. Menurut Kusumadewi (2003), Kecerdasan buatan atau *Artifical Intelligence* merupakan salah satu bagian ilmu komputer yang membuat computer dapat berfikir dan melakukan suatu pekerjaan layaknya manusia.

Kecerdasan buatan menyediakan dasar dasar ilmu yang dapat diterapkan pada berbagai bidang teknologi, antara lain

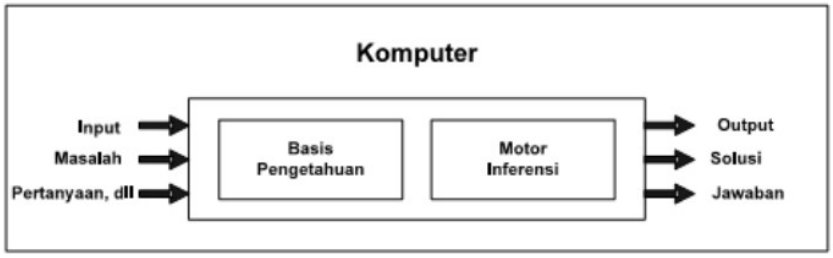
a. Sistem Pakar (*expert system),*
b. Pengolahan Bahasa Alami (*natural language processing),*
c. Pengenalan Suara/Ucapan *(Speech Recognition),*
d. Robotika dan Sistem Sensor *(Robotic and Sensory System),*
e. Imprementasi Gambar dan Object *(Computer Vision),*
f. Proses Pembelajaran/Tutor *(Intelligent Computer -aided Instruction),*
g. Permainan *(Game Playing).*

Pada masa sekarang, kecerdasan buatan paling popular dalam bidang pengembangan *Academic AI* dan *Game AI* (Millington, 2009). Dalam penerapannya dalam game, umumnya AI banyak diterapkan untuk memberikan sensasi nyata kepada user dengan mengendalikan tingkat kesulitan yang terjadi dalam permainan berdasarkan input yang diterima. Hal itu didapat dari basis data, pengetahuan, dan kumpulan informasi yang diperoleh dari analisa pengalaman dan input yang dilakukan oleh user, atau dapat diperoleh sejak pembuatan kecerdasan itu sendiri yang ditanamkan oleh pembuatnya.

Untuk menciptakan sebuah kecerdasan buatan, terdapat 2 komponen utama yang sangat dibutuhkan, antara lain :

a. *Knowledge Based* (Basis Pengetahuan), berisi tentang teori, hubungan antara satu dengan attribute lainnya, teori, serta fakta fakta yang sesuai.

b. *Inferensi Engine* (Motor Inferensi), yaitu kemampuan suatu engine dalam menarik kesimpulan berdasarkan pengalaman



Gambar 2.2 Penerapan Konsep Kecerdasan Buatan Pada Sistem Komputer

Sumber : Kusumadewi, 2003

2.4 Dynamic Difficulty Scaling

Dynamic Difficulty Scaling adalah sebuah dinamika untuk mengatur tingkat kesulitan dalam *game* secara *real time* yang dalam penerapannya memberikan tingkat kesulitan yang sesuai dengan *input* dari kemampuan *player* saat melakukan aksi dalam *game (V. Chapman, 2004)*. Pola dinamis pada *Dynamic Difficulty Scaling* membuat tingkat kesulitan dapat berubah dengan dinamis secara otomatis naik atau turun tergantung progress atau kemampuan *player.*

Level permainan sangatlah berperngaruh pada jalannya *game*, dimana hal itu dapat meningkatkan kepuasan pemain dalam melaksanakan aksinya saat bermain *game.* Umumnya penerapan *difficult* pada *game* terdiri dari 3 tingkat, yaitu *easy* (mudah), *medium* (sedang), dan *hard* (susah). Namun tak sedikit pula yang menambahkan tingkat kesulitan sendiri untuk *beginner* (pemula) dan *master* (ahli). Hal ini terbentuk untuk memberikan lebih banyak variasi kepada *player* dalam menentukan tingkat kesulitan *game.*

2.5 Unity3D

Unity3D adalah suatu perangkat lunak yang mendukung pembuatan *game multi platform* berbasis 3D dengan desain yang mudah digunakan. *Unity Technologies* didirikan pada tahun 2004 oleh David Helgason (CEO), Nicholas Francis (CCO), dan Joachim Ante (CTO) di Copenhagen, Denmark. Grafis yang mendukung OpenGL dan directX untuk menghasilkan grafis tingkat tinggi juga merupakan salah satu keunggulan dari Unity3D. Unity dapat digunakan sebagai sarana pembuatan video *game* 3D, real time animasi 3D, dan visualisasi arsitektur serta hal serupa yang interaktif lainnya (Roedavan R, 2014).

Fitur-fitur Unity adalah sebagai berikut :

1. *Rendering*
   Graphics engine yang digunakan adalah Direct3D (Windows, Xbos 360), OpenGL (Max, Windows, Linux, PS3), OpenGL ES (Android, iOS), dan proprietary APIs (Wii).

   Unity mendukung pengambilan asset dari aplikasi lain melalui pengaturan *graphical user interface Unity,* dengan format desain 3ds Max, Maya, Sortimage, Blender, modo, ZBrush, Cheetah3D, Adobe Photoshop, Adobe Fireworks, dan Allegorithmic Substance kedalam game project.

2. ShaderLab
   ShaderLab adalah Bahasa yang digunakan untuk memberikan deklaratif "*programming*" dari *fixed-function pipeline*, *shader* dapat deprogram dalam GLSL atau Cg. Shader digunakan sebagai pendeteksi video card terbaik saat ini yang dapat *compatible* dengan aplikasi.

3. *Scripting*
   Scripting dalam game engine Unity defaultnya dapat dibuat dengan Mono, sebuah implementasi open-source dari .NET Framework. Bahasa yang dapat digunakan adalah C#, Boo (pengembangan *syntax* dari phyton), dan UnityScript (Bahasa kustomisasi dari ECMAScript dalam bentuk JavaScript). Setelah rilisnya versi 3.0, kini Unity menyertakan versi MonoDevelop yang terkustomisasi untuk melakukan *debug script.*

4. *Asset Tracking*
   Server Unity Asset merupakan sebuah solusi terkontrol untuk developer Game asset dan script. Server Unity Asset dilengkapi dengan PostgreSQL sebagai *backend,* FMOD library sebagai *system audio*, Theora codec, *engine* daratan dan vegetasi sebagai *video playback.* Beast sebagai *built-in lighmapping* dan *global illumination*, RakNet sebagai *multiplayer networking*, dan navigasi *mesh* pencari jalur *built-in.*

5. *Platform*
   Platform yang saat ini didukung adalah BlackBerry 10, Windows, Windows Phone, Mac, iOS, Unity Web Player, Linux, Android, Adobe Flash, PlayStation, PlayStation Vita, Xbox 360, Wii U dan Wii.

6. *Asset Store*
   Asset Store memberikan lebih dari 4400 asset packages beserta seluruh komponen didalamnya yang meliputi *3D models, textures*

dan materials, paricle, tutorial dan project, musik dan efek suara, *scripting package, editor extensions* dan *servis online.*

7. *Physic*
   Sistem penambahan kemampuan untuk mensimulasikan *real-time cloth* pada *arbitrary and skinned meshes, collision layers*, dan *thick ray cast.*

BAB 3 METODOLOGI PENELITIAN

Pada bab ini akan dijelaskan mengenai prosedur dan kegiatan-kegiatan yang akan dilakukan dalam pengerjaan skripsi. Dimulai dari kajian pustaka, analisis kebutuhan, pengumpulan dan pengolahan data, desain sistem, implementasi, pengujian, analisis hasil, dan penarikan kesimpulan dan saran.

Gambar 3.1 Diagram alur metodologi penelitian

3.1 Kajian Pustaka

Kajian pustaka merupakan tahap penelusuran pengetahuan dalam rangka menyusun dasar teori yang digunakan sebagai dasar pengetahuan dalam penelitian ini. Penelusuran pengetahuan ini bersumber dari buku, jurnal, dan internet (website resmi) yang berkaitan dengan informasi yang dibutuhkan dalam mengerjakan penelitian ini. Teori yang akan digunakan oleh penulis sebagai landasan dasar pengetahuan dalam melaksanakan penelitian ini.

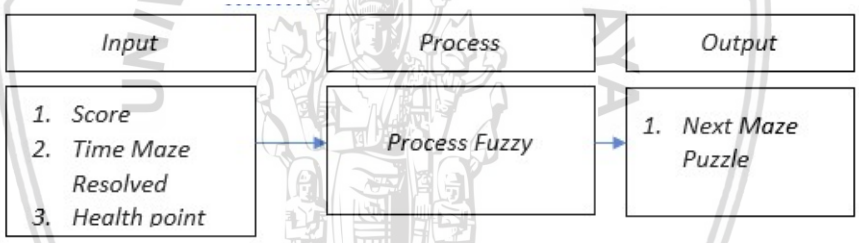
3.2 Pengumpulan dan Pengolahan Data

Pengumpulan data merupakan tahap pengumpulan informasi terkait isi atau konten yang digunakan dalam penelitian ini. Data yang dibutuhkan yaitu data jurnal dan artikel yang mendukung dalam pengimplementasian algoritme *logika fuzzy,* serta beberapa artikel mengenai rancangan data terkait yang kemudian diolah menjadi rumus algoritme yang nantinya digunakan dalam pembuatan program.

3.3 Desain Penerapan Fuzzy

Tahap desain sistem dilakukan ketika semua kebutuhan sistem dan data telah diperoleh melalui pengumpulan dan pengolahan data. Pada tahap ini akan dibangun beberapa diagram perancangan sistem, yang meliputi diagram alir aplikasi, diagram alir logika *fuzzy*, fungsi keanggotaan, *knowledge base*, serta perhitungan manual yang dilakukan.

Rancangan system yang digunakan dalam menyelesaikan masalah dapat digambarkan pada gambar 3.2.



Gambar 3.2 Diagram rancangan sistem

Berdasarkan gambar 3.2 ditunjukkan bahwa perancangan dari sistem yang dilakukan adalah dengan memberikan suatu *input* yang berupa variabel *score, time maze resolved,* dan *health point*, *input* tersebut yang kemudian akan diproses pada proses perhitungan *fuzzy,* yang pada akhirnya akan mengeluarkan *output next maze puzzle* atau level *maze* yang akan muncul pada level selanjutnya. *Next maze puzzle* yang keluar adalah bisa merupakan *level* yang berbeda, yang dimaksud dengan level disini adalah besarnya *range maze* yang di*generate*.

3.4 Implementasi

Pada tahap implementasi akan dilakukan implementasi terhadap hasil rancangan aplikasi yang telah dilakukan pada tahap desain sistem. Penulis akan memanfaatkan algoritme *logika fuzzy* sebagai dasar algoritme dalam pengembangan aplikasi *game* labirin ini. Pada tahap implementasi akan ditampilkan beberapa *source code* dari blok model pada aplikasi *game* labirin

yang mewakili fitur pada aplikasi ini. *Source code* tersebut akan dapat menjelaskan beberapa proses yang terjadi dalam aplikasi game labirin yang dikembangkan oleh penulis.

Dengan memanfaatkan perangkat keras berupa: Processor 3,4GHz, RAM 12 GB dengan Sistem Operasi Windows 10 64-bit, dan Perangkat Unity. Tahapan dalam implementasi program meliputi pembuatan antarmuka pengguna serta pengimplementasian algoritme. Implementasi yang dilakukan akan menghasilkan game labirin dengan algoritme logika *Fuzzy* sebagai laga perilaku *enemy*.

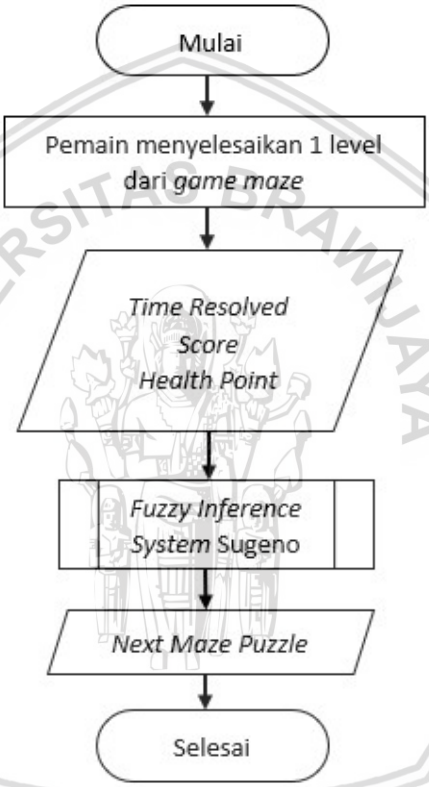
3.5 Pengujian dan Analisis

Pengujian performa yang dipakai meliputi pengujian validasi *behavior* dan performa dari permainan. Pengujian validasi *behavior* dilakukan dengan cara mencari 5 orang relawan dalam menguji permainan untuk melihat hasil keluaran *next level maze* yang muncul serta menganalisis hasil dari *balance level* yang terjadi. Sedangkan pengujian performa dilakukan dengan cara menguji besarnya *frames per second* (fps) yang dihasilkan selama permainan.

BAB 4 PERANCANGAN

4.1 Diagram Alir Kinerja Aplikasi

Pada tahap ini akan dijelaskan mengenai alur kinerja dari aplikasi dalam menentukan *output difficulty* dari permainan labirin dengan menggunakan metode *Fuzzy Inference System* Sugeno. Diagram alir kinerja aplikasi ditunjukkan pada gambar 4.1.

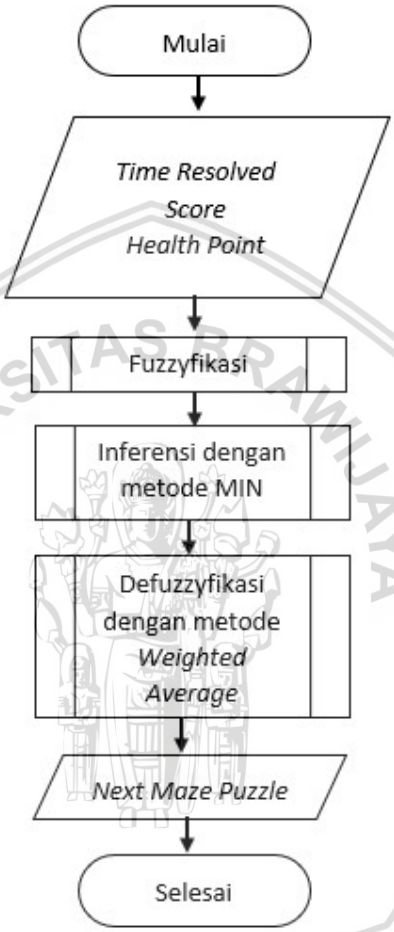


Gambar 4.1 Diagram alir kinerja aplikasi

Berdasarkan gambar 4.1 alur kinerja aplikasi dimulai dari pemain menyelesaikan 1 *level* dari *game* labirin dari *start* hingga *finish*. Setelah pemain sampai pada titik *finish*, kemudian akan dihitung nilai dari *Time Resolved* yang merupakan waktu penyelesaian dari level tersebut, *Score* yang merupakan total skor dari perolehan pemain mengambil koin pada *level* tersebut, dan *Health Point* yang merupakan sisa dari *health point* yang dimiliki pemain. Kemudian dimulai proses *Fuzzy Inference System* Sugeno yang menghasilkan nilai dari kemungkinan *Next Maze Puzzle* yang akan ditampilkan pada *level* labirin selanjutnya.

4.2 Diagram Alir Kinerja Logika *Fuzzy Inference System Sugeno*

Pada tahap ini akan dijelaskan mengenai proses perhitungan dengan metode *Fuzzy Inference System* Sugeno. Diagram alir kinerja logika *fuzzy* ditunjukkan pada gambar 4.2.



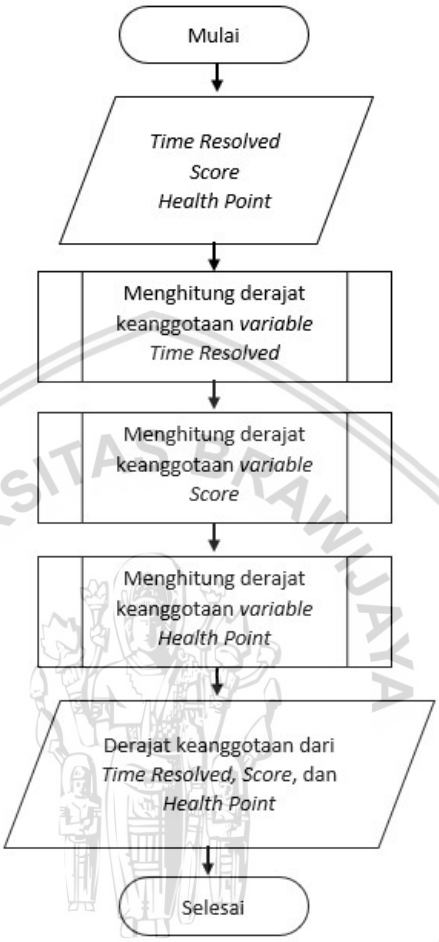
Gambar 4.2 Diagram alir rancangan metode Fuzzy Inference System Sugeno

Berdasarkan gambar 4.2 proses perhitungan menggunakan metode *Fuzzy Inference System* Sugeno dengan masukan variabel *Time Resolved*, *Score*, dan *Health Point*, kemudian dimulailah proses fuzzyfikasi, inferensi, dan defuzzyfikasi yang menghasilkan nilai dari *Next Maze Puzzle*.

4.2.1 Diagram Alir Fuzzyfikasi

Pada proses fuzzyfikasi dilakukan proses pemetakan nilai *crisp* (numerik) ke dalam himpunan fuzzy serta menentukan derajat keanggotaan dari setiap variabel (*Time Resolved, Score*, dan *Health*

Point) dalam himpunan *fuzzy*. Diagram alir fuzzyfikasi ditunjukkan pada gambar 4.3.



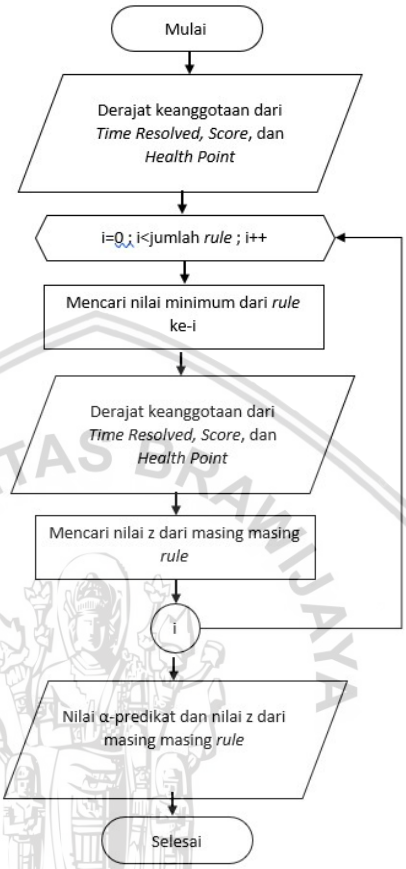
Gambar 4.3 Diagram alir proses fuzzyfikasi.

Berdasarkan gambar 4.3 ditunjukkan alir kinerja dari proses fuzzyfikasi. Dari variabel yang masuk, dilakukan perhitungan derajat keanggotaan dari setiap variabelnya, yaitu yang menghitung derajat keanggotaan dari *Time Resolved*, selanjutnya menghitung derajat keanggotaan dari *Score*, dan terakhir menghitung derajat keanggotaan dari *Health Point*. Nilai dari ketiga derajat keanggotaan itu yang nantinya dikembalikan dan diproses lagi pada proses implikasi.

4.2.2 Diagram Alir Inferensi

Pada proses inferensi (mencari nilai α-predikat dan nilai z) dengan fungsi implikasi MIN. Fungsi implikasi MIN dilakukan untuk mencari nilai minimum dai derajat keanggotaan masing-masing *rule* yang sudah didapatkan sebelumnya pada proses fuzzyfikasi.

Sedangkan nilai z diperoleh dari konsekuen masing-masing *rule*. Diagram alir inferensi ditunjukkan pada gambar 4.4.



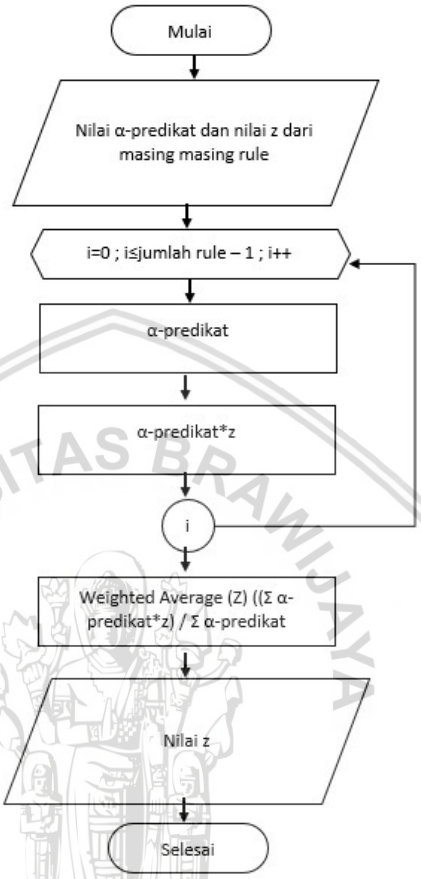
Gambar 4.4 Diagram alir proses inferensi.

Berdasarkan gambar 4.4 ditunjukkan alir proses inferensi dengan fungsi implikasi MIN yang digunakan dalam logika *fuzzy*. Pertama mendapatkan masukan variabel *Time Resoved*, *Score*, dan *Health Point*, kemudian melakukan perulangan for dengan batas 0 hingga jumlah banyaknya rule yang dipakai. Kemudian dilakukan fungsi implikasi MIN yang akan mencari nilai minimum yang dimiliki oleh setiap derajat keanggotaan masing-masing rule yang sudah diperoleh pada proses fuzzyfikasi. Selanjutnya yaitu mencari nilai z dari masing-masing rule yang ada.

4.2.3 Diagram Alir Defuzzyfikasi

Pada proses defuzzyfikasi dilakukan untuk mendapatkan nilai z dengan meode *Weighted Average*, hasil penjumlahan dari perkalian antara nilai α-predikat dan nilai z pada masing masing *rule* dibagi dengan penjumlahan nilai α-predikatdari masing-masing rule. Nilai z

digunakan untuk menentukan apakah level kesulitan dari maze akan naik, turun, atau tetap pada labirin selanjutnya.



Gambar 4.5 Diagram alir proses defuzzyfikasi

Dari gambar 4.5 ditunjukkan alir proses dari defuzzyfikasi. Yaitu proses untuk mendapatkan nilai Z yang diperoleh dengan menggunakan metode *Weighted Average*. Metode ini bermaksud untuk mendapatkan hasil penjumlahan dari α-predikat dikali z dari masing masing *rule* dan dibagi dengan penjumlahandari α-predikat dari masing masing rule. Berikut merupakan rumus yang digunakan dalam proses defuzzyfikasi

$$Z = \frac{\sum_{i=1}^{N} apredikat_i \, z_i}{\sum_{i=1}^{N} apredikat_i}$$

Keterangan :

Z = nilai output fuzzy

$Apredikat_i$ = bobot hasil implikasi ke-i

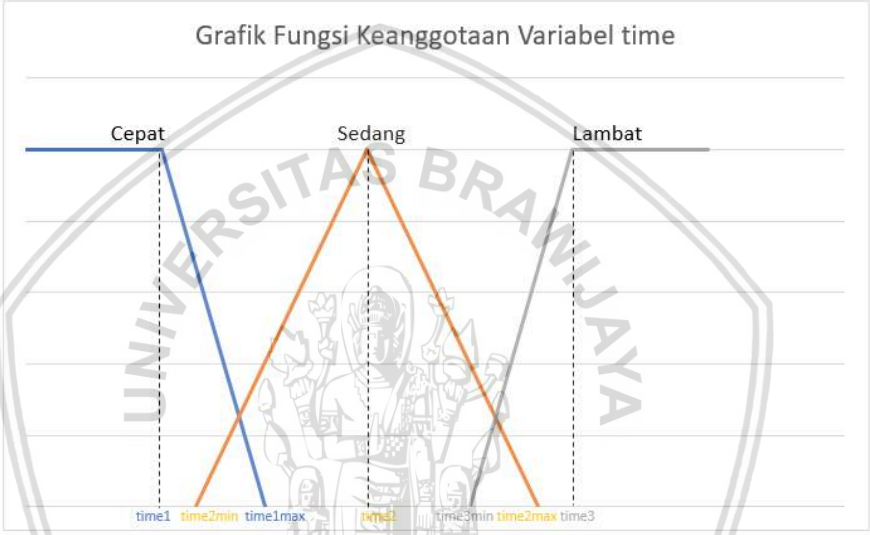
z_i = nilai output dari aturan ke-i

4.3 Fungsi Keanggotaan

Berisi tentang fungsi keanggotaan serta grafik *fuzzy* dari tiap variabel.

4.3.1 *Time Resolved*

Time Resolved diinisialisasi sebagai variabel "time" dan diklasifikasikan ke dalam tiga kategori yaitu lambat, sedang, dan cepat. Grafik fungsi keanggotaan variabel *time* ditunjukkan pada gambar 4.6.



Gambar 4.6 Grafik fungsi keanggotaan variabel time

Tabel himpunan fuzzy dari variabel *time* ditunjukkan pada tabel 4.1.

Tabel 4.1 Tabel himpunan fuzzy variabel time

No	Himpunan *Fuzzy Time Resolved*	*Range*
1	Cepat	x≤time1max
2	Sedang	time2min≤x≤time2max
3	Lambat	Time3min≤x

Nilai persamaan:

$$\mu_{Cepat}(X) = \begin{cases} 1, 0 \le x \le time1 \\ \frac{time1max - x}{time1max - time1}, time1 < x < time1max \\ 0, x \ge time1max \end{cases} \quad \text{.....(4-1)}$$

$$\mu_{Sedang}(X) = \begin{cases} 0, \ x \leq time2min \ || \ x \geq time2max \\ \frac{x - time2min}{time2 - time2min}, time2min < x \leq time2 \\ \frac{time2max - x}{time2max - ti}, time2 < x < time2max \end{cases} \text{.....(4-2)}$$

$$\mu_{Lambat}(X) = \begin{cases} 1, x \geq time3 \\ \frac{x - time2min}{time3 - time3min}, time3min < x < time3 \\ 0, x \leq time3min \end{cases} \text{.....(4-3)}$$

keterangan nilai variabel:

$time1 = 2 * pathCount * 1 / 4;$

$time2 = 2 * pathCount * 2 / 4;$

$time3 = 2 * pathCount * 3 / 4;$

$time1max = time1 + ((time2 - time1) * 3 / 4);$

$time2min = time1 + ((time2 - time1) * 1 / 4);$

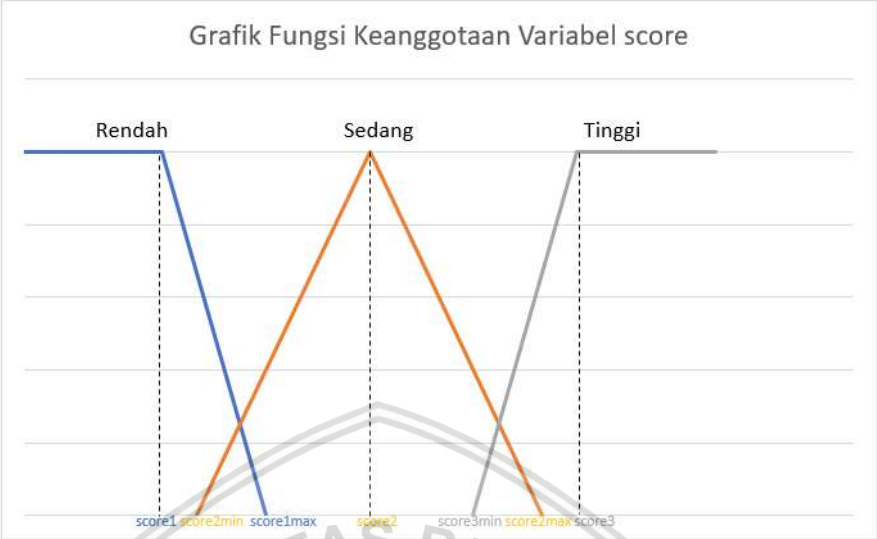
$time2max = time2 + ((time3 - time2) * 3 / 4);$

$time3min = time2 + ((time3 - time2) * 1 / 4);$

Variabel pathCount merupakan total dari langkah yang dapat dilewati hingga titik *finish*. variabel x merupakan nilai dari waktu penyelesaian *maze* hingga titik *finish*.

4.3.2 *Score*

Score diinisialisasikan sebagai variabel "score" dan diklasifikasikan ke dalam tiga kategori yaitu Sedikit, sedang, dan Banyak. Grafik fungsi keanggotaan variabel *score* ditunjukkan pada gambar 4.7.



Gambar 4.7 Grafik fungsi keanggotaan variabel score

Tabel himpunan fuzzy dari variabel *score* ditunjukkan pada tabel 4.2.

Tabel 4.2 Tabel Rule Base variabel score

No	Himpunan *Fuzzy score*	*Range*
1	Rendah	x≤score1max
2	Sedang	score2min≤x≤score2max
3	Tinggi	Score3min≤x

Nilai persamaan:

$$\mu_{Rendah}(X) = \begin{cases} 1, 0 \le x \le score1 \\ \frac{score1max-}{score1max-sco}, score1 < x < score1max \\ 0, x \ge score1max \end{cases} \text{.....(4-4)}$$

$$\mu_{Sedang}(X) = \begin{cases} 0, \ x \le score2min \ || \ x \ge score2max \\ \frac{x-score2mi}{score2-score2mi}, score2min \ < x \le score2 \\ \frac{score2max-x}{score2max-score}, score2 < x < score2max \end{cases} \text{.....(4-5)}$$

$$\mu_{Tinggi}(X) = \begin{cases} 1, x \ge score3 \\ \frac{x-score3min}{score3-score3}, score3min < x < score3 \\ 0, x \le score3min \end{cases} \text{.....(4-6)}$$

keterangan nilai variabel:

$$score1 = totCoin * 1;$$

$$score2 = totCoin * 2;$$

$$score3 = totCoin * 3;$$

$$score1max = score1 + ((score2 - score1) * 3 / 4);$$

$$score2min = score1 + ((score2 - score1) * 1 / 4);$$

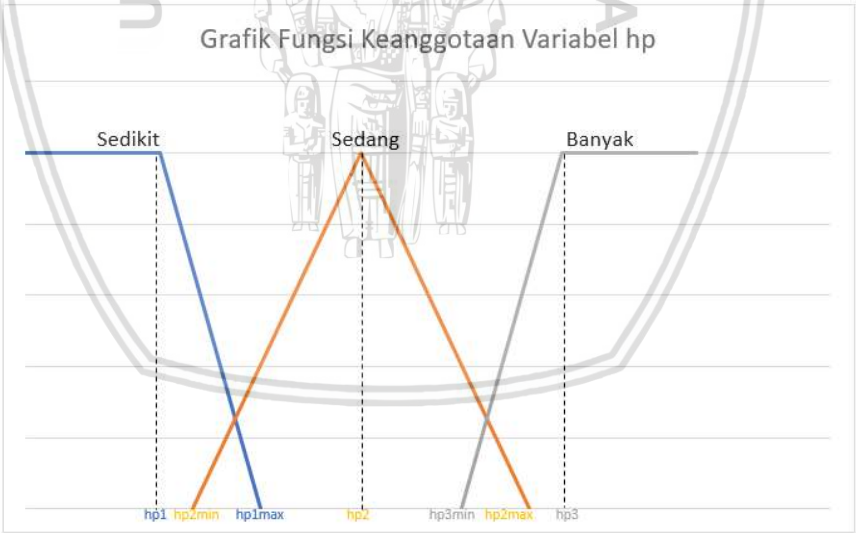
$$score2max = score2 + ((score3 - score2) * 3 / 4);$$

$$score3min = score2 + ((score3 - score2) * 1 / 4);$$

Variabel totCoin merupakan nilai dari total coin yang ada dalam permainan (hanya coin bernilai plus). Variabel x merupakan hasil perhitungan dari (coinPlus yang diperoleh*4) – (trapCoin yang diperoleh*2).

4.3.3 Health Point

Health Point diinisialisasi sebagai variabel "hp" dan diklasifikasikan ke dalam tiga kategori yaitu Sedikit, sedang, dan Banyak. Grafik fungsi keanggotaan variabel *hp* ditunjukkan pada gambar 4.8.



Gambar 4.8 Grafik fungsi keanggotaan variabel hp

Tabel himpunan fuzzy dari variabel *hp* ditunjukkan pada tabel 4.1.

Tabel 4.3 Tabel Rule Base variabel hp

No	Himpunan *Fuzzy score*	*Range*
1	Sedikit	x≤hp1max
2	Sedang	hp2min≤x≤hp2max
3	Banyak	hp3min≤x

Nilai persamaan:

$$\mu_{Sedikit}(X) = \begin{cases} 1, 0 \leq x \leq hp1 \\ \frac{hp1max - x}{hp1max - hp1}, hp1 < x < hp1max \\ 0, x \geq hp1max \end{cases} \text{.....(4-7)}$$

$$\mu_{Sedang}(X) = \begin{cases} 0, \ x \leq hp2min \ || \ x \geq hp2max \\ \frac{x - hp2min}{hp2 - hp2min}, hp2min \ < x \leq hp2 \\ \frac{hp2max - x}{hp2max - hp}, hp2 < x < hp2max \end{cases} \text{.....(4-8)}$$

$$\mu_{Banyak}(X) = \begin{cases} 1, x \geq hp3 \\ \frac{x - hp3min}{hp3 - hp3min}, hp3min < x < hp3 \\ 0, x \leq hp3min \end{cases} \text{.....(4-9)}$$

keterangan nilai variabel:

$hp1 = radius * 1 / 4;$

$hp2 = radius * 2 / 4;$

$hp3 = radius * 3 / 4;$

$hp1max = hp1 + ((hp2 - hp1) * 3 / 4);$

$hp2min = hp1 + ((hp2 - hp1) * 1 / 4);$

$hp2max = hp2 + ((hp3 - hp2) * 3 / 4);$

$hp3min = hp2 + ((hp3 - hp2) * 1 / 4);$

Variabel radius merupakan nilai *range* dari game *maze* atau merupakan nilai dari level *game* saat ini. Variabel x adalah nilai dari perhitungan variabel maksimal health point – total jebakan yang diinjak oleh pemain.

4.4 *Knowledge Base*

Berisi tentang aturan-aturan yang digunakan, aturan yang dipakai diperoleh dari pengetahuan penulis yang kemudian diterapkan dalam sebuah

rule base. *Rule base* ini yang nantinya digunakan sebagai aturan dasar dari logika *fuzzy*. Rule base dari fuzzy yang digunakan ditunjukkan pada table 4.4.

Tabel 4.4 Tabel Rule Base fuzzy

No	*Time Resolved*	*Score*	*Health Point*	*Next Maze Puzzle*
1	Cepat	Tinggi	Banyak	Naik Level
2	Cepat	Tinggi	Sedang	Naik Level
3	Cepat	Tinggi	Sedikit	Naik Level
4	Cepat	Sedang	Banyak	Naik Level
5	Cepat	Sedang	Sedang	Naik Level
6	Cepat	Sedang	Sedikit	Level Tetap
7	Cepat	Rendah	Banyak	Level Tetap
8	Cepat	Rendah	Sedang	Level Tetap
9	Cepat	Rendah	Sedikit	Level Tetap
10	Sedang	Tinggi	Banyak	Naik Level
11	Sedang	Tinggi	Sedang	Level Tetap
12	Sedang	Tinggi	Sedikit	Level Tetap
13	Sedang	Sedang	Banyak	Level Tetap
14	Sedang	Sedang	Sedang	Level Tetap
15	Sedang	Sedang	Sedikit	Level Tetap
16	Sedang	Rendah	Banyak	Level Tetap
17	Sedang	Rendah	Sedang	Level Tetap
18	Sedang	Rendah	Sedikit	Turun Level
19	Lambat	Tinggi	Banyak	Level Tetap
20	Lambat	Tinggi	Sedang	Level Tetap
21	Lambat	Tinggi	Sedikit	Level Tetap
22	Lambat	Sedang	Banyak	Level Tetap
23	Lambat	Sedang	Sedang	Turun Level

24	Lambat	Sedang	Sedikit	Turun Level
25	Lambat	Rendah	Banyak	Turun Level
26	Lambat	Rendah	Sedang	Turun Level
27	Lambat	Rendah	Sedikit	Turun Level

Dalam setiap aturan digunakan operator ***AND*** untuk mendapatkan perdikat dengan mencari nilai terkecil dari setiap aturan yang ada. Agar lebih mudah dalam memproses implikasi, komposisi, dan defuzzyfikasi, dapat diasumsikan dengan memberikan bobot pada setiap komposisi dari setiap variabel (Mahzar, 2013). Didapatkan hasil sebagai berikut :

- *Time Resolved* (time) diberikan bobot 3 kali lipat dari masing masing himpunan, dengan himpunan cepat berbobot 3, sedang berbobot 2, dan lambat berbobot 1.
- *Score* (score) diberikan bobot 2 kali lipat dari masing masing himpunan, dengan himpunan rendah berbobot 1, sedang berbobot 2, dan tinggi berbobot 3.
- *Health Point* (hp) diberikan bobot sesuai dengan masing masing himpunan, dengan himpunan sedikit berbobot 1, sedang berbobot 2, dan banyak berbobot 3.

Dari pernyataan diatas, untuk mendapatkan nilai minimum 0 dan maksimum 1 diperoleh rumus sebagai berikut :

$$z = ((3 * time) + (2 * score) + hp - 6) / 12 \quad \text{.....(4-10)}$$

Dari hasil persamaan 4-10 dituliskan mengenai ke 27 aturan tersebut beserta bobot dari himpunannya sebagai berikut:

1. (z1) Jika time cepat(3), score tinggi(3), dan hp banyak(3), maka next maze puzzle akan naik level (1)
2. (z2) Jika time cepat(3), score tinggi(3), dan hp sedang(2), maka next maze puzzle akan naik level (0,9167)
3. (z3) Jika time cepat(3), score tinggi(3), dan hp sedikit(1),), maka next maze puzzle akan naik level (0,8333)
4. (z4) Jika time cepat(3), score sedang(2), dan hp banyak(3), maka next maze puzzle akan naik level (0,8333)

5. (z5) Jika time cepat(3), score sedang(2), dan hp sedang(2), maka next maze puzzle akan naik level (0,75)
6. (z6) Jika time cepat(3), score sedang(2), dan hp sedikit(1), maka next maze puzzle akan level tetap (0,6667)
7. (z7) Jika time cepat(3), score rendah(1), dan hp banyak(3), maka next maze puzzle akan level tetap (0,6667)
8. (z8) Jika time cepat(3), score rendah(1), dan hp sedang(2), maka next maze puzzle akan level tetap (0,5833)
9. (z9) Jika time cepat(3), score rendah(1), dan hp sedikit(1), maka next maze puzzle akan level tetap (0,5)
10. (z10) Jika time sedang(2), score tinggi(3), dan hp banyak(3), maka next maze puzzle akan naik level (0,75)
11. (z11) Jika time sedang(2), score tinggi(3), dan hp sedang(2), maka next maze puzzle akan level tetap (0,6667)
12. (z12) Jika time sedang(2), score tinggi(3), dan hp sedikit(1), maka next maze puzzle akan level tetap (0,5833)
13. (z13) Jika time sedang(2), score sedang(2), dan hp banyak(3), maka next maze puzzle akan level tetap (0,5833)
14. (z14) Jika time sedang(2), score sedang(2), dan hp sedang(2), maka next maze puzzle akan level tetap (0,5)
15. (z15) Jika time sedang(2), score sedang(2), dan hp sedikit(1), maka next maze puzzle akan level tetap (0,4167)
16. (z16) Jika time sedang(2), score rendah(1), dan hp banyak(3), maka next maze puzzle akan level tetap (0,4167)
17. (z17) Jika time sedang(2), score rendah(1), dan hp sedang(2), maka next maze puzzle akan level tetap (0,3333)
18. (z18) Jika time sedang(2), score rendah(1), dan hp sedikit(1), maka next maze puzzle akan turun level (0,25)
19. (z19) Jika time lambat(1), score tinggi(3), dan hp banyak(3), maka next maze puzzle akan level tetap (0,5)
20. (z20) Jika time lambat(1), score tinggi(3), dan hp sedang(2), maka next maze puzzle akan level tetap (0,4167)

21. (z21) Jika time lambat(1), score tinggi(3), dan hp sedikit(1), maka next maze puzzle akan level tetap (0,3333)
22. (z22) Jika time lambat(1), score sedang(2), dan hp banyak(3), maka next maze puzzle akan level tetap (0,3333)
23. (z23) Jika time lambat(1), score sedang(2), dan hp sedang(2), maka next maze puzzle akan turun level (0,25)
24. (z24) Jika time lambat(1), score sedang(2), dan hp sedikit(1), maka next maze puzzle akan n turun aik level (01667)
25. (z25) Jika time lambat(1), score rendah(1), dan hp banyak(3), maka next maze puzzle akan turun level (0,1667)
26. (z26) Jika time lambat(1), score rendah(1), dan hp sedang(2), maka next maze puzzle akan turun level (0,8333)
27. (z27) Jika time lambat(1), score rendah(1), dan hp sedikit(1), maka next maze puzzle akan turun level (0)

Dari proses diatas dapat disimpulkan kriteria dari penentuan next level maze yang berlaku adalah sebagai berikut :

- Naik level jika z≥0,75
- Level tetap jika 0.75>z>0,25
- Turun level jika z≤0,25

4.5 Perhitungan Manual

Contoh kasus:

Dari hasil clear permainan labirin dengan generate ukuran 10x10 diperoleh data sebagai berikut:

Hasil data *generate*:

- pathCount : 199
- totCoin : 7
- radius : 10

Hasil *clear maze*:

- Time Resolved : 169
- Score : 24
- Health Point : 6

pathCount merupakan total dari langkah yang dapat dilewati hingga titik *finish*. totCoin merupakan nilai dari total coin yang ada dalam permainan (hanya coin bernilai plus). radius merupakan nilai *range* dari game *maze* atau merupakan nilai dari level *game* saat ini. Time Resolved merupakan nilai dari waktu yang dibutuhkan pemain dalam menyelesaikan 1 level permainan. Score merupakan nilai dari *score* yang diperoleh oleh pemain. Health Point merupakan nilai dari sisa *health point* yang masih dimiliki oleh pemain.

Berikut adalah langkah langkah yang dilakukan untuk menentukan apakah tingkat kesulitan labirin selanjutnya akan naik, tetap, atau turun dengan metode *fuzzy* dengan masukan yang telah tertera.

1. Menentukan fungsi keanggotaan tiap *variabel*

   a. *Time Resolved* (time)

   $$time1 = 2 * 199 * 1 / 4 = 99{,}5$$

   $$time2 = 2 * 199 * 2 / 4 = 199{,}0$$

   $$time3 = 2 * 199 * 3 / 4 = 298{,}5$$

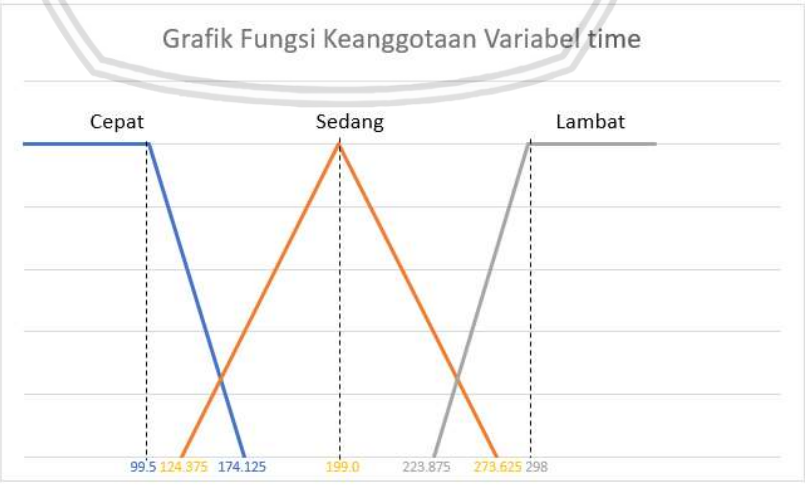
   $$time1max = 99.5 + ((199{,}0 - 99{,}5) * 3 / 4) = 174{,}125$$

   $$time2min = 99.5 + ((199{,}0 - 99{,}5) * 1 / 4) = 124{,}375$$

   $$time2max = 199.0 + ((298{,}5 - 199{,}0) * 3 / 4) = 273{,}625$$

   $$time3min = 199.0 + ((298{,}5 - 199{,}0) * 1 / 4) = 223{,}875$$

   Hasil perhitungan fungsi keanggotaan variabel *time* dapat digambarkan sebagai grafik yang ditunjukkan pada gambar 4.9.



Gambar 4.9 Grafik fungsi keanggotaan variabel time

b. *Score* (score)

$score1 = 7 * 1 = 7{,}0$

$score2 = 7 * 2 = 14{,}0$

$score3 = 7 * 3 = 21{,}0$

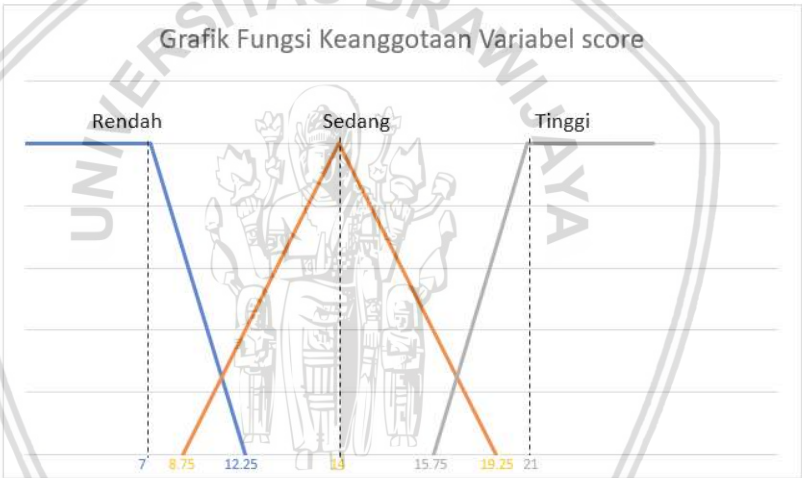
$score1max = 7 + ((14 - 7) * 3 / 4) = 12{,}25$

$score2min = 7 + ((14 - 7) * 1 / 4) = 8{,}75$

$score2max = 14 + ((21 - 14) * 3 / 4) = 19{,}25$

$score3min = 14 + ((21 - 14) * 1 / 4) = 15{,}75$

Hasil perhitungan fungsi keanggotaan variabel *score* dapat digambarkan sebagai grafik yang ditunjukkan pada gambar 4.10.



Gambar 4.10 Grafik fungsi keanggotaan variabel score

c. *Health Point* (hp)

$hp1 = 10 * 1 / 4 = 2{,}5$

$hp2 = 10 * 2 / 4 = 5$

$hp3 = 10 * 3 / 4 = 7{,}5$

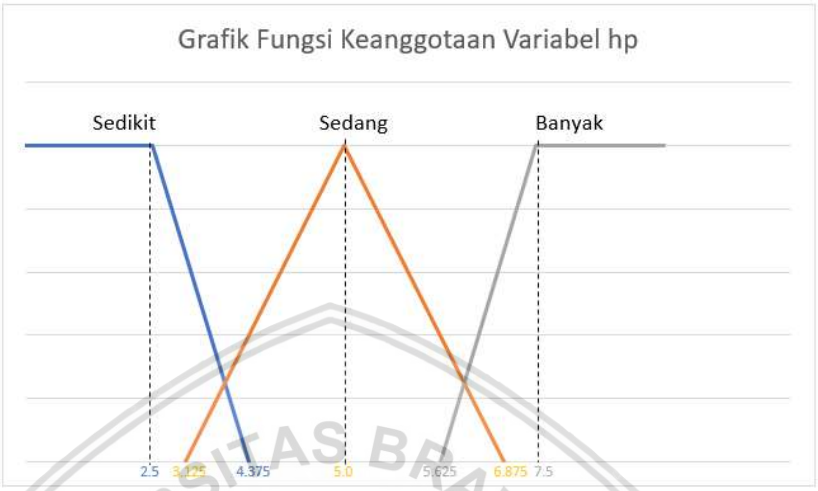
$hp1max = 2.5 + ((5 - 2.5) * 3 / 4) = 4{,}375$

$hp2min = 2.5 + ((5 - 2.5) * 1 / 4) = 3{,}125$

$hp2max = 5 + ((7.5 - 5) * 3 / 4) = 6{,}875$

$hp3min = 5 + ((7.5 - 5) * 1 / 4) = 5{,}625$

Hasil perhitungan fungsi keanggotaan variabel *hp* dapat digambarkan sebagai grafik yang ditunjukkan pada gambar 4.11.



Gambar 4.11 Grafik fungsi keanggotaan variabel hp

2. Menentukan derajat keanggotaan tiap variabel

   a. *Time Resolved* (time), terdiri dari tiga macam himpunan, yaitu cepat, sedang, dan lambat. Untuk himpunan cepat digunakan persamaan(4-1), untuk himpunan sedang digunakan persamaan (4-2), dan untuk himpunan lambat digunakan persamaan (4-3). Jika diketahui nilai dari variabel time adalah 169, maka derajat keanggotaan variabel time terhadap masing-masing himpunan *fuzzy* adalah:

$$\mu_{Cepat} = 99{,}5 < x < 174{,}125$$

$$= \frac{174{,}125 - 169}{174{,}125 - 99{,}5} = 0.0686$$

$$\mu_{Sedang} = 124{,}375 < x \leq 199$$

$$= \frac{169 - 124{,}375}{199 - 124{,}375} = 0{,}5979$$

$$\mu_{Lambat} = x \leq 223.875$$

$$= 0$$

   b. *Score* (score) terdiri dari tiga macam himpunan, yaitu rendah, sedang, dan tinggi. Untuk himpunan rendah digunakan persamaan(4-4), untuk himpunan sedang digunakan persamaan (4-5), dan untuk himpunan tinggi digunakan persamaan (4-6). Jika diketahui nilai dari variabel score adalah 18, maka derajat keanggotaan variabel score terhadap masing-masing himpunan *fuzzy* adalah:

$\mu_{Rendah} = 18 < 15{,}75$

$= 0$

$\mu_{Sedang} = 14 < 18 < 19{,}25$

$= \frac{19{,}25 - 18}{19{,}25 - 14} = 0{,}238$

$\mu_{Tinggi} = 15{,}75 < 18 < 21$

$= \frac{18 - 15{,}75}{21 - 15{,}75} = 0{,}4285$

c. *Health Point* (hp) terdiri dari tiga macam himpunan, yaitu sedikit, sedang, dan banyak. Untuk himpunan sedikit digunakan persamaan(4-7), untuk himpunan sedang digunakan persamaan (4-8), dan untuk himpunan banyak digunakan persamaan (4-9). Jika diketahui nilai dari variabel hp adalah 6, maka derajat keanggotaan variabel hp terhadap masing-masing himpunan *fuzzy* adalah:

$\mu_{Sedikit} = 6 > 4{,}375$

$= 0$

$\mu_{Sedang} = 5 < 6 < 6{,}875$

$= \frac{6{,}875 - 6}{6{,}875 - 5} = 0{,}4666$

$\mu_{Banyak} = 5{,}625 < 6 < 7{,}5$

$= \frac{6 - 5{,}625}{7{,}5 - 5{,}625} = 0{,}2$

3. Mencari nilai α-predikat dari setiap aturan dengan menggunakan fungsi implikasi MIN sehingga didapatkan nilai maksimum dari setiap kombinasi dari aturan yang ada. Kemudian langkah selanjutnya yaitu mendapatkan nilai z atau hasil inferensi dari setiap aturan. Dimisalkan mengambil 2 aturan.

   z13 : *IF* time sedang *AND* score sedang *AND* hp banyak *THEN* level tetap.

   α-predikat13 = min(0,5979 , 0.238 , 0.2) = 0,2

   berdasarkan pengetahuan dari *knowledge base* nilai untuk z13 = 0,5833

   z19 : *IF* time lambat *AND* score rendah *AND* hp banyak *THEN* level tetap.

   α-predikat19 = min(0 , 0, 0,2) = 0

berdasarkan pengetahuan dari *knowledge base* nilai untuk z19 = 0,5

4. Proses defuzzyfikasi dilakukan dengan menggunakan metode Weighted Average untuk mencari nilai dari Z. Hasil dari proses defuzzyfikasi digunakan sebagai acuan dalam penarikan kesimpulan apakah pada *next level maze* tersebut, maze akan naik level, level tetap, atau turun level.

$$Z = \frac{\sum_{i=1}^{N} apredikat_i \, z_i}{\sum_{i=1}^{N} apredikat_i}$$

$$Z = \frac{apredikat1 * z1 + apredikat\ *z2 + apredikat3 * z3 + \ldots + apredikat27 * z2}{apredikat1 + apredikat2 + apredikat3 + \ldots\ldots + apredikat2}$$

$$Z = \frac{0,2 * 0,5833 + 0 * 0,5}{0,2 + 0}$$

$$Z = \frac{0,1167 +}{0,2} = 0,5833$$

Keterangan :

Z = nilai output fuzzy

$Apredikat_i$ = bobot hasil implikasi ke-i

z_i = nilai output dari aturan ke-i

Dari hasil defuzzyfikasi didapatkan nilai 0,5833, maka nilai untuk *next level maze* yaitu adalah level tetap.

BAB 5 IMPLEMENTASI

5.1 Lingkungan Implementasi

5.1.1 Spesifikasi Perangkat Keras

Berikut merupakan spesifikasi perangkat keras yang digunakan selama proses pengembangan system.

- *Processor* : Intel® Core™ i7-6700HQ CPU @ 2.60GHz ~ 3,1GHz
- *Memory* : 12288MB RAM
- GPU : NVIDIA GeForce GTX 950M
- VRAM : 2019MB

5.1.2 Spesifikasi Perangkat Lunak

Berikut merupakan spesifikasi perangkat keras yang digunakan selama proses pengembangan system.

- *Operational System* : Windows 10 Home Single Language 64-bit.
- *Programming Language* : C#
- *Programming App* : Unity 2017.1.0f3 (64 bit)
- *Text Editor* : Visual Studio 2017

5.2 Implementasi Kode Program

Implementasi algoritme *fuzzy* dari *Dynamic Difficulty Scaling* pada game labirin ditunjukkan pada tabel 5.1.

Tabel 5.1 Tabel Rule Base

No	Source Code
1	`using System;`
2	`using System.Collections;`
3	`using System.Collections.Generic;`
4	`using UnityEngine;`
5	`using UnityEngine.UI;`
6	`#if UNITY_EDITOR`
7	`using UnityEditor;`
8	`#endif`
9	`using System.IO;`
10	`using UnityEngine.SceneManagement;`
11	
12	`public class Fuzzy : MonoBehaviour{`
13	`    public int pathCounttxt=0, totCointxt=0,`
14	`radiustxt=0;`

```
15      private float hp1, hp1max, hp2, hp2min, hp2max,
16  hp3, hp3min;
17      private float score1, score1max, score2, score2min,
18  score2max, score3, score3min;
19      private float time1, time1max, time2, time2min,
20  time2max, time3, time3min;
21      private float[] rule = new float[27];
22      private float[] predicat = new float[27];
23      private float[] ztime = new float[3];
24      private float[] zscore = new float[3];
25      private float[] zhp = new float[3];
26      private float tempTop = 0, tempBottom = 0, Z;
27      // Use this for initialization
28      void Start () {
29          countLimit(pathCounttxt, totCointxt,
30  radiustxt);
31        }
32
33      public void fuzzyStart(float hp, float score, float
34  time){
35          countHp(hp);
36          countScore(score);
37          countTime(time);
38          int x = 0;
39          for (float i = 3; i > 0; i--)
40          {
41              for (float j = 3; j > 0; j--)
42              {
43                  for (float k = 3; k > 0; k--)
44                  {
45                      rule[x] = (3 * i + 2 * j + k - 6) /
46  12;
47                      x++;
48                  }
49              }
50          }
51          x = 0;
52          for (int i = 0; i < 3; i++)
53          {
54              for (int j = 0; j < 3; j++)
55              {
56                  for (int k = 0; k < 3; k++)
57                  {
58                      if (ztime[i] <= zscore[j] &&
59  ztime[i] <= zhp[k])
60                      {
61                          predicat[x] = ztime[i];
62                          x++;
63                      }
64                      else if (zscore[j] <= ztime[i] &&
65  zscore[j] <= zhp[k])
66                      {
67                          predicat[x] = zscore[j];
68                          x++;
69                      }
70                      else
71                      {
```

```
72                          predicat[x] = zhp[k];
73                          x++;
74                      }
75                  }
76              }
77          }
78          for (int i = 0; i < 27; i++)
79          {
80              tempTop += predicat[i] * rule[i];
81              tempBottom += predicat[i];
82          }
83          Z = tempTop / tempBottom;
84          countIndex(Z);
85      }
86
87      public Fuzzy()
88      {
89
90      }
91
92      void countLimit(int pathCount, int totCoin, int
93  radius)
94      {
95          hp1 = radius * 1 / 4;
96          hp2 = radius * 2 / 4;
97          hp3 = radius * 3 / 4;
98          hp1max = hp1 + ((hp2 - hp1) * 3 / 4);
99          hp2min = hp1 + ((hp2 - hp1) * 1 / 4);
100         hp2max = hp2 + ((hp3 - hp2) * 3 / 4);
101         hp3min = hp2 + ((hp3 - hp2) * 1 / 4);
102         score1 = totCoin * 1;
103         score2 = totCoin * 2;
104         score3 = totCoin * 3;
105         score1max = score1 + ((score2 - score1) * 3 /
106 4);
107         score2min = score1 + ((score2 - score1) * 1 /
108 4);
109         score2max = score2 + ((score3 - score2) * 3 /
110 4);
111         score3min = score2 + ((score3 - score2) * 1 /
112 4);
113         time1 = 2 * pathCount * 1 / 4;
114         time2 = 2 * pathCount * 2 / 4;
115         time3 = 2 * pathCount * 3 / 4;
116         time1max = time1 + ((time2 - time1) * 3 / 4);
117         time2min = time1 + ((time2 - time1) * 1 / 4);
118         time2max = time2 + ((time3 - time2) * 3 / 4);
119         time3min = time2 + ((time3 - time2) * 1 / 4);
120     }
121
122     void countHp(float hp)
123     {
124         if (hp < hp1max && hp >= hp1)
125         {
126             zhp[2] = (hp1max - hp) / (hp1max - hp1);
127         }
128         else if (hp < hp1)
```

```
129         {
130             zhp[2] = 1;
131         }
132         else
133         {
134             zhp[2] = 0;
135         }
136         if (hp > hp2min && hp <= hp2)
137         {
138             zhp[1] = (hp - hp2min) / (hp2 - hp2min);
139         }
140         else if (hp >= hp2 && hp < hp2max)
141         {
142             zhp[1] = (hp2max - hp) / (hp2max - hp2);
143         }
144         else
145         {
146             zhp[1] = 0;
147         }
148         if (hp > hp3min && hp <= hp3)
149         {
150             zhp[0] = (hp - hp3min) / (hp3 - hp3min);
151         }
152         else if (hp > hp3)
153         {
154             zhp[0] = 1;
155         }
156         else
157         {
158             zhp[0] = 0;
159         }
160     }
161
162
163     void countScore(float score)
164     {
165         if (score < score1max && score >= score1)
166         {
167             zscore[2] = (score1max - score) /
168 (score1max - score1);
169         }
170         else if (score < score1)
171         {
172             zscore[2] = 1;
173         }
174         else
175         {
176             zscore[2] = 0;
177         }
178         if (score > score2min && score <= score2)
179         {
180             zscore[1] = (score - score2min) / (score2 -
181 score2min);
182         }
183         else if (score >= score2 && score < score2max)
184         {
185
```

```
186             zscore[1] = (score2max - score) /
187 (score2max - score2);
188         }
189         else
190         {
191             zscore[1] = 0;
192         }
193         if (score > score3min && score <= score3)
194         {
195             zscore[0] = (score - score3min) / (score3 -
196 score3min);
197         }
198         else if (score > score3)
199         {
200             zscore[0] = 1;
201         }
202         else
203         {
204             zscore[0] = 0;
205         }
206     }
207
208     void countTime(float time)
209     {
210         if (time < time1max && time >= time1)
211         {
212             ztime[0] = (time1max - time) / (time1max -
213 time1);
214         }
215         else if (time < time1)
216         {
217             ztime[0] = 1;
218         }
219         else
220         {
221             ztime[0] = 0;
222         }
223         if (time > time2min && time <= time2)
224         {
225             ztime[1] = (time - time2min) / (time2 -
226 time2min);
227         }
228         else if (time >= time2 && time < time2max)
229         {
230             ztime[1] = (time2max - time) / (time2max -
231 time2);
232         }
233         else
234         {
235             ztime[1] = 0;
236         }
237         if (time > time3min && time <= time3)
238         {
239             ztime[2] = (time - time3min) / (time3 -
240 time3min);
241         }
242         else if (time > time3)
```

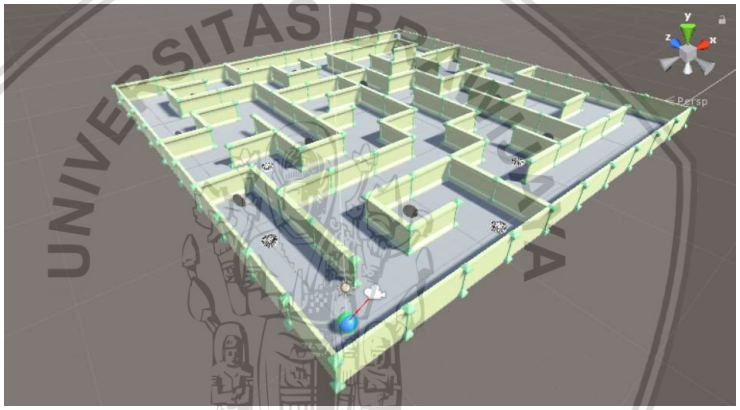
```
243         {
244             ztime[2] = 1;
245         }
246         else
247         {
248             ztime[2] = 0;
249         }
250     }
251 
252     void countIndex(float Z)
253     {
254 
255         string path =
256 "Assets/MazeGenerator/Scripts/Level.txt";
257         //baca text
258         StreamReader reader = new StreamReader(path);
259         int levelTxt = int.Parse(reader.ReadToEnd());
260         reader.Close();
261         //tulis text
262         float count=Z;
263         if (count <= 0.25 && levelTxt>5)
264         {
265             Debug.Log("turun level");
266             levelTxt -= 1;
267             StreamWriter writer = new
268 StreamWriter(path, false);
269             writer.Write(levelTxt+"");
270             writer.Close();
271             SceneManager.LoadScene("fuzzy");
272         }
273         else if (count > 0.25 && count < 0.75)
274         {
275             Debug.Log("level tetap");
276             StreamWriter writer = new
277 StreamWriter(path, false);
278             writer.Write(levelTxt + "");
279             writer.Close();
280             SceneManager.LoadScene("fuzzy");
281         }
282         else if (count >= 0.75)
283         {
284             Debug.Log("naik level");
285             levelTxt += 1;
286             StreamWriter writer = new
287 StreamWriter(path, false);
288             writer.Write(levelTxt+"");
289             writer.Close();
290             SceneManager.LoadScene("fuzzy");
291         }
292         else
293         {
294             SceneManager.LoadScene("fuzzy");
295         }
296     }
297 
298 }
```

Berikut adalah penjelasan dari script diatas :

- pathCounttxt, totCointxt, radiustxt merupakan nilai *generate* yang diperoleh dari *maze generate*.
- hp1, hp1max, hp2, hp2min, hp2max, hp3, hp3min merupakan variabel untuk menyimpan fungsi keanggotaan dari variabel hp.
- score1, score1max, score2, score2min, score2max, score3, score3min merupakan variabel untuk menyimpan fungsi keanggotaan dari variabel score.
- time1, time1max, time2, time2min, time2max, time3, time3min merupakan variabel untuk menyimpan fungsi keanggotaan dari variabel time.
- float[] rule merupakan variabel yang digunakan untuk menyimpan aturan aturan yang dipakai dalam pembuatan logika *fuzzy*.
- float[] predicat merupakan variabel yang digunakan untuk menyimpan nilai dari α-predikat.
- float[] ztime merupakan variabel yang digunakan untuk menyimpan nilai dari keanggotaan setiap variabel time.
- float[] zscore merupakan variabel yang digunakan untuk menyimpan nilai dari keanggotaan setiap variabel score.
- float[] zhp merupakan variabel yang digunakan untuk menyimpan nilai dari keanggotaan setiap variabel hp.
- tempTop, tempBottom, Z merupakan variabel yang digunakan dalam proses implikasi *fuzzy*.
- Baris 33-85 (FuzzyStart) merupakan method yang digunakan untuk proses perhitungan fuzzy. Baris 35-37 digunakan untuk membuat aturan dari logika *fuzzy*. Baris 39-77 digunakan untuk menghitung nilai dari α-predikat. Baris 78-84 digunakan untuk mencari nilai Z.
- Baris 92-120 (countLimit) digunakan untuk membuat fungsi keanggotaan dari setiap variabel yaitu time, score, dan hp.
- Baris 122-160 (countHp) digunakan untuk menghitung nilai derajat keanggotaan dari variabel hp.
- Baris 163-206 (countScore) digunakan untuk menghitung nilai derajat keanggotaan dari variabel score.

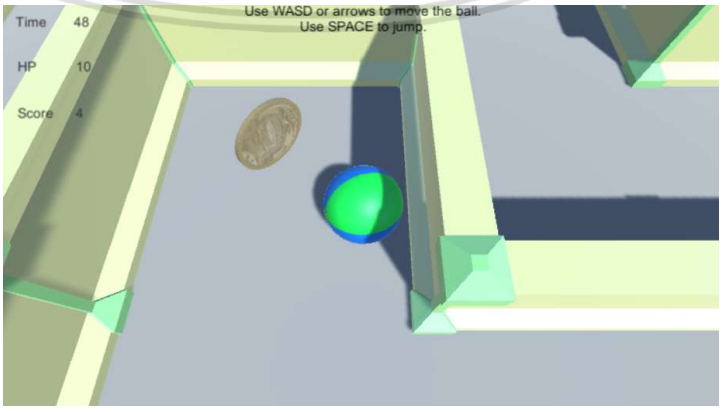
- Baris 208-250 (countTime) digunakan untuk menghitung nilai derajat keanggotaan dari variabel time.
- Baris 252-296 (countIndex) digunakan untuk proses defuzzyfikasi serta sasil dari aturan tersebut merupakan penentuan dari *next level maze* apakah naik level, level tetap, atau turun level. Untuk merefresh *level* digunakan StreamReader untuk membaca serta mengubah isi dari file txt yang berisi tentang level permainan saat ini dengan inputan system naik (plus), tetap (sama dengan), atau turun (minus). Kemudian system akan mereload *scene game maze* dengan masukan level terbaru dari hasil *fuzzy* yang telah didapat.

Adapun tampilan yang menunjukkan hasil dari implementasi metode fuzzy pada *game* labirin yang ditunjukkan pada gambar 5.1 hingga 5.5.



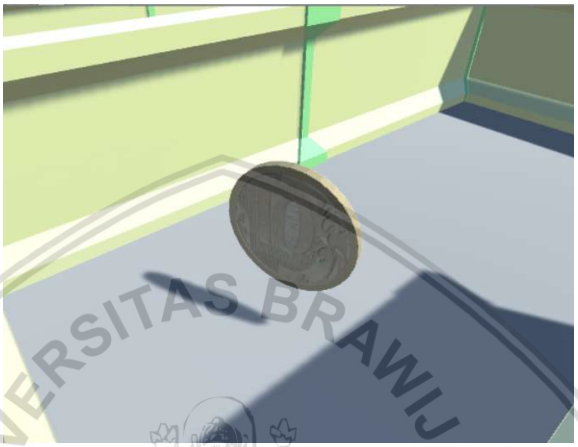
Gambar 5.1 Gambar game labirin dari perspektif unity scene

Gambar 5.1 menunjukkan pandangan *game* labirin dari perspektif *scene* dari Unity untuk melihat seluruh ruangan labirin serta objek didalamnya telah berhasil tergenerate dan terbentuk.



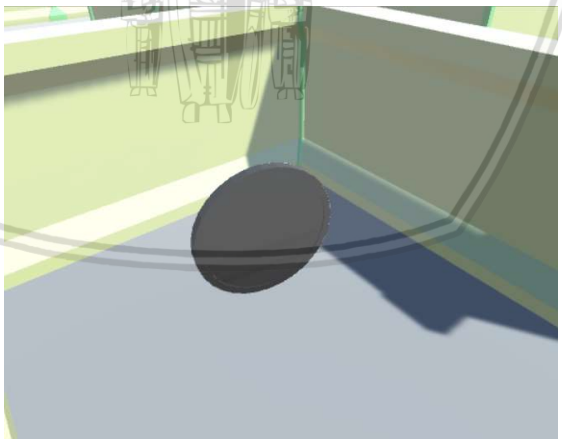
Gambar 5.2 Gambar game labirin dari perspektif pemain

Gambar 5.2 menunjukkan pandangan *game* labirin dari sudut pandang pemain yang bermain *game* labirin. Terdapat 4 macam text, yaitu *Time* untuk menunjukkan lamanya permainan berlangsung dalam detik, HP yang menunjukkan *health point* yang dimiliki saat ini, *Score* yang menunjukkan peroleh *score* yang diperoleh saat ini, serta papan pemberitahuan yang ada di tengah untuk memberikan panduan *basic* kepada pemain.



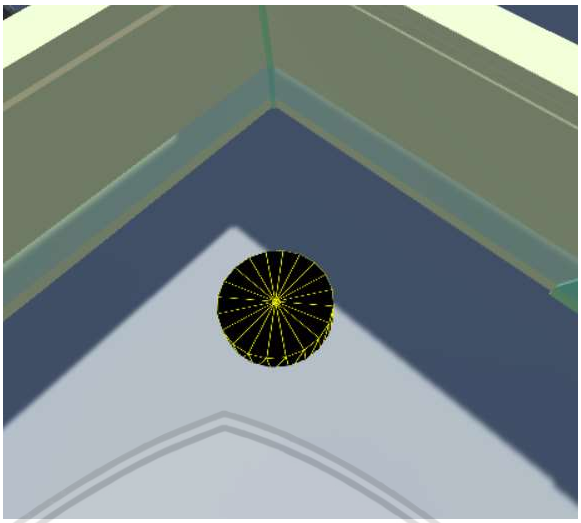
Gambar 5.3 Gambar coin

Gambar 5.3 menunjukkan gambar coin yang dipakai dalam menambah nilai plus pada score saat diambil, setiap coin yang didapat bernilai *plus* 4 *point*.



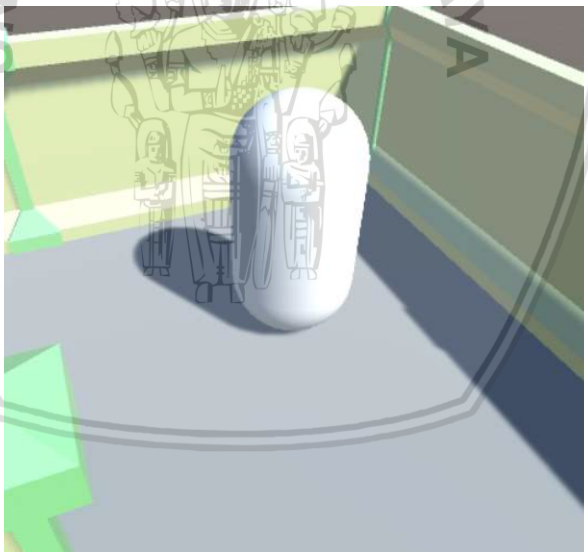
Gambar 5.4 Gambar trap coin

Gambar 5.4 menunjukkan gambar *trap coin* yang dipakai dalam mengurangi score saat tertabrak, setiap *trap coin* yang didapat bernilai *minus* 2 *point*. Berbeda dari coin yang bernilai *plus,* coin bernilai *minus* berwarna hitam kelam dan tidak memiliki shadow.



Gambar 5.5 Gambar jebakan

Gambar 5.5 menunjukkan gambar jebakan yang terdapat pada beberapa jalan yang harus dihindari oleh pemain, jika terkena jebakan maka *health point* pemain akan berkurang.



Gambar 5.6 Gambar goal

Gambar 5.6 menunjukkan gambar *goal* atau *finish* dari *game* labirin ini. Setelah pemain mencapai titik ini, maka proses fuzzy akan dilakukan untuk menentukan tingkat keberhasilan pemain serta mengenerate labirin yang akan muncul selanjutnya.

BAB 6 PENGUJIAN DAN ANALISIS

6.1 Lingkungan Pengujian

Berikut merupakan spesifikasi perangkat yang digunakan selama proses pengembangan system.

- *System Model* : ASUS ROG GL55VX
- *Processor* : Intel® Core™ i7-6700HQ CPU @ 2.60GHz ~ 3.1GHz
- *Memory* : 12288MB RAM
- GPU : NVIDIA GeForce GTX 950M
- VRAM : 2019MB
- DirectX : DirectX 12
- *Operating System* : Windows 10 Home Single Language 64-bit

6.2 Pengujian Kinerja

Pengujian Kinerja yang dipakai meliputi pengujian validasi behavior dan performa dari logika *fuzzy*. Pengujian validasi behavior dilakukan dengan cara menyelesaikan level dari map yang ditentukan serta melihat hasil keluaran *next level maze* yang muncul setelah *level maze* tersebut selesai. Sedangkan pengujian performa dilakukan dengan cara menguji besarnya *frames per second* (fps) yang dihasilkan selama permainan.

Pengujian *behavior* dilakukan dengan mengujikan *game* labirin ini sebanyak 10 kali terhadap 5 orang random untuk mendapatkan hasil grafik dari permainan pemain untuk melihat apakah balance level tersebut sudah cocok untuk diterapkan pada *game* labirin ini. Adapun hasil dari pengujian *behavior* yang ditunjukkan pada tabel 6.1.

Tabel 6.1 Tabel Hasil Pengujian Behavior

Player No.	Map	Level	Range	Coin	Time	Score	Hp	Output fuzzy	Output Game
1	1	10	10x10	10	318	18	4	0,225	Turun level
	2	9	9x9	9	221	26	6	0,4901	Level tetap
	3	9	9x9	8	157	20	5	0,6667	Level tetap

	4	9	9x9	9	163	26	7	0,75	Naik level
	5	10	10x10	9	285	22	8	0,3981	Level tetap
	6	10	10x10	10	303	30	6	0,4416	Level tetap
	7	10	10x10	10	73	2	7	0,6667	Level tetap
	8	10	10x10	8	259	28	8	0,5734	Level tetap
	9	10	10x10	9	169	24	7	0,7554	Level naik
	10	11	11x11	10	240	24	6	0,55	Level tetap
2	1	10	10x10	10	237	26	7	0,684	Level tetap
	2	10	10x10	9	201	30	9	0,75	Naik level
	3	11	11x11	8	193	22	9	0,8241	Naik level
	4	12	12x12	11	230	28	9	0,7688	Naik level
	5	13	13x13	13	248	30	8	0,7033	Level tetap
	6	13	13x13	15	265	26	10	0,6638	Level tetap
	7	13	13x13	12	301	32	11	0,7222	Level tetap
	8	13	13x13	13	273	36	11	0,8149	Naik level
	9	14	14x14	12	512	26	6	0,3125	Level tetap
	10	14	14x14	12	493	30	7	0,4526	Level tetap
3	1	10	10x10	9	201	26	7	0,75	Naik level
	2	11	11x11	11	265	32	8	0,75	Naik level
	3	12	12x12	10	347	28	8	0,6348	Level tetap

	4	12	12x12	11	352	30	9	0,6326	Level tetap
	5	12	12x12	11	312	26	7	0,57	Level tetap
	6	12	12x12	11	320	34	9	0,75	Naik level
	7	13	13x13	12	470	24	6	0,25	Turun level
	8	12	12x12	10	317	30	7	0,6805	Level tetap
	9	12	12x12	11	291	32	10	0,7499	Level tetap
	10	12	12x12	12	339	38	10	0,6938	Level tetap
4	1	10	10x10	10	259	20	6	0,3737	Level tetap
	2	10	10x10	9	263	24	5	0,4462	Level tetap
	3	10	10x10	9	231	24	7	0,6541	Level tetap
	4	10	10x10	10	222	26	6	0,6354	Level tetap
	5	10	10x10	8	199	22	7	0,75	Naik level
	6	11	11x11	11	287	28	7	0,5411	Level tetap
	7	11	11x11	10	319	30	7	0,5364	Level tetap
	8	11	11x11	11	295	28	8	0,6509	Level tetap
	9	11	11x11	10	270	30	8	0,75	Naik level
	10	12	12x12	11	354	24	9	0,4082	Level tetap
5	1	10	10x10	9	165	22	6	0,6885	Level tetap
	2	10	10x10	10	147	30	8	0,8863	Naik level
	3	11	11x11	10	175	32	9	0,8988	Naik level

	4	12	12x12	12	294	36	12	0,75	Naik level
	5	13	13x13	11	268	34	11	0,8297	Naik level
	6	14	14x14	13	340	40	11	0,755	Naik level
	7	15	15x15	15	421	40	12	0,7222	Level tetap
	8	15	15x15	15	380	44	13	0,7786	Naik level
	9	16	16x16	14	501	36	13	0,6904	Level tetap
	10	16	16x16	16	529	40	14	0,6666	Level tetap

Berdasarkan table 6.1 ditunjukkan table hasil pengujian yang dilakukan terhadap 5 pemain acak. Dalam table diberikan pengetahuan tentang range/radius dari *game* labirin serta jumlah *coin* hasil dari generate labirin, pemain diharuskan untuk menyelesaikan permainan dan mendapatkan nilai dari *Time Resolved, Score,* dan *Health Point.* Kemudian pada kolom terakhir ditunjukkan hasil dari perhitungan dengan metode *fuzzy* dalam bentuk output.

Pengujian performa dilakukan dengan memainkan *game* labirin pada 5 *map* yang berbeda serta dengan 5 kali percobaan pada setiap *map* dengan nilai parameter yang cukup besar. Adapun hasil dari pengujian performa yang ditunjukkan pada table 6.2.

Tabel 6.2 Tabel Hasil Pengujian Performa

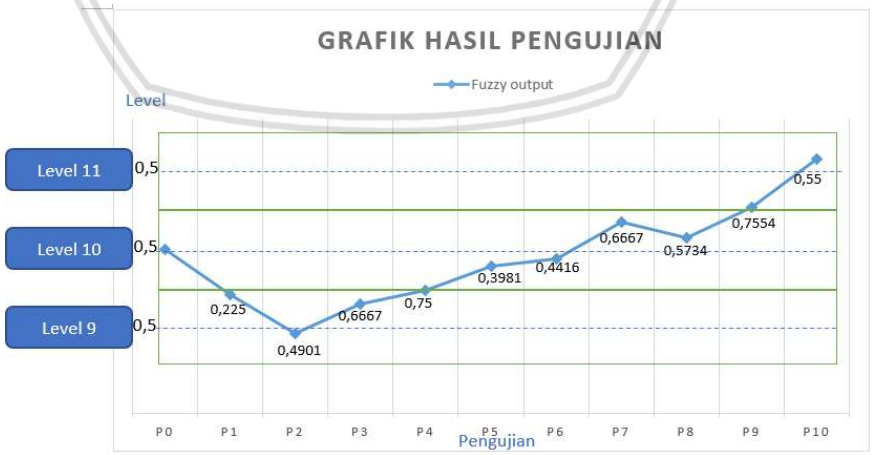
No	*Frames per second* (fps)				
	5x5	*10x10*	*20x20*	*30x30*	*50x50*
1	97,02	95,12	94,25	95,93	96,24
2	95,66	99,79	99,43	93,48	94,85
3	102,26	101,91	97,12	89,43	91,01
4	96,91	93,25	92,99	97,23	98,23
5	107,22	95,76	94,66	95,33	87,94

Rata-rata	99,814	97,166	95,69	94,28	93,65

Dari table 6.2 ditunjukkan bahwa hasil pengujian performa dari *game* labirin setelah ditambahkan implementasi *fuzzy* untuk *dynamic difficulty scaling* memiliki nilai performa terbaik pada saat pengujian ke 5 dengan labirin berukuran 5x5 dengan nilai 107.22, serta performa terburuk berada pada saat pengujian ke 4 dengan labirin berukuran 50x50 dengan nilai 87.65. Berdasarkan pengujian performa fps yang dilakukan, ditunjukkan hasil performa dari *game* hanya terpengaruh sedikit oleh ukuran dari labirin yang di*generate.*

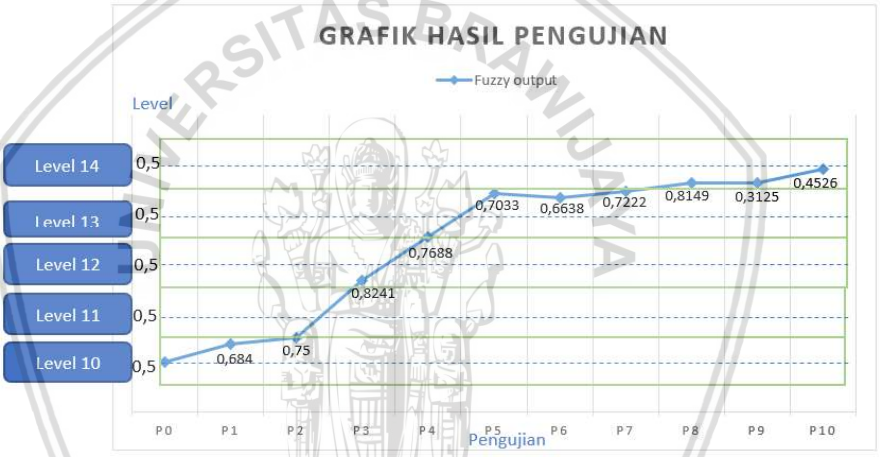
6.3 Hasil Analisis

Dari hasil pengujian behavior yang dilakukan, hasil analisis dari setiap percobaan ditunjukkan pada gambar 6.1-6.5. Nilai 0,5 merupakan nilai batas normal dari setiap level yang ditandai dengan garis putus-putus. Garis batas hijau menandakan batas kemampuan pemain yang memiliki nilai 0,25 dan 0,75, jika pemain berada diatas 0,75 maka pemain dinyatakan telah berada di atas rata-rata kemampuan bermain dan diarahkan untuk menuju ke level selanjutnya dengan hasil naik level, sedangkan untuk pemain yang berada dibawah 0,25 maka pemain dinyatakan berada di bawah rata-rata yang mengharuskan pemain turun level pada permainan selanjutnya. P0 merupakan level awal yang diberikan kepada pemain untuk memulai permainan, sedangkan p1 hingga p10 merupakan pengujian yang dilakukan sebanyak 10 kali.



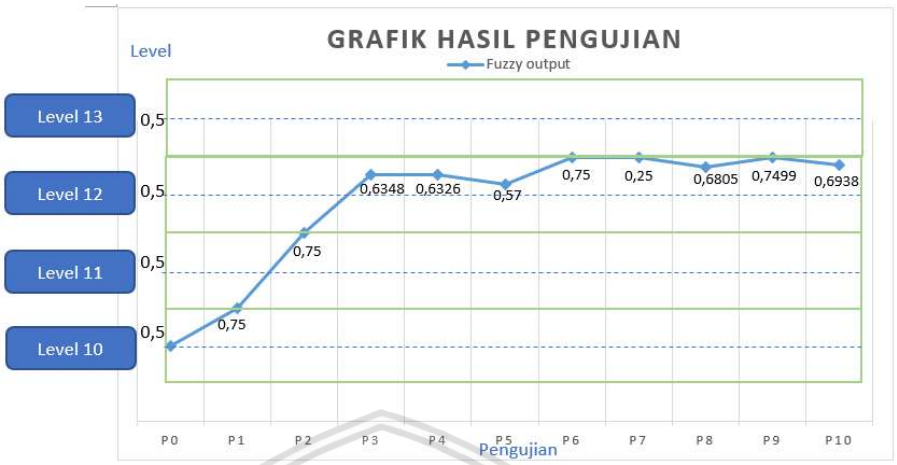
Gambar 6.1 Grafik hasil pengujian *player* 1

Berdasarkan Gambar 6.1 ditunjukkan hasil dari pengujian yang dilakukan oleh pemain pertama. Titik pada gambar tersebut menjelaskan tentang besarnya nilai *fuzzy* yang didapatkan oleh pemain saat berada pada level tersebut. Diketahui bahwa titik awal pemain diberikan level 10 karena dengan anggapan pemain belum mengetahui kemampuan bermain dalam *game* labirin ini. Terlihat pada table 6.1, pada pengujian ke 1, pemain mendapatkan nilai output yang berada dibawah rata rata yaitu 0,225, mengakibatkan dirinya harus turun satu level dibawah level saat ini. Seiring berjalannya permainan, pemain mulai mengerti dan belajar tentang strategi permainan yang membuatnya dapat naik level pada pengujian ke 4. Pada pengujian ke 5 hingga ke 8, pemain masih berusaha untuk mengingkatkan strategi bermainnya hingga pada pengujian ke 9 pemain berhasil naik level ke level selanjutnya.



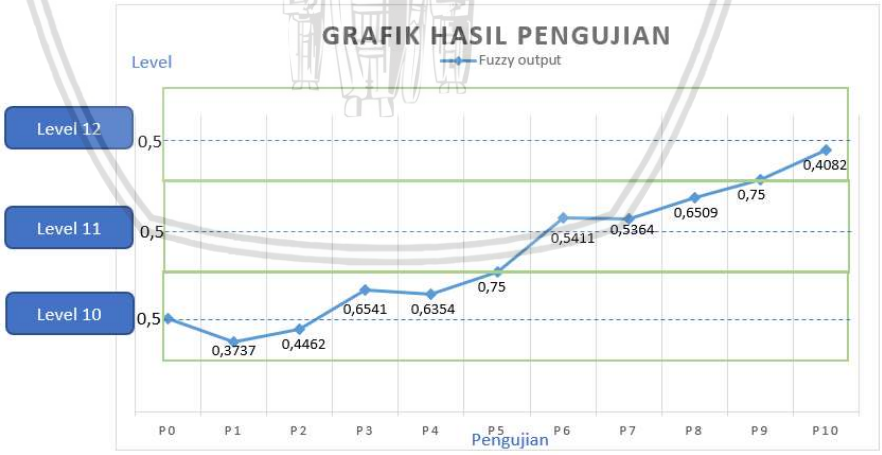
Gambar 6.2 Grafik hasil pengujian *player* 2

Berdasarkan Gambar 6.2 ditunjukkan hasil dari pengujian yang dilakukan oleh pemain kedua. Titik pada gambar tersebut menjelaskan tentang besarnya nilai *fuzzy* yang didapatkan oleh pemain saat berada pada level tersebut. Diketahui bahwa titik awal pemain diberikan level 10 karena dengan anggapan pemain belum mengetahui kemampuan bermain dalam *game* labirin ini. Terlihat pada table 6.1, pada pengujian ke 1, pemain masih berada di titik rata rata yaitu dengan nilai 0,684. Nilai awal tersebut terbilang cukup bagus hingga ternyata terlihat pada pengujian ke 2 – 4 ternyata pemain selalu berada pada titik rata rata dan membuatnya naik ke tingkat selanjutnya, langkahnya sempat terhenti saat berusaha naik dari level 13 ke level 14.



Gambar 6.3 Grafik hasil pengujian player 3

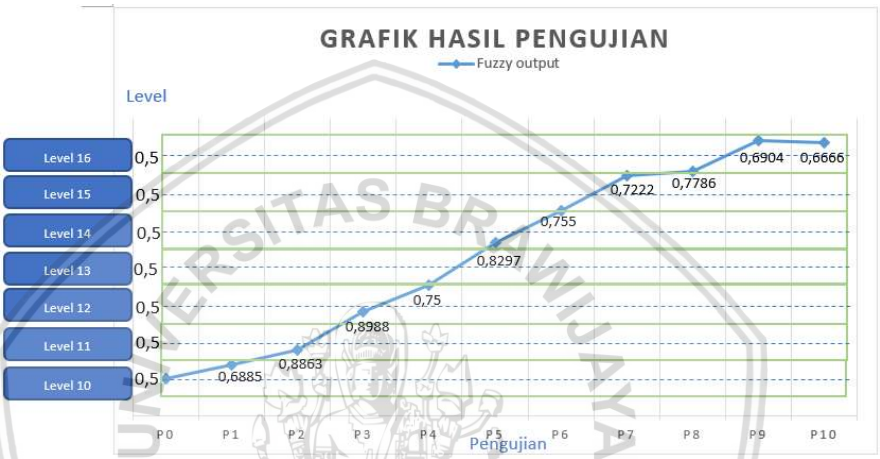
Berdasarkan Gambar 6.3 ditunjukkan hasil dari pengujian yang dilakukan oleh pemain ketiga. Titik pada gambar tersebut menjelaskan tentang besarnya nilai *fuzzy* yang didapatkan oleh pemain saat berada pada level tersebut. Diketahui bahwa titik awal pemain diberikan level 10 karena dengan anggapan pemain belum mengetahui kemampuan bermain dalam *game* labirin ini. Terlihat pada table 6.1, pada pengujian ke 1, pemain langsung naik ke *level* selanjutnya hingga akhirnya pada pengujian ke 4 pemain harus susah payah untuk menyelesaikan permainan yang membawa pemain berada pada garis rata rata level tersebut.



Gambar 6.4 Grafik hasil pengujian *player* 4

Berdasarkan Gambar 6.4 ditunjukkan hasil dari pengujian yang dilakukan oleh pemain keempat. Titik pada gambar tersebut menjelaskan tentang besarnya nilai *fuzzy* yang didapatkan oleh pemain saat berada pada level tersebut. Diketahui bahwa titik awal pemain diberikan level 10 karena dengan

anggapan pemain belum mengetahui kemampuan bermain dalam *game* labirin ini. Terlihat pada table 6.1, pada pengujian ke 1, pemain berusaha untuk beradaptasi dengan *game* yang di setting dengan level 10. Pada percobaan ke 5 barulah pemain dapat naik ke level selanjutnya dengan bersusah payah. Pada percobaan ke 6 pemain kesulitan kembali untuk naik ke level selanjutnya, namun masih dalam standart rata-rata. Baru pada percobaan ke 9 pemain baru dapat naik ke *level* selanjutnya.



Gambar 6.5 Grafik hasil pengujian *player* 5

Berdasarkan Gambar 6.2 ditunjukkan hasil dari pengujian yang dilakukan oleh pemain pertama. Titik pada gambar tersebut menjelaskan tentang besarnya nilai *fuzzy* yang didapatkan oleh pemain saat berada pada level tersebut. Diketahui bahwa titik awal pemain diberikan level 10 karena dengan anggapan pemain belum mengetahui kemampuan bermain dalam *game* labirin ini. Terlihat pada table 6.1, pada pengujian ke 1 hingga ke 6 dapat terlihat pemain selalu berada diatas rata-rata yang ditentukan, pemain beranggapan level awal yang diberikan terlalu mudah. Baru pada pengujian ke 7 pemain sedikit merasakan kesulitan berada diatas rata-rata *level* yang ditentukan.

Pengimplementasian metode *fuzzy* untuk *dynamic difficulty scaling* pada *game* labirin yang diuji dalam 5 kali percobaan dengan sample random mendapatkan nilai yang cukup balance bagi pemain dengan kemampuan bermain yang berbeda beda. kemampuan pemain dapat diasah dengan pengalaman bermain, sehingga level permainan akan merespon kemampuan dari pemain. Jika pemain terlalu berada dibawah rata rata maka level permainan akan diturunkan, jika pemain berada didalam rata rata maka level

permainan akan tetap pada level tersebut, dan jika permain berada diatas rata rata maka level permainan akan dinaikan.

BAB 7 PENUTUP

7.1 Kesimpulan

Berdasarkan hasil dari pengujian yang telah dilakukan, dapat disimpulkan beberapa poin sebagai berikut.

1. Penerapan algoritme *fuzzy* dilakukan dengan cara mengambil data input dari pemain yang telah menyelesaikan suatu level dari *game* labirin, yang di ambil sebagai input yaitu nilai dari waktu penyelesaian permainan (*time resolved),* total *score* yang di dapat (*score),* dan sisa *health point* dari pemain (*hp).* Pedoman yang dipakai dalam menentukan kriteria fungsi keanggotaan adalah nilai dari total jalan yang dapat diambil (*pathCount),* total dari coin yang bernilai positif (*coinPlus*), dan *range area* dari level labirin (*range)*. Dari ketiga inputan tersebut dilakukan proses fuzzyfikasi untuk menggolongkan fungsi keanggotaan dari setiap variabel. Dari setiap fungsi keanggotaan yang diperoleh kemudian dimasukkan pada setiap *rule base* dari *knowledge base* untuk proses implikasi dengan menggunakan metode MIN, hasil tersebut kemudian dilakukan proses defuzzyfikasi untuk mendapatkan nilai keluaran dari perhitungan fuzzy, hasil dari *fuzzy* tersebut yang dijadikan sebagai tolok ukur sebagai penentu level pada level selanjutnya dari *game* labirin.
2. Dari pengujian performa yang dilakukan, nilai dari fps permainan setelah penambahan dynamic difficulty scaling tergolong dalam kategori stabil dalam artian baik, yang artinya pengimplementasian logika fuzzy ini tidak mengganggu atau memberatkan permainan. Berdasarkan pengujian performa fps yang dilakukan, ditunjukkan hasil performa dari game hanya terpengaruh sedikit oleh ukuran dari labirin yang di*generate*. Nilai dari pengujian fps pun tergolong dalam kategori layak untuk dimainkan mengingat batas minimal standart fps adalah 30fps, sedangkan saat pengujian nilai terkecil adalah 87,65.

7.2 Saran

Dynamic difficulty scaling yang dilakukan adalah untuk memberikan fitur *balance* level masih mengharuskan pemain bermain pada awal memiliki kemungkinan jauh dalam *standart* kemampuan pemain. Kenaikan level atau penurunan level yang dilakukan masih berada pada 1 tingkat dan belum dapat melompat ke 2 atau lebih pada tingkat selanjutnya secara langsung. untuk menyelaraskan sistem dengan kemampuan pemain, pemain harus bermain dalam beberapa waktu dan beberapa kali untuk mencapai tingkat standart yang dimilikinya.

Oleh karena itu disarankan untuk dapat dikembangkan pada penelitian selanjutnya agar dapat memiliki lebih dari 1 pemikiran dan kemungkinan untuk sistem dapat membaca kemampuan pemain dengan lebih spesifik agar dapat memberikan tingkatan yang sesuai dengan pemain.

DAFTAR PUSTAKA

Arif, Y. M., Kurniawan F., Nugroho, F., 2011. *Desain Perubahan Perilaku pada NPC Game Menggunakan Logika Fuzzy*. Seminar on Electrical, Informatics and Education, p. A2.110 -A2.111, 2011.

Ahsani, N. H., Jonemao, E. M. A., Candra, D., 2014. *Penerapan Algoritma Carlo Tree Search Pada Permainan Komputer Maze Treasure*. Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, Universitas Brawijara.

Anggiawan, S. D., 2011. *Pembuatan Game Logic Puzzle untuk Edukasi Logika Anak*. Surabaya: Teknologi Multimedia Broadcasting, Institut Teknologi Sepuluh November.

Dogan, A., 2014. *Prevention and Treatments of Games Addiction: NonPharmacological Approaches for Game Addiction. In S. Gunuc, ed. Epidemiology of Game Addiction*. California: OMICS Group.

Handriyantini, Eva, 2009. *Permainan Edukatif (Educational Games) Berbasis Komputer untuk Siswa Sekolah Dasar.* Malang: Sekolah Tinggi Informasi & Komputer Indonesia.

Kusumadewi, S., 2010, *Aplikasi Logika Fuzzy untuk pendukung keputusan*, Edisi 2, Graha Ilmu, Yogyakarta.

Kusumadewi, S., 2003, *Artificial Intelligence (Teknik dan Aplikasinya)*, Yogyakarta : Graha Ilmu.

Millington, I., Funge, J., 2009. *Artificial Intelligence For Games Second Edition.* Tersedia di <http://lecturer.ukdw.ac.id/~mahas/dossier/gameng_AIFG.pdf> [Diakses 27 Oktober 2017].

Mutiah, Diana, 2010. *Psikologi Bermain Anak Usia Dini*. Jakarta: Kencana Prenada Media Group.

Moreira, R. V., Regan, L., Mandryk, Gutwin, C., 2015, *Now You Can Compete With Anyone : Balancing Players of Different Skill Levels in a First-Person Shooter Game*, Chi, 2015, 2255–64.

Nur, H., 2015.*Pergerakan dan Perilaku Non Player Characters pada Permainan First Person Shooter dengan Algoritma Boids dan Logika Fuzzy*. Banjarbaru.

Purba, K. Radion, Hasanah, Nur R., Muslim, Azis, M., 2013. *Implementasi Logika Fuzzy Untuk Mengatur Perilaku Musuh dalam Game Bertipe Action-RPG*, Jurnal EECCIS Vol. 7, No. 1, Juni 2013.

Roedavan, R. (2014). *Unity Tutorial Game Engine.*Bandung: Informatika.

Septian, R. F., 2014. *Implementasi Fuzzy Logic Metode Mamdani Untuk Pengembangan Intelligent Non-Player Character Pada Game Strategy*. Universitas Pendidikan Indonesia.

Suyanto, 2007. *Artificial Intelligence Searching, Reasoning, Planning and Learning, Penerbit Informatika*, Bandung.

Widiastuti, N. I., 2013. *Model Perilaku Berjalan Agen-Agen Menggunakan Fuzzy Logic,* Jurnal Ilmiah Komputer dan Informatika, Vol. 1, No.1, 2013.

Yuliani, S., 2015. *Apa Itu Softskill?.* Tersedia di <http://scdc.binus.ac.id/bslc/2017/08/apa-itu-soft-skills/> [Diakses 18 Oktober 2017].