GAME RESCUE IDOL DENGAN MENGGUNAKAN METODE FINITE STATE MACHINE (FSM)

SKRIPSI

Disusun Oleh :

DANAR PRADIKTIA YUNANTA

13.18.047

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI INDUSTRI

INSTITUT TEKNOLOGI NASIONAL MALANG 2017

LEMBAR PERSETUJUAN DAN PENGESAHAN

GAME RESCUE IDOL DENGAN MENGGUNAKAN METODE FINITE STATE MACHINE (FSM)

SKRIPSI

Disusun dan Diajukan Untuk Melengkapi dan Memenuhi Syarat Untuk Memperoleh Gelar Sarjana Komputer Strata Satu (S-1)

Disusun Oleh :

DANAR PRADIKTIA YUNANTA

NIM : 13.18.047

Diperiksa dan Disetujui

Dosen Pembimbing I	Dosen Pembimbing II
Suryo Adi Wibowo, ST., MT.	**Febriana Santi Wahyuni, S.Kom, M.Kom**
NIP.P. 1031000438	**NIP.P. 1031000425**

Mengetahui

Ketua Program Studi Teknik Informatika S-1

Joseph Dedy Irawan, ST., MT.
NIP. 197404162005011002

PROGRAM STUDI TEKNIK INFORMATIKA S-1

FAKULTAS TEKNOLOGI INDUSTRI

INSTITUT TEKNOLOGI NASIONAL MALANG 2017

LEMBAR KEASLIAN

PERNYATAAN KEASLIAN SKRIPSI

Saya yang bertanda tangan dibawah ini :

Nama : Danar Pradiktia Yunanta

NIM : 13.18.047

Program Studi : Teknik Informatika S-1

Fakultas : Fakultas Teknologi Industri

Menyatakan dengan sesungguhnya bahwa skripsi saya yang berjudul :

"Game Rescue Idol Dengan Menggunakan metode Finite State Machine (FSM)" Adalah skripsi sendiri bukan duplikasi serta mengutip atau menyadur seluruhnya karya orang lain kecuali dari sumber aslinya.

Malang, 03 Agustus 2017

Yang membuat pernyataan

Danar Pradiktia Yunanta

NIM. 13.18.047

GAME RESCUE IDOL DENGAN MENGGUNAKAN METODE FINITE STATE MACHINE (FSM)

Danar Pradiktia Yunanta
(13.18.047)
Program Studi Teknik Informatika S-1
Fakultas Teknologi Industri, Institut Teknologi Nasional Malang
Email : pradiktiadanar@gmail.com

Dosen Pembimbing : 1. Suryo Adi Wibowo, ST., MT.
2. Febriana Santi Wahyuni, S.Kom., M.Kom.

Abstraks

Terdapat media untuk bermain *game* yaitu konsol *game computer* dan *smartphone. Computer* salah satunya media yang paling banyak digunakan para pengembang industri *game* untuk mengeluarkan produknya. Dalam *game*, musuh dan karakter merupakan salah satu elemen yang paling penting dalam *video game*, untuk menjadikan *game* semakin menarik, karakter dan NPC *(Non Playable Character*), salah satunya musuh dalam *game* harus dapat meniru sifat atau perilaku manusia dalam dunia nyata. Unt uk melakukan hal tersebut diperlukan kecerdasan buatan pada karakter NPC (*Non Playable Character*) pada musuh yaitu kecerdasan buatan FSM (*Finite State Machine*) yang menggunakan kecerdasan buatan FSM (*Finite State Machine*) pada karakter musuh agar dapat mendeteksi keberadaan player, dan juga kecerdasan buatan *Fuzzy Logic* yang di gunakan untuk *output* aksi serangan musuh terhadap *player*.

Dalam perancangan *game* ini, penulis menggunakan aplikasi berupa *Unity3D* dengan bahasa pemrograman C#. Penulis dalam membuat game ini menerapkan metode kecerdasan buatan *Finite State Machine* yang menggunakan 3 hal yaitu keadaan, kejadian, dan aksi. Penulis juga menerapkan kecerdasan buatan *Fuzzy logic* yang digunakan pada karakter bos *(Non Playable Character*) untuk menentukan aksi dengan menggunakan 2 masukan berupa *health point* pada *player* dan jarak *player* ke bos.

Dari pembuatan *game* yang telah dilakukan, implementasi FSM (*Finite State Machine*) dapat diterapkan pada *game* 2 dimensi bergenre *adventure* dengan indikasi musuh dapat mengejar dan menyerang *player* dengan kondisi tertentu. Kecerdasan buatan yang diterapkan pada musuh sudah berjalan seluruhnya.Semua fungsi tombol pada *menu* utama sudah berjalan seluruhnya.Semua fungsi tombol yang diterapkan untuk melakukan aksi pada *player* sudah berjalan.Pengujian *game* di 3 laptop yaitu Asus,Lenovo, dan Acer dan juga 2 OS yaitu *windows* 8 dan 10 dapat berjalan dan sesuai. Pengujian *user* dengan 20 orang di Malang dapat dikatakan bahwa 42% mengatakan baik, 40% mengatakan cukup, dan 18% mengatakan kurang.Pengujian *cheat* yang sudah dibuat dapat berjalan seluruhnya.

Kata kunci : Game, Unity3D, Finite State Machine, Fuzzy Logic, NPC

KATA PENGANTAR

Segala puji Tuhan karena dengan rahmat dan hidayah-Nya lah penulis dapat menyelesaikan Skripsi yang berjudul **Game Rescue Idol Dengan Menggunakan Metode Finite State Machine (FSM)** tepat pada waktunya.

Terwujudnya laporan ini, tentunya tidak lepas dari bantuan-bantuan yang telah penulis terima. Pada kesempatan ini, penulis menyampaikan terima kasih kepada yang terhormat :

1. Bapak Dr. Ir. Lalu Mulyadi, MTA. Selaku Rektor Institut Teknologi Nasional Malang.
2. Bapak Dr. Ir. F. Yudi Limpraptono, MT, selaku Dekan Fakultas Teknologi Industri Insitut Teknologi Nasional Malang.
3. Bapak Joseph Dedy Irawan, ST, MT, selaku Ketua Program Studi Teknik Informatika, Institut Teknologi Nasional Malang.
4. Bapak Suryo Adi Wibowo, ST, MT, selaku Sekretaris Program Studi Teknik Informatika, Institut Teknologi Nasional Malang.
5. Bapak Suryo Adi Wibowo, ST, MT, selaku Dosen Pembimbing I yang selalu memberikan bimbingan dan masukan.
6. Ibu Febriana Santi Wahyuni, S.Kom, M.Kom selaku Dosen Pembimbing II yang selalu memberikan bimbingan dan masukan.
7. Semua dosen Program Studi Teknik Informatika yang telah membantu dalam penulisan dan masukan.
8. Semua teman teman berbagai angkatan yang telah memberikan do'a dan dukungan dalam penyelesaian skripsi.

Malang, Agustus 2017

Penulis

DAFTAR ISI

HALAMAN JUDUL.....i

LEMBAR PERSETUJUAN DAN PENGESAHAN ii

LEMBAR KEASLIAN iii

ABSTRAKSI.....iv

KATA PENGANTAR.....v

DAFTAR ISI.....vi

DAFTAR GAMBAR.....vii

DAFTAR TABEL viii

BAB I PENDAHULUAN 1

1.1 Latar Belakang 1

1.2 Rumusan Masalah 2

1.3 Batasan Masalah.....2

1.4 Tujuan.....2

1.5 Metodologi Penelitian 2

1.6 Sistematika Penulisan.....3

BAB II TINJAUAN PUSTAKA.....5

2.1 Penelitian Terkait.....5

2.2 Game.....5

2.2.1 Jenis-Jenis *Game* 5

a. *Game ction*.....5

b. *Game Action-Adventure*.....5

c. *Game* Simulasi, Kontruksi dan Manajemen.....6

d. *Role Playing Game*.....7

e. *Game* Strategi.....7

f. *Game Racing*.....7

g. *Game Puzzle*.....7

2.3 Kecerdasan Buatan (*Artificial Intelligence*).....7

2.3.1 Jenis-Jenis Kecerdasan Buatan.....8

a. *Decision Making*.....8

b. *Path Finding*.....10

2.4 Unity3D.....12

2.5 CorelDraw.....14

2.6 Bahasa Pemrograman C#.....15

BAB III ANALISIS DAN PERANCANGAN SISTEM.....16

3.1 Analisa Game.....16

3.1.1 Analisis Target User.....16

3.1.2 Analisis Kebutuhan Perangkat.....17

a. Analisis Kebutuhan Perangkat Lunak.....17

b. Analisis Kebutuhan Perangkat Keras.....17

3.1.3 Analisis Konsep Game.....17

a. Tujuan.....17

b. Start.....17

c. Middle 17

d. Ending 18

3.2 Perancangan Game 18

3.2.1 Story Line.....19

3.2.2 Perancangan Level 19

3.2.3 Flowchart Game 21

3.2.4 Perancangan Struktur Menu 21

3.2.5 Perancangan Metode FSM (*finite state machine*) 22

3.2.6 *Fuzzy Logic* untuk tindakan karakter bos 23

3.2.7 Perancangan Karakter 26

3.2.8 Perancangan Item dan Background.....27

BAB IV IMPLEMENTASI DAN PENGUJIAN.....28

4.1 Implementasi *Game* 28

4.1.1 Langkah-Langkah Membuat *Project* Baru di Unity3D 28

4.1.2 Membuat *Folder* di Unity3D 29

4.1.3 Membuat *Script* di Unity3D 31

4.1.4 Membuat *Scene* di Unity3D 31

4.1.5 Membuat *Animation* dan *Controller* di Unity3D 32

4.1.6 Mengatur *Animation* di Unity3D 34

4.1.7 Membuat *Game Object* "*World"* di Unity3D 37

4.1.8 Mengatur Kamera di Unity3D 39

4.1.9 Mengatur Patrol Musuh dan Target ke *Player* di Unity3D 40

4.1.10 Mengatur *Player Attack* dan *Enemy Attack* di Unity3D 41

4.1.11 Membuat Menu Utama di Unity3D 43

4.2 Pengujian *Gameplay* 46

4.2.1 *Menu* Utama *Game* 46

4.2.2 *Menu Help* 46

4.2.3 *Menu About* 47

4.2.4 *Menu Game Over* 47

4.2.5 *Menu Next Level* 48

4.2.6 *Enemy* Patrol ke *Player* 48

4.2.7 *Enemy Attack* ke *Player* 49

4.2.8 Penerapan *Player Attack* ke *Enemy* 50

4.2.9 Penerapan *Damage* dari Jebakan ke *Player* 51

4.2.10 Penerapan *Attack* Bos ke *Player* 51

4.2.11 Penerapan Item Tambah Nyawa 52

4.2.12 Penerapan Item Koin 52

4.3 Pengujian 53

4.3.1 Pengujian FSM (*Finite State Machine*) Pada Musuh 53

4.3.2 Pengujian Menu Utama 54

4.3.3 Pengujian Kontrol Player 54

4.3.4 Pengujian Laptop 55

4.3.5 Pengujian Fungsionalitas 55

4.3.6 Pengujian User 56

4.3.7 Pengujian Cheat 57

BAB V Penutup 58

5.1 Kesimpulan 58

5.2 Saran 58

DAFTAR PUSTAKA 60

DAFTAR GAMBAR

Gambar 2.1 Contoh Diagram State 10

Gambar 3.1 *Level* 1 20

Gambar 3.2 *Level* 2 20

Gambar 3.3 *Flowchart Game* 21

Gambar 3.4 Struktur Menu 22

Gambar 3.5 Alur *Finite State Machine* 23

Gambar 3.6 Grafik fungsi keanggotaan health point 23

Gambar 3.7 Grafik fungsi keanggotaan jarak pada player 24

Gambar 3.8 Grafik Fungsi Keanggotaan Output Aksi 25

Gambar 4.1Memulai Membuat Projek 28

Gambar 4.2 Create Project 28

Gambar 4.3 Tampilan Unity3D 29

Gambar 4.4 Assets 29

Gambar 4.5 Membuat Folder 29

Gambar 4.6 Folder Baru 30

Gambar 4.7 Assets 30

Gambar 4.8 Membuat Script 30

Gambar 4.9 Script Baru 31

Gambar 4.10 Assets 31

Gambar 4.11. Membuat Scene 31

Gambar 4.12 Scene Baru 32

Gambar 4.13 Folder Sprite 32
Gambar 4.14 Folder Character 32
Gambar 4.15 Memilih Sprites 33
Gambar 4.16 Mendrag Sprites 33
Gambar 4.17 Animation dan Controller Baru 33
Gambar 4.18 Memunculkan Tab Animator 34
Gambar 4.19 Tab Animator 34
Gambar 4.20 Mengatur Animation 34
Gambar 4.21 Membuat Transisi 35
Gambar 4.22 Mengarahkan Ke Animasi Lain 35
Gambar 4.23 Mengatur Transisi 35
Gambar 4.24 Membuat Parameter 36
Gambar 4.25 Menambahkan Parameter di Kondisi 36
Gambar 4.26 Hasil Transisi 36
Gambar 4.27 Membuat Game Object 37
Gambar 4.28 Game Object “World” 37
Gambar 4.29 Mengatur Background 37
Gambar 4.30 Menambahkan Box Collider 2D 38
Gambar 4.31 Mengatur Kamera 38
Gambar 4.32 Diterapkan Di Script 38
Gambar 4.33 Edge Collider 39
Gambar 4.34 Tampilan Edge Collider 39
Gambar 4.35 Setting Di Box Collider 2D 40

Gambar 4.36 Tampilan Sight 40
Gambar 4.37 Tampilan Script Sight 40
Gambar 4.38 Edge Collider Pada SwordCollider Enemy 40
Gambar 4.39 Tampilan SwordCollider pada Enemy 41
Gambar 4.40 Edge Collider Pada SwordCollider Player 41
Gambar 4.41 Tampilan SwordCollider Pada Player 41
Gambar 4.42 Menentukan Animasi Yang Dipakai Pada Player 42
Gambar 4.43 Membuat Canvas 42
Gambar 4.44 Membuat Button 43
Gambar 4.45 Event Trigger 43
Gambar 4.46 Text 43
Gambar 4.47 Event System 44
Gambar 4.48 Menu Utama 45
Gambar 4.49 Menu Help 45
Gambar 4.50 Menu About 46
Gambar 4.51 Game Over 46
Gambar 4.52 Next Level 47
Gambar 4.53 Musuh Melalukan Patrol Dan Menyerang Player 47
Gambar 4.54 Musuh Melakukan Patrol Dan Menyerang Player 48
Gambar 4.55 Musuh Menyerang Player Dengan Serangan Dekat 48
Gambar 4.56 Musuh Menyerang Player Dengan Serangan Jauh 49
Gambar 4.57 Karakter Player Menyerang Musuh Dengan Pedang 49
Gambar 4.58 Musuh Mati 50

Gambar 4.59 Darah Player Berkurang Terkena Jebakan 50

Gambar 4.60 Interaksi Player Dengan Bos 51

Gambar 4.61 Player Menyentuh Item Hp 51

Gambar 4.62 Player Menyentuh Item Koin 52

DAFTAR TABEL

Table 3.1 Fungsi keanggotaan *health point* 20

Table 3.2 Fungsi keanggotaan jarak 21

Table 3.3 Fungsi keanggotaan aksi 22

Table 3.4 Keterangan Karakter 26

Table 3.5 Keterangan *Item* dan *Background* 27

Table 4.1 Pengujian *Artificial Intelligence* 53

Table 4.2 Pengujian Menu Utama 53

Table 4.3 Pengujian *Control Player* 54

Table 4.4 Pengujian Laptop 55

Table 4.5 Pengujian Fungsionalitas 55

Table 4.6 Pengujian User 56

Table 4.7 Pengujian Cheat 57

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi dan informasi yang semakin pesat khususnya pada teknologi *computer* memberikan pengaruh besar pada berbagai bidang kehidupan, salah satunya adalah perkembangan *game*. Teknologi tidak harus dibatasi hanya untuk penggunaan mesin. Bersamaan dengan perkembangan teknologi dibidang elektronika munculah media baru yaitu komputer. Hal inilah yang membuat perkembangan *game* di *computer* sangat cepat. Game bukan hanya sekedar permainan untuk hiburan atau meningkatkan kreatifitas dan tingkat intelektual para penggunanya, pada saat ini tidak sedikit yang menjadikan *game* sebagai sebuah lapangan pekerjaan.

Metode yang dipakai dalam pembuatan *game "*Rescue Idol" ini menggunakan metode *FSM* atau *Finite State Machine dan Fuzzy Logic*. Dimana metode tersebut bagian dari metode kecerdasan buatan atau *artificial intelligence* untuk pengambilan keputusan karakter *NPC (Non Playable Character)* yaitu karakter yang digerakan oleh kecerdasan buatan yang digunakan untuk mendukung *game* tersebut

Pada *game* biasanya *FSM* dan *fuzzy logic* digunakan pada karakter *NPC (Non Playable Character)* yaitu karakter yang digerakan oleh kecerdasan buatan yang digunakan untuk mendukung *game* tersebut. Seperti karakter musuh, *fuzzy logic* dapat digunakan untuk menentukan gerakan dan aksi dari musuh tanpa melibatkan pengguna *game.* (Utomo, 2015).

Dari uraian di atas penulis ingin mengimplementasikan penggunaan metode *FSM* dan *Fuzzy Logic* untuk melakukan penelitian dalam pembuatan game untuk tugas akhir skripsi dengan judul "Rescue Idol" yang merupakan game 2D dengan *Genre Adventures* menggunakan *Unity3D* sebagai *game engine*. *Game* ini bercerita tentang seorang fans yang mempunyai keinginan untuk menyelamatkan idolanya yang diculik oleh penjahat. Dalam *game* ini penulis berharap nantinya *game* ini dapat digunakan untuk media hiburan interaktif dan meningkatkan daya kreatifitas pemain.

1.2 Latar Belakang

Dari latar belakang tersebut maka dicari suatu pemecah masalah yaitu :

1. Bagaimana merancang *game Rescue Idol* 2D dengan menggunakan *game engine* Unity3D ?.
2. Bagaimana mengimplementasikan metode *Finite State Machine* pada karakter musuh?.
3. Bagaimana mengimplementasikan metode *Fuzzy Logic* pada tindakan bos.

1.3 Batasan Masalah

Dalam penyusunan skripsi agar sistematis dan mudah dimengerti, maka akan diterapkan beberapa batasan masalah :

1. Jumlah misi dalam menyelesaikan game ini akan dibuat sebanyak dua level
2. Metode Finite State Machine diimplementasikan pada karakter musuh untuk menyerang karakter pemain jika memasuki jarak atau range dari musuh.
3. Metode Fuzzy Logic diimplementasikan pada bos musuh pada rintangan terakhir di tiap tahapan level.
4. *Game* ini dibuat dengan model 2D.
5. *Game* ini dibuat menggunakan *tools* dan *game engine* Unity3D.
6. Cerita dari game ini merupakan cerita fiksi yang dibuat oleh penulis.

1.4 Tujuan

Tujuan dari pembuatan game ini adalah Menerapkan metode *Finite State Machine* dan *Fuzzy Logic* untuk menentukan gerakan pada musuh dan tindakan bos di *game* Rescue Idol.

1.5 Metodologi Penelitian

Langkah-langkah yang digunakan dalam penyusunan skripsi disini menggunakan metode meliputi :

a. Studi Literatur

Pada tahap ini dipelajari literatur dan perencanaan serta konsep awal untuk merancang *game* yang akan dibuat yaitu didapat dari referensi buku, internet, maupun sumber-sumber yang lain.

b. Pengumpulan data dan analisis

Pada tahap ini adalah proses pengumpulan data yang dibutuhkan untuk pembuatan *game*, serta melakukan analisa atau pengamatan pada data yang sudah terkumpul untuk selanjutnya diolah lebih lanjut.

c. Analisa dan perancangan system

Setelah selesai pada tahap pengumpulan data dan analisis maka tahap selanjutnya adalah melakukan analisa dan perancangan sistem. Pada tahap ini adalah proses perancangan dari *game* yang akan dibuat untuk selanjutnya akan diproses lebih lanjut.

d. Pembuatan game

Setelah tahap perancangan sistem maka tahap selanjutnya adalah pembuatan *game*. Pada tahap ini rancangan yang sebelumnya telah dibuat akan diterapkan pada program yang akan dibuat. Pembuatan *game* ini menggunakan pemrograman C# dan *Fuzzy Logic Mamdani* dan FSM sebagai metode penalaran pada program ini.

e. Uji Coba Game

Setelah *game* selesai dibuat maka dilakukan pengujian program untuk mengetahui apakah game tersebut telah bekerja dengan benar dan sesuai dengan sistem yang dibuat.

f. Pembuatan Kesimpulan

Pada tahap akhir ini adalah pembuatan kesimpulan atau ringkasan dari skripsi ini dan kesimpulan tentang program yang telah dibuat.

1.6 Sistematika Penulisan

Untuk mempermudah memahami pembahasan pada penulis skripsi ini, maka sistematika penulisan yang diperoleh sebagai berikut :

BAB I : PENDAHULUAN

Berisi latar belakang, perumusan masalah, batasan masalah, tujuan penelitian, metode penelitian, dan sistemika penulisan.

BAB II : TINJAUAN PUSTAKA

Berisi tinjauan pustaka mengenai permasalahan yang berhubungan dengan penelitian ini.

BAB III : ANALISIS DAN PERANCANGAN

Berisi mengenai perancangan *game "*Rescue Idol" yang meliputi : analisis *game*, analisis kebutuhan perangkat, perancangan *game*, perancangan struktur menu, perancangan *Finite State Machine* dan *Fuzzy Logic*, perancangan karakter, dan *flowchart.*

BAB IV : IMPLEMENTASI DAN PENGUJIAN

Berisi implementasi metode penalaran pada *game* "Rescue Idol" serta melakukan pengujian terhadap *game* tersebut.

BAB V : PENUTUP

Berisi kesimpulan dari hasil penelitian yang dilakukan dan saran yang dapat digunakan untuk bahan pengembangan penelitian berikutnya.

BAB II
TINJAUAN PUSTAKA

2.1 Penelitian Terkait

Salah satu unsur yang berperan penting dalam sebuah *game* adalah kecerdasan buatan atau *Artificial Intelligence* (AI) yang merupakan suatu program komputer yang bertindak dan berpikir seperti manusia dan juga bertindak dan berpikir secara rasional pada saat yang bersamaan. Dengan kecerdasan buatan, elemen-elemen dalam *game* dapat berperilaku seperti manusia. *Game playing* (permainan game) merupakan bidang AI yang sangat populer berupa permainan antara manusia melawan mesin yang memiliki intelektual untuk berpikir. Komputer dapat bereaksi dan menjawab tindakan-tindakan yang diberikan oleh lawan mainnya (Utari, Dely. 2012).

Game "Rescue Idol" ini mengadopsi tipe *side scrolling game*, yaitu game dengan sudut pandang kamera dari sebelah samping. Dan game ini menggunakan 2 metode, Untuk karater musuh menggunakan metode FSM *(Finite State Machine),* yaitu kecerdasan yang berguna untuk menentukan berbagai macam respon NPC berdasarkan interaksi yang dilakukan oleh pemain, hal ini disebabkan karena FSM dapat digunakan untuk mendesain dan menentukan respon perilaku yang dilakukan terhadap perubahan kondisi. (Rahadian, Suyatno, & Maharani, 2016).

Untuk karakter bos musuh di *game* "Rescue Idol" menggunakan metode Fuzzy Logic untuk menentukan gerakan dan aksi yang dikeluarkan berdasarkan *health point* dan jarak, Fuzzy Logic sendiri adalah peningkatan dari logika Boolean yang berhadapan dengan konsep kebenaran sebagian. Saat logika klasik menyatakan bahwa segala hal dapat diekspresikan dalam istilah biner (0 atau 1, hitam atau putih, ya atau tidak), logika fuzzy menggantikan kebenaran boolean dengan tingkat kebenaran. (Wardani, K. E. 2015).

2.2 *Game*

Game merupakan salah satu hiburan karena mampu mengurangi tingkat kepenatan seseorang dari rutinitas pekerjaan setiap hari. *Game* juga mampu meningkatkan kecerdasan seseorang kertika *game* tersebut memerlukan tingkat

ketangkasan dari seorang pemain. Beberapa game yang beredar saat ini terdapat unsur mendidik, ketangkasan dan ada pula unsur kekerasan, maka ketika game itu diperjual belikan terdapat batas umue pemakainya. (Ardi, 2012)

Game atau lebih tepatnya disebut *video game* adalah permaian yang menggunakan interaksi dengan antarmuka pengguna melalui gambar yang dihasilkan oleh piranti *video*. Permainan *video* umumnya menyediakan system penghargaan yang dicapai dalam menyelesaikan tugas-tugas yang ada dalam permainan. (Satriyana, 2012).

2.2.1 Jenis-Jenis *Game*

Seiring dengan perkembangan industry *game*, jenis-jenis game semakin bervariasi. Perbedaan jenis game terletak pada *gameplay*, interaksi, dan kategori. *Gameplay* merupakan system yang berjalan pada *game*. *System* tersebut meliputi *story line*, cara bermain, menu, area permainan, dan sebgainya. Bisa jadi dua *game* memiliki persamaan, namun yang membedakannya adalah bagaimana pemain memainkan dan berinteraksi dengan game. Berdasarkan hal-hal tersebut, game dikategorikan menjadi berbagai macam. Jenis-jenis game meliputi :

a. *Game Action*

Game ini merupakan macam game yang paling popular. *Game* jenis ini membutuhkan kemampuan reflex pemain dan timing yang tepat. Salah satu subgenre *action* yang paling popular adalah first person shooter. *Game* jenis ini memerlukan kecepatan berpikir, *game* ini dibuat seolah-olah pemain yang berada dalam suasana tersebut. Contoh *game genre* ini, misalnya Half Life, Crysis, Paladin.

b. *Game Action-Adventure*

Game ini memadukan gameplay aksi dan petualangan, pemain diajak untuk menelusuri goa bawah tanah sambil mengalahkan musuh, mencari artefak kuno, menyebrangi sungai dan sebagainya. Saat ini kebanyakan *genre* ini sudah mengadopsi 3 Dimensi. Contoh *game genre* ini, misalnya Tomb Rider, Hitman, Splinter Cell.

c. *Game* Simulasi, Kontruksi dan Manajemen

Pemain dalam *game* ini diberi keleluasaan untuk membangun, berekspansi, dan mengatur komunitas fiksi atau proyek tertentu dengan bahan baku yang terbatas.

Contoh *game genre* ini, misalnya SimCity, The Sims, dan Spore.

d. *Role Playing Game*

Dalam *game role playing game* dapat memilih satu karakter untuk dimainkan, seiring dengan naiknya *level game*, karakter tersebut dapat berubahm bertambah skill-nya, bertambah senjatanya, bertambah hewan peliharaannya dan lain sebagainya. Contoh game genre ini, misalnya Final Fantasy, World Of Warcraft, Fallout.

e. *Game* Strategi

Awal mula *genre* ini adalah board game, genre strategi menitik beratkan pada kemampuan berpikir dan organisasi. Game strategi dibedakan menjadi dua yaitu *Turn Based Strategy* dan *Real Time Strategy* mengharuskan pemain membuat keputusan dan secara bersamaan pihak lawan juga beraksi hingga menimbulkan serangkaiaan kejadian waktu yang sebenarnya. Contoh *game genre* ini *Age Of Empires*, *Rise Of Nation*. Sementara itu *genre Turn Based Strategy* mengharuskan pemain berganti menjalankan taktiknya. Saat pemain mengambil langkah, pihak lawan menunggu, demikian juga sebaliknya. Contoh game genre ini *Heroes Of Might and Magic*, *Master Of Orion*.

f. *Game Racing*

Dalam *game* ini pemain dapat memilih kendaraan, menandai lalu melaju diarena balap. Tujuannya hanya satu, yaitu mencapai garis finish tercepat. Contoh game genre ini Need For Speed, Grand Turismo, Daytona, Top Gear.

Genre game ini membawa olahraga ke dalam *computer* atau konsol, biasanya *gameplay* dibuat semirip mungkin dengan kondisi olahraga yang sebenarnya. Contoh *game genre* ini adalah FIFA, Winning Eleven, PES, NBA, Tony Hawk Pro.

g. *Game Puzzle*

Game Puzzle menyajikan teka-teki, menyamakan warna bolam perhitungan matematika, menyusun balok, dan sebagainya.contoh *game genre* ini Tetris, Bejeweled, Minesweeper, dan Bomberman.

2.3 Kecerdasan Buatan (*Artificial Intelligence*)

Kecerdasan buatan (*artificial intelligence*) merupakan salah satu bagian dari ilmu *computer* yang mempelajari bagaimana mesin (komputer) dapat melakukan

pekerjaan seperti dan sebaik yang dilakukan oleh manusia bahkan bisa lebih baik dari yang dilakukan oleh manusia.

Menurut McCarhy, (1956) AI (*artificial intelligence*) adalah untuk mengetahui dan memodelkan proses-proses berpikir manusia dan mendesain mesin agar dapat meirukan perilaku manusia. Cerdas, berarti memiliki pengetahuan ditambah pengalaman. Penalaran (bagaimana membuat keputusan dan mengambil tindakan), moral yang baik.

Manusia cerdas (pandai) dalam menyelesaikan permasalah karena manusia mempunyai pengetahuan dan pengalaman. Pengetahuan diperoleh dari belajar. Semakin banyak bekal pengetahuan yang dimiliki tentut lebih mampu menyelesaikan permasalah. Tapi bekal pengatahuan tidak cukup, manusia juga diberi akal untuk melakukan penaran, mengampil kesimpulan berdasankan pengetahuan dan pengalaman yang dimiliki.

Tanpa memiliki kemampuan untuk menalar dengan baik, manusia dengan segudang pengalaman dan pengetahuan tidak dapat menyelesaikan masalah dengan baik. Demikian juga dengan kemampuan menalar yang sangat baik, tanpa bekal pengetahuan dan pengalaman yang memadai manusia tidak dapat menyelesaikan masalah dengan baik (Dahria, 2008).

2.3.1 Jenis-Jenis Kecerdasan Buatan

Adapun jenis-jenis kecerdasan buatan meliputi :

a. *Decision Making*

Adapun kecerdasan buatan decision making meliputi :

1. *Fuzzy Logic*

Fuzzy secara bahasa diartika sebgai kabur atau samar-samar. Suatu nilai dapat bernilai besar atau salah secara bersamaan, dalam *fuzzy* dikenal dengan derajat keanggotaan yang memiliki rentang nilai 0 (nol) hingga 1 (satu). Berbeda dengan himpunan tegas yang memiliki nilai 1 atau 0 (ya atau tidak).

Logika *fuzzy* merupakan suatu logika yang memiliki nilai kekaburan atau kesamaran antara benar atau salah. Dalam teori logika *fuzzy* suatu nilai bias bernilai benar atau salah secara bersamaan. Namun berapa besar keberaddan dan kesalahan

suatu tergantung pada bobot keanggotaan yang dimilikinya. Logika *fuzzy* memiliki derajat keanggotaan 0 hingga 1. Berbeda dengan logika digital yang hanya memiliki dua nilai 1 atau 0. Logika *fuzzy* digunakan untuk menterjemahkan suatu besaran yang diekpresikan menggunakan bahasa misalkan besaran kecepatan laju kendaran yang diekpresikan dengan pelan, agak cepat, cepat dan sangat cepat (Hasanah, 2014).

Penerapan *fuzzy logic* pada *game bonny's tooth booth* dengan *genre virtual pet* dimana untuk mengjasilkan perilaku *pet* agar dapat berprilaku adaptif seperti *pet* sungguhan, sehingga tidak hanya dari pengaturan waktu saja sebagai penentu prilakunya. Perancangan prilaku pet pada permainan ini menentukan kondisi nafsu makan dan haus dengan *energy* dan suhu sebagai parameter inputnya dimana nantinya akan menghasilkan output perilaku kesehatan. Konsep dari permainan virtual pet dengan misi permainan menjaga kebersihan gigi pada permaian *bonnys tooth booth*.

Penerapan *fuzzy logic* pada penelitian ini berfokus mengenai *scoring game* pada NPC (*non player character*) dan nilai parameter inputan dari Nilai Kesehatan (HP), Nilai Serang (AP), Nilai Pertahanan (DP), *damage* untuk menghitung nilai pada *game*, sehingga tidak hanya pada *player scoring* dibutuhkan suatu *scoring* tetapi diperlukan suatu parameter nilai untuk NPC (*non* pl*a*yer *character*) dengan menggunakan kecerdasan buatan yaitu *fuzzy* (Putri,2014).

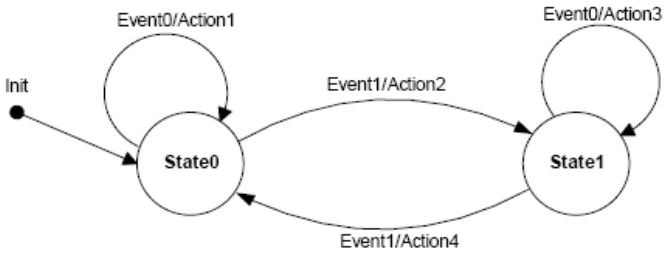
2. *Finite State Machine* (FSM)

Penelitian berjudul penerapan metode finite state machine pada game "The Relationship", pada metode ini finite state machine pada game berhuna untuk menentukan berbagai macam respon *Non Player Character* berdasarkan interaksi yang

dilakukan oleh pemain, hal ini disebabkan karena *Finite State Machine* dapat digunakan untuk mendesain dan menentukan respon perilaku yang dilakukan terhadap perubahan kondisi. Salah satu hasil dari penerapan metode *Finite State Machine* pada penelitian ini adalah diawali pada *state* awal, sebuah *NPC* berada dalam keadaan diam, kemudian ketika player utama menabrak *NPC,* maka *NPC* akan berbicara, jika pembicaraan selesai kemudian *NPC* akan kembali pada keadaan diam (rahadian,2016).

Penelitian yang berjudul Rancang Bangun *Game Survival Armybot* Menggunakan *Finite State Machine (FSM).* Implementasi dari *Finite State Machine*

dapat diterapkan pada *game 3D Adventure* dengan indikasi musuh dalam *game* dapat mengejar dan menyerang karakter pemain dalam kondisi tertentu. Hasil dari penerapan *Finite State Machine* pada *game* tersebut adalah jika player berada dalam jarak yang dekat dengan musuh, maka musuh akan menyiapkan senjata. dalam jarak tertentu, musuh dapat mengejar player dan dalam jarak tertentu juga musuh dapat menyerang player (Utomo,2015).



Gambar 2.1. Contoh Diagram *State* Sederhana

Penelitian yang berjudul Pembuatan *Game* 3D *Fighting* Dengan Menggunakan *Finite State Machine* Sebagai Strategi Karakter, pada *game* ini, *Finite State Machine* digunakan untuk menentukan bagaimana karakter bereaksi atau memutuskan tindakan berdasarkan situasi atau kondisi tertentu. Pada *Finite State Machine* ini, penulis menggunakan tujuh buah *state* yang biasanya harus ada pada suatu *game* dengan jenis *fighting*. Hasil dari penerapan *Finite State Machine* pada *game* ini adalah terciptanya karakter musuh dimana dalam keadaan tertentu dapat memutuskan gerakan berupa maju, serang, bertahan, diam, jurus 1, jurus 2, tangkis dan mundur. Dengan demikian player dapat melawan musuh berupa *NPC* yang dapat memutuskan gerakanya sendiri (Suharian,2008).

Penelitian yang berjudul rancang bangun game edukasi ular tangga pada aplikasi mobile, pada game ini penerapan metode finite state machine terletak pada karakter

NPC. Penerapan finite state machine pada karakter NPC bertujuan agar karakter NPC dapat memutuskan aksi serangan yang dia lakukan saat bermain. Dengan demikian player utama dapat memainkan game ular tangga dengan melawan NPC terasa seperti

melawan manusia nyata. *State* yang terdapat pada karakter *NPC* diantaranya mengacak dadu, jalan, naik dan turun (Shaleh, 2009).

b. *Path Finding*

Adapun kecerdasan buatan path finding meliputi :

1. *Algoritma A Star* (A*)

Algoritma *A Star* (A*) merupakan Algoritma *Best First Search* yang menggabungkan *Uniform Cost Search* dan *Greedy Best-First Search*. Biaya yang diperhitungkan didapat dari biaya sebenarnya ditambah dengan biaya perkiraan, dalam notasi matematika dituliskan sebagai berikut: ( ) = ( ) + ( ). Berikut merupakan pengertian dari notasi matematika algoritma. A* F adalah nilai suatu kotak. H adalah estimasi nilai yang dibutuhkan untuk menempuh jarak dari kotak selanjutnya untuk menempuh poin tujua dan G nilai yang dibutuhkan untuk menempuh jarak dari poin awal menuju kotak selanjutnya. Algoritma A* adalah salah satu algoritma dalam kecerdasan buatan yang digunakan untuk mencari rute terdekat. Algoritma A* menyelesaikan masalah yang menggunakan graf untuk perluasan ruang statusnya. Dengan menerapkan suatu heuristik, algoritma ini membuang langkah-langkah yang tidak perlu dengan pertimbangan bahwa langkah-langkah yang dibuang sudah pasti merupakan langkah yang tidak mencapai solusi yang diinginkan. Algoritma A* membangkitkan simpul yang paling mendekati solusi. Simpul ini kemudian disimpan suksesornya ke dalam *list* sesuai dengan urutan yang paling mendekati solusi terbaik. Kemudian, simpul pertama pada *list* diambil, dibangkitkan suksesornya dan kemudian suksesor ini disimpan ke dalam *list* sesuai dengan urutan yang terbaik untuk solusi (Suyanto, 2015).

2. *Algoritma Minimax*

Algoritma *minimax* (juga sering disebut *Minmax*) adalah sebuah algortima yang mendasari pola piker langkah penyelesaian masalah dalam beberapa jenis komputer, seperti tic-tac-toe, Othello, checkers, catur, dll. Pada dasarnya, algoritma minmax sangat andal untuk menyelesaikan segala masalah dalam pencarian langkah untuk permainan komputer dengan jumlah kemungkinan penyelesaian yang kecil, seperti pada permainan tic-tac-toe. Tetapi, jika algoritma minimax digunakan pada permainan yang jumlah kemungkinanya yang besar seperti pada permainan catur, algoritma minimax ini memerlukan waktu yang sangat lama untuk membangun pohon penyelesaian. Oleh karena itu, bebrapa metode optimasi telah dikembangkan untuk membatasi melonjaknya jumlah simpul dalam pembangunan pohon penyelesaian. Berbagai jenis metode telah ditemukan untuk mengoptimasi algoritma minimax, seperti alpha betasearch, negascout, null move search, MTD, dll. Dengan menggunakan metode

optimasi inilah proses komputasi penyelesaian masalah pada permainan catur dapat diminimalisasi (Ilman, 2008).

3. *Breadth Search First* (BFS)

Breadth search first adalah suatu metode yang melakukan pencarian secara melebar yang mengunjungi secara preorder yaitu mengunjungi suatu simpul kemudian mengunjungi simpul yang bertetangga dengan simpul tersebut dahulu. Selanjutnya, simpul yang belum dikunjungi dan betetangga dengan simpul-simpul yang tadi dikunjungi, demikian seterusnya. Jika graf berbentuk pohon graf berakar, maka semua simpul pada arah dikunjungi lebih dahulu sebelum simpul-simpul pada arah d+1 (Prasetyo, 2014).

4. *Depth Search First* (DFS)

Depth first search (DFS) adalah suatu metode pencarian pada pohon dengan menelusuri satu cabang sebuah pohon sampai menemukan solusi. Pencarian dilakuakan pada satu node dalam setiap *level* dari yang paling kiri dan dilanjutkan pada *node* sebelah kanan. Jika solusi ditemukan maka tidak diperlukan proses backtracking yaitu penelusuran balik untuk mendapatkan jalur yang diinginkan. Pada metode DFS pemakaian memori tidak banyak karena node-node pada lintasan yang aktif saja yang disimpan. Selain itu jika solusi yang dicari berada pada level yang dalam dan paling kiri, maka DFS akan menemukannya secara cepat (Prasetyo, 2014).

5. *Algoritma Dijkstra*

Alogritma dijkstra merupakan salah satu algoritma yang efektif dalam memberikan lintasan terpendek dari suatu lokasi ke lokasi lain. Prinsip dari algoritma Dijkstra adalah dengan pencarian dua lintasan yang paling kecil. Algoritma Dijkstra memiliki iterasi untuk mencari titik yang jaraknya dari titik awal adalah paling pendek. Pada setiap iterasi, jarak titik yang diketahui (dari titik awal) diperbarui apabila terbyata didapat titik yang baru yang memberikan jarak terpendek. Syarat algoritma ini adalah bobot sisinya yang harus non-negatif (sholichin, 2013).

2.4 Unity3D

Unity merupakan suatu aplikasi yang digunakan untuk mengembangkan game multi platform yang didesain untuk mudah digunakan. Unity itu bagus dan penuh perpaduan dengan aplikasi yang profesional. Editor pada Unity dibuat degan user interface yang sederhana. Editor ini dibuat setelah ribuan jam yang mana telah

dihabiskan untuk membuatnya menjadi nomor satu dalam urutan rankking teratas untuk editor game. Grafis pada unity dibuat dengan grafis tingkat tinggi untuk OpenGL dan directX. Unity mendukung semua format file, terutamanya format umum seperti semua format dari art applications. Unity cocok dengan versi 64-bit dan dapat beroperasi pada Mac OS x dan windows dan dapat menghasilkan game untuk Mac, Windows, Wii, iPhone, iPad dan Android.

Unity secara rinci dapat digunakan untuk membuat video game 3D, real time animasi 3D dan visualisasi arsitektur dan isi serupa yang interaktif lainnya. Editor Unity dapat menggunakan plugin untuk web player dan menghasilkan game browser yang didukung oleh Windows dan Mac. Plugin web player dapat juga dipakai untuk widgets Mac. Unity juga akan mendukung console terbaru seperti PlayStation 3 dan Xbox 360. Pada tahun 2010, telah memperoleh Technology Innovation Award yang diberikan oleh Wall Street Journal dan tahun 2009, Unity Technology menjadi 5 perusahaan game terbesar. Tahun 2006, menjadi juara dua pada Apple Design Awards.

Server aset dari Unity dapat digunakan semua scripts dan aset game sebagai solusi dari versi kontrol dan dapat mendukung proyek yang terdiri atas banyak gigabytes dan ribuan dari file multi-megabyte. Editor Unity dapat menyimpan metadata dan versi mereka, itu dapat berjalan , pembaharuan dan didalam perbandingan versi grafis. Editor Unity dapat diperbaharui dengan sesegera mungkin seperti file yang telah dimodifikasi. Server aset Unity juga cocok pada Mac, Windows dan Linux dan juga berjalan pada PostgreSQL, database server opensource.

Perizinan atau license dari Unity ada dua bentuk. Ada Unity dan Unity Pro. Versi Unity tersedia dalam bentuk gratis, sedang versi Unity Pro hanya dapat dibeli. Versi Unity Pro ada dengan fitur bawaan seperti efek post processing dan render efek texture. Versi Unity merupakan yang gratis memperlihatkan aliran untuk game web dan layar splash untuk game yang berdiri sendiri. Unity dan Unity Pro menyediakan tutorial, isi, contoh project, wiki, dukungan melalui forum dan perbaruan kedepannya. Unity digunakan pada iPhone, iPod dan iPad operating system yang mana iOS ada sebagai add-ons pada Unity editor yang telah ada lisensinya, dengan cara yang sama juga pada Android.

Unity 3D adalah salah satu *software* yang bagus untuk mengembangkan *game* 3D dan selain itu juga merupakan *software* atau aplikasi yang interaktif dan atau dapat

juga digunakan untuk membuat animasi 3 dimensi. *Unity* lebih tepat dijelaskan sebagai salah satu *software* untuk mengembangkan *video game* atau disebut juga *game engine* , yang sebanding dengan *game engine* yang lain contohnya saja: *Director* dan *Torque.*

game engine. *Unity* sebanding dengan mereka (*Director* dan *Torque*) dikarenakan mereka semua sama – sama menggunakan grafis yang digunakan untuk pengembangan aplikasi 3D. (Nugraha, 2014)

2.5 CorelDraw

CorelDraw adalah program untuk menggambar yang berbasis vector, pemakai dapat dengan mudah membuat suatu desain karya seni yang professional, mulai dari logo sederhana sampai pada ilustrasi teknik yang rumit. Karena itu aplikasi CorelDraw 12 sangat cocok untuk membuat suatau rancangan gambar atau desain grafis, seperti membuat latar belakang, tombol, dan gambar-gambar yang diperlukan selama pembuatan rancangan sehingga didapat tampilan pewarnaan yang menarik (Waskito, 2014)

CorelDraw adalah software grafis serbaguna yang biasa dipakai untuk ilustrasi dan publikasi. Sehingga banyak digunakan untuk aplikasi percetakan di media kertas, kain, outdoor, elektronik dll. CorelDraw merupakan aplikasi grafis yang dengan format vector (koordinat), tidak seperti Macromedia Adobe Photoshop yang lebih mengutamakan format bitmap (pixel). (Hendi Hendratman 2009) dalam (Miskowati, 2013)

2.6 Bahasa Pemrograman C#

C# (dibaca: *C sharp*) merupakan sebuah bahasa pemrograman yang berorientasi objek yang dikembangkan oleh Microsoft sebagai bagian dari inisiatif kerangka *.NET Framework.* Bahasa pemrograman ini dibuat berbasiskan bahasa C++ yang telah dipengaruhi oleh aspek-aspek ataupun fitur bahasa yang terdapat pada bahasa-bahasa pemrograman lainnya seperti *Java, Delphi, Visual Basic,* dan lain-lain) dengan beberapa penyederhanaan.

C# bersifat sederhana, karena bahasa ini didasarkan kepada bahasa C dan C++. Jika anda familiar dengan C dan C++ atau bahkan Java, anda akan menemukan aspek-aspek yang begitu familiar, seperti *statements, expression, operators,* dan beberapa

fungsi yang diadopsi langsung dari C dan C++, tetapi dengan berbagai perbaikan yang membuat bahasanya menjadi lebih sederhana.

Object Oriented Language *C#* memenuhi syarat-syarat sebagai sebuah bahasa pemrograman yang bersifat *Object Oriented*, yaitu *encapsulation*, *inheritance* dan *polymorphism*. C# bisa digunakan untuk membuat berbagai macam aplikasi, seperti aplikasi pengolah kata, grafik, *spreadsheets*, atau bahkan membuat kompiler untuk

sebuah bahasa permrograman**.** C# tidak memiliki terlalu banyak *keyword*, sehingga dapat mengurangi kerumitan.

Kode C# ditulis dengan pembagian masing *Class-Class (classes)* yang terdiri dari beberapa *routines* yang disebut sebagai *member methods*. *Class-Class* dan metode-metode ini dapat digunakan kembali oleh program atau aplikasi lain. Hanya dengan memberikan informasi yang dibutuhkan oleh *class* dan metode yang dimaksud, maka kita akan dapat membuat suata kode yang dapat digunakan oleh satu atau beberapa aplikasi dan program *(reusable code).* Setelah kita membaca dan mengetahui apa itu bahasa C# dan mengapa di baca *c sharp*, sekarang kita harus mengetahui apa saja kelebihan dan kekurangan bahasa C#. (Wardani, K. E. 2015).

BAB III
ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis *Game*

Analisis *game* merupakan proses identifikasi serta evaluasi terhadap *game* sejenis dan *game* yang dibangun. Dalam *game adventure*, pemain diharuskan untuk menyelesaikan semua level yang ada dalam *game*.

Setiap *level* memiliki tingkat kesulitan tertentu dan memiliki syarat tertentu seperti menyelesaikan misi di setiap *level* untuk dapat menuju ke *level* selanjutnya. Pada *game adventure*, terdapat kriteria yakni pada saat pemain menjalankan *game* akan melewati hutan, sungai, gunung dan lainnya.

Game Rescue Idol ini merupakan *game* dengan *genre adventure* dengan *sideview* atau *sidescroller* yaitu *game* yang terlihat dari samping dan *game* seperti ini merupakan *game* 2 dimensi yang terdiri dari 2 *level* yang harus diselesaikan untuk memenangkan *game* ini dan target user dari *game* ini adalah usia 12 tahun ke atas. Berikut ini adalah analisis sistem yang ada pada *game* ini :

1. *System* single player
2. Grafik game 2 dimensi.
3. Interaksi menggunakan *keyboard*.
4. Tindakan pada karakter musuh di implementasikan menggunakan *finite state machine* dan *fuzzy logic*.
5. *Game* ini dibuat dengan menggunakan *software unity3D* dengan menggunakan bahasa pemrograman C#.

3.1.1 Analisis Target *User*

Analisis penggunaan dimana digunakan untuk mengetahui spesifikasi pengguna pemain agar dapat memainkan permaianan dari *game* Rescue Idol yaitu:

1. Pengguna untuk usia 12 tahun ke atas. 20.
2. Berlatar belakang pendidikan mulai dari SMP, SMA, hingga dewasa.
3. Pengguna mengerti cara pengoperasian *computer*.

3.1.2 Analisis Kebutuhan Perangkat

Analisis kebutuhan perangkat menggambarkan kebutuhan yang dimiliki oleh sistem, diantaranya kebutuhan perangkat keras, perangkat lunak, serta *user* sebagai bahan analisis kekurangan dan kebutuhan yang harus dipenuhi dalam perancangan sistem yang akan diterapkan meliputi :

a. Analisis Kebutuhan Perangkat Lunak

1. Spesifikasi perangkat lunak yang dibutuhkan pengembang yang digunakan dalam membangun *game* Rescue Idol yaitu meliputi :

a. Sistem Operasi Windows 7
b. *Software Unity* versi 5.3.1
c. *Software CorelDraw*
d. *Visual studio 2010 atau versi diatasnya*

2. Sedangkan spesifikasi perangkat lunak bagi pemain yang memainkan *game* Rescue Idol adalah dengan sistem operasi *Windows* 8 dan 8.1.

b. Analisis Kebutuhan Perangkat Keras

Spesifikasi *hardware* atau perangkat keras yang dibutuhkan oleh pengembang yaitu meliputi :

a. Prosesor dengan kecepatan 2.3 Ghz
b. RAM 2 gigabyte.
c. *Hardisk* 20 gigabyte.
d. VGA Card 1024 megabyte.
e. Monitor.
f. *Mouse* dan *Keyboard*.
g. *Speaker*.

3.1.3 Analisis Konsep Game

a. Tujuan

Tujuan dari *game* “Rescue Idol” yaitu menyelamatkan idola dari musuh-musuh yang ada pada tiap level.

b. Start

1. *Player* memulai dari level 1 dengan memiliki *health point* (hp) dalam bentuk bar.
2. *Player* memilik 3 nyawa.
3. *Player* mempunyai aksi berupa serangan.

4. Jika *health point player* habis, player mengulang di level 1.
5. Jika nyawa habis *(game over), player* mengulang dari level 1.
6. *Player* mengalahkan musuh-musuh yang diterapkan kecerdasan buatan FSM dengan serangan dekat.
7. *Player* dapat menghindar dari jebakan yang ada.
8. *Player* dapat menambah *health point* jika menyentuh *item* yang dapat menambah *health point player.*
9. *Player* dapat menambah skor jika menyentuh item koin.

c. Middle

1. *Player* menuju level 2 untuk menyelamatkan idolanya dari tangan bos.
2. *Player* juga memilik *health point* dalam bentuk bar
3. *Player* memiliki 3 nyawa.
4. *Player* mempunyai aksi berupa serangan.
5. *Player* akan mati jika *health point* habis.
6. Jika *health point player* habis, player mengulang di level 2.
7. Jika nyawa habis *(game over), player* mengulang dari level 1.
8. *Player* dapat menghindar dari jebakan yang ada.
9. *Player* mengalahkan musuh-musuh yang diterapkan kecerdasan buatan FSM dengan serangan dekat dan serangan jauh.
10. *Player* mengalahkan bos yang diterapkan kecerdasan buatan *Fuzzy Logic.*
11. *Player* dapat menambah *health point* jika menyentuh *item* yang dapat menambah *health point player.*
12. *Player* dapat menambah skor jika menyentuh item koin.

d. Ending

Player berhasil mengalahkan bos dan menyelamatkan idolanya.

3.2 Perancangan *Game*

Perancangan sistem adalah suatu bagian dari metodologi pengembangan suatu perangkat lunak yang dilakukan untuk memberikan gambaran secara terperinci tentang *Game* Rescue Idol. Perancangan *Game* Rescue Idol meliputi :

3.2.1 *Story Line*

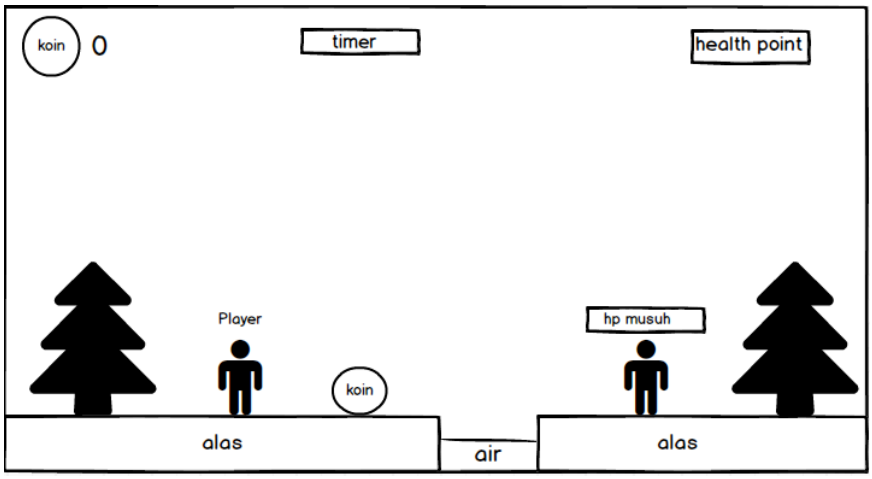
Game ini menceritakan tentang seorang fans yang mempunyai idola. Untuk alur cerita game ini yaitu :

1. Alkisah di ceritakan seorang anak yang mempunyai seorang idola yang sangat dia idolakan.
2. Pada suatu hari anak itu mendapat kabar kalau sang idola diculik oleh kawanan penjahat.
3. Pada hari itu juga anak tersebut memulai petualangannya menyelamatkan sang idola yang diculik.
4. Mereka harus menempuh perjalanan yang sulit dan panjang melewati gunung dan hutan.
5. Mereka harus melawan beberapa penjahat dan halang rintang.
6. Hingga akhirnya mereka akan melawan bos dari penjahat yang menyandra idolanya.
7. Setelah berhasil menyelamatkan idolanya yang disandra oleh bos.
8. Mereka kemudian bahagia karena dapat menyelamatkan idolanya yang diculik dari kawanan penjahat.

3.2.2 Perancangan *Level*

a. *Level* 1

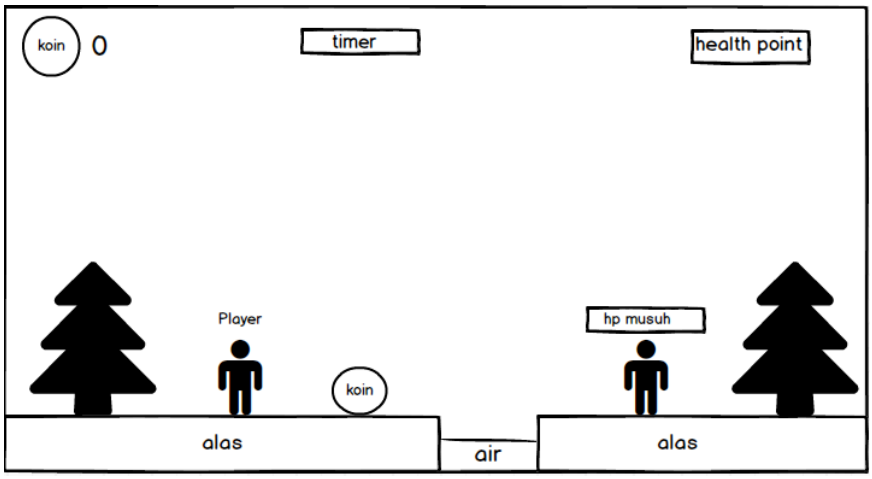
1. *Goal* pada *level* 1 *game Rescue Idol* ini adalah, pemain harus melewati jebakan dan musuh yang sudah diterapkan kecerdasan buatan *finite state machine* dengan serangan jarak dekat.
2. Setelah berhasil melewati jebakan dan musuh-musuh tersebut maka pemain bisa menuju ke level selanjutnya yang nantinya misi untuk menyelamatkan idola yang disandra.
3. Di level 1 ini terdapat *item* untuk menambah *health point* dan juga terdapat koin untuk menambah *score*.
4. *Player* menyerang dengan serangan dekat menggunakan pedang.



Gambar 3.1 Level 1

b. *Level* 2

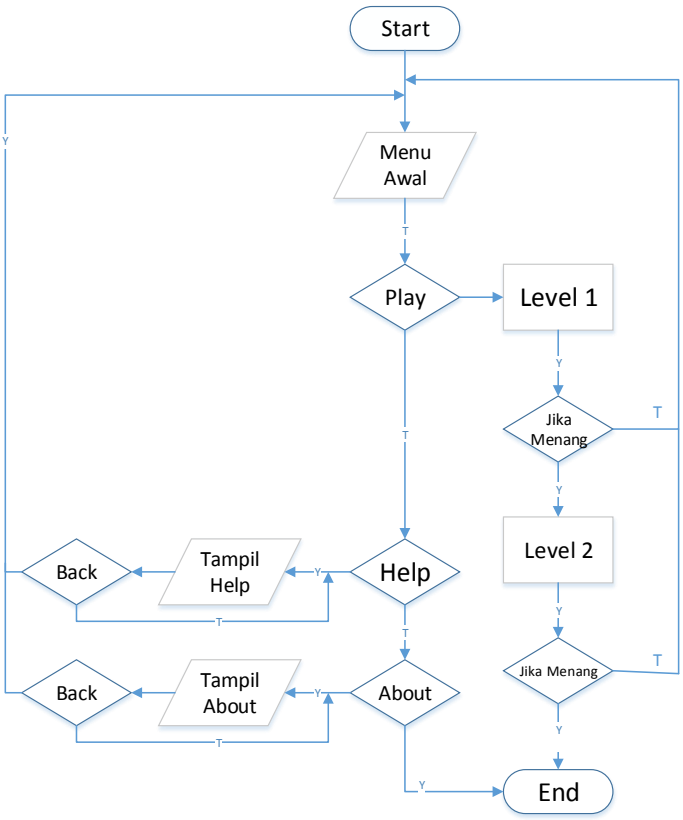
1. *Goal* pada *level* 2 game *Rescue Idol*, pemain harus melewati jebakan dan musuh yang sudah diterapkan kecerdasan buatan *finite state machine* dengan serangan jarak dekat atau jarak jauh yang ada pada level 2 untuk menuju ke bos.
2. Pada saat menghadapi bos pemain harus mampu mengalahkan bos karena, disanalah sang idola disandra yang harus diselamatkan.
3. Jika pemain selesai mengalahkan bos maka permainan telah usai dan berhasil menyelamatkan idolanya tersebut.
4. Di level 2 terdapat *item* untuk menambah *health point* dan juga terdapat koin untuk menambah *score*.
5. *Player* menyerang dengan serangan dekat menggunakan pedang.



Gambar 3.2 Level 2

3.2.3 *Flowchart Game*

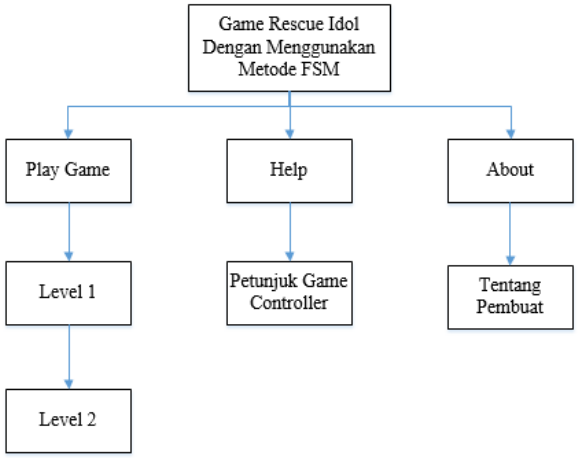
Pada tahap perancangan diagram alir game berfungsi untuk mengetahui alur proses dari program dimulai dari *start* program hingga *end* program. Program dimulai dari *star*t kemudian masuk ke 3 menu utama yaitu *Play* , *Help*, dan *About*. Jika permainan memulai *Play* maka langsung saja menuju *level* 1, jika selesai maka melanjutkan ke *level* 2. Jika permainan memilih *Help* maka akan muncul bantuan dalam bermain. Jika permainan memilih *About* maka akan muncul tampilan profil pembuat game. Flowchart game seperti Gambar 3.3.



Gambar 3.3 *Flowchart Game*

3.2.4 Perancangan Struktur Menu

Pada *Game Rescue Idol* Dengan Menggunakan Metode FSM terdiri dari beberapa menu, yaitu *Play*, *Help*, dan *About*. Diagram struktur menu seperti gambar 3.4

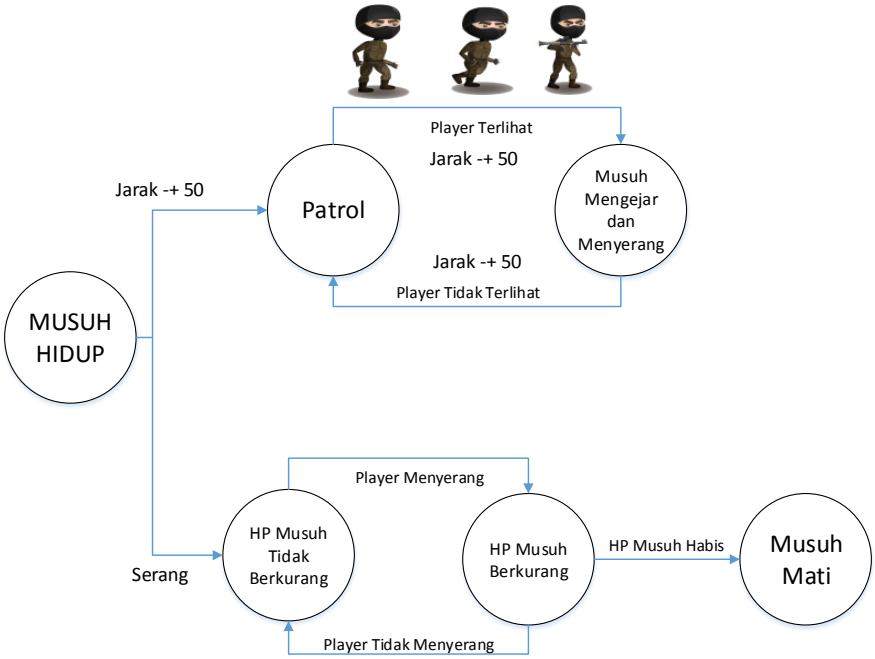


Gambar 3.4 Struktur Menu

Gambar 3.4 menjelaskan tentang struktur menu pada *game* "Rescue Idol" yang berisi *menu* "*play game"* menuju ke *level* 1 lalu ke *level* 2, "*help"*menuju tampilan petunjuk *game controller* , dan *"about"* menuju tampilan tentang pembuat *game*.

3.2.5 Perancangan Metode FSM (*finite state machine*)

Finite state machine merupakan salah satu logika penalaran yang memperlihatkan prilaku system dengan berdasarkan tiga hal, yaitu state (keadaan), event (kejadian), dan action (aksi). Pada suatu saat, system akan berada pada salah satu state yang aktif. Sistem dapat beralih atau bertransisi menuju state lain jika mendapatkan masukan atau event tertentu. Transisi keadaan ini umumnya juga disertai oleh aksi yang dilakukan oleh *system* ketika menanggapi masukan yang terjadi. Pada game *Rescue Idol* ini metode *Finite State Machine* diterapkan pada karakter musuh. Alur kecerdasan buatan *finite state machine* seperti gambar 3.5.

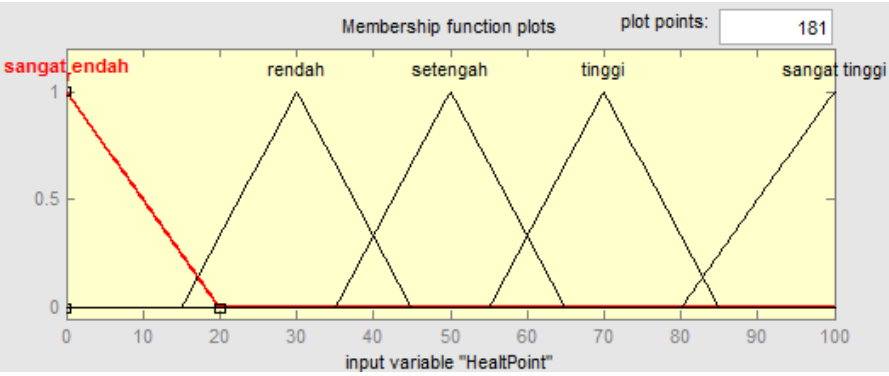


Gambar 3.5 Alur *Finite State Machine* Pada Musuh

3.2.7 *Fuzzy Logic* untuk tindakan karakter bos

Dalam *game* ini *fuzzy logic* digunakan untuk menentukan tindakan yang akan dilakukan oleh bos terhadap *player*. Terdapat 2 masukan yaitu *health point player* dan juga jarak *player* ke bos musuh.

Untuk grafik fungsi keanggotaan *health point* Fungsi keanggotaan pada *health point* dibagi menjadi 5 inputan yaitu sangat rendah, rendah, sedang, tinggi, sangat tinggi seperti Gambar 3.6.



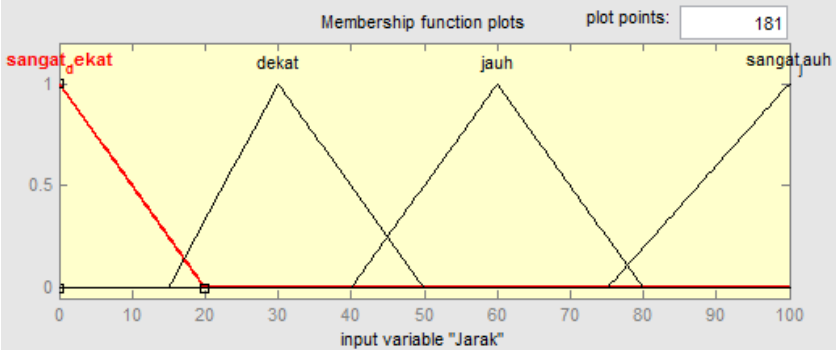
Gambar 3.6 Grafik Fungsi Keanggotaan *Health Point*

Tabel 3.1 Fungsi Keanggotaan *Healht Point*

Fungsi Keanggotaan	Domain	Parameter
Sangat Rendah	0 – 20	[0,20]
Rendah	15 – 45	[15,30,45]
Sedang	35 – 65	[35,50,65]
Tinggi	55 – 85	[55,70,85]
Sangat Tinggi	80 - 100	[80,100]

Berdasarkan tabel 3.1 menunjukkan keterangan fungsi keanggotaan *health point* dengan 5 fungsi keanggotaan yaitu sangat rendah dengan domain (0-20) dengan parameter (0,20), rendah dengan domain (15-45) dengan parameter (15,30,45), sedang dengan domain (35-65) dengan parameter (35,50,65), tinggi dengan domain (55-85) dengan parameter (55,70,85), dan sangat tinggi dengan domain (80-100) dengan parameter (80,100).

Untuk fungsi keanggotaan jarak dibagi menjadi 4 *fuzzy input* yaitu sangat dekat,dekat, jauh, dan sangat jauh, grafik keanggotaan dari jarak seperti Gambar 3.7



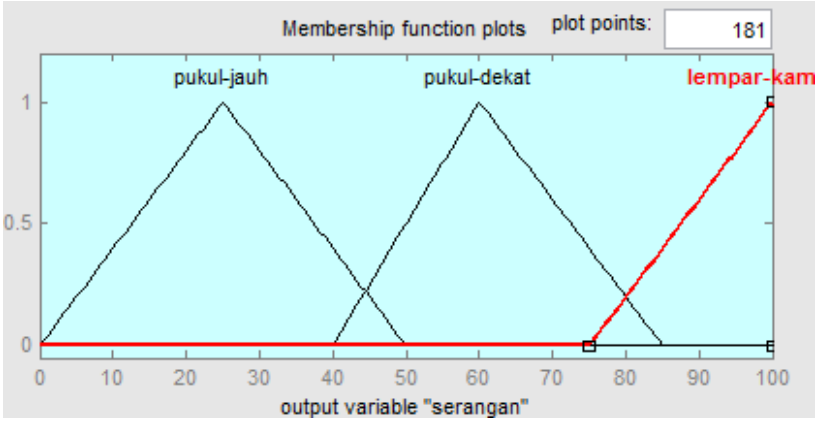
Gambar 3.7 Grafik Fungsi Keanggotaan Jarak *Player*

Tabel 3.2 Fungsi Keanggotaan Jarak

Fungsi Keanggotaan	Domain	Parameter
Sangat Dekat	0 – 20	[0, 20]
Dekat	15 – 50	[15, 30, 50]
Jauh	40 – 80	[40, 60, 80]
Sangat Jauh	75 - 100	[75,100]

Berdasarkan tabel 3.2 menunjukkan keterangan fungsi keanggotaan jarak dengan 4 fungsi keanggotaan yaitu sangat dekat dengan domain (0-20) dengan parameter (0,20), dekat dengan domain (15-50) dengan parameter (15,30,50), jauh dengan domain (40-80) dengan parameter (40,60,80) dan sangat jauh dengan domain (75-100) dengan parameter (75,100).

Untuk grafik *membership function output* aksi dibagi menjadi 3 yaitu pukul jauh, pukul dekat, dan tembak jauh, grafik fungsi keanggotaan untuk *output* aksi seperti Gambar 3.8.



Gambar 3.8 Grafik Fungsi Keanggotaan Output Aksi

Tabel 3.3 Fungsi Keanggotaan Aksi

Fungsi Keanggotaan	Domain	Parameter
Pukul Jauh	0 – 50	[10, 25, 50]
Pukul Dekat	40 – 85	[40, 60, 85]
Lempar Jauh	75 - 100	[75, 100]

Berdasarkan tabel 3.3 menunjukkan keterangan fungsi keanggotaan aksi dengan 3 fungsi keanggotaan yaitu pukul dekat dengan domain (0-50) dengan parameter (10,25,50), pukul jauh dengan domain (40-85) dengan parameter (40,60,85), dan lempar jauh dengan domain (75-100) dengan parameter (75,100).

Fuzzy Rule yaitu :

1. **JIKA** health_point Sangat Rendah **DAN** Jarak Sangat Dekat **MAKA** Pukul Dekat
2. **JIKA** health_point Sangat Rendah **DAN** Jarak Dekat **MAKA** Pukul Dekat
3. **JIKA** health_point Sangat Rendah **DAN** Jarak Jauh **MAKA** Pukul Dekat
4. **JIKA** health_point Sangat Rendah **DAN** Jarak Sangat Jauh **MAKA** Pukul Dekat
5. **JIKA** health_point Rendah **DAN** Jarak Sangat Dekat **MAKA** Pukul Jauh
6. **JIKA** health_point Rendah **DAN** Jarak Dekat **MAKA** Pukul Jauh
7. **JIKA** health_point Rendah **DAN** Jarak Jauh **MAKA** Lempar Jauh
8. **JIKA** health_point Rendah **DAN** Jarak Sangat Jauh **MAKA** Lempar Jauh
9. **JIKA** health_point Setengah **DAN** Jarak Sangat Dekat **MAKA** Pukul Dekat

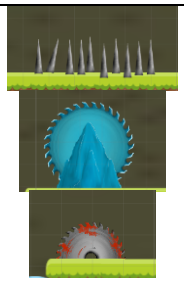
10. **JIKA** health_point Setengah **DAN** Jarak Dekat **MAKA** Pukul Dekat
11. **JIKA** health_point Setengah **DAN** Jarak Jauh **MAKA** Lempar Jauh
12. **JIKA** health_point Setengah **DAN** Jarak Sangat Jauh **MAKA** Lempar Jauh
13. **JIKA** health_point Tinggi **DAN** Jarak Sangat Dekat **MAKA** Pukul Dekat
14. **JIKA** health_point Tinggi **DAN** Jarak Dekat **MAKA** Pukul Dekat
15. **JIKA** health_point Tinggi **DAN** Jarak Jauh **MAKA** Lempar Jauh
16. **JIKA** health_point Tinggi **DAN** Jarak Sangat Jauh **MAKA** Lempar Jauh
17. **JIKA** health_point Sangat Tinggi **DAN** Jarak Sangat Dekat **MAKA** Pukul Dekat
18. **JIKA** health_point Sangat Tinggi **DAN** Jarak Dekat **MAKA** Pukul Dekat
19. **JIKA** health_point Sangat Tinggi **DAN** Jarak Jauh **MAKA** Tembak Jauh
20. **JIKA** health_point Sangat Tinggi **DAN** Jarak Sangat Jauh **MAKA** Tembak Jauh

3.2.7 Perancangan Karakter

Perancangan karakter merupakan pembahasan mengenai karakter yang terlibat dalam *Game Rescue Idol*. Karakter pada *game Rescue Idol* pada Table 3.4.

Tabel 3.4 Keterangan Karakter

No	Gambar	Keterangan
1.		Player
2.		Musuh pada *level* 1 dan 2 dengan *Health Point* 30
3.		Musuh penjahat pada *level* 2 dan bos pada level 1 dengan *Health Point* 50
4.		Bos penjahat pada *level* terakhir yaitu di *level* 2 dengan *Health Point* 100

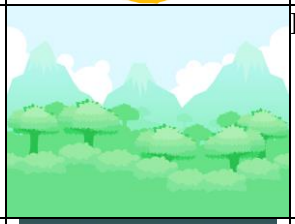
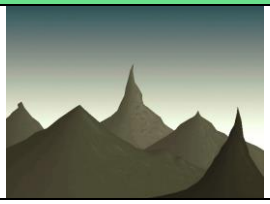
5.		Idola yang diselamatkan
6.		Jebakan pada level 1 dan 2

Berdasarkan tabel 3.4 menunjukkan keterangan karakter, musuh pada level 1 dan 2, bos pada level 2 dengan *health point* 100, idola yang diselamatkan, dan jebakan pada level 1 dan 2.

3.2.8 Perancangan Item dan Background

Perancangan *item* dan *background* merupakan pembahasan mengenai *item* dan *background* yang terlibat dalam *Game Rescue Idol*. Item dan *background* pada *game Rescue Idol* pada Tabel 3.5.

Tabel 3.5 Keterangan *Item* dan *Background*

No	Gambar	Keterangan
1		Bisa menambah *health point* 10
2		Bisa menambah koin
3		Background level 1
4		Background level 2

Berdasarkan tabel 3.5 menunjukkan keterangan *item* dan *background* yang bisa menambah *health point*, koin, dan juga *background* di *level* 1 dan 2.

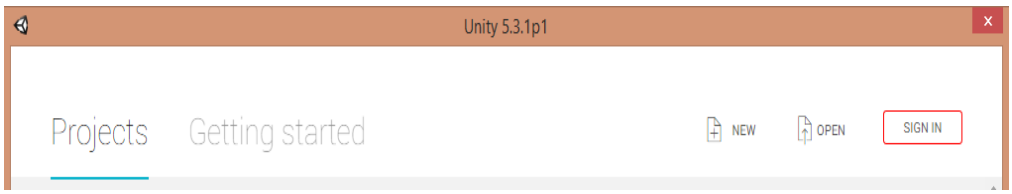
BAB IV
IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi *Game*

4.1.1 Langkah-Langkah Membuat *Project* Baru di Unity3D

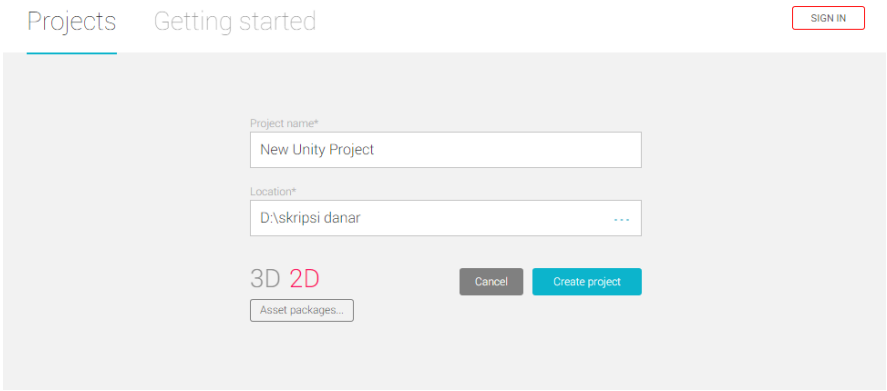
Sebelum membuat program pada Unity3D, hal yang harus dilakukan terlebih dahulu adalah membuat *project* baru pada Unity3D. Berikut adalah langkah-langkah pembuatan *project* baru :

1. Install terlebih dahulu aplikasi Unity3D
2. Membuka aplikasi Unity3D
3. Membuat projek baru di Unity3D dengan cara mengklik "*New*", beri nama projek, pilih "*Create Project*".



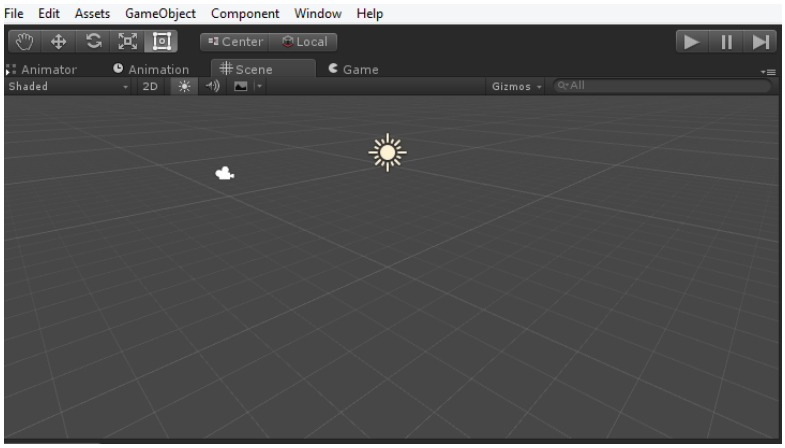
Gambar 4.1 Memulai Membuat Projek

Pada gambar 4.1 penjelasan tentang akan membuat projek baru di Unity3D



Gambar 4.2 *Create Project*

Pada gambar 4.2 penjelasan akan membuat projek baru dengan menentukan nama *file*, disimpan di *folder*, dan memilih "2D".

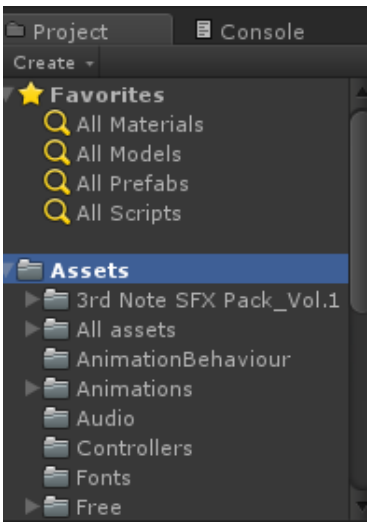


Gambar 4.3 Tampilan Unity3D

Pada gambar 4.3 ini merupakan tampilan pada Unity3D versi 5.3

4.1.2 Membuat *Folder* di Unity3D

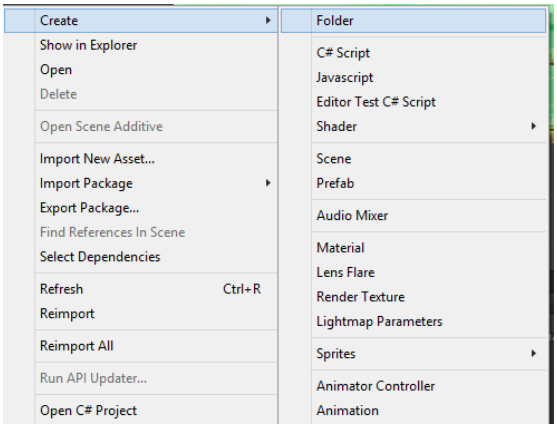
1. Memilih ”*Assets*”.



Gambar 4.4 *Assets*

Pada gambar 4.4 merupakan tampilan assets yang ada di Unity3D 5.3

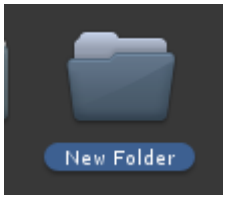
2. Mengklik kanan lalu pilih “*folder”*.



Gambar 4.5 Membuat *Folder*

Pada gambar 4.5 menjelaskan tentang cara membuat *folder* di Unity3D dengan cara mengklik "*create*" lalu memilih "*folder*".

3. Maka otomatis akan terbuat *folder* baru.

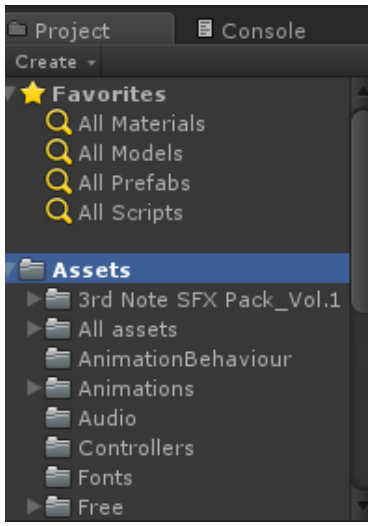


Gambar 4.6 *Folder* Baru

Pada gambar 4.6 merupakan hasil dari pembuatan folder di Unity3D 5.3.

4.1.3 Membuat *Script* di Unity3D

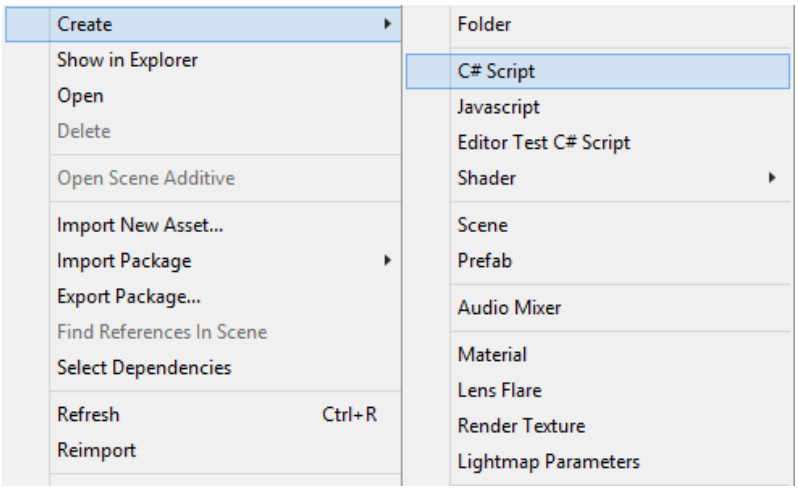
1. Memilih *Assets*



Gambar 4.7 *Assets*

Pada gambar 4.7 merupakan tampilan assets yang ada di Unity3D 5.3

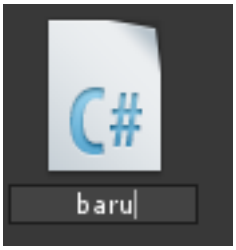
2. Mengklik kanan lalu pilih C# Script



Gambar 4.8 Membuat *Script*

Pada gambar 4.8 menjelaskan tentang cara membuat C# script dengan cara mengklik kanan pada assets, memilih "*create",* memilih "*C# Script".*

3. Maka otomatis akan terbuat *script* baru

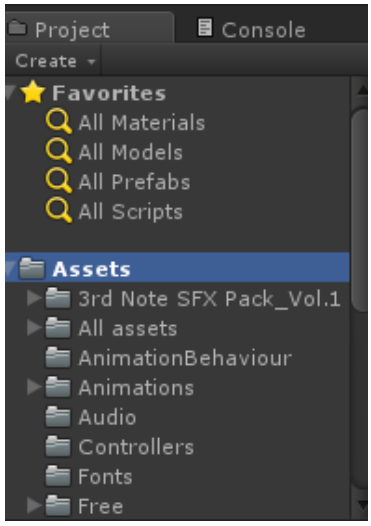


Gambar 4.9 *Script* Baru

Pada gambar 4.9 merupakan hasil dari pembuatan C# Script

4.1.4 Membuat *Scene* di Unity3D

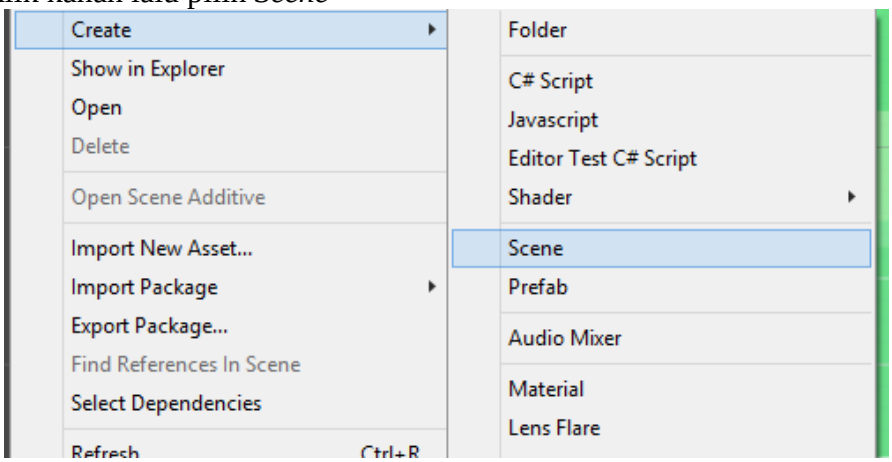
1. Memilih *assets*



Gambar 4.10 *Assets*

Pada gambar 4.10 merupakan tampilan assets yang ada di Unity3D 5.3

2. Mengklik kanan lalu pilih *Scene*



Gambar 4.11 Membuat *Scene*

Pada gambar 4.11 menjelaskan cara membuat *Scene* baru di Unity3D 5.3 dengan cara mengklik kanan, memilih c*reate*, memilih *scene.*

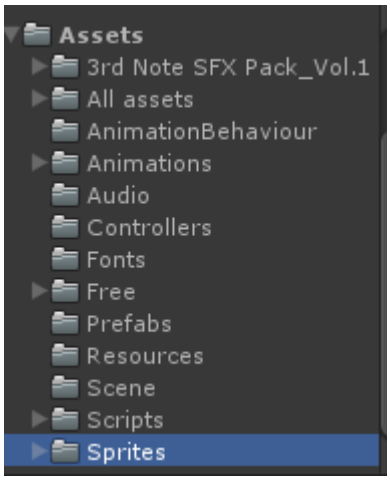
3. Maka otomatis akan terbuat *scene* baru

Gambar 4.12 *Scene* Baru

Pada gambar 4.12 merupakan tampilan hasil dari pembuatan *Scene* baru di Unity3D 5.3. *Scene* itu sendiri tempat untuk membuat halaman projek baru di Unity3D.

4.1.5 Membuat *Animation* dan *Controller* di Unity3D

1. Memilih *folder sprites*



Gambar 4.13 *Folder Sprites*

Pada gambar 4.13 menjelaskan awal untuk membuat animasi pada *player* dengan cara memilih *folder* yang bernama *Sprites.*

2. Memilih *folder Character*

Gambar 4.14 *Folder Characters*

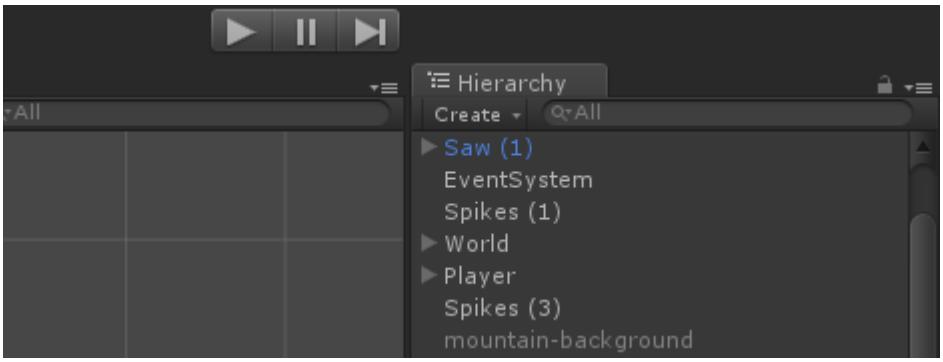
Pada gambar 4.14 menjelaskan untuk memilih animasi pada *player* yang sudah tersimpan di *folder Characters.*

3. Memilih *sprites* mana yang akan digunakan

Gambar 4.15 Memilih *Sprites*

Pada gambar 4.15 menjelaskan memilih *sprites* yang akan digunakan dalam membuat animasi di *player*. *Sprites* merupakan suatu gambar yang nantinya akan menjadi suatu animasi jika sudah dimasukkan ke dalam *scene.*

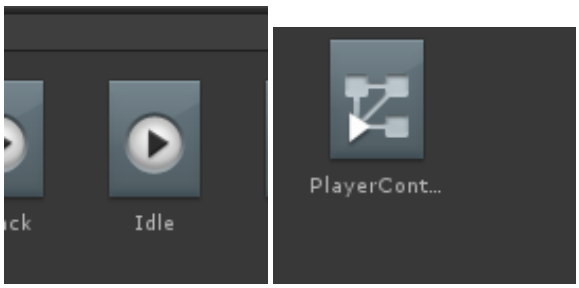
4. Mendrag sprites tersebut ke tab *menu Hierarchy.*



Gambar 4.16 Mendrag *Sprites*

Pada gambar 4.16 menjelaskan cara untuk menggeser *sprites* tersebut ke dalam *Hierarchy*. *Hierarchy* merupakan tab menu yang berisi objek-objek apa saja yang dibuat di dalam *scene*.

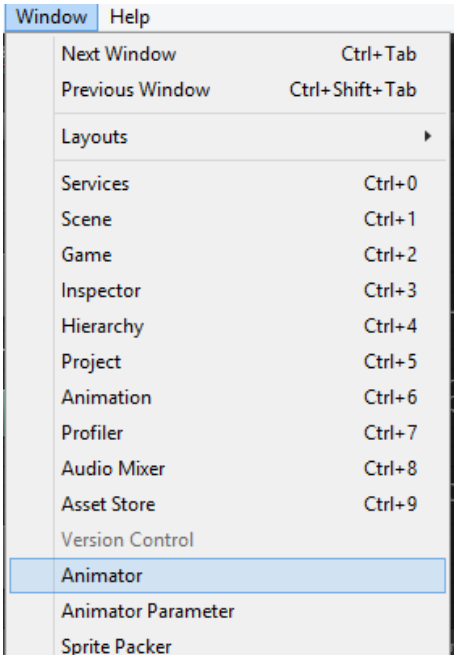
5. Maka otomatis akan terbuat "*Animation*" dan "*Controller"* baru



Gambar 4.17 *Animation* dan *Controller* Baru

Pada gambar 4.17 merupakan hasil dari pembuatan animasi dan kontrol pada *player*. *Controller* nantinya akan berupa kontrol pada *player* dan animasi yang nantinya akan menentukan pergerakan *player*.

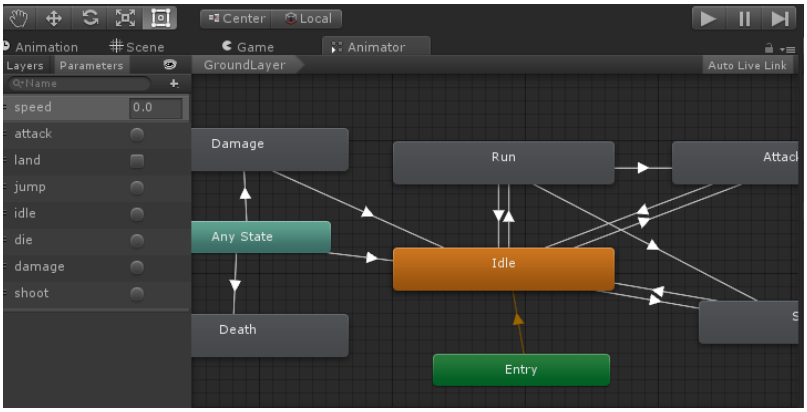
6. Memunculkan tab *Animator* dengan cara mengklik tab *Window*, pilih *Animator*



Gambar 4.18 Memunculkan Tab *Animator*

Pada gambar 4.18 merupakan tampilan untuk memunculkan tab *Animator*. Di tab *Animator* ini bisa digunakan untuk mengedit animasi-animasi yang ada.

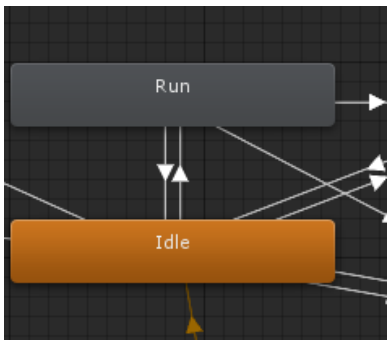
7. Tampilan tab *Animator*



Gambar 4.19 Tab *Animator*

Pada gambar 4.19 merupakan tampilan pada tab *Animator* yang nantinya bisa digunakan untuk mengedit animasi yang telah dibuat.

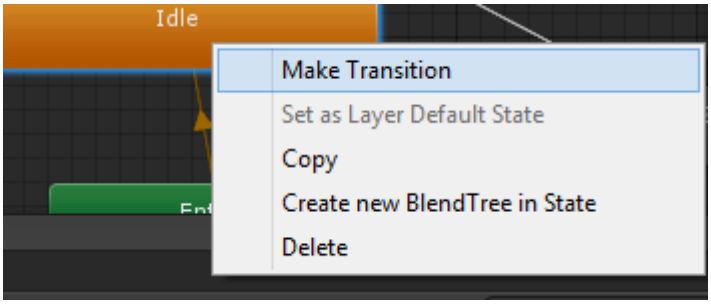
1. Membuka tab *Animator*, pilih *animation* yang akan di atur



Gambar 4.20 Mengatur *Animation*

Pada gambar 4.20 merupakan tampilan untuk mengatur animasi yang sudah dibuat lalu memilih animasi mana yang akan diatur.

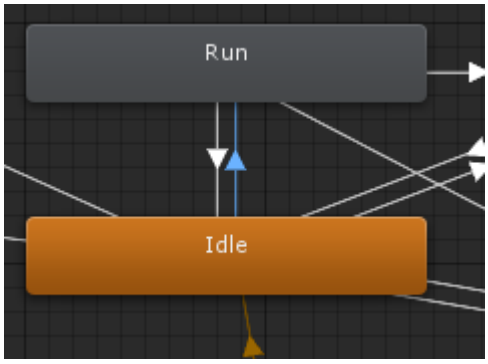
2. Membuat transisi ke animasi yang lain dengan cara mengklik kanan pada animasi "*idle*", lalu memilih "*Make Transition.*



Gambar 4.21 Membuat Transisi

Pada gambar 4.21 merupakan cara untuk menghubungkan animasi satu ke lainnya agar nantinya dapat membentuk suatu animasi yang baru.

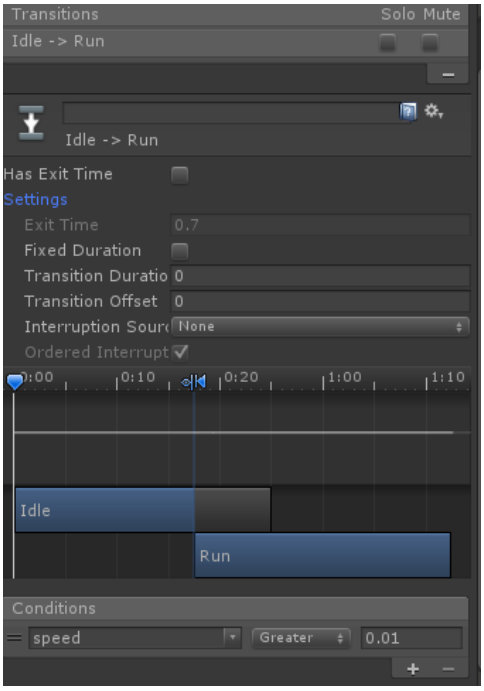
3. Arahkan ke animasi yang lain



Gambar 4.22 Mengarahkan Ke Animasi Lain

Pada gambar 4.22 tampilan arah dari animasi "*idle"* ke animasi "*run"*.

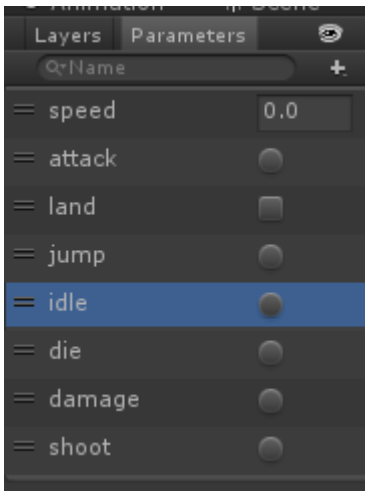
4. Mengatur transisi



Gambar 4.23 Mengatur Transisi

Pada gambar 4.23 merupakan penjelasan untuk mengatur transisi dari animasi "*idle"* ke "*run"* dengan menentukan durasi transisinya dan juga kondisinya.

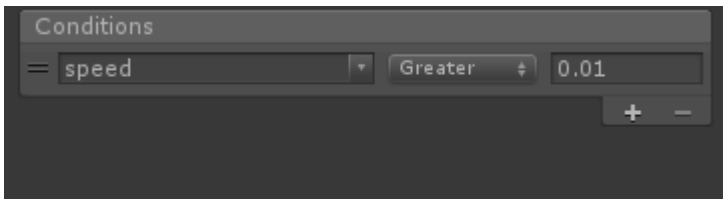
5. Membuat Parameter pada animasi agar nantinya dapat dipanggil di *script* dengan mengklik parameter, mengklik " + ", lalu memilih tipe paramater apa yang akan digunakan.



Gambar 4.24 Membuat Parameter

Pada gambar 4.24 menjelaskan membuat parameter di "*animator"* yang nantinya parameter tersebut di letakkan di dalam mengatur transisi dan dapat dijalan di *script.*

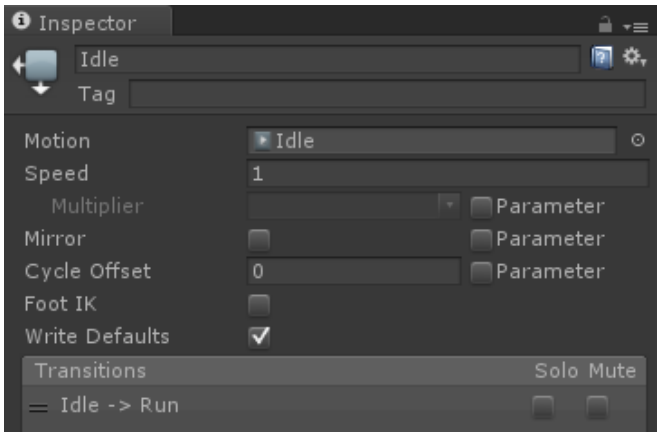
6. Jika sudah maka langsung memasukkan parameternya di kondisi pada transisi



Gambar 4.25 Menambahkan Parameter Di Kondisi

Pada gambar 4.25 merupakan hasil dari menambahkan parameter yang sudah dibuat lalu di letakkan di dalam kondisi di *setting* transisi.

7. Hasil dari transisi

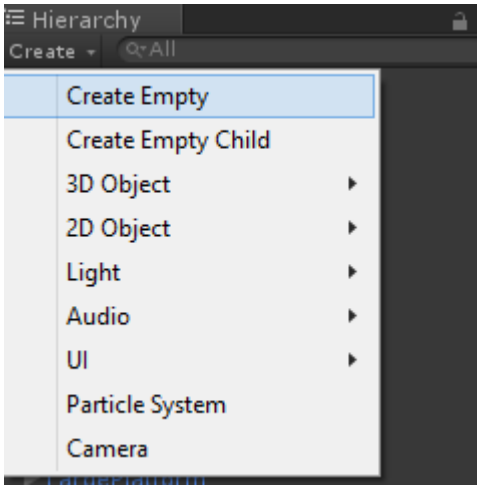


Gambar 4.26 Hasil Transisi

Pada gambar 4.26 merupakan tampilan dari hasil transisi dari *"idle"* ke "*run"*.

4.1.7 Membuat *Game Object* "*World"* di Unity3D

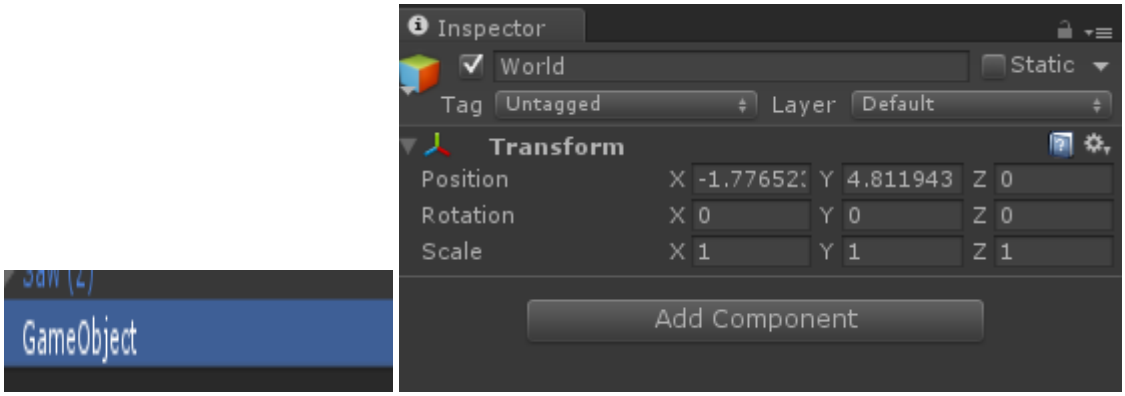
1. Mengklik *Create*, pilih *Create Empty*



Gambar 4.27 Membuat *Game Object*

Pada gambar 4.27 menjelaskan cara untuk membuat *game object* yang bernama "*world"*. *Game object* tersebut nantinya akan digunakan untuk membuat suatu object benda yang nantinya akan digunakan di *scene.*

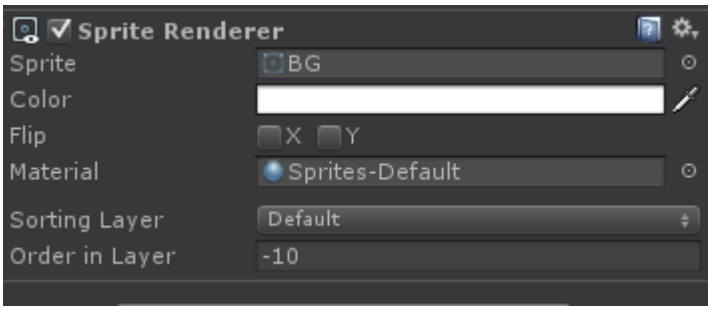
2. Tampilan *game object.* Disini pembuat menamakan *game object* dengan nama "*World*". Didalam *"World"* ini terdapat beberapa *game object* lain seperti *ground,background* dll.



Gambar 4.28 *Game Object "World"*

Pada gambar 4.28 merupakan hasil dari pembuatan objek dengan pengaturan tata letak yang sudah ditentukan.

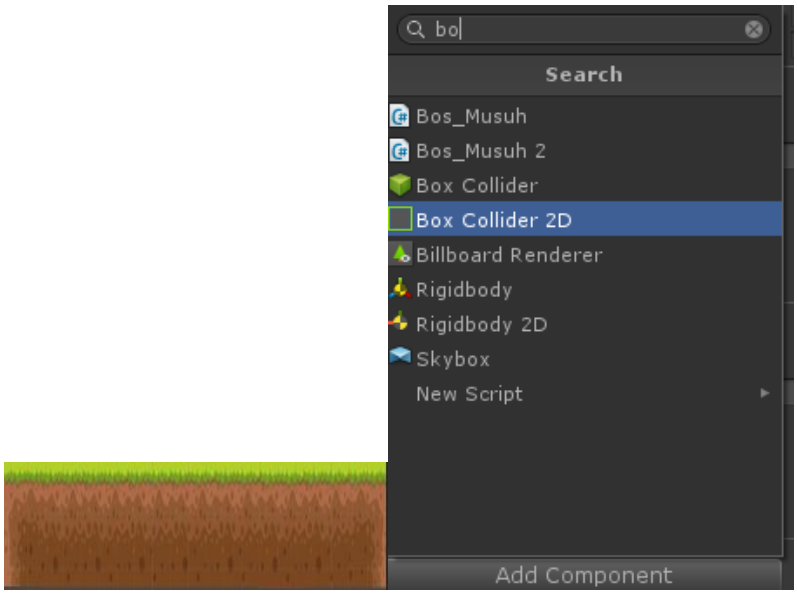
3. Memasukkan *background* dengan cara mendrag ke "*World"* lalu pilih "*Order In Layer"* agar *background* terlihat di *scene.*



Gambar 4.29 Mengatur *Background*

Pada gambar 4.29 merupakan pengaturan dari *game object* yang bernama *background* agar dapat muncul di dalam *scene.*

4. Memasukkan *Ground* atau alas dengan cara mendrag ke "*World"* lalu menambahkan komponen bernama "*Box Collider 2D*".

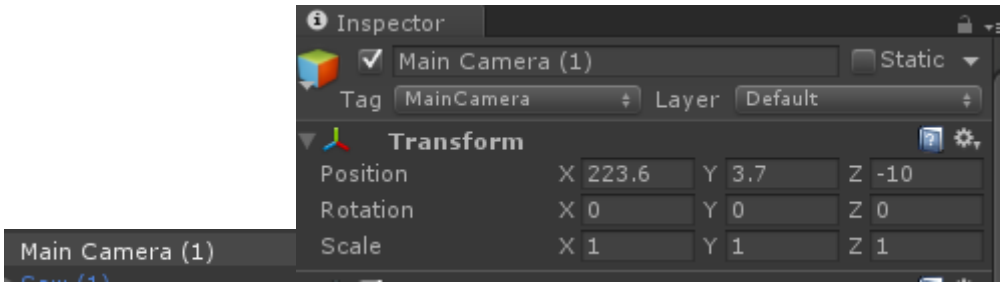


Gambar 4.30 Menambahkan *Box Collider 2D*

Pada gambar 4.30 merupakan cara untuk menambahkan "*box collider 2D"* pada objek yang sudah dibuat agar nantinya *player* dapat berjalan di alas atau *ground.*

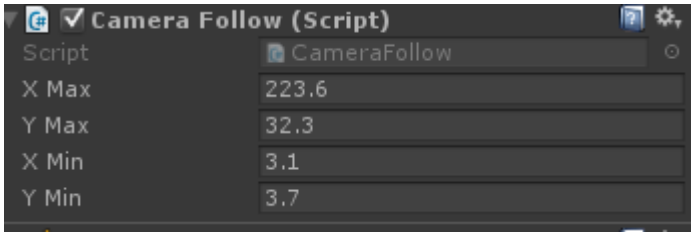
4.1.8 Mengatur Kamera di Unity3D

1. Mengklik "*Main Camera",* menentukan posisi sumbu x dan sumbu y lalu di terapkan pada *script camera follow.*



Gambar 4.31 Mengatur Kamera

Pada gambar 4.31 merupakan tampilan untuk menentukan posisis kamera agar dapat mengikuti pergerakan *player.*

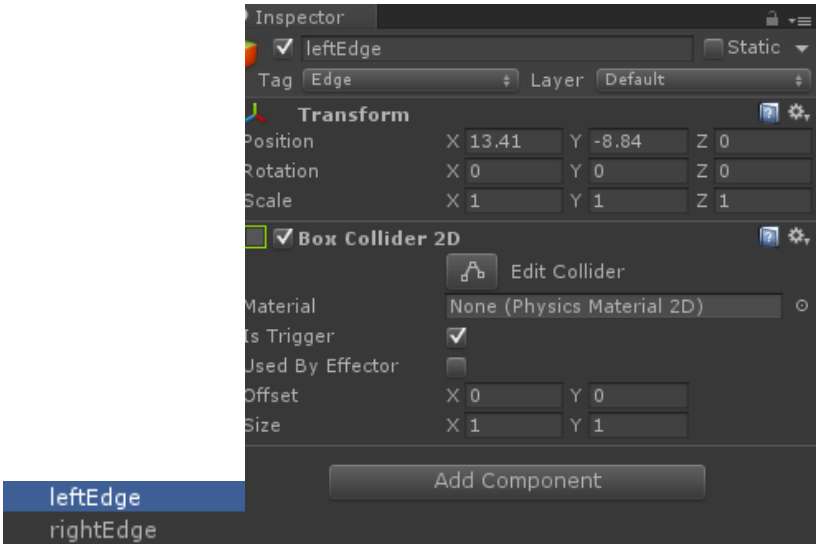


Gambar 4.32 Diterapkan Di *Script*

Pada gambar 4.32 merupakan tampilan dari *script* yang ada di kamera agar dapat berjalan di kamera utama dengan menentukan sumbu X dan Y nya berdasarkan pengaturan dari kameranya.

4.1.9 Mengatur Patrol Musuh dan Target ke *Player* di Unity3D

1. Membuat *game object* dengan nama "*left edge"* dan "*right edge"* dengan ditambahkan komponen berupa *box collider 2D.*



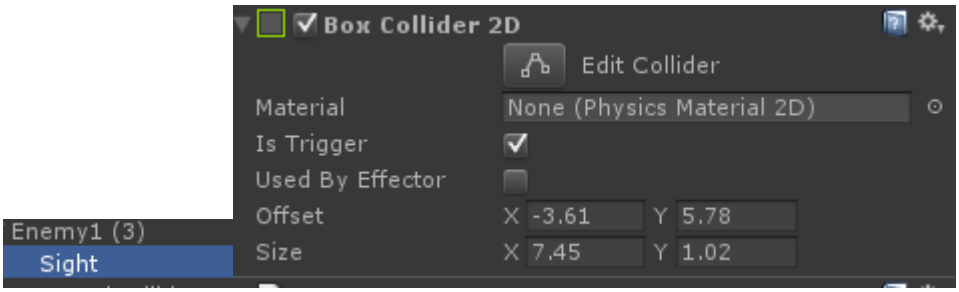
Gambar 4.33 *Edge Collider*

Pada gambar 4.33 merupakan pengaturan dari *game object* yang diberi nama *"left edge"* dan "*right edge"*

Gambar 4.34 Tampilan *Edge Collider*

Pada gambar 4.34 merupakan hasil dari pengaturan "*left edge"* dan "*right edge"* agar musuh dapat melalukan patrol di dalam daerah yang sudah ditentukan.

2. Membuat *game object* dengan nama "*sight"* dan ditambahkan komponen *box collider 2D* dan juga diatur sedemikian rupa.



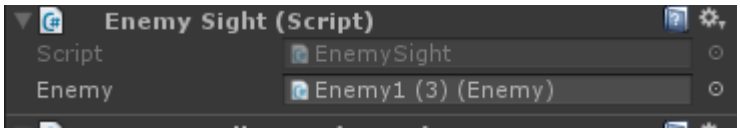
Gambar 4.35 *Setting Di Box Collider 2D*

Pada gambar 4.35 merupakan pengaturan untuk menentukan jarak serang musuh ke *player* dengan objek bernama "*sight"* dan juga diberikan box collider 2D.

Gambar 4.36 Tampilan *Sight*

Pada gambar 4.36 merupakan hasil dari pengaturan sight yang telah dibuat dengan pengaturan *box collider 2D* yang sudah ditentukan.

3. Selanjutnya menerapkan di *script "Enemy Sight".*

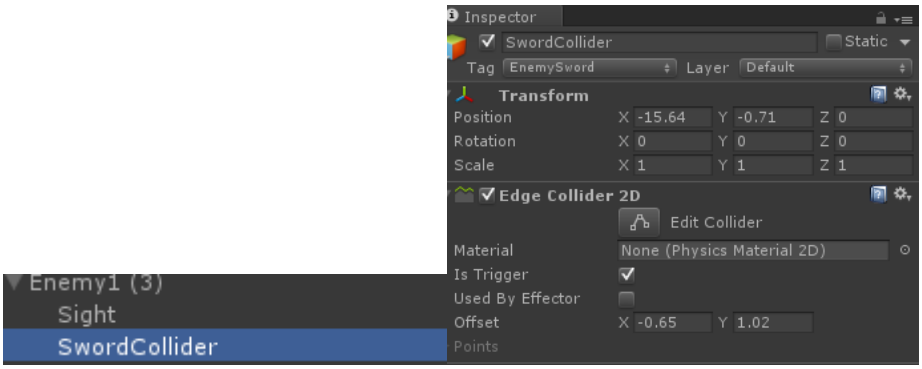


Gambar 4.37 Tampilan *Script Sight*

Pada gambar 4.37 merupakan *script* untuk mengatur pola jarak serang musuh berdasarkan *game object* yang sudah ditentukan dengan *box collider 2D.*

4.1.10 Mengatur *Player Attack* dan *Enemy Attack* di Unity3D

1. Membuat *game object* dengan nama "*SwordCollider"* dengan ditambahkan komponen "*Edge Collider 2D"*

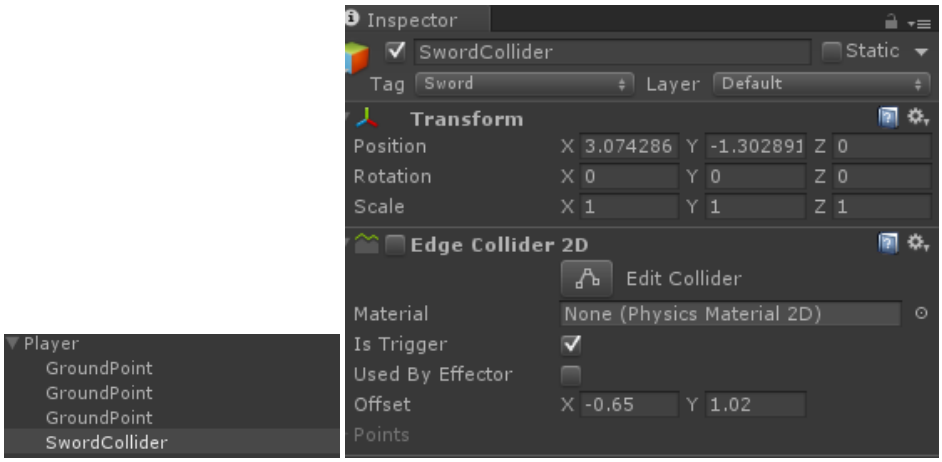


Gambar 4.38 *Edge Collider* Pada "*SwordCollider" Enemy*

Pada gambar 4.38 merupakan pengaturan dari "*Edge Collider 2D"* yang diletakkan di *game object* dengan nama "*SwordCollider". SwordCollider* ini digunakan untuk melakukan serangan musuh ataupun *player* dengan jarak dekat. *Edge Collider 2D* fungsinya sama dengan *Box Collider 2D.*

Gambar 4.39 *Tampilan* "*SwordCollider"* Pada *Enemy* Dan *Player*

Pada gambar 4.39 merupakan hasil dari pembuatan "*SwordCollider"* dengan di dalamnya sudah ada "*Edge Collider 2D"* yang sudah diatur.



Gambar 4.40 *Edge Collider* Pada "*SwordCollider" Player*

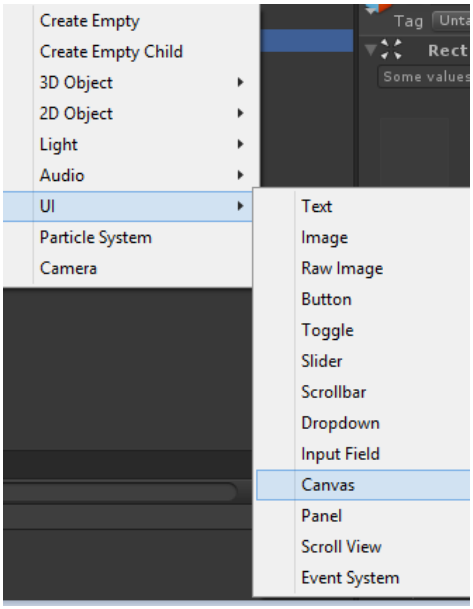
Pada gambar 4.40 merupakan pengaturan dari "*Edge Collider 2D"* yang diletakkan di *game object* dengan nama "*SwordCollider"*. *SwordCollider* ini digunakan untuk melakukan serangan musuh ataupun *player* dengan jarak dekat. *Edge Collider 2D* fungsinya sama dengan *Box Collider 2D.*

2. Selanjutnya pada "*SwordCollider" player* akan di buat tag yang bernama "*Sword".*

3. Lalu menentukan animasi pada *player* di tab *Animation*, pilih "*Attack".*

4.1.11 Membuat Menu Utama di Unity3D

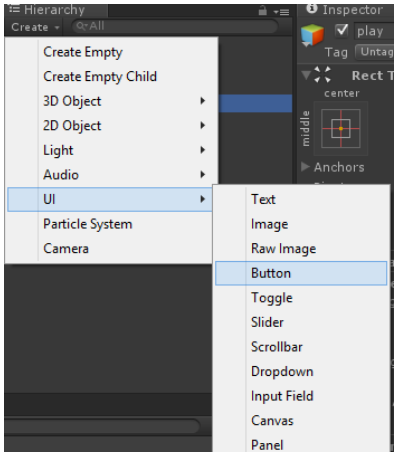
1. Membuat *canvas* dengan cara *create*, pilih UI, pilih *Canvas*



Gambar 4.41 Membuat *Canvas*

Pada gambar 4.41 merupakan cara untuk membuat suatu "*Canvas" di* Unity3D. *Canvas* bisa disebut dengan halaman untuk membuat tampilan UI yang biasanya bukan untuk objek benda yang bergerak.

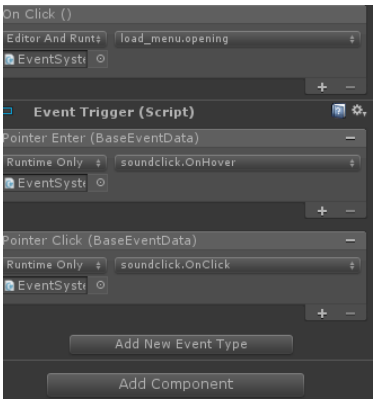
2. Membuat 3 *button* yaitu "*Play*", "*Help*", "*About*" dengan mengklik *create*, pilih UI, pilih *Canvas*.



Gambar 4.42 Membuat Button

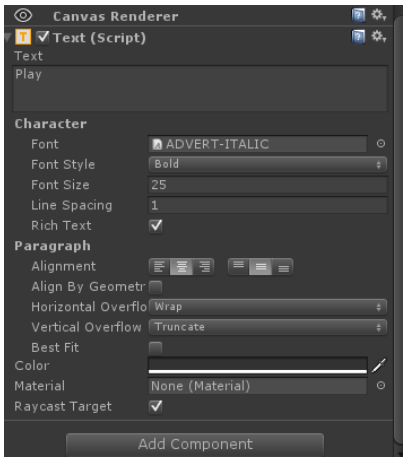
Pada gambar 4.44 merupakan cara untuk membuat suatu "*Button"* atau tombol pada Unity3D untuk *Menu* utama pada projek *game. Button* digunakan untuk dapat mengarahkan satu *scene* di *menu* ke *scene* lain yang ada di menu, misalkan jika mengklik *button* "*Help",* maka akan menuju *scene Help.*

3. Membuat event trigger dan membuat text yang nantinya di letakkan di dalam ketiga button



Gambar 4.43 Event Trigger

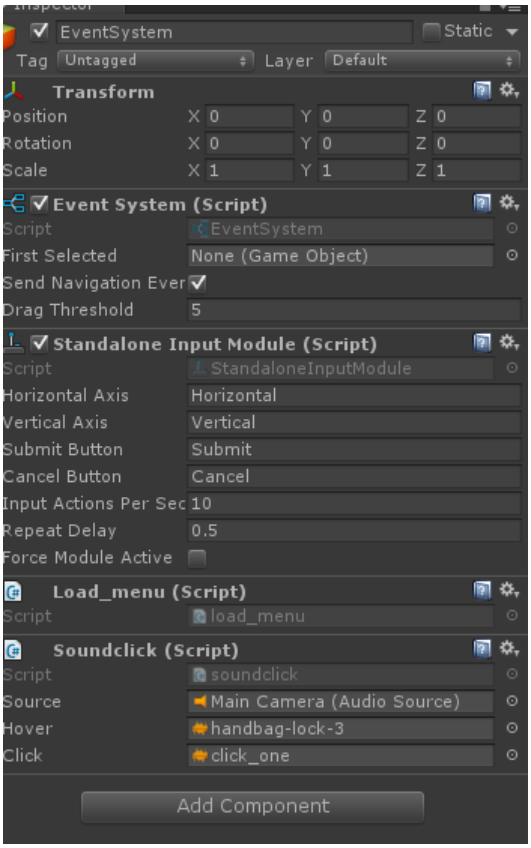
Pada gambar 4.43 merupakan "*Event Trigger"* pada *Canvas Menu* utama. Fungsi dari *script event trigger* adalah memanggil *scene* lain yang ada di *menu* utama yang diletakkan di *button.*



Gambar 4.44 Text

Pada gambar 4.44 merupakan tampilan *text* untuk membuat *text* di objek pada *menu.*

4. Meletakkan *script load menu* dan *sound click* di *Event System*



Gambar 4.45 Event System

Pada gambar 4.45 merupakan tampilan *"Event System"* pada *scene menu* yang nantinya digunakan untuk tempat memasukkan *script "load menu" Script "load menu"* ini yang digunakan untuk memanggil *scene* lain.

4.2 Pengujian *Gameplay*

Pengujian *gameplay* adalah pengujian bagaimana *game* tersebut berjalan sesuai dengan rancangan *sistem* yang telah dibuat.

4.2.1 *Menu* Utama *Game*

Pada Gambar 4.46 adalah desain GUI menu utama dari *game* "*Rescue Idol*". Menu utama dari *game* ini dapat diakses oleh *user* ketika permainan akan dimulai. Pada menu utama ini *user* dapat memilih menu yang diinginkan. Jika *user* memilih menu *play* maka *user* akan memulai permainan, sebelum *user* memulai permainan, disarankan terlebih dahulu untuk memilih menu *help* yang berisi petunjuk permainan, jadi *user* tidak akan kesulitan dalam memainkan *game* ini.

Gambar 4.46 *Menu* Utama

4.2.2 *Menu Help*

Pada bagian ini merupakan menu agar *user* dapat mengetahui fungsi tombol apa saja yang akan digunakan pada *Game* Rescue Idol seperti Gambar 4.47.

Gambar 4.47 *Menu Help*

4.2.3 *Menu About*

Pada bagian ini merupakan menu untuk mengetahui profil pembuat *Game* Rescue Idol dimana didalamnya terdapat foto profil pembuat, nama pembuat, jurusan, dan universitas pembuat game, dan angkatan, tampilan dari menu about seperti Gambar 4.48

Gambar 4.48 *Menu About*

4.2.4 *Menu Game Over*

Pada bagian ini adalah tampilan yang muncul ketika *player* mati pada suatu level.Jika mengklik *Retry* maka otomatis akaan menuju *level* 1 dan jika mengklik *exit* maka akan kembali ke *menu* seperti gambar 4.49.

Gambar 4.49 *Game Over*

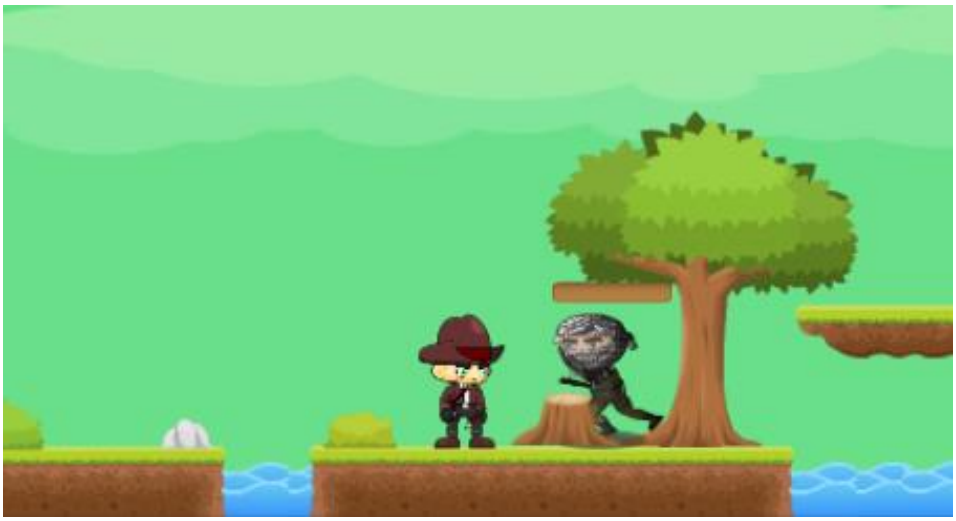
4.2.5 *Menu Next Level*

Pada gambar 4.50 merupakan *next level* agar *user* dapat memilih apakah *user* memilih *menu* atau *next level.* Jika *menu,* maka akan kembali ke *menu* utama. Jika *next level* maka akan menuju ke *level* 2.

Gambar 4.50 *Next Level*

4.2.6 *Enemy* Patrol ke *Player*

Pada gambar 4.51 adalah penerapan interaksi antara karakter dengan musuh dimana musuh melihat keberadaan *player* dan melakukan *output* serangan jarak dekat, jika dari hasil serangan musuh mengenai *player* maka *health bar player* akan berkurang.



Gambar 4.51 Musuh Melakukan Patrol Dan Menyerang *Player*

Pada gambar 4.52 adalah penerapan interaksi antara karakter dengan musuh dimana musuh melihat keberadaan *player* dan melakukan output serangan jarak dekat atau jarak jauh, jika dari hasil serangan musuh mengenai *player* maka *health bar player* akan berkurang.

Gambar 4.52 Musuh Melakukan Patrol Dan Menyerang *Player*

4.2.7 *Enemy Attack* ke *Player*

Pada gambar 4.53 adalah penerapan interaksi antara karakter dengan musuh dimana musuh melihat keberadaan player yang sangat dekat posisinya dan melakukan output serangan pukul dekat, jika hasil serangan musuh mengenai player maka health bar player akan berkurang.

Gambar 4.53 Musuh Menyerang *Player* Dengan Serangan Dekat

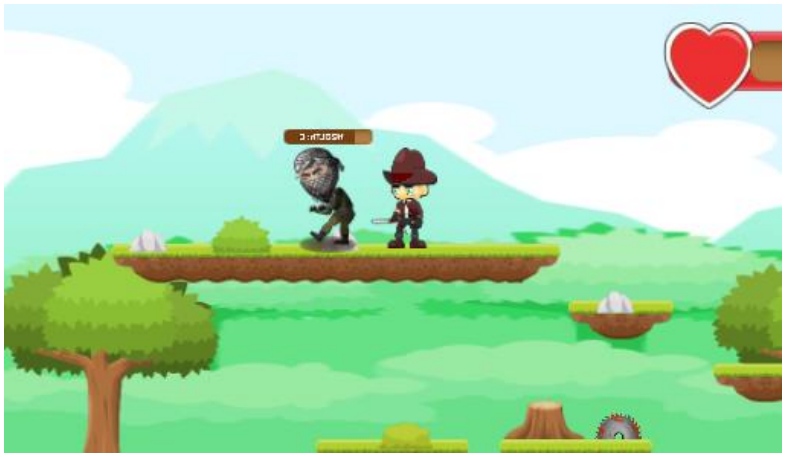
Pada gambar 4.54 adalah penerapan interaksi antara karakter dengan musuh dimana musuh melihat keberadaan player yang dekat posisinya dan melakukan output

serangan tembak jauh, jika hasil serangan musuh mengenai player maka health bar player akan berkurang.

Gambar 4.54 Musuh Menyerang *Player* Dengan Serangan Jauh

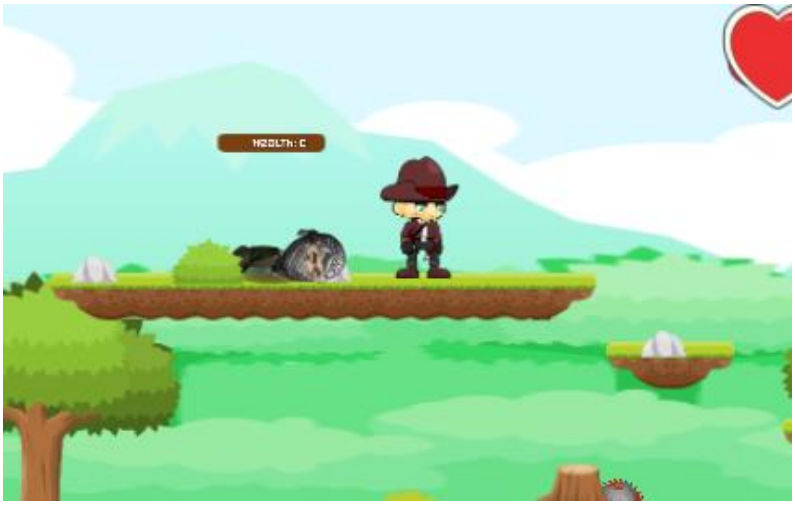
4.2.8 Penerapan *Player Attack* ke *Enemy*

Pada gambar 4.55 adalah penerapan ketika *player* menyerang musuh dengan pedang, jika serangan *player* mengenai musuh, maka *health bar* musuh akan berkurang.



Gambar 4.55 Karakter *Player* Menyerang Musuh Dengan Pedang

Pada gambar 4.56 adalah penerapan ketika karakter *player* menyerang musuh dan *health bar* musuh habis maka musuh akan mati, dalam jangka waktu 3 detik karakter musuh akan hilang.



Gambar 4.56 Musuh Mati

4.2.9 Penerapan *Damage* dari Jebakan ke *Player*

Pada Gambar 4.57 adalah penerapan *damage* ketika darah *player* berkurang karena terkena serangan dari jebakan yang terdapat pada setiap *level.*

Gambar 4.57 Darah *Player* Berkurang Terkena Jebakan

4.2.10 Penerapan *Attack* Bos ke *Player*

Pada Gambar 4.58 adalah penerapan interaksi antara *player* dengan bos dimana bos mempunyai beberapa aksi serangan untuk *player* dimana hasil aksi serangan bos terhadap *player* merupakan yakni dari kecerdasan buatan *fuzzy logic* yang diterapkan pada bos.

Gambar 4.58 Interaksi *Player* Dengan Bos

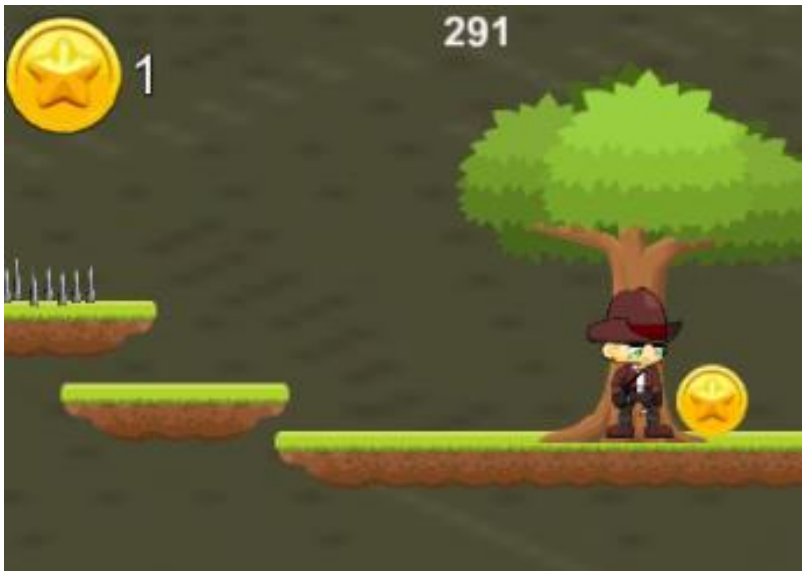
4.2.11 Penerapan Item Tambah Nyawa

Pada Gambar 4.59 penerapan ketika darah *player* bertambah ketika menyentuh *item* yang berfungsi menambah *hp*, yang terdapat pada setiap *level.*

Gambar 4.59 *Player* Menyentuh *Item Hp*

4.2.12 Penerapan Item Koin

Pada Gambar 4.60 penerapan ketika *player* menyentuh *item* berupa koin yang berfungsi untuk menambah koin nominal 1 yang terdapat pada setiap *level.*



Gambar 4.60 *Player* Menyentuh *Item* Koin

4.3 Pengujian

Pada tahapan pengujian penulis menampilkan hasil dari pengujian yang telah dilakukan, dimana pengujian tersebut meliputi pengujian fungsional sistem, apakah semua fungsi dalam *game* "Rescue Idol" telah berfungsi semua atau belum.

4.3.1 Pengujian FSM (*Finite State Machine*) Pada Musuh

Pada bagian ini merupakan hasil pengujian metode *finite state machine* pada musuh telah berjalan dengan baik, dimana jika *player* terdeteksi oleh pengelihatan musuh maka musuh akan mengejar lalu akan memberikan *damage* pada *player*. Tampilan pengujian *finite state machine* pada musuh dapat dilihat pada tabel 4.1

Tabel 4.1 Pengujian *Artificial Intelligence*

NO	Keadaan	Kejadian	Tindakan	Hasil
1	Player Tidak Terlihat	Jika player tidak masuk daerah musuh	Maka musuh akan tetap pada posisi diam atau patrol	Sesuai
	Player Terlihat	Jika player masuk pada daerah musuh	Maka musuh akan mengejar dan menyerang player	Sesuai

	Player Tidak Menyerang	Jika player tidak masuk daerah musuh dan tidak menyerang	Maka musuh akan tetap pada posisi diam atau patrol dan HP musuh tidak berkurang	Sesuai
	Player Menyerang	Jika player berada pada daerah musuh	Maka musuh akan menyerang dengan serangan dekat atau jauh	Sesuai
	HP Musuh Habis	Jika player menyerang musuh dan HP musuh habis	Maka musuh akan mati	Sesuai

Berdasarkan dari tabel 4.1,dapat diambil kesimpulan bahwa kecerdasan buatan yang diterapkan pada musuh sudah berhasil seluruhnya.

4.3.2 Pengujian Menu Utama

Pengujian *menu* utama adalah pengujian setiap fungsi dari tombol yang sudah diterapkan di *menu* utama. Hasil pengujian *menu* pada tabel 4.2

Tabel 4.2 Pengujian *Menu* Utama

No	Tombol	Fungsi	Hasil
1	Play	Memulai permainan	Sesuai
2	Help	Menampilkan kontrol pada player	Sesuai
3	About	Menampilkan profil pembuat game	Sesuai

Berdasarkan pengujian tabel 4.2,dapat disimpulkan bahwa semua fungsi tombol pada *menu* utama sudah berhasil seluruhnya.

4.3.3 Pengujian Kontrol Player

Pengujian *control player* adalah pengujian setiap fungsi dari tombol yang sudah diterapkan untuk menggerakan karakter utama. Hasil pengujian *player* pada tabel 4.3

Tabel 4.3 Pengujian *Control Player*

No	Tombol	Fungsi	Hasil
1	←	Menggerakkan ke kiri	Sesuai
2	→	Menggerakkan ke kanan	Sesuai
3	**S**	*Player* meloncat	Sesuai
4	**D**	*Player* melakukan aksi berupa serangan (*attack)*	Sesuai

Berdasarkan pengujian tabel 4.3,dapat disimpulkan bahwa semua fungsi tombol yang diterapkan untuk melakukan aksi pada *player* sudah sesuai.

4.3.4 Pengujian Laptop

Pada fungsionalitas ini game dicoba dijalankan pada Windows. Hasil dari pengujian dapat dilihat pada Tabel 4.4.

Tabel 4.4 Pengujian Laptop.

No	Tipe Laptop	Spesifikasi			Berhasil	Gagal
		RAM	Processors	Harddisk		
1	Acer Aspire E5-53 G	8 GB	AMD A-12	1 TB	√	-
2	Asus A455LF	10 GB	Intel Core I-3	1 TB	√	-
3	Lenovo G-400 S	4 GB	Intel Core I-3	500 GB	√	-

Pada hasil pengujian beberapa model dan tipe laptop diatas didapat kesimpulan sesuai dan tidak ada kendala saat di mainkan di berbagai laptop dengan spesifikasi laptop yang sesuai dengan *minimum requirement* yang sudah ditetapkan.

4.3.5 Pengujian Fungsionalitas

Pada fungsionalitas ini game dicoba dijalankan pada Windows. Hasil dari pengujian dapat dilihat pada tabel 4.5.

Tabel 4.5 Pengujian Fungsional.

No	Item Uji	Windows 8		Windows 10	
		Berhasil	Gagal	Berhasil	Gagal
1	Menu Utama	√	-	√	-
2	*Button Start*	√	-	√	-
3	*Button Help*	√	-	√	-
4	*Button About*	√	-	√	-
7	Masuk Level 1 Game	√	-	√	-
8	Masuk Level 2 Game	√	-	√	-
9	Bar Health Point karakter	√	-	√	-
10	Bar Health Point Musuh	√	-	√	-
11	Loncat dan animation state loncat pada karakter	√	-	√	-
12	Attack dan animation state attack pada karakter	√	-	√	-
13	HP musuh berkurang ketika terkena attack	√	-	√	-
14	Collision Detection pada Musuh	√	-	√	-

NO	Item Uji	Windows 8		Windows 10	
		Berhasil	Gagal	Berhasil	Gagal
15	Item tambah HP pada player	√	-	√	-
16	Item Koin	√	-	√	-
17	Musuh Patroli pada daerah yang ditentukan	√	-	√	-
18	Musuh mengikuti pada daerah yang ditentukan	√	-	√	-
19	Musuh menyerang pada jarak yang ditentukan	√	-	√	-
20	Game tamat ketika player menyentuh Idola	√	-	√	-
21	Background Sound	√	-	√	-
22	Sound Effect	√	-	√	-

Berdasarkan pengujian tabel 4.5 dapat disimpulkan bahwa pengujian fungsionalitas di *Windows* 8 dan 10 berhasil dijalankan sesuai dengan yang diuji.

4.3.6 Pengujian User

Pengujian dilakukan untuk mengetahui apakah sistem sudah berjalan dengan baik atau belum. Pengujian dilakukan terhadap 20 orang. Hasil dari pertanyaan terhadap responden dapat dilihat pada Tabel 4.6.

Tabel 4.6 Pengujian Terhadap User

No	Pertanyaan	Penilaian		
		Baik	Cukup	Kurang
1	Desain Karakter Game	9	7	4
2	Desain Animasi pada Game	9	6	5
3	Kontrol pada game	9	10	1
4	Fitur game	10	6	4
5	Informasi game (Cerita, Narasi Game)	5	11	4

No	Pertanyaan	Penilaian		
		Baik	Cukup	Kurang
6	Game sudah menarik	9	8	3

Berdasarkan tabel 4.6 menunjukkan bahwa 42% mengatakan baik, 40% mengatakan cukup, dan 18% mengatakan kurang.

4.3.7 Pengujian Cheat

Pengujian dilakukan untuk mengetahui apakah cheat sudah berjalan dengan baik atau belum. Pengujian dilakukan dengan mencoba shortcut cheat di semua Stage. Hasil dari pengujian dapat dilihat pada Tabel 4.7.

Tabel 4.7 Pengujian Fungsi Cheat

No	Tombol Shortcut	Fungsi	Penjelasan	Ket
1	1	Menuju ke Level 1	Tombol angka 3 pada *keyboard* jika tekan akan otomatis menuju ke level 1	Sesuai
2	2	Menuju ke Level 2	Tombol angka 2 pada *keyboard* jika tekan akan otomatis menuju ke level 2	Sesuai

BAB V
PENUTUP

5.1 Kesimpulan

Setelah pembuatan *Game Rescue Idol*, maka penulis dapat mengambil kesimpulan :

1. Berdasarkan tabel 4.1 implementasi FSM (*Finite State Machine*) dapat diterapkan pada *game* ”Rescue Idol” dengan indikasi musuh dapat mengejar dan menyerang *player* dengan kondisi tertentu.
2. Berdasarkan tabel 4.2 fungsi tombol pada *menu* utama bisa berjalan seperti mengklik *menu* “*play*” akan memulai permainan,mengklik *menu* ”*about*” akan menuju tampilan pembuat, dan mengklik *menu “help”* akan menuju tampilan *controll player*.
3. Berdasarkan tabel 4.3 fungsi tombol yang diterapkan di *player* dapat berjalan dengan cara menekan tombol kiri (←), kanan (→), S (meloncat), dan D (serang).
4. Berdasarkan tabel 4.4 pengujian *game* bisa berjalan di 3 laptop dengan spesifikasi RAM 4 GB – 10 GB, Processor Intel dan AMD, dan Harddisk 500 GB dan 1 TB.
5. Berdasarkan tabel 4.5 pengujian fungsionalitas pada *Windows* 8 dan 10 bisa berjalan dengan baik sesuai dengan yang diuji.
6. Berdasarkan tabel 4.6 pengujian dari 20 *user* diketahui bahwa 42% mengatakan, 40% mengatakan cukup, dan 18% mengatakan kurang.
7. Berdasarkan tabel 4.7 pengujian *cheat* pada *game* bisa berjalan jika mengklik tombol 2 akan menuju *level* 2 dan jika mengklik tombol 3 akan menuju *level* 1.

5.2 Saran

Setelah dilakukan pengujian terhadap *Game Rescue Idol* maka masih ada kekurangan sehingga untuk pengembangan lebih lanjut disarankan :

1. Dapat dikembangkan sebagai mobile game berbasis *platform android* atau ios.
2. *Fuzzy logic* hanya dapat diimplementasikan pada karakter bos, selanjutnya *fuzzy logic* dapat diimplementasikan pada semua karakter game.

DAFTAR PUSTAKA

Ardi, R., 2012. Pembuatan Game First Person Shooter (FPS)"Operation Zygma Force" X10. Yogyakarta : STIMIK AMIKOM.

Dahria, M., 2008. Kecerdasan Buatan Jurnal SAINTIKOM, p.185.

Hasanah, S.N dan Widiastuti, N.I, 2014. Representasi Emosi Menggunakan Fuzzy Pada Permainan Bonny's Tooth Booth.

Ilman, A., 2008. Penerapan Algoritma Minimax Dengan Optimasi MTD Pada Permainan Catur. Bandung : Insitut Teknologi Bandung.

Setiawan, I., 2016. Perancangan Software Embedded System Berbasis FSM Online: http.elektro.undip.ac.id/iwan/perancangan, p.20.

Septian, F.R., 2014. Implementasi Fuzzy Logic Metode Mamdani Untuk Pengembangan Intelligent Non-Player Character pada Game Strategy.

Sholichin, R, Mohamad Yasindan Lucky, Dkk., 2013. Implementasi Algoritma Dijkstra Dalam Pencarian Lintasan Terpendek Lokasi Rumah Sakit, Hotel dan Terminal Kota Malang Berbasis Web. Malang : Universitas Negeri Malang.

Satriyanta, Bonaventura Putra., 2012. Analisis dan Pembuatan Game Ular Tangga Interaktif untuk Pembelajaran Matematika dengan adobe flash Cs3. Yogyakarta : Amikom.

LAMPIRAN

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
PROGRAM STUDI TEKNIK INFORMATIKA S-1
Jl. Karanglo, Km. 2 Malang

BERITA ACARA UJIAN SKRIPSI
FAKULTAS TEKNOLOGI INDUSTRI

NAMA : Danar Pradiktia Yunanta
NIM : 13.18.047
JURUSAN : Teknik Informatika S-1
JUDUL : Game Rescue Idol Dengan Menggunakan Metode Finite State Machine (FSM)

Dipertahankan dihadapan Majelis Penguji Skripsi Jenjang Strata Satu (S-1) pada :
Hari : Sabtu
Tanggal : 5 Agustus2017
Nilai : 79 (B+)

Panitia Ujian Skripsi :

Ketua Majelis Penguji

Joseph Dedy Irawan, ST, MT
NIP. 197404162005011002

Anggota Penguji :

Dosen Penguji I	**Dosen Penguji II**
Sandy Nataly Mantja, S.Kom	**Moh. Miftakhur Rokhman, S.Kom.M.Kom**
NIP.P. 1030800418	**NIP.P. 1031500479**

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
PROGRAM STUDI TEKNIK INFORMATIKA S-1
Jl. Karanglo, Km. 2 Malang

FORMULIR PERBAIKAN SKRIPSI

Dalam pelaksanaan ujian skripsi jenjang Strata 1 Program Studi Teknik Informatika, maka perlu adanya perbaikan skripsi untuk mahasiswa :

NAMA : Danar Pradiktia Yunanta
NIM : 13.18.047
JURUSAN : Teknik Informatika S-1
JUDUL : Game Rescue Idol Dengan Menggunakan Metode Finite State Machine (FSM)

No	Penguji	Tanggal	Uraian	Paraf
1.	Penguji I	5 Agustus 2017	1. Hasil pengujian ke 20 user diletakkan di BAB IV,kesimpulan dan abstrak 2. Sama dengan azmi 3. Revisi tanggal 21-25 agustus 2017	
2.	Penguji II	5 Agustus 2017	1. Perbaiki timing menggunakan progression 2. Perbaiki senjata bos yang muncul terlebih dahulu 3. Tambahkan alur cerita pada pembuka/awal 4. Input score dan waktu tersimpan dimunculkan di akhir	

Dosen Penguji I

Sandy Nataly Mantja, S.Kom
NIP.P. 1030800418

Dosen Penguji II

Moh. Miftakhur Rokhman, S.Kom.M.Kom
NIP.P. 1031500479

Dosen Pembimbing I

Suryo Adi Wibowo, ST, MT.
NIP. P.1031000438

Dosen Pembimbing II

Febriana Santi Wahyuni, S.Kom, M.Kom
NIP. P. 1031000425

INSTITUT TEKNOLOGI NASIONAL MALANG
Fakultas Teknologi Industri
Program Studi Teknik Informatika S1

FORMULIR BIMBINGAN SKRIPSI

Nama : Danar Pradiktia Yunanta
NIM : 1318047
Masa Bimbingan : 28 April 2017 S/D 2 Agustus 2017
Judul Skripsi : GAME RESCUE IDOL DENGAN MENGGUNAKAN METODE FINITE STATE MACHINE (FSM)

No.	TANGGAL	URAIAN	PARAF PEMBIMBING
1.	08-05-2017	Kumpulkan background,karakter, komponen-komponen pendukung yang lain	
2.	10-05-2017	Kembangkan dengan metode Fuzzy Logic	
3.	12-05-2017	ACC Seminar Progres	
4.	11-07-2017	Revisi Makalah	
5.	12-07-2017	ACC Semhas	
6.	31-07-2017	Revisi Bab I + II lanjutkan Bab III + IV	
7.	01-08-2017	ACC Bab I – II , Revisi Bab III + IV	
8.	02-08-2017	ACC Bab III + IV Lanjutkan Bab V	
9.	03-08-2017	ACC Bab V Revisi Daftar Pustaka+Lampiran	
10.	04-08-2017	ACC Uji Kompre	

Malang, Agustus 2017
Dosen Pembimbing II

Febriana Santi Wahyuni, S.Kom, M.Kom
NIP.P. 1031000425

PERKUMPULAN PENGELOLA PENDIDIKAN UMUM DAN TEKNOLOGI NASIONAL MALANG

INSTITUT TEKNOLOGI NASIONAL MALANG

FAKULTAS TEKNOLOGI INDUSTRI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
PROGRAM PASCASARJANA MAGISTER TEKNIK

PT. BNI (PERSERO) MALANG
BANK NIAGA MALANG

Kampus I : Jl. Bendungan Sigura-gura No. 2 Telp. (0341) 551431 (Hunting), Fax. (0341) 553015 Malang 65145
Kampus II : Jl. Raya Karanglo, Km 2 Telp. (0341) 417636 Fax. (0341) 417634 Malang

Malang, 3 April 2017

Nomor : ITN-924/IV.INF/TA/2017
Lampiran : ---
Perihal : *Bimbingan Skripsi*

Kepada : Yth. Bpk/Ibu Suryo Adi Wibowo,ST.MT
Dosen Pembina Program Studi Teknik Informatika S-1
Institut Teknologi Nasional
Malang

Dengan Hormat,
Sesuai dengan permohonan dan persetujuan dalam proposal skripsi untuk mahasiswa :

Nama : DANAR PRADIKTIA YUNANTA
Nim : 1318047
Prodi : Teknik Informatika S-1
Fakultas : Teknologi Industri

Maka dengan ini pembimbingan kami serahkan sepenuhnya kepada Saudara/i selama waktu 6 (enam) bulan, terhitung mulai tanggal :

3 April 2017 S/D 3 Oktober 2017

Sebagai satu syarat untuk menempuh Ujian Akhir Sarjana Teknik, Program Studi Teknik Informatika S-1.
Demikian agar maklum dan atas perhatian serta bantuannya kami sampaikan terima kasih.

Mengetahui
Program Studi Teknik Informatika S-1
K e t u a,

Joseph Dedy Irawan,ST., MT.
NIP : 197404162005021002

Form S-4a

BAN-PT

PERKUMPULAN PENGELOLA PENDIDIKAN UMUM DAN TEKNOLOGI NASIONAL MALANG

INSTITUT TEKNOLOGI NASIONAL MALANG

FAKULTAS TEKNOLOGI INDUSTRI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
PROGRAM PASCASARJANA MAGISTER TEKNIK

PT. BNI (PERSERO) MALANG
BANK NIAGA MALANG

Kampus I : Jl. Bendungan Sigura-gura No. 2 Telp. (0341) 551431 (Hunting), Fax. (0341) 553015 Malang 65145
Kampus II : Jl. Raya Karanglo, Km 2 Telp. (0341) 417636 Fax. (0341) 417634 Malang

Malang, 3 April 2017

Nomor : ITN-924/IV.INF/TA/2017
Lampiran : ---
Perihal : *Bimbingan Skripsi*

Kepada : Yth. Bpk/Ibu Febriana Santi Wahyuni,S.Kom.M.Kom
Dosen Pembina Program Studi Teknik Informatika S-1
Institut Teknologi Nasional
Malang

Dengan Hormat,
Sesuai dengan permohonan dan persetujuan dalam proposal skripsi untuk mahasiswa :

Nama : DANAR PRADIKTIA YUNANTA
Nim : 1318047
Prodi : Teknik Informatika S-1
Fakultas : Teknologi Industri

Maka dengan ini pembimbingan kami serahkan sepenuhnya kepada Saudara/i selama waktu 6 (enam) bulan, terhitung mulai tanggal :

3 April 2017 S/D 3 Oktober 2017

Sebagai satu syarat untuk menempuh Ujian Akhir Sarjana Teknik, Program Studi Teknik Informatika S-1.
Demikian agar maklum dan atas perhatian serta bantuannya kami sampaikan terima kasih.

Mengetahui
Program Studi Teknik Informatika S-1
K e t u a,

Joseph Dedy Irawan,ST., MT.
NIP : 197404162005021002

Form S-4a

ISO 9001:2008 Certificate No. QU16/0232
BAN-PT

INSTITUT TEKNOLOGI NASIONAL
Jl. Bendungan Sigura-Gura No. 2
MALANG

Malang, 3 April 2017

Lampiran : 1(Satu) berkas
Perihal : Kesediaan sebagai Pembimbing Skripsi

Kepada : Yth. Bpk/Ibu Suryo **Adi Wibowo,ST.MT**
Dosen Pembina Prodi Teknik Informatika S-1
Institut Teknologi Nasional
MALANG

Yang bertanda tangan dibawah ini:

Nama : DANAR PRADIKTIA YUNANTA
Nim : 1318047
Prodi : Teknik Informatika S-1

Dengan ini mengajukan permohonan, kiranya Bapak/Ibu bersedia menjadi Dosen Pembimbing Utama / ~~Pendamping~~ *), untuk penyusunan Skripsi dengan judul (Proposal Terlampir) :

Game Rescue Idol Dengan Menggunakan Metode FSM

Adapun tugas tersebut sebagai salah satu syarat untuk menempuh Ujian Akhir Sarjana Teknik.
Demikian permohonan kami dan atas kesediaan Bapak/Ibu kami sampaikan terima kasih.

Prodi T. Informatika S-1
K e t u a,

Joseph Dedy Irawan,ST., MT.
NIP : 197404162005021002

Hormat Kami,

DANAR PRADIKTIA YUNANTA

Form S-3a

INSTITUT TEKNOLOGI NASIONAL
Jln. Bendungan Sigura-gura No. 2
Jln. Raya Karanglo Km2
M A L A N G

PERNYATAAN KESEDIAAN DALAM PEMBIMBINGAN SKRIPSI

Sesuai permohonan dari mahasiswa/i :

Nama : DANAR PRADIKTIA YUNANTA

Nim : 1318047

Program Studi : Teknik Informatika

Dengan ini menyatakan bersedia / ~~tidak bersedia~~ *) membimbing skripsi dari mahasiswa tersebut dengan judul :

Game Rescue Idol Dengan Menggunakan Metode FSM

Demikian Surat Pernyataan ini kami buat agar dipergunakan seperlunya.

Malang, _____

Hormat Kami,

Suryo Adi Wibowo,ST.MT

Catatan :
Setelah disetujui agar formulir ini diserahkan mahasiswa/i yg bersangkutan kepada Jurusan untuk diproses lebih lanjut
*) coret yang tidak perlu

Form S-3b

INSTITUT TEKNOLOGI NASIONAL
Jl. Bendungan Sigura-Gura No. 2
MALANG

Malang, 3 April 2017

Lampiran : I(Satu) berkas
Perihal : Kesediaan sebagai Pembimbing Skripsi

Kepada : Yth. Bpk/Ibu **Febriana Santi Wahyuni,S.Kom.M.Kom**
Dosen Pembina Prodi Teknik Informatika S-1
Institut Teknologi Nasional
MALANG

Yang bertanda tangan dibawah ini:

Nama : DANAR PRADIKTIA YUNANTA
Nim : 1318047
Prodi : Teknik Informatika S-1

Dengan ini mengajukan permohonan, kiranya Bapak/Ibu bersedia menjadi Dosen Pembimbing ~~Utama~~ / Pendamping *), untuk penyusunan Skripsi dengan judul (Proposal Terlampir) :

Game Rescue Idol Dengan Menggunakan Metode FSM

Adapun tugas tersebut sebagai salah satu syarat untuk menempuh Ujian Akhir Sarjana Teknik.
Demikian permohonan kami dan atas kesediaan Bapak/Ibu kami sampaikan terima kasih.

Prodi T. Informatika S-1
K e t u a,

Joseph Dedy Irawan,ST., MT.
NIP. : 197404 62005021002

Hormat Kami,

DANAR PRADIKTIA YUNANTA

Form S-3a

INSTITUT TEKNOLOGI NASIONAL
Jln. Bendungan Sigura-gura No. 2
Jln. Raya Karanglo Km2
M A L A N G

PERNYATAAN KESEDIAAN DALAM PEMBIMBINGAN SKRIPSI

Sesuai permohonan dari mahasiswa/i :

Nama	:	DANAR PRADIKTIA YUNANTA
Nim	:	1318047
Program Studi	:	Teknik Informatika

Dengan ini menyatakan bersedia / tidak bersedia *) membimbing skripsi dari mahasiswa tersebut dengan judul :

Game Rescue Idol Dengan Menggunakan Metode FSM

Demikian Surat Pernyataan ini kami buat agar dipergunakan seperlunya.

Malang, _____

Hormat Kami,

Febriana Santi Wahyuni,S.Kom.M.Kom

Catatan :
Setelah disetujui agar formulir ini diserahkan mahasiswa/i
yg bersangkutan kepada Jurusan untuk diproses lebih lanjut
*) coret yang tidak perlu

Form S-3b

INSTITUT TEKNOLOGI NASIONAL MALANG
Fakultas Teknologi Industri
Program Studi Teknik Informatika S1

BERITA ACARA SEMINAR JUDUL SKRIPSI
PROGRAM STUDI TEKNIK INFORMATIKA S-1

No				
	Nama Mahasiswa : DANAR PRADIKTIA Y			Nim 1318047
1	Keterangan	Tanggal 12-3-2017	Waktu	Tempat T. INF
2	Pelaksanaan			Ruang LAB RPL
	Spesifikasi Judul (berilah tanda silang)**)			
3	a. Jaringan komputer b. Multimedia c. Basis data d. Pemrograman & RPL X Lainnya..... GAME			
4	Judul proposal yang diseminarkan mahasiswa	GAME RESCUE TOOL DG MENGGUNAKAN METODE FSM		
5	Perubahan judul yang diusulkan oleh kelompok dosen keahlian	.....		
6	Catatan :			
7	Catatan :			
	Persetujuan judul skripsi			
	Disetujui, Dosen keahlian I (.....)	Disetujui, Dosen keahlian II	Disetujui, Dosen keahlian III	
	Mengetahui, Ketua Prodi T.Informatika Joseph Dedy Irawan. ST.MT NIP. 19740416 200501 1 002	Moderator I PEBRIANA	Moderator II	

INSTITUT TEKNOLOGI NASIONAL MALANG
Fakultas Teknologi Industri
Program Studi Teknik Informatika S1

FORMULIR PERBAIKAN SEMINAR JUDUL

Dalam pelaksanaan Seminar Judul Jenjang Strata 1 Jurusan Teknik Informatika, maka perlu adanya perbaikan untuk mahasiswa :

Nama : DOMAR PRAPTIA Y
NIM : 13180047
Perbaikan Meliputi :

Malang, 18-3-2017

(SANDI PATAH)

INSTITUT TEKNOLOGI NASIONAL MALANG
Fakultas Teknologi Industri
Program Studi Teknik Informatika S1

FORMULIR PERBAIKAN UJIAN SKRIPSI

Dalam pelaksanaan Ujian Skripsi Jenjang Strata 1 Jurusan Teknik Informatika, maka perlu adanya perbaikan untuk mahasiswa :

Nama : DAMAR PRAJNYA Y
NIM : 13.18.047
Perbaikan Meliputi :

1. HASIL PENGUJIAN KE 2 DAN DIKETAHUKAN DI BAB IV KESIMPULAN ABSTRAK
2. PENULISAN PENGUJI ALUMI
3. REVISI TGL 24-25 AGT 2017

Hasil pengujian dimasukkan ke abstrak dan kesimpulan

Malang, 7-8-17

( )

INSTITUT TEKNOLOGI NASIONAL MALANG
Fakultas Teknologi Industri
Program Studi Teknik Informatika S1

FORMULIR PERBAIKAN UJIAN SKRIPSI

Dalam pelaksanaan Ujian Skripsi Jenjang Strata 1 Jurusan Teknik Informatika, maka perlu adanya perbaikan untuk mahasiswa :

Nama : Danar Pradiktia Yunanta
NIM : 13 18 047
Perbaikan Meliputi :

- Perbaiki timing menggunakan progress time
- Perbaiki senjata boss yg muncul terlebih dahulu.
- Tambahkan alur cerita pada pembuka/awal. (Storyboard).
- Input nama User, Score dan ~~tim~~ waktu tersimpan, dimunculkan di akhir.

Malang, 5-8-2017

(M. M. Rakhman)

Lampiran 1. Script pemain (player)

```
using UnityEngine;
using System.Collections;
using System;
using UnityEngine.UI;


public delegate void DeadEventHandler();
public class Player : Character
{
    private static Player instance;
    public event DeadEventHandler Dead;
    public static Player Instance
    {
        get
        {
            if (instance == null)
            {
                instance = GameObject.FindObjectOfType<Player>();
            }
            return instance;
        }
    }
    [SerializeField]
    private Transform[] groundPoints;
    [SerializeField]
    private float groundRadius;
    [SerializeField]
    private LayerMask whatIsGround;
    [SerializeField]
    private float jumpForce;
    private bool immortal = false;
    private SpriteRenderer spriteRenderer;
    [SerializeField]
    private float immortalTime;
    [SerializeField]
    private bool airControl;
    [SerializeField]
    int coin1;
    [SerializeField]
    private float nyawa;
      [SerializeField]
    private GameObject peluruPrefab;
    [SerializeField]
    private Transform pelurupost;
    public Rigidbody2D MyRigidbody { get; set;}
    public bool Jump { get; set; }
    public bool OnGround { get; set; }
    private Vector2 startPos;
    public override bool IsDead
    {
        get
        {
            if (healthStat.CurrentVal <= 0)
            {
                OnDead();
            }
```

```
            return healthStat.CurrentVal <= 0;
                    }
                }
 // Use this for initialization
   public override void Start()
    {
        base.Start();
        startPos = transform.position;
        spriteRenderer = GetComponent<SpriteRenderer>();
        MyRigidbody = GetComponent<Rigidbody2D>();
    }
    void Update()
    {
        if (!TakingDamage && !IsDead)
        {
            if (transform.position.y <= -14f)
            {
                Death();
            }
            HandleInput();
        }
    }
    // Update is called once per frame
    void FixedUpdate()
    {
        if (!TakingDamage && !IsDead)
        {
            float horizontal = Input.GetAxis("Horizontal");
            OnGround = IsGrounded();
            HandleMovement(horizontal);
         Flip(horizontal);
            HandleLayers();
            GameOver();
        }
    }
    public void OnDead()
    {
        if (Dead != null)
        {
            Dead();
        }
    }
    private void HandleMovement(float horizontal)
    {
        if (MyRigidbody.velocity.y < 0)
        {
            MyAnimator.SetBool("land", true);
        }
        if (!Attack && (OnGround || airControl))
        {
            MyRigidbody.velocity = new Vector2(horizontal * movementSpeed,
MyRigidbody.velocity.y);
        }
        if (Jump && MyRigidbody.velocity.y == 0)
        {
            MyRigidbody.AddForce(new Vector2(0, jumpForce));
        }
        MyAnimator.SetFloat("speed", Mathf.Abs(horizontal));
```

```
}
    private void HandleInput()
    {
        if (Input.GetKeyDown(KeyCode.S))
        {
            soundScript.PlaySound("Jump");
            MyAnimator.SetTrigger("jump");
        }
        if (Input.GetKeyDown(KeyCode.D))
        {
            soundScript.PlaySound("Att");
            MyAnimator.SetTrigger("attack");
        }
        if (Input.GetKeyDown(KeyCode.F))
        {
            soundScript.PlaySound("PlayerHit");
            MyAnimator.SetTrigger("shoot");
            Peluru(0);
        }
        // script cheat
        if (Input.GetKeyDown(KeyCode.Alpha2))
        {
            Application.LoadLevel("Stage3");
        }
        if (Input.GetKeyDown(KeyCode.Alpha3))
        {
            Application.LoadLevel("Tamat");
        }
    }
    private void Flip(float horizontal)
    {
        if (horizontal > 0 && !facingRight || horizontal < 0 && facingRight)
        {
            ChangeDirection();
        }
    }
private bool IsGrounded()
    {
        if (MyRigidbody.velocity.y <= 0)
        {
            foreach (Transform point in groundPoints)
            {
                Collider2D[] colliders =
Physics2D.OverlapCircleAll(point.position, groundRadius, whatIsGround);

                for (int i = 0; i < colliders.Length; i++)
                {
                    if (colliders[i].gameObject != gameObject)
                    {
                        return true;
                    }
                }
            }
        }
        return false;
    }
```

```
private void HandleLayers()
    {
        if (!OnGround)
        {
            MyAnimator.SetLayerWeight(1, 1);
        }
        else
        {
            MyAnimator.SetLayerWeight(1, 0);
        }
    }
private IEnumerator IndicateImmortal()
    {
        while (immortal)
        {
            spriteRenderer.enabled = false;
            yield return new WaitForSeconds(.1f);
            spriteRenderer.enabled = true;
            yield return new WaitForSeconds(.1f);
        }
    }
    public override IEnumerator TakeDamage()
    {
        if (!immortal)
        {
            healthStat.CurrentVal -= 10;

            if (!IsDead)
            {
                MyAnimator.SetTrigger("damage");
                immortal = true;
                StartCoroutine(IndicateImmortal());
                yield return new WaitForSeconds(immortalTime);

                immortal = false;
            }
            else
            {
                MyAnimator.SetLayerWeight(1, 0);
                MyAnimator.SetTrigger("die");
            }
        }
    }
    public override void Death()
    {
        nyawa -= 1;
        MyRigidbody.velocity = Vector2.zero;
        MyAnimator.SetTrigger("idle");
        MyAnimator.ResetTrigger("die");
        healthStat.CurrentVal = healthStat.MaxVal;
        transform.position = startPos;
    }
    private void GameOver()
    {
        if (nyawa == 0)
        {
            Destroy(gameObject, 0.75f);
```

```
Application.LoadLevel(7);
        }
    }

    private void OnCollisionEnter2D(Collision2D other)
    {

        if (other.gameObject.tag == "Coin")
        {
            GameManager.Instance.CollectedCoins++;
            Destroy(other.gameObject);
            soundScript.PlaySound("Coin");
        }
        if (other.gameObject.tag == "P3K")
        {
            healthStat.CurrentVal += 10;
            Destroy(other.gameObject);
           soundScript.PlaySound("Item2");
        }
        if (other.gameObject.tag == "Enemy")
        {
            healthStat.CurrentVal -= 25;
        }
    }
    //public void Initialize(Vector2 direction)
    //{
    //    this.direction = direction;
    //}
    public void Peluru(int value)
    {
        if (facingRight)
        {
             GameObject tmp =  (GameObject)Instantiate(peluruPrefab,
pelurupost.position, Quaternion.Euler(new Vector3(0,0,-90)));
             tmp.GetComponent<Knife>().Initialize(Vector2.right);
        }
        else
        {
            GameObject tmp = (GameObject)Instantiate(peluruPrefab,
pelurupost.position, Quaternion.Euler(new Vector3(0, 0, 90)));
            tmp.GetComponent<Knife>().Initialize(Vector2.left);
        }
    }
}
```

Lampiran 2. Script Fuzzy Logic pada Bos

```
using UnityEngine;
using System.Collections;
using DotFuzzy;
using System.Collections.Generic;

public class Bos_Musuh : MonoBehaviour
{
    public float skala_model;
    public float speed = 20f;
    public float movespeed;
    private Vector3 posisiAwal;
    private bool facingright;
    private int areaPatroli;
    private Animator anim;
    private Transform groundCheck, hajar;
    private bool patrol;
    private GameObject player;
    float CoolDown = 0;
    public Transform target;
    public GameObject effect;
    //public CoinManager conman;
    //public int coinpoint = 20;
    Rigidbody2D rb2D;
    public AudioClip hit_sound;
    public float darah = 100;
    private float darahAwal;
    Transform hpbar;
    GameObject hpbarobj;
    public Rigidbody2D anak;
    public Rigidbody2D kapak;
    public float spawnTime = 5f;
    public float spawnDelay = 3f;
    private Animator player_anim;
    //DotFuzzy Deklarasi
    LinguisticVariable HealthPoint;
    LinguisticVariable Jarak;
    LinguisticVariable Aksi;
    FuzzyEngine fuzzyEngine;
    //[SerializeField]
   // private List<string> damageSource;
    void Start()
    {
        darahAwal = darah;
        hpbar = transform.Find("darahbar");
        target = GameObject.Find("Player").transform;
        //conman =
GameObject.FindGameObjectWithTag("Score").GetComponent<CoinManager>();
        hpbarobj = hpbar.gameObject;
        posisiAwal = transform.position;
        facingright = true;
        rb2D = GetComponent<Rigidbody2D>();
        areaPatroli = 100;
anim = GetComponent<Animator>();
patrol = true;
```

```
player = GameObject.Find("Player");
        groundCheck = transform.Find("GroundCheck");
        hajar = transform.Find("Demage");
        anim = GetComponent<Animator>();

        //Fuzzy
        fuzzyEngine = new FuzzyEngine();
        //==================== INPUT ====================//
        //Fuzzy Tambah Fungsi Keanggotaan HP
        HealthPoint = new LinguisticVariable("HealthPoint");
        HealthPoint.MembershipFunctionCollection.Add(new
MembershipFunction("Rendah", 0, 0, 25, 30));
        HealthPoint.MembershipFunctionCollection.Add(new
MembershipFunction("Setengah", 20, 40, 40, 55));
        HealthPoint.MembershipFunctionCollection.Add(new
MembershipFunction("Tinggi", 50, 65, 65, 80));
        HealthPoint.MembershipFunctionCollection.Add(new
MembershipFunction("SangatTinggi", 75, 85, 100, 100));
        //Fuzzy Tambah Fungsi Keanggotaan Jarak
        Jarak = new LinguisticVariable("Jarak");
        Jarak.MembershipFunctionCollection.Add(new
MembershipFunction("SangatDekat", 0, 0, 15, 20));
        Jarak.MembershipFunctionCollection.Add(new MembershipFunction("Dekat",
15, 30, 30, 50));
        Jarak.MembershipFunctionCollection.Add(new MembershipFunction("Jauh",
40, 60, 60, 80));
        Jarak.MembershipFunctionCollection.Add(new
MembershipFunction("SangatJauh", 75, 80, 100, 100));
        //Fuzzy Tambah Fungsi Keanggotaan Aksi
        //==================== OUTPUT ====================//
        Aksi = new LinguisticVariable("Aksi");
        Aksi.MembershipFunctionCollection.Add(new
MembershipFunction("PanggilAnak", 0, 0, 20, 25));
        Aksi.MembershipFunctionCollection.Add(new MembershipFunction("Lempar",
20, 50, 50, 70));
        Aksi.MembershipFunctionCollection.Add(new MembershipFunction("Attack",
60, 70, 100, 100));

        fuzzyEngine.LinguisticVariableCollection.Add(HealthPoint);
        fuzzyEngine.LinguisticVariableCollection.Add(Jarak);
        fuzzyEngine.LinguisticVariableCollection.Add(Aksi);
        fuzzyEngine.Consequent = "Aksi";
//Fuzzy Rule
        ////////////////////////////////////////////
        ////////////////////// SATU ////////////////
        ////////////////////////////////////////////
        fuzzyEngine.FuzzyRuleCollection.Add(new FuzzyRule("IF (HealthPoint IS
Rendah) AND (Jarak IS SangatDekat) THEN Aksi IS PanggilAnak"));
        fuzzyEngine.FuzzyRuleCollection.Add(new FuzzyRule("IF (HealthPoint IS
Rendah) AND (Jarak IS Dekat) THEN Aksi IS PanggilAnak"));
        fuzzyEngine.FuzzyRuleCollection.Add(new FuzzyRule("IF (HealthPoint IS
Rendah) AND (Jarak IS Jauh) THEN Aksi IS PanggilAnak"));
        fuzzyEngine.FuzzyRuleCollection.Add(new FuzzyRule("IF (HealthPoint IS
Rendah) AND (Jarak IS SangatJauh) THEN Aksi IS PanggilAnak"));
```

```
     ////////////////////////////////////////////
       ////////////////////// DUA ////////////////
       ////////////////////////////////////////////
       fuzzyEngine.FuzzyRuleCollection.Add(new FuzzyRule("IF (HealthPoint IS
Setengah) AND (Jarak IS SangatDekat) THEN Aksi IS Attack"));
       fuzzyEngine.FuzzyRuleCollection.Add(new FuzzyRule("IF (HealthPoint IS
Setengah) AND (Jarak IS Dekat) THEN Aksi IS Attack"));
       fuzzyEngine.FuzzyRuleCollection.Add(new FuzzyRule("IF (HealthPoint IS
Setengah) AND (Jarak IS Jauh) THEN Aksi IS Lempar"));
       fuzzyEngine.FuzzyRuleCollection.Add(new FuzzyRule("IF (HealthPoint IS
Setengah) AND (Jarak IS SangatJauh) THEN Aksi IS Lempar"));
       ////////////////////////////////////////////
       ////////////////////// TIGA ////////////////
       ////////////////////////////////////////////
       fuzzyEngine.FuzzyRuleCollection.Add(new FuzzyRule("IF (HealthPoint IS
Tinggi) AND (Jarak IS SangatDekat) THEN Aksi IS Attack"));
       fuzzyEngine.FuzzyRuleCollection.Add(new FuzzyRule("IF (HealthPoint IS
Tinggi) AND (Jarak IS Dekat) THEN Aksi IS Attack"));
       fuzzyEngine.FuzzyRuleCollection.Add(new FuzzyRule("IF (HealthPoint IS
Tinggi) AND (Jarak IS Jauh) THEN Aksi IS Attack"));
       fuzzyEngine.FuzzyRuleCollection.Add(new FuzzyRule("IF (HealthPoint IS
Tinggi) AND (Jarak IS SangatJauh) THEN Aksi IS Lempar"));
       ////////////////////////////////////////////
       ////////////////////// EMPAT ////////////////
       ////////////////////////////////////////////
       fuzzyEngine.FuzzyRuleCollection.Add(new FuzzyRule("IF (HealthPoint IS
SangatTinggi) AND (Jarak IS SangatDekat) THEN Aksi IS Attack"));
       fuzzyEngine.FuzzyRuleCollection.Add(new FuzzyRule("IF (HealthPoint IS
SangatTinggi) AND (Jarak IS Dekat) THEN Aksi IS Attack"));
       fuzzyEngine.FuzzyRuleCollection.Add(new FuzzyRule("IF (HealthPoint IS
SangatTinggi) AND (Jarak IS Jauh) THEN Aksi IS Attack"));
       fuzzyEngine.FuzzyRuleCollection.Add(new FuzzyRule("IF (HealthPoint IS
SangatTinggi) AND (Jarak IS SangatJauh) THEN Aksi IS Lempar"));
    }
void Update()
    {
        patroli();
        attack();
        Darah();
    }
void Darah()
    {
        if (darah > (darahAwal * 0.5) && darah <= (darahAwal * 0.75))
        {
            if (hpbarobj.transform.Find("darahbar_4"))
            {
                Destroy(hpbarobj.transform.Find("darahbar_4").gameObject);
            }
        }
if (darah > (darahAwal * 0.25) && darah <= (darahAwal * 0.5))
        {
            if (hpbarobj.transform.Find("darahbar_3"))
            {
                Destroy(hpbarobj.transform.Find("darahbar_3").gameObject);
            }
        }
```

```
if (darah > 0 && darah <= (darahAwal * 0.25))
        {
            if (hpbarobj.transform.Find("darahbar_2"))
            {
                Destroy(hpbarobj.transform.Find("darahbar_2").gameObject);
            }
        }
        if (darah <= 0)
        {
            if (hpbarobj.transform.Find("darahbar_1"))
            {
                Destroy(hpbarobj.transform.Find("darahbar_1").gameObject);
            }
        }
    }
    void patroli()
    {
        if (patrol == true)
        {
            if (facingright == true && posisiAwal.x - areaPatroli <
transform.position.x)
            {
                anim.SetBool("Jalan", true);
                rb2D.velocity = new Vector2(-5, rb2D.velocity.y);
            }
            else if (facingright == true && posisiAwal.x - areaPatroli >
transform.position.x)
            {
                anim.SetBool("Jalan", false);
                CoolDown -= Time.deltaTime;
                if (CoolDown < 0)
                {
                    CoolDown = 3;
                }
                if (CoolDown == 3)
                {
                    Flip();
                }
                else if (CoolDown < 3 && Vector3.Distance(target.position,
transform.position) <= 25)
                {
                    anim.SetBool("Jalan", true);
                }
            }
            else if (facingright == false && posisiAwal.x + areaPatroli >
transform.position.x)
            {
                anim.SetBool("Jalan", true);
                rb2D.velocity = new Vector2(5, rb2D.velocity.y);
            }
            else if (facingright == false && posisiAwal.x + areaPatroli <
transform.position.x && patrol == true)
            {
                anim.SetBool("Jalan", false);
                CoolDown -= Time.deltaTime;
                if (CoolDown < 0)
                {
                    CoolDown = 3;
                }
                if (CoolDown == 3)
```

```
                {
                    Flip();
                }
                else if (CoolDown < 3 && Vector3.Distance(target.position,
transform.position) <= 25)
                {
                    anim.SetBool("Jalan", true);
                }
            }
        }
    }
void attack()
    {
        CoolDown -= Time.deltaTime;
        patrol = false;
        if (CoolDown < 0)
        {
            HealthPoint.InputValue = darah;
            if (Vector3.Distance(target.position, transform.position)*6>=100)
            //if ((transform.position.x - target.position.x - 3) * 10 >= 100)
            {
                Jarak.InputValue = 100;
            }
            else if (Jarak.InputValue > 100)
            {
                anim.SetBool("Serang", false);
                anim.SetBool("Jalan", false);
                anim.SetBool("Lempar", false);
            }
            else
            {
                //Jarak.InputValue = (transform.position.x -
player.transform.position.x) * 10;
                Jarak.InputValue = (Vector3.Distance(target.position,
transform.position)* 5 + 10);
            }

            ////////////////////////////////////////////////////////
            ////////////////////// PANGGIL PRAJURIT ////////////////
            ////////////////////////////////////////////////////////
            if ((fuzzyEngine.Defuzzify() >= 0) && (fuzzyEngine.Defuzzify() <=
30))
            {
                Instantiate(anak,
GameObject.Find("lemparsenjata").GetComponent<Transform>().position,
Quaternion.identity);
            }
            ////////////////////////////////////////////////////////
            ////////////////////// LEMPAR SENJATA //////////////////
            ////////////////////////////////////////////////////////
            else if ((fuzzyEngine.Defuzzify() >= 26) && (fuzzyEngine.Defuzzify()
<= 70))
            {
                anim.SetBool("Serang", false);
                anim.SetBool("Jalan", false);
                anim.SetBool("Lempar", true);

                //StartCoroutine(play_hit_sound());
            }

            ////////////////////////////////////////////////////////
            ////////////////////// SERANG PUKUL ////////////////////
```

```
            ////////////////////////////////////////////////////////
            else if ((fuzzyEngine.Defuzzify() >= 71) && (fuzzyEngine.Defuzzify()
<= 100))
            {
                //CoolDown = 2;
                anim.SetBool("Lempar", false);
                patrol = false;
                if (Vector3.Distance(target.position, transform.position) <= 28
&& Vector3.Distance(target.position, transform.position) > 7)
                {
                    anim.SetBool("Jalan", true);
                    rb2D.velocity = new Vector2(-5, rb2D.velocity.y);
                }
                if (Vector3.Distance(target.position, transform.position) <= 7)
                {
                    anim.SetBool("Jalan", false);
                    anim.SetBool("Serang", true);
                    rb2D.velocity = new Vector2( -5, rb2D.velocity.y);
                }
            }
            Debug.Log("hp: " + HealthPoint.InputValue + " jarak : " +
Jarak.InputValue + " output : " + fuzzyEngine.Defuzzify());
            CoolDown = 2;
        }
    }

  void lempar()
    {
        if (facingright)
        {
            Rigidbody2D bulletInstance = Instantiate(kapak,
GameObject.Find("lemparsenjata").GetComponent<Transform>().position,
Quaternion.Euler(new Vector3(180f, 0, 180f))) as Rigidbody2D;
            bulletInstance.velocity = new Vector2(-speed, 0);
        }
        else
        {
            Rigidbody2D bulletInstance = Instantiate(kapak,
GameObject.Find("lemparsenjata").GetComponent<Transform>().position,
Quaternion.Euler(new Vector3(0, 0, 0))) as Rigidbody2D;
            bulletInstance.velocity = new Vector2(speed, 0);
        }
    }

    void OnExplode()
    {
        Quaternion randomRotation = Quaternion.Euler(0f, 0f, Random.Range(0f,
360f));
        int i = Random.Range(0, 3);
        Instantiate(effect, hajar.position, randomRotation);
    }


    void Spawn()
    {
        if (facingright)
        {
            //anim.SetBool("Attack1", true);
            Rigidbody2D bulletInstance = Instantiate(kapak,
GameObject.Find("lemparsenjata").GetComponent<Transform>().position,
Quaternion.Euler(new Vector3(180f, 0, 180f))) as Rigidbody2D;
            bulletInstance.velocity = new Vector2(-speed, 0);
```

```
        }
        else
        {
            //anim.SetBool("Attack1", true);
            Rigidbody2D bulletInstance = Instantiate(kapak,
GameObject.Find("lemparsenjata").GetComponent<Transform>().position,
Quaternion.Euler(new Vector3(0, 0, 0))) as Rigidbody2D;
            bulletInstance.velocity = new Vector2(speed, 0);
        }
    }
public void Flip()
    {
        var s = transform.localScale;
        s.x *= -1;
        transform.localScale = s;
        facingright = !facingright;
    }
    IEnumerator play_hit_sound()
    {
        yield return new WaitForSeconds(0.7f);
        AudioSource.PlayClipAtPoint(hit_sound, transform.position);
    }

    void OnTriggerEnter2D(Collider2D col)
    {
        if (col.gameObject.tag == "Sword")
        {
            darah -= 25;
        }
        if (darah <= 0)
        {
            //conman.AddPoints(coinpoint);
            //SaveScene();

            anim.SetBool("Mati", true);
            anim.SetBool("Jalan", false);
            anim.SetBool("Lempar", false);
            anim.SetBool("Serang", false);
        }
        if (col.gameObject.tag == "Peluru")
        {
            darah -= 25;
        }
        if (darah <= 0)
        {
            //conman.AddPoints(coinpoint);
            //SaveScene();

            anim.SetBool("Mati", true);
            anim.SetBool("Jalan", false);
            anim.SetBool("Lempar", false);
            anim.SetBool("Serang", false);
        }
    }

   // void SaveScene()
   // {
    //    PlayerPrefs.SetInt("score",conman.score);
   // }

IEnumerator ReloadGame()
   {
```

```
        // ... pause briefly
      yield return new WaitForSeconds(0.0f);
        // ... and then reload the level.
      Application.LoadLevel(6);

    }


    IEnumerator serang(float time)
    {
        anim.Stop();
        yield return (new WaitForSeconds(time));
    }

    void enemyattack()
    {
        anim.SetBool("Serang", true);
        patrol = false;
        anim.SetBool("Jalan", false);

        CoolDown -= Time.deltaTime;
        if (CoolDown < 0)
        {
            Debug.Log("Hit The Target !");
            anim.SetBool("Serang", true);
            CoolDown = 1f;
        }
        else
        {
            anim.SetBool("Serang", false);
        }
    }
void followplayer()
    {
        if (Vector3.Distance(target.position, transform.position) <= 100 &&
Vector3.Distance(target.position, transform.position) > 10)
        {
        patrol = false;
        if (target.position.x < transform.position.x) // jika player sebelah kiri
        {
            if (facingright == false) // jika player sebelah kana dan musuh hadap
kanan
            {
                Flip();
            }
            transform.position -= transform.right * movespeed * Time.deltaTime;
// player is left of enemy, move left
            anim.SetBool("Jalan", true);
        }
        if (target.position.x > transform.position.x) // jika player sebelah
kanan
        {
            if (facingright == true) // jika player sebelah kana dan musuh hadap
kiri
            {
                Flip();
            }
            transform.position += transform.right * movespeed * Time.deltaTime;
// player is left of enemy, move right
            anim.SetBool("Jalan", true);
        }
```

```
        if (Vector3.Distance(target.position, transform.position) <= 10)
        {
            //anim.SetBool("Jalan", false);
            anim.SetBool("Serang", true);
            //enemyattack();
        }
        else if (Vector3.Distance(target.position, transform.position) <= 20 &&
patrol == false)
        {
            anim.SetBool("Jalan", true);
            anim.SetBool("Serang", false);
            patrol = true;
        }
        }
    }

}
```

Lampiran 3. Script Musuh

```
using UnityEngine;
using System.Collections;

public class Enemy : Character
{

    private IEnemyState currentState;

    public GameObject Target { get; set; }

    [SerializeField]
    private float meleeRange;

    [SerializeField]
    private float shootRange;

    private Vector3 startPos;

    [SerializeField]
    private Transform leftEdge;
    [SerializeField]
    private Transform rightEdge;

    private Canvas healthCanvas;

    private bool dropItem = true;

    public bool InMeleeRange
    {
        get
        {
            if (Target != null)
            {
                return Vector2.Distance(transform.position, Target.transform.position)
<= meleeRange;
            }

            return false;
        }
    }

   public bool InshootRange
    {
        get
        {
            if (Target != null)
            {
                return Vector2.Distance(transform.position, Target.transform.position)
<= shootRange;
            }

            return false;
        }
    }

    // Use this for initialization
```

```
    public override void Start()
    {
        base.Start();
        Player.Instance.Dead += new DeadEventHandler(RemoveTarget);
        ChangeState(new IdleState());

        healthCanvas = transform.GetComponentInChildren<Canvas>();
    }

    // Update is called once per frame
    void Update()
    {
        if (!IsDead)
        {
            if (!TakingDamage)
            {
                currentState.Execute();
            }

            LookAtTarget();

        }


    }
public void RemoveTarget()
    {
        Target = null;
        ChangeState(new PatrolState());
    }

    private void LookAtTarget()
    {
        if (Target != null)
        {
            float xDir = Target.transform.position.x - transform.position.x;

            if (xDir < 0 && facingRight || xDir > 0 && !facingRight)
            {
                ChangeDirection();
            }
        }
    }
    public void ChangeState(IEnemyState newState)
    {
        if (currentState != null)
        {
            currentState.Exit();
        }

        currentState = newState;

        currentState.Enter(this);
    }

    public void Move()
    {
        if (!Attack)
        {
            if ((GetDirection().x > 0 && transform.position.x < rightEdge.position.x)
|| (GetDirection().x < 0 && transform.position.x > leftEdge.position.x))
            {
```

```
                MyAnimator.SetFloat("speed", 1);

                transform.Translate(GetDirection() * (movementSpeed *
Time.deltaTime));
            }
            else if (currentState is PatrolState)
            {
                ChangeDirection();
            }
        }
    }

public Vector2 GetDirection()
    {
        return facingRight ? Vector2.right : Vector2.left;
    }


    public override void OnTriggerEnter2D(Collider2D other)
    {
        base.OnTriggerEnter2D(other);
        currentState.OnTriggerEnter(other);
    }

    public override IEnumerator TakeDamage()
    {
        if (!healthCanvas.isActiveAndEnabled)
        {
            healthCanvas.enabled = true;
        }

        healthStat.CurrentVal -= 10;

        if (!IsDead)
        {
            MyAnimator.SetTrigger("damage");
        }
        else
        {

            MyAnimator.SetTrigger("die");
            yield return null;
        }
    }

public override bool IsDead
    {
        get
        {
            return healthStat.CurrentVal <= 0;
        }
    }

    //public void coindeath()
    //{
    //    if (healthStat.CurrentVal == 0)
    //    {


    //    }
    //}
```

```
    public override void Death()
    {
        GameObject coin = (GameObject)Instantiate(GameManager.Instance.CoinPrefab, new
Vector3(transform.position.x, transform.position.y + 2), Quaternion.identity);
       // Instantiate(GameManager.Instance.CoinPrefab, new
Vector3(transform.position.x, transform.position.y + 2), Quaternion.identity);
        Physics2D.IgnoreCollision(coin.GetComponent<Collider2D>(),
GetComponent<Collider2D>());
        Destroy(gameObject);
        healthStat.CurrentVal = healthStat.MaxVal;
        transform.position = startPos;
        healthCanvas.enabled = false;
    }
}
```

```
    [SerializeField]
    private Transform leftEdge;

    [SerializeField]
    private Transform rightEdge;

    private Canvas healtCanvas;

    public bool InMeleeRange
    {
        get
        {
            if (Target != null)
            {
                return Vector2.Distance(transform.position,
Target.transform.position) <= meleeRange ;
            }

            return false;
        }
    }

    public bool InThrowRange
    {
        get
        {
            if (Target != null)
            {
                return Vector2.Distance(transform.position,
Target.transform.position) <= throwRange ;
            }

            return false;
        }
    }

    public override void Start()
    {
        base.Start();
        Player.Instance.Dead += new DeadEventHandler(RemoveTarget);
        ChangeState (new IdleState());

                        // healtCanvas                                  =
                             transform.GetComponentInChildren<Canvas>();
    }

    void Update()
    {
        if (!IsDead)
        {
            if (!TakingDamage)
            {
                currentState.Execute();
            }

            LookAtTarget();
        }

    }
```

```
    public void RemoveTarget()
    {
        Target = null;
        ChangeState(new PatrolState());
    }

    private void LookAtTarget()
    {
        if (Target != null)
        {
            float xDir = Target.transform.position.x
- transform.position.x;

            if (xDir < 0 && facingRight || xDir > 0 && !facingRight)
            {
                ChangeDirection();
            }
        }
    }

    public void ChangeState(IEnemyState newState)
    {
        if (currentState != null)
        {
            currentState.Exit();
        }

        currentState = newState;
        currentState.Enter(this);
    }

    public void Move()
    {

        if (!attack)
        {
            if ((GetDirection().x > 0 && transform.position.x <
rightEdge.position.x) || (GetDirection().x < 0 && transform.position.x
> leftEdge.position.x))
            {
                MyAnimator.SetFloat("speed", 1);
                transform.Translate(GetDirection() * (movementSpeed *
Time.deltaTime));
            }

            else if (currentState is PatrolState)
            {
                ChangeDirection();
            }
        }

    }

    public Vector2 GetDirection()
    {
        return facingRight ? Vector2.right : Vector2.left;
    }
```

```
    public override  void OnTriggerEnter2D(Collider2D other)
    {
        Debug.Log("hit");
        base.OnTriggerEnter2D(other);
        currentState.OnTriggerEnter(other);
    }

    public override IEnumerator TakeDamage()
    {
       //if (!healtCanvas.isActiveAndEnabled)
      // {
         //  healtCanvas.enabled = true;
        //}

        healthStat.CurrentVal  -= 10;

        if (!IsDead)
        {
            MyAnimator.SetTrigger("damage");
        }
        else
        {
            MyAnimator.SetTrigger("die");
            yield return null;
        }
    }

    public override bool IsDead
    {
        get
        {
            return healthStat.CurrentVal <= 0;
        }
    }

    public override void Death()
    {
        Destroy(gameObject);
        MyAnimator.ResetTrigger("die");
        //healtCanvas.enabled = false;

    }

   // public override void
  // {
       // Transform tmp = transform.FindChild("Enemy
Health Bar").transform;
      // Vector3 pos = tmp.position;
       //tmp.SetParent(null);

       //base.ChangeDirection();

      // tmp.SetParent(transform);
      // tmp.position = pos;
 //  }
}
```

Kuesioner

Nama : Panda Nugraha

Umur : 22

Alamat : Tranggulek

Berdasarkan pengalaman anda memainkan game "Rescue Idol", berilah penilaian sesuai dengan pendapat anda, dengan cara memberi tanda centang ("√") pada kolom penilaian yang sudah disediakan.

No	Pertanyaan	Penilaian		
		Baik	Cukup	Kurang
1	Desain Karakter Game		√	
2	Desain Animasi pada Game	√		
3	Kontrol pada game		√	
4	Fitur game	√		
5	Informasi game (Cerita, Narasi Game)		√	
6	Game sudah menarik	√		

Terima kasih atas kesediaannya mengisi kuesioner ini.

Malang, Agustus 2017

(.....)

Nama dan tanda tangan

Kuesioner

Nama : Rio Arifando

Umur : 24

Alamat : Malang

Berdasarkan pengalaman anda memainkan game "Rescue Idol", berilah penilaian sesuai dengan pendapat anda, dengan cara memberi tanda centang ("√") pada kolom penilaian yang sudah disediakan.

No	Pertanyaan	Penilaian		
		Baik	Cukup	Kurang
1	Desain Karakter Game		√	
2	Desain Animasi pada Game		√	
3	Kontrol pada game		√	
4	Fitur game			√
5	Informasi game (Cerita, Narasi Game)	√		
6	Game sudah menarik		√	

Terima kasih atas kesediaannya mengisi kuesioner ini.

Malang, Agustus 2017

(.....)

Nama dan tanda tangan

Kuesioner

Nama : Moh Tarmizi Lakara

Umur : 22

Alamat : Jl. Al-Jihad No. 20A -Palu

Berdasarkan pengalaman anda memainkan game "Rescue Idol", berilah penilaian sesuai dengan pendapat anda, dengan cara memberi tanda centang ("√") pada kolom penilaian yang sudah disediakan.

No	Pertanyaan	Penilaian		
		Baik	Cukup	Kurang
1	Desain Karakter Game	√		
2	Desain Animasi pada Game	√		
3	Kontrol pada game		√	
4	Fitur game	√		
5	Informasi game (Cerita, Narasi Game)		√	
6	Game sudah menarik			√

Terima kasih atas kesediaannya mengisi kuesioner ini.

Malang, Agustus 2017

(Tarmizi. L)

Nama dan tanda tangan

Kuesioner

Nama : Ary Muhamad Rumalsey

Umur : 22

Alamat : Jl. Pandopo. Bula

Berdasarkan pengalaman anda memainkan game "Rescue Idol", berilah penilaian sesuai dengan pendapat anda, dengan cara memberi tanda centang ("√") pada kolom penilaian yang sudah disediakan.

No	Pertanyaan	Penilaian		
		Baik	Cukup	Kurang
1	Desain Karakter Game	√		
2	Desain Animasi pada Game	√		
3	Kontrol pada game		√	
4	Fitur game	√		
5	Informasi game (Cerita, Narasi Game)			√
6	Game sudah menarik		√	

Terima kasih atas kesediaannya mengisi kuesioner ini.

Malang, Agustus 2017

(Ary M. Rumalsey)

Nama dan tanda tangan

Kuesioner

Nama : M. Fahrul
Umur : 20
Alamat : Malang

Berdasarkan pengalaman anda memainkan game "Rescue Idol", berilah penilaian sesuai dengan pendapat anda, dengan cara memberi tanda centang ("√") pada kolom penilaian yang sudah disediakan.

No	Pertanyaan	Penilaian		
		Baik	Cukup	Kurang
1	Desain Karakter Game	√		
2	Desain Animasi pada Game	√		
3	Kontrol pada game		√	
4	Fitur game		√	
5	Informasi game (Cerita, Narasi Game)			√
6	Game sudah menarik	√		

Terima kasih atas kesediaannya mengisi kuesioner ini.

Malang, Agustus 2017

(.....)
Nama dan tanda tangan

Kuesioner

Nama : fauzan

Umur : 21

Alamat : Jl. Jongbiru blok M no 13

Berdasarkan pengalaman anda memainkan game "Rescue Idol", berilah penilaian sesuai dengan pendapat anda, dengan cara memberi tanda centang ("√") pada kolom penilaian yang sudah disediakan.

No	Pertanyaan	Penilaian		
		Baik	Cukup	Kurang
1	Desain Karakter Game		√	
2	Desain Animasi pada Game	√		
3	Kontrol pada game	√		
4	Fitur game		√	
5	Informasi game (Cerita, Narasi Game)			√
6	Game sudah menarik		√	

Terima kasih atas kesediaannya mengisi kuesioner ini.

Malang, Agustus 2017

(.....)

Nama dan tanda tangan

Kuesioner

Nama : Gama

Umur : 20

Alamat : Jl. Joyoboyo No. 228

Berdasarkan pengalaman anda memainkan game "Rescue Idol", berilah penilaian sesuai dengan pendapat anda, dengan cara memberi tanda centang ("√") pada kolom penilaian yang sudah disediakan.

No	Pertanyaan	Penilaian		
		Baik	Cukup	Kurang
1	Desain Karakter Game			√
2	Desain Animasi pada Game			√
3	Kontrol pada game		√	
4	Fitur game	√		
5	Informasi game (Cerita, Narasi Game)	√		
6	Game sudah menarik		√	

Terima kasih atas kesediaannya mengisi kuesioner ini.

Malang, Agustus 2017

(.....)

Nama dan tanda tangan

Kuesioner

Nama : Rahana

Umur : 19

Alamat : Perumahan Permata biru No 20

Berdasarkan pengalaman anda memainkan game "Rescue Idol", berilah penilaian sesuai dengan pendapat anda, dengan cara memberi tanda centang ("√") pada kolom penilaian yang sudah disediakan.

No	Pertanyaan	Penilaian		
		Baik	Cukup	Kurang
1	Desain Karakter Game	√		
2	Desain Animasi pada Game	√		
3	Kontrol pada game		√	
4	Fitur game	√		
5	Informasi game (Cerita, Narasi Game)	√		
6	Game sudah menarik		√	

Terima kasih atas kesediaannya mengisi kuesioner ini.

Malang, Agustus 2017

(.....)

Nama dan tanda tangan

Kuesioner

Nama : Manang

Umur : 20

Alamat : Mgronggo RW 178

Berdasarkan pengalaman anda memainkan game "Rescue Idol", berilah penilaian sesuai dengan pendapat anda, dengan cara memberi tanda centang ("√") pada kolom penilaian yang sudah disediakan.

No	Pertanyaan	Penilaian		
		Baik	Cukup	Kurang
1	Desain Karakter Game			✓
2	Desain Animasi pada Game		✓	
3	Kontrol pada game	✓		
4	Fitur game	✓		
5	Informasi game (Cerita, Narasi Game)		✓	
6	Game sudah menarik	✓		

Terima kasih atas kesediaannya mengisi kuesioner ini.

Malang, Agustus 2017

(.....)

Nama dan tanda tangan

Kuesioner

Nama : Jatmiko

Umur : 24

Alamat : Jatipalem Blo 20

Berdasarkan pengalaman anda memainkan game "Rescue Idol", berilah penilaian sesuai dengan pendapat anda, dengan cara memberi tanda centang ("√") pada kolom penilaian yang sudah disediakan.

No	Pertanyaan	Penilaian		
		Baik	Cukup	Kurang
1	Desain Karakter Game			√
2	Desain Animasi pada Game			√
3	Kontrol pada game		√	
4	Fitur game	√		
5	Informasi game (Cerita, Narasi Game)		√	
6	Game sudah menarik	√		

Terima kasih atas kesediaannya mengisi kuesioner ini.

Malang, Agustus 2017

(.....)

Nama dan tanda tangan

Kuesioner

Nama : Arki Agustyawan

Umur : 23

Alamat : Ngasem No 17

Berdasarkan pengalaman anda memainkan game "Rescue Idol", berilah penilaian sesuai dengan pendapat anda, dengan cara memberi tanda centang ("√") pada kolom penilaian yang sudah disediakan.

No	Pertanyaan	Penilaian		
		Baik	Cukup	Kurang
1	Desain Karakter Game		√	
2	Desain Animasi pada Game			√
3	Kontrol pada game		√	
4	Fitur game			√
5	Informasi game (Cerita, Narasi Game)		√	
6	Game sudah menarik	√		

Terima kasih atas kesediaannya mengisi kuesioner ini.

Malang, Agustus 2017

(.....)

Nama dan tanda tangan

Kuesioner

Nama : Mahmud Fahri Lutfulu

Umur : 29

Alamat : Jl. gampong rejo no 15

Berdasarkan pengalaman anda memainkan game "Rescue Idol", berilah penilaian sesuai dengan pendapat anda, dengan cara memberi tanda centang ("√") pada kolom penilaian yang sudah disediakan.

No	Pertanyaan	Penilaian		
		Baik	Cukup	Kurang
1	Desain Karakter Game	√		
2	Desain Animasi pada Game		√	
3	Kontrol pada game	√		
4	Fitur game			√
5	Informasi game (Cerita, Narasi Game)		√	
6	Game sudah menarik	√		

Terima kasih atas kesediaannya mengisi kuesioner ini.

Malang, Agustus 2017

(.....)

Nama dan tanda tangan

Kuesioner

Nama : Octavian W-I-I

Umur : 22

Alamat : Jl. Karangsio

Berdasarkan pengalaman anda memainkan game "Rescue Idol", berilah penilaian sesuai dengan pendapat anda, dengan cara memberi tanda centang ("√") pada kolom penilaian yang sudah disediakan.

No	Pertanyaan	Penilaian		
		Baik	Cukup	Kurang
1	Desain Karakter Game		√	
2	Desain Animasi pada Game			√
3	Kontrol pada game	√		
4	Fitur game		√	
5	Informasi game (Cerita, Narasi Game)			√
6	Game sudah menarik		√	

Terima kasih atas kesediaannya mengisi kuesioner ini.

Malang, Agustus 2017

(.....auto.....)

Nama dan tanda tangan

Kuesioner

Nama : Muh. Hanif Alyaoni

Umur : 23

Alamat : Dsn Kupur, Keb. Kepuhrejo, Kec. Karangploso

Berdasarkan pengalaman anda memainkan game "Rescue Idol", berilah penilaian sesuai dengan pendapat anda, dengan cara memberi tanda centang ("√") pada kolom penilaian yang sudah disediakan.

No	Pertanyaan	Penilaian		
		Baik	Cukup	Kurang
1	Desain Karakter Game	√		
2	Desain Animasi pada Game	√		
3	Kontrol pada game	√		
4	Fitur game		√	
5	Informasi game (Cerita, Narasi Game)		√	
6	Game sudah menarik		√	

Terima kasih atas kesediaannya mengisi kuesioner ini.

Malang, Agustus 2017

(M. Hanif A.)

Nama dan tanda tangan

Kuesioner

Nama : Rahmad Al Fath

Umur : 22

Alamat : Jl. Blimbing, belakang masjid Sabilillah no 7

Berdasarkan pengalaman anda memainkan game "Rescue Idol", berilah penilaian sesuai dengan pendapat anda, dengan cara memberi tanda centang ("√") pada kolom penilaian yang sudah disediakan.

No	Pertanyaan	Penilaian		
		Baik	Cukup	Kurang
1	Desain Karakter Game	√		
2	Desain Animasi pada Game		√	
3	Kontrol pada game	√		
4	Fitur game	√		
5	Informasi game (Cerita, Narasi Game)		√	
6	Game sudah menarik	√		

Terima kasih atas kesediaannya mengisi kuesioner ini.

Malang, Agustus 2017

(Rahmad Al Fath)

Nama dan tanda tangan

Kuesioner

Nama : Jaka Pahmadi

Umur : 21

Alamat : Jl. RTa. Mibuo kw. 25 no5.

Berdasarkan pengalaman anda memainkan game "Rescue Idol", berilah penilaian sesuai dengan pendapat anda, dengan cara memberi tanda centang ("√") pada kolom penilaian yang sudah disediakan.

No	Pertanyaan	Penilaian		
		Baik	Cukup	Kurang
1	Desain Karakter Game		√	
2	Desain Animasi pada Game			√
3	Kontrol pada game	√		
4	Fitur game	√		
5	Informasi game (Cerita, Narasi Game)		√	
6	Game sudah menarik			√

Terima kasih atas kesediaannya mengisi kuesioner ini.

Malang, Agustus 2017

(Jaka Pahmadi)

Nama dan tanda tangan

Kuesioner

Nama : Muh. Fadhil Rahman

Umur : 21

Alamat : Lombok

Berdasarkan pengalaman anda memainkan game "Rescue Idol", berilah penilaian sesuai dengan pendapat anda, dengan cara memberi tanda centang ("√") pada kolom penilaian yang sudah disediakan.

No	Pertanyaan	Penilaian		
		Baik	Cukup	Kurang
1	Desain Karakter Game	√		
2	Desain Animasi pada Game	√		
3	Kontrol pada game	√		
4	Fitur game		√	
5	Informasi game (Cerita, Narasi Game)		√	
6	Game sudah menarik	√		

Terima kasih atas kesediaannya mengisi kuesioner ini.

Malang, Agustus 2017

(Muh. Fadhil R.)

Nama dan tanda tangan

Kuesioner

Nama : Wahyudi Noviarto

Umur : 23

Alamat : Jl Simpang Grayaham I /44

Berdasarkan pengalaman anda memainkan game "Rescue Idol", berilah penilaian sesuai dengan pendapat anda, dengan cara memberi tanda centang ("√") pada kolom penilaian yang sudah disediakan.

No	Pertanyaan	Penilaian		
		Baik	Cukup	Kurang
1	Desain Karakter Game	√		
2	Desain Animasi pada Game		√	
3	Kontrol pada game		√	
4	Fitur game		√	
5	Informasi game (Cerita, Narasi Game)	√		
6	Game sudah menarik		√	

Terima kasih atas kesediaannya mengisi kuesioner ini.

Malang, Agustus 2017

(Wahyudi Noviarto)

Nama dan tanda tangan

Kuesioner

Nama : Maris

Umur : 21

Alamat : Jl. Mauni No 118

Berdasarkan pengalaman anda memainkan game "Rescue Idol", berilah penilaian sesuai dengan pendapat anda, dengan cara memberi tanda centang ("√") pada kolom penilaian yang sudah disediakan.

No	Pertanyaan	Penilaian		
		Baik	Cukup	Kurang
1	Desain Karakter Game		√	
2	Desain Animasi pada Game	√		
3	Kontrol pada game	√		
4	Fitur game		√	
5	Informasi game (Cerita, Narasi Game)		√	
6	Game sudah menarik			√

Terima kasih atas kesediaannya mengisi kuesioner ini.

Malang, Agustus 2017

(.....)

Nama dan tanda tangan

Kuesioner

Nama : Danang
Umur : 21
Alamat : Malang

Berdasarkan pengalaman anda memainkan game "Rescue Idol", berilah penilaian sesuai dengan pendapat anda, dengan cara memberi tanda centang ("√") pada kolom penilaian yang sudah disediakan.

No	Pertanyaan	Penilaian		
		Baik	Cukup	Kurang
1	Desain Karakter Game			√
2	Desain Animasi pada Game		√	
3	Kontrol pada game			√
4	Fitur game	√		
5	Informasi game (Cerita, Narasi Game)	√		
6	Game sudah menarik	√		

Terima kasih atas kesediaannya mengisi kuesioner ini.

Malang, Agustus 2017

(.....)

Nama dan tanda tangan