PEMBUATAN KARAKTER MUSUH DENGAN KECERDASAN BUATAN PADA GAME TOP DOWN SHOOTER "FROM THE DOWNTOWN"

SKRIPSI

MUHAMMAD RIDWAN LUBIS

201401116

PROGRAM STUDI S-1 ILMU KOMPUTER

FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI

UNIVERSITAS SUMATERA UTARA

MEDAN

2024

PEMBUATAN KARAKTER MUSUH DENGAN KECERDASAN BUATAN PADA GAME TOP DOWN SHOOTER "FROM THE DOWNTON"

SKRIPSI

Diajukan untuk melengkapi tugas dan memenuhi syarat memperoleh ijazah Sarjana Ilmu Komputer

MUHAMMAD RIDWAN LUBIS
201401116

PROGRAM STUDI S-1 ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA
MEDAN
2024

PERSETUJUAN

Judul	: PEMBUATAN KARAKTER MUSUH DENGAN KECERDASAN BUATAN PADA GAME *TOP DOWN SHOOTER "FROM THE DOWNTOWN"*
Kategori	: SKRIPSI
Nama	: MUHAMMAD RIDWAN LUBIS
Nomor Induk Mahasiswa	: 201401116
Program Studi	: SARJANA (S-1) ILMU KOMPUTER
Fakultas	: ILMU KOMPUTER DAN TEKNOLOGI INFORMASI UNIVERSITAS SUMATERA UTARA
Komisi Pembimbing	:

Pembimbing 2	Pembimbing 1
Dr. Jos Timanta Tarigan S.Kom., M.Sc	Handrizal S.Si., M.Comp.Sc
NIP. 198501262015041001	NIP. 197706132017061001

Diketahui/Disetujui Oleh
Program Studi S-1 Ilmu Komputer
Ketua,

Dr. Amalia ST., M.T.
NIP. 197812212014042001

PERNYATAAN

PEMBUATAN KARAKTER MUSUH DENGAN KECERDASAN BUATAN PADA GAME *TOP DOWN SHOOTER "FROM THE DOWNTOWN"*

SKRIPSI

Saya mengakui bahwa skripsi ini adalah hasil karya saya sendiri, kecuali beberapa kutipan dan ringkasan yang masing-masing telah disebutkan sumbernya.

Medan, 9 Juli 2024

Muhammad Ridwan Lubis
201401116

PENGHARGAAN

Bismillahirrahmanirrahim, puji dan syukur dipanjatkan kepada Allah *Subhanahu Wa Ta'ala* berkat limpahan rahmat-Nya sehingga penulis dapat berada pada tahap penyusunan skripsi ini sebagai syarat untuk mendapatkan gelar Sarjana Komputer di Program Studi S-1 Ilmu Komputer, Fakultas Ilmu Komputer dan Teknologi Informasi, Universitas Sumatera Utara. Shalawat dan salam dicurahkan kepada Rasulullah *Shalallaahu 'Alayhi Wasallam* yang telah membimbing umat manusia dari zaman kegelapan menuju zaman terang benderang saat ini.

Dengan penuh rasa hormat pada kesempatan ini penulis mengucapkan terima kasih sebesar-besarnya kepada Mama tecinta, Nurcahaya Lubis atas segala bentuk kasih sayang serta doa-doa yang dipanjatkan untuk penulis. Dan terima kasih kepada Ayah, Abdul Hadi Nasution atas dukungan dan kasih sayang yang membersamai di setiap langkah penulis. Terima kasih untuk setiap dukungan yang telah diberikan hingga penulis dapat berada di titik ini.

Penyusunan skripsi ini tidak terlepas dari bantuan, dukungan, dan bimbingan dari banyak pihak. Oleh karena itu, penulis mengucapkan banyak terima kasih kepada:

1. Bapak Prof. Dr. Muryanto Amin S.Sos., M.Si. selaku Rektor Universitas Sumatera Utara.
2. Ibu Dr. Maya Silvi Lydia B.Sc., M.Sc. selaku Dekan Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara.
3. Ibu Dr. Amalia, S.T., M.T. selaku Ketua Program Studi S-1 Ilmu Komputer Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara.
4. Bapak Handrizal S.Si., M.Comp.Sc selaku dosen Pembimbing I yang telah memberikan banyak bimbingan dan masukan yang berharga kepada penulis selama proses penyusunan skripsi ini.
5. Bapak Dr. Jos Timanta Tarigan S.Kom., M.Sc. selaku dosen Pembimbing II yang telah memberikan bimbingan serta masukannya kepada penulis yang memudahkan penulis selama proses penyusunan skripsi ini.
6. Ibu Desilia Selvida S.Kom., M.Kom selaku Dosen Pembimbing Akademik yang telah memberikan saran, motivasi dan banyak dukungan kepada penulis selama perkuliahan.

7. Seluruh Bapak dan Ibu Dosen Program Studi S-1 Ilmu Komputer, yang telah membimbing penulis selama masa perkuliahan hingga akhir masa studi.
8. Teristimewa kepada Orang Tua penulis Muhammad Irwan dan Julidar yang telah memberikan penulis kasih sayang yang tiada henti, ilmu yang bermanfaat, dan berbagai doa bagi penulis sehingga penulis dapat menjalani perkuliahan dengan baik hingga penyusunan skripsi ini.
9. Saudara tercinta Marwan, Zulfa dan Alif yang selalu mendukung penulis dalam menjalani kehidupan masa kuliah hingga sampai menyelesaikan tugas akhir.
10. Keluarga Besar dari pihak Mama dan Ayah yang selalu percaya, menyemangati dan mendoakan penulis dalam setiap langkah yang diambil untuk mendapatkan gelar S-1.
11. Terima kasi sebesar besarnya kepada teman teman yaitu M.Said Agung, M.Al Fatih Zanqi dan Nabil Siregar yang telah menemani saya dalam proses pengerjaan skripsi.
12. Teman seperjuangan Kaum Mujahirudin yang telah memberikan banyak semangat dan kata-kata mutiara kepada penulis untuk menyelesaikan skripsi ini.
13. Fachriza Adrian dan keluarga yang telah memberikan tempat untuk belajar bersama dalam penyelesaian skripsi.
14. Jefta Steven Filemon Sinaga dan keluarga yang selalu memberikan tempat belajar yang nyaman di luar kampus yang membantu penulis untuk menyelesaikan skripsi ini.

Dan seluruh pihak yang telah memberi dukungan serta doa baik yang tidak dapat penulis sebutkan satu per-satu. Semoga Allah *Subhanahu Wa Ta'ala* senantiasa melimpahkan keberkahan serta kemudahan atas semua dukungan yang telah diberikan kepada penulis dan semoga hasil penelitian ini dapat memberi manfaat maupun inspirasi di masa yang akan datang.

Medan, 9 Juli 2024

Penulis,

Muhammad Ridwan Lubis

ABSTRAK

Permainan *"From The Downtown*" merupakan game bergenre *top-down shooter* yang mengintegrasikan elemen *survival*, petualangan, dan RPG. Penelitian ini difokuskan pada pengembangan karakter musuh dengan menggunakan kecerdasan buatan melalui metode *Finite State Machine* (FSM). Metode FSM dipilih karena mampu memberikan perilaku yang lebih dinamis dan realistis pada karakter musuh sehingga menambah kompleksitas dan tantangan dalam permainan. Dalam pengembangan ini empat tipe karakter musuh utama telah diciptakan *Melee, Range, Chaser*, dan *Charger*. Setiap karakter memiliki perilaku dan kemampuan unik yang di implementasikan melalui FSM. Musuh tipe *Melee* menyerang pemain dengan serangan jarak dekat, musuh tipe *Range* melakukan serangan dari jarak jauh, musuh tipe *Chaser* mengejar pemain, dan musuh tipe *Charger* menyerang dengan menabrak pemain. Dengan penerapan FSM karakter musuh mampu bereaksi secara adaptif terhadap tindakan pemain sehingga menciptakan tantangan yang seimbang dan menarik. Penelitian ini juga melibatkan proses pengujian untuk memastikan bahwa karakter musuh berfungsi sesuai dengan desain dan memberikan pengalaman bermain yang optimal. Implementasi FSM dapat meningkatkan kualitas interaksi dalam game sehingga memberikan kepuasan lebih kepada pemain dari game "*From The Downtown*". Hasil penelitian ini menyimpulkan bahwa penggunaan FSM pada karakter musuh berhasil meningkatkan dinamika permainan dan memberikan tantangan yang lebih bervariasi kepada pemain. Karakter musuh dengan FSM menunjukkan perilaku yang lebih cerdas dan adaptif sehingga pengalaman bermain menjadi lebih menarik dan menantang. Oleh karena itu FSM terbukti efektif dalam pengembangan kecerdasan buatan untuk karakter musuh dalam game *top-down shooter*.

Kata Kunci: *Finite State Machine*, Kecerdasan Buatan, *Game Top Down Shoote*

ABSTRACT

CREATION OF ENEMY CHARACTERS WITH ARTIFICIAL INTELLIGENCE IN THE TOP DOWN SHOOTER GAME “FROM THE DOWNTOWN”

The game "From The Downtown" is a top-down shooter that integrates elements of survival, adventure, and RPG. This research focuses on the development of enemy characters using artificial intelligence through the Finite State Machine (FSM) method. FSM was chosen because it provides more dynamic and realistic behavior to enemy characters adding complexity and challenge to the game. In this development four main types of enemy characters have been created Melee, Range, Chaser, and Charger. Each character has unique behaviors and abilities implemented through FSM. Melee enemies attack players with close-range attacks, Range enemies attack from a distance, Chaser enemies pursue players, and Charger enemies attack by ramming into players. With the application of FSM enemy characters can adaptively react to player actions creating balanced and engaging challenges. This research also involves a testing process to ensure that the enemy characters function as designed and provide an optimal gaming experience. The implementation of FSM enhance the quality of interactions within the game thereby offering greater satisfaction to players of "From The Downtown". The results of this research conclude that the use of FSM in enemy characters successfully increases the dynamics of the game and provides more various challenges for players. Enemy characters with FSM exhibit more intelligent and adaptive behavior making the gameplay experience more engaging and challenging. Therefore, FSM has proven to be effective in the development of artificial intelligence for enemy characters in top-down shooter games.

Keywords: *Finite State Machine, Artificial Intelligence, Game Top Down Shooter*

DAFTAR ISI

PERSETUJUAN ii

PERNYATAAN iii

PENGHARGAAN iv

ABSTRAK vi

ABSTRACT vii

DAFTAR ISI viii

DAFTAR TABEL x

DAFTAR GAMBAR xi

DAFTAR LAMPIRAN xii

BAB 1 PENDAHULUAN 1

1.1 Latar Belakang 1

1.2 Rumusan Masalah 3

1.3 Batasan Masalah 3

1.4 Tujuan Penelitian 3

1.5 Manfaat Penelitian 4

1.6 Metodologi Penelitian 4

1.7 Penelitian Relevan 5

1.8 Sistematika Penulisan 5

BAB 2 LANDASAN TEORI 7

2.1 Desain Karakter Musuh 7

2.2 *Finite State Machine* 8

2.3 *Top-Down Shooter Game* 8

2.3.1 Top-Down 9

2.3.2 Shooter Game 9

BAB 3 ANALISIS DAN PERANCANGAN 11

3.1 Analisis 11

3.1.1 *Analisis masalah* 11

3.1.2 *Analisis kebutuhan* 11

3.2 Arsitektur Umum Sistem 13

3.3 Perancangan Karakter Musuh 14

3.4* Perancangan *Finite State Machine enemy 16

3.4.1 Finite State Machine Enemy Flying Zombie 16

3.4.2 Finite State Machine Enemy Chaser 17

3.4.3 Finite State Machine Enemy Range 18

3.4.4 Finite State Machine Enemy Charger 19

BAB 4 IMPLEMENTASI DAN PENGUJIAN 20

4.1 Implementasi Karakter Musuh 20

4.2 Implementasi *Finite State Machine* 21

4.2.1 Finite State Machine Flying Zombie 21

4.2.2 Finite State Machine Chaser 23

4.2.3 Finite State Machine Range 24

4.2.4 Finite State Machine Charger 26

4.3 Pengujian 27

BAB 5 KESIMPULAN DAN SARAN 34

5.1 Kesimpulan 34

5.2 Saran 34

DAFTAR PUSTAKA 35

DAFTAR TABEL

Tabel 4. 1 Hasil tes pada *Flying Zombie* 30
Tabel 4. 2 Hasil tes pada *Chaser* 31
Tabel 4. 3 Hasil tes pada *Range* 31
Tabel 4. 4 Hasil tes pada *Charger* 32

DAFTAR GAMBAR

Gambar 2. 1 Top Down pada Game Project Zomboid 9
Gambar 2. 2 Game Shooter Left 4 Dead 2 10
Gambar 3. 1 Arsitektur Umum Sistem 13
Gambar 3. 2 Musuh Flying Zombie 14
Gambar 3. 3 Musuh *Chaser* 15
Gambar 3. 4 Karakter Musuh *Range* 15
Gambar 3. 5 Karakter Musuh *Charger* 16
Gambar 3. 6 *Melee Finite State Machine* 16
Gambar 3. 7 *Chaser Finite State Machine* 17
Gambar 3. 8 *Range Finite State Machine* 18
Gambar 3. 9 *Charger Finite State Machine* 19
Gambar 4. 1 Objek Musuh 20
Gambar 4. 2 *Animator Controller Musuh* 21
Gambar 4. 3 *AIBrain* dari *Flying Zombie* 22
Gambar 4. 4 *AIBrain* dari *Chaser* 23
Gambar 4. 5 *AIBrain* dari *Range* 25
Gambar 4. 6 *AIBrain* dari *Charger* 26
Gambar 4. 7 Tampilan awal *player* 28
Gambar 4. 8 Pengujian musuh *Flying Zombie* 28
Gambar 4. 9 Pengujian musuh *Chaser* 29
Gambar 4. 10 Pengujian musuh *Range* 29
Gambar 4. 11 Pengujian musuh *Charger* 30

DAFTAR LAMPIRAN

Lampiran 1 Kode Program 28
Lampiran 2 Biodata 38

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Pengguna mendapatkan hiburan dan tantangan dari permainan game. Tantangan membuat game menjadi mudah diselesaikan dan cepat membosankan jika tidak ada. Namun, jika tantangan terlalu sulit, pemain dapat menjadi frustrasi dan menyerah dengan cepat. Ini berhubungan dengan konsep *flow-state machine*, yang menekankan bahwa kemampuan pemain dan tingkat tantangan dalam game harus seimbang. Biasanya, setiap game menawarkan berbagai Pilihan tingkat kesulitan untuk memberikan tantangan kepada pemain, seperti pilihan mudah, sedang , dan sulit . Sayangnya, model pengaturan ini cenderung statis, yang dapat menyebabkan ketidakcocokan antara kemampuan pemain dan tantangan yang dihadapinya dalam game. (Sofyan et al., 2019).

NPC (*Non-Player Character*) pada game memiliki peran penting yang dapat mempengaruhi kesulitan dan kompleksitas di dalam game. Untuk itu NPC harus diberikan *behavior* atau kebiasaan yang dapat membuat karakter tersebut bergerak sendiri sesuai dengan keadaan-keadaan tertentu. Hal ini juga bergantung pada enemy atau musuh yang terdapat pada game, semakin kompleks *behavior* musuh tersebut maka akan membuat game semakin menarik untuk dimainkan. Dengan menambahkan *behavior* ke dalam NPC pada game membuat game memiliki kehidupan tersendiri yang dibangun di dalamnya. Dalam penelitian yang dibuat oleh Muhammad Luthfi Setiawan, Arbansyah dan Sayekhti Harits Suryawan dalam pembuat game tersebut pergerakan pada NPC (*Non-Player Character*) yang dibuat hanya mentargetkan pemain dan tidak memiliki tindakan atau keputusan yang jelas, sehingga terlihat tidak jelas. Dalam situasi yang menargetkan pemain, NPC dirancang untuk memilih jalur terdekat dan menghindari objek yang ada di map game. (Setiawan et al., 2023).

Saat ini, dalam game online satu pemain, pemain harus melawan musuh yang dikendalikan oleh komputer atau AI, yang membuat permainan menjadi lebih menantang. Dalam penelitian yang ditulis oleh Wennedy Surya Pratama,

Amak Yunus, dan Wasum, karakter musuh dapat mengambil keputusan berdasarkan tindakan pemain. Beberapa pilihan tindakan musuh adalah tidak menyerang pemain yang berada di luar jangkauan mereka, dan menyerang pemain yang memasuki jangkauan tersebut, yang akan mengurangi poin pemain jika terkena serangan (Wennedy et al., 2019).

Kemudian pada penelitian yang dilakukan oleh Fahmi Nurfauzi, Purba Daru Kusuma dan Ashri Dinimahawarati dalam game tersebut menerangkan terdapat karakter NPC bernama Eric yang memiliki behavior yang dapat diam, mengikuti pemain, menyerang, menunggu dan diam untuk melacak para *player* ketika berada dalam jarak yang jauh dan ketika *player* mendekati jarak eric maka eric akan bertindak dengan algoritma yang sudah dirancang (Fahmi et al., 2023).

Video game "*From The Downtown*" yang akan dikembangkan di dalam penelititan ini dengan perspektif *top-down shooter*. *Top-down shooter* merupakan jenis video game bertemakan perang dan mengalahkan musuh dengan menggunakan alat senjata dimana pemain mengontrol karakter dengan sudut pandang dari atas karakter (Adams, 2010). Perspektif *top down* adalah permainan di mana kamera mengarah ke bawah, memberikan tampilan yang menyerupai peta. Perspektif ini sangat berguna dalam permainan bergenre RPG (*Role Playing Game*) karena memungkinkan pemain melihat berbagai elemen permainan secara luas, memberikan banyak informasi, dan membantu dalam pengambilan keputusan (Johny et al., 2021).

"*From The Downtown*" merupakan vidio *game* bergenre *survival*, pertualangan dan RPG, pada genre petualangan terdapat banyak game yang memilih tema tersebut sehingga mendapatkan minat yang cukup tinggi untuk dimainkan, genre tersebut diminati karena dalam permainan memiliki alur cerita yang akan diperani oleh pemain dan pertualangan di identikkan dengan pencarian barang, eksplorasi dan pemecahan teka-teki (Alaa, 2020). pengembangan game "*From The Downtown*" ini akan dibangun karakter dengan game bergenre pertualangan yang berupa melawan *zombie* yang dapat bergerak dengan cepat ketika bertemu dengan pemain, menembakkan racun pada pemain atau perilaku lainnya.

1.2 Rumusan Masalah

Dalam Permainan *Survive* biasanya memiliki karakter yang berbeda-beda, terdapat karakter di beberapa game yang mempunyai sifat karakteristik yang sama sehingga para pemain mudah mengetahui sifat dari musuh ke pemain. Maka dari itu pada penelitian ini ingin lebih meningkatkan lagi karakteristik musuh yang lebih terjangkau seperti musuh memiliki sifat yang berbeda-beda dan musuh mempunyai skillnya masing masing.

1.3 Batasan Masalah

Batasan masalah dalam penelitian ini adalah sebagai berikut:

1. Penelitian ini akan membuat 4 tipe karakter musuh yaitu *Melee* menyerang *player* dengan memukulnya, *Range* menyerang *player* dari jarak jauh, *Chaser* mengejar *player* dan memukulnya, *Charger* musuh akan menabrak *player* dengan *Damage* cukup besar.
2. Pada penelitian ini pengembangan karakteristik musuh untuk game *Top-Down Shooter "From the Downtown"* pada *platform* berbasis PC.
3. Game ini akan menggunakan *Top-Down Shooter*.
4. Penelitian ini dimainkan *single player*.
5. Game dirancang dengan Game *Engine Unity* dan bahasa pemograman C#.
6. Pada penelitian ini menggunakan *Finite State Machine.*

1.4 Tujuan Penelitian

Penelitian ini bertujuan mengembangkan karakteristik musuh dalam meningkatkan pengalaman bermain dan merancang musuh memiliki kekuatan, kelemahan, perilaku yang berbeda- beda bertujuan memperkaya naratif atau cerita dalam game Pemain akan merasa tertarik untuk memainkan kembali permainan.

1.5 Manfaat Penelitian

Adapun meningkatkan pengalaman bermain bagi pemain dengan mengembangkan musuh-musuh yang menarik, menantang, dan dinamis. Hal ini dapat mencakup pembuatan musuh-musuh dengan perilaku yang cerdas, kekuatan dan kelemahan yang unik dan menyediakan tantangan yang seimbang bagi pemain, di mana musuh-musuh tidak terlalu mudah atau terlalu sulit untuk diatasi.

1.6 Metodologi Penelitian

Beberapa metode yang diterapkan dalam penelitian ini adalah sebagai berikut:

1. Studi Pustaka

   Penelitian ini dimulai dengan mencari referensi dari berbagai sumber terpercaya dan melakukan peninjauan pustaka melalui buku-buku, jurnal, *e-book*, artikel ilmiah, makalah ataupun situs internet yang berhubungan dengan *Game Development*, *Enemy Behavior*, *Top Down Shooter* dan *Unreal Engine.*
2. Analisa dan Perancangan

   Tahap ini dilakukan sebuah perancangan dari game yang akan dibuat sesuai dengan permasalahan dan perancangan game. Perancangan game ini akan dirancang dengan pembuatan *flowchart* dan *prototype game.*
3. Implementasi

   Tahap ini, membuat sebuah game dengan menggunakan *Unity* serta aset yang ada di *Unity* seperti *More Mountains*, dan menggunakan bahasa pemrograman C# sesuai dengan diagram alir yang telah dirancang.
4. Pengujian

   Tahap ini, dilakukan uji coba peningkatan karakteristik pada musuh yang telah dibuat berhasil dijalankan, serta mampu menjalankan game tersebut dengan baik.
5. Dokumentasi

   Penelitian yang telah dilakukan, didokumentasikan mulai dari tahap analisa sampai kepada pengujian dalam bentuk skripsi.

1.7 Penelitian Relevan

Beberapa penelitian terdahulu yang relevan dengan penelitian ini, antara lain:

1. Berdasarkan analisis oleh Yulia Windi Astuti et al. dengan judul "Perilaku *Non Player Character* (NPC) Pada Game FPS" "*ZOMBIE COLONIAL WARS*" Menggunakan *Finite State Machine* (FSM) ", disimpulkan bahwa perilaku NPC ditentukan oleh interaksi yang dilakukan pemain selama bermain menggunakan metode *finite state machine*. Beberapa contoh situasi yang terjadi yaitu saat pemain terlihat oleh musuh dan masuk dalam radius serang, musuh akan bergerak menuju pemain dan menyerangnya. Selain itu, ketika musuh terkena tembakan dari pemain, musuh akan mati (Astuti et al., 2019).
2. Dalam jurnal " Kajian Desain Kostum *Game Life after* Upaya Mengeksplorasi Gameplay yang Imersif " oleh Natasha Laurentia dan Toto Haryadi, terdapat penekanan pada pentingnya imersi dalam gameplay. Imersi adalah perasaan untuk benar-benar terlibat dan merasakan keberadaan di dalam dunia game sehingga pemain terlibat secara emosional dalam memainkan game tersebut dan pemain akan merasakan candu terhadap gameplay yang membuat pemain memainkan game dengan waktu yang lama (Laurentia & Haryadi, 2021).

1.8 Sistematika Penulisan

Sistematika penulisan skripsi yang digunakan dalam penelitian ini adalah sebagai berikut:

BAB 1 PENDAHULUAN

Bab ini mencakup penjelasan mengenai latar belakang pemilihan judul, rumusan dan batasan masalah, tujuan, manfaat, dan metodologi penelitian, penelitian relevan, dan sistematika penulisan skripsi.

BAB 2 LANDASAN TEORI

Bab ini menjelaskan beberapa teori yang berkaitan dengan penelitian, yaitu *Finite State Machine*, dan *Top-Down Shooter Game.*

BAB 3 ANALISIS DAN PERANCANGAN

Bab ini menjelaskan mengenai analisis dan dilakukan perancangan diagram yang diperlukan, seperti diagram alir (*flowchart*).

BAB 4 IMPLEMENTASI DAN PENGUJIAN

Bab ini berisi penjelasan mengenai implementasi sistem ke dalam game yang kemudian dilakukan pengujian.

BAB 5 KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan yang dapat diperoleh berdasarkan pemaparan pada setiap bab serta saran yang diberikan peneliti sebagai masukan untuk penelitian selanjutnya.

BAB 2

LANDASAN TEORI

2.1 Desain Karakter Musuh

Desain karakter merupakan langkah langkah menciptakan karakter yang bersifat fiksi dan memiliki penampilan visual yang unik maupun karakteristik yang khas. Desain karakter biasanya digunakan dalam berbagai media seperti buku, film, animasi, komik, dan permainan video untuk membantu pengguna dalam memahami karakter tersebut dan menjalin koneksi individual dengan mereka. Desain karakter yang bagus harus mampu mencerminkan sifat, kepribadian, dan konflik yang relevan dengan penggunanya, sehingga karakter tersebut dapat menjadi menarik dan mudah dikenali.

Pada karakter yang dalam jurnal "Analisis Semiotika Desain Karakter *Silverash* Pada Game *Arknights*", dimana karakter dari permainan video *Arknights*. *SilverAsh* tersebut merupakan karakter yang memiliki pengaruh terhadap mitologis dari hewan seperti macan tutul salju dan hewan tersebut memiliki sifat yang kuat, tegas, dan pemimpin. Analisis dalam desain karakter SilverAsh dilakukan menggunakan teori Roland Barthes, Roland Barthes mengungkap makna-makna yang terkandung dalam karakter tersebut berdasarkan aspek visualnya, seperti bentuk, warna, gestur, dan prinsip desain lainnya, yang menjadikan karakter tersebut mempunyai berbagai macam sifat (Waluyo & Patria, 2022). Game "*From The Downtown*" sendiri mempunyai beberapa sifat yang dimiliki musuh untuk menghadapi para pemain. Selama dalam permainan musuh mempunyai 4 sifat yaitu *Flying Zombie, Range, Chaser, Charger* dimana pemain nantinya akan berhadapan dengan musuh-musuh tersebut.

2.2 *Finite State Machine*

Dalam game, *finite state machine* adalah metode perancangan perilaku musuh yang menggunakan tiga elemen utama: keadaan, kejadian, dan perilaku. Musuh dapat beralih ke keadaan lain ketika menerima input tertentu, baik dari perangkat eksternal maupun dari kemampuan musuh itu sendiri. Dalam kebanyakan kasus, transisi ini diikuti oleh tindakan yang dilakukan oleh musuh sebagai tanggapan terhadap input tersebut. Tindakan ini dapat berupa tindakan sederhana atau terdiri dari rangkaian proses yang cukup kompleks.

Terdapat beberapa metode dalam pembuatan game 2D yang menggunakan AI (*Artificial Intelligence*) yaitu :

1. Penggunaan metode *Finite State Machine* (FSM) dalam pembuatan sebuah game selama permainan akan mengatur perilaku karakter musuh.
2. Implementasi kecerdasan buatan (AI) pada karakter musuh dapat meniru perilaku manusia dan berinteraksi dengan pemain secara realistis.
3. Pengujian AI (*Artificial Intelligence*) pada game memastikan fungsi *Finite State Machine* (FSM) berjalan dengan tingkat keberhasilan 100%.
4. Penerapan FSM digunakan untuk mendeteksi keberadaan karakter dalam game.
5. *Finite State Machine* digunakan untuk mendukung interaksi antara karakter *player* dan musuh dalam game.

Beberapa metode tersebut memberikan gambaran bagaimana tentang sistem AI bekerja, khususnya metode *Finite State Machine*, digunakan dalam pembuatan game untuk menciptakan pengalaman bermain yang menarik dan realistis (Rumakey et al., 2020).

2.3 *Top-Down Shooter Game*

Top-Down Shooter Game merupakan salah satu genre yang ada dalam video game. Penjelasan *Top-Down Shooter Game* dapat dibagi menjadi dua bagian yaitu *Top-down dan Shooter Game*.

2.3.1 Top-Down

Top-down dalam game mengacu pada desain dan pemrograman dari sudut pandang mata burung ataupun dari atas. Ini umumnya digunakan dalam game strategi, memungkinkan pemain untuk mengelola sumber daya dan unit di area yang luas. Pendekatan ini berfokus pada strategi keseluruhan dalam desain dan membangun game, serta memungkinkan pengembangan karakter untuk berinteraksi dengan dunia game dari perspektif yang sama. Ini berguna untuk game strategi dan lainnya yang memerlukan pemahaman menyeluruh tentang dunia game (Adams, 2010). Contoh "game *top-down* adalah *Project Zomboid* seperti pada Gambar 2.1 dibawah ini".

Gambar 2. 1 *Top Down* pada *Game Project Zomboid*

2.3.2 Shooter Game

Shooter game adalah jenis permainan video yang berfokus pada penembakan atau pertempuran sebagai mekanisme utama, di mana pemain mengendalikan karakter bersenjata untuk mengalahkan musuh, menyelesaikan misi, atau mencapai tujuan tertentu. Permainan ini dapat dibedakan berdasarkan perspektif pemain seperti *first-person shooter* (FPS) yang dimainkan dari sudut pandang karakter dan *third-person shooter* (TPS) yang menampilkan karakter dari sudut pandang orang ketiga.

Selain itu, *shooter game* juga bervariasi berdasarkan latar dan genrenya termasuk *Shooter*, fiksi ilmiah, atau horor (Liu et al., 2021). Contoh "game *shooter* adalah *Left 4 Dead 2* seperti pada Gambar 2.2 dibawah ini".

Gambar 2. 2 Game *Shooter Left* 4 *Dead* 2

BAB 3

ANALISIS DAN PERANCANGAN

3.1 Analisis

Tahap analisis merupakan awal dari sebelum melakukan sebuah perancangan dan pengembangan dari penelitian. Pada tahap ini, peneliti dapat memahami beragam masalah yang dihadapi selama pembuatan musuh. Pada pembuatan musuh adanya kecerdasan buatan (AI) yang akan digunakan sebagai sifat dari musuh sehingga musuh akan berinteraksi dengan *player*, adanya pergerakan musuh berjalan dengan lancer disebabkan oleh *Finite State Machine* yang akan mengatur proses pergerakan musuh. Dengan Tahap ini penting agar sistem yang dikembangkan dapat memenuhi kebutuhan yang ada dan mencapai tujuan akhir yang diinginkan.

3.1.1 Analisis masalah

Adanya video game dengan genre *top-down shooter*, kehadiran musuh yang cerdas dapat secara penting untuk meningkatkan kualitas sebuah permainan. Pada musuh memiliki peran penting dalam menjalankan karakter sebagai tantangan kepada *player*. Musuh memberikan tantangan yang beragam macam, tidak hanya dari segi kekuatan, tetapi juga adanya kecerdasan untuk mengantisipasi gerakan dan serangan pemain. Kecerdasan buatan (AI) pada musuh dalam game dapat menghadirkan pengalaman lebih menantang bagi *player*.

3.1.2 Analisis kebutuhan

Pada karakter musuh untuk mengimplementasikan *Finite State Machine* dibutuhkan sebagai sistem dapat berjalan dengan lancar. Ada dua komponen utama dalam analisis kebutuhan: kebutuhan fungsional dan non-fungsional.

1. Kebutuhan fungsional

   Kebutuhan fungsional Pada Game adanya *finite state machine* yaitu agar musuh dapat bergerak berbagai sifat yang diperlukan agar musuh dapat berperilaku sesuai dengan alur permainan. Beberapa kebutuhan fungsional pada musuh:

a. Musuh harus bisa mendeteksi keberadaan pemain dalam radius tertentu.
b. Pemain yang terdeteksi oleh musuh harus bergerak untuk mengejar pemain.
c. Ketika berada dalam jarak serangan, musuh harus beralih dengan menyerang pemain.
d. Ketika pemain tidak terdeteksi, musuh harus bergerak dalam mode patroli.
e. Ketika Musuh dalam kondisi mati harus bisa beralih ke keadaan mati ketika darah mereka tidak ada.

Dalam kebutuhan *Finite State Machine* membantu agar musuh tetap berjalan dengan sesuai alur dan mengatur prilaku yang sesuai ketika game sedang berjalan.

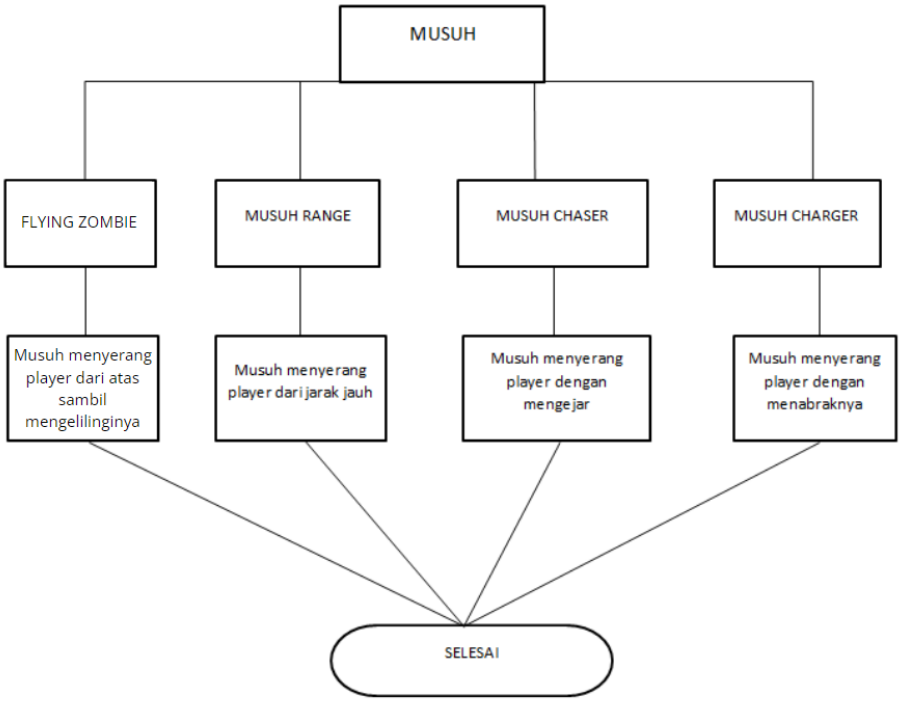
2. Kebutuhan non-fungsional

Kebutuhan non-fungsional untuk musuh adanya cakupan pada aspek-aspek yang tidak berhubungan dengan fungsi spesifik, tetapi lebih ke kualitas dan performa sistem. Berikut adalah beberapa kebutuhan non-fungsional yang penting:

a. Kinerja pada game memastikan tidak ada lag atau penurunan frame rate selama permainan.
b. Pada tahap menangani sejumlah besar musuh secara bersamaan tanpa adanya mengurangi performa game.
c. *Finite State Machine* harus bekerja secara konsisten, tanpa mengalami kegagalan atau bug yang mengganggu *gameplay*.
d. Musuh merespon input dan kejadian dalam game secara *real-time* atau dalam waktu yang sangat singkat, memberikan pengalaman bermain yang mulus.

3.2 Arsitektur Umum Sistem

Rangkaian alur pada karakter musuh merupakan rancangan arsitektur umum yang merupakan gambaran dari keseluruhan sistem yang akan dibangun sehingga setiap karakter musuh memiliki sifatnya masing masing, berikut arsitektur pada kecerdasan musuh :



Gambar 3. 1 Arsitektur Umum Sistem

Terdapat pada gambar arsitektur umum, dimana musuh mempunyai skillnya masing masing seperti musuh *Flying Zombie* dengan skillnya ketika musuh berhadapan dengan *player* maka musuh akan menyerangnya dari jarak atas sambal mengelilinginya, kemudian musuh *Chaser* berinteraksi dengan *player* akan mengejar dengan lari yang sangat kencang sehingga *player* mengalami gangguan ketika dikejar oleh musuh, pada musuh Range menyerang *player* dengan cara mengeluarkan racun dari mulutnya sehingga ketika terkena racun tersebut *player* akan terkena *Damage* yang perlahan mengurang, untuk musuh Charger skillnya berbeda dari ketiga musuh tersebut yang dimana Charger akan mendatangin *player* dengan cara melompat ke arah *player* yang dapat mengurangi *Damage* yang cukup sakit.

3.3 Perancangan Karakter Musuh

Pada dalam permainan game akan dibuat 4 jenis musuh yang setiap karakternya mempunyai sifat berbeda beda. Setiap karakter mempunyai jaraknya masing masing yaitu setiap musuh memiliki jarak lari dan jarak serang yang berbeda sehingga ketika dalam permainan musuh akan melakukan interaksi kepada pemain.

Pada game ini terdapat 4 musuh yaitu *Flying Zombie, Chaser, Range dan Charger*.

a. Flying Zombie

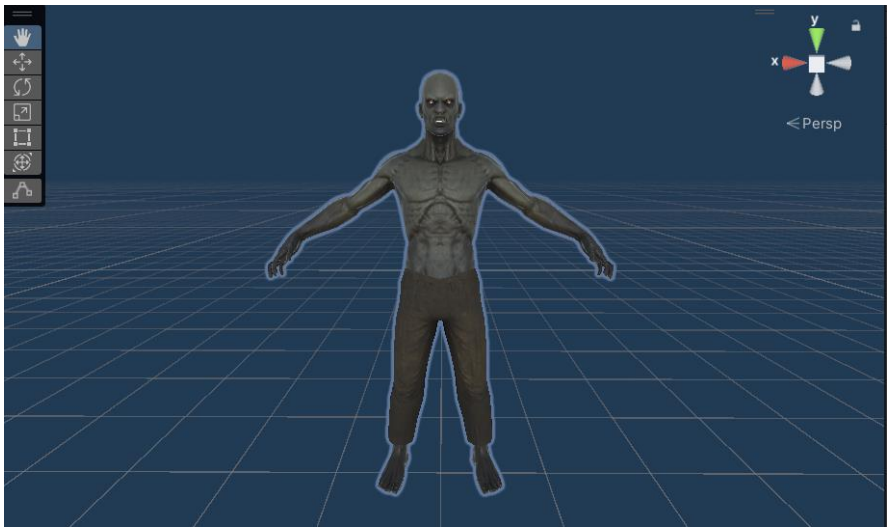
Musuh *Flying Zombie* merupaka karakter musuh dengan skill menyerang secara terbang mengelilingi *player*. Ketika *player* terkena zona dari *Flying Zombie* maka musuh akan terus mengelilingi *player* dan menembakkan racun ke *player*. Berikut dari "tampilan *Flying Zombie* pada gambar 3.2 dibawah ini".

Gambar 3. 2 Musuh *Flying Zombie*

b. Chaser

Musuh *chaser* yaitu menyerang pemain dengan jarak cukup dekat. *Chaser* memiliki sifat yang lebih agresif ketika *player* dalam jarak

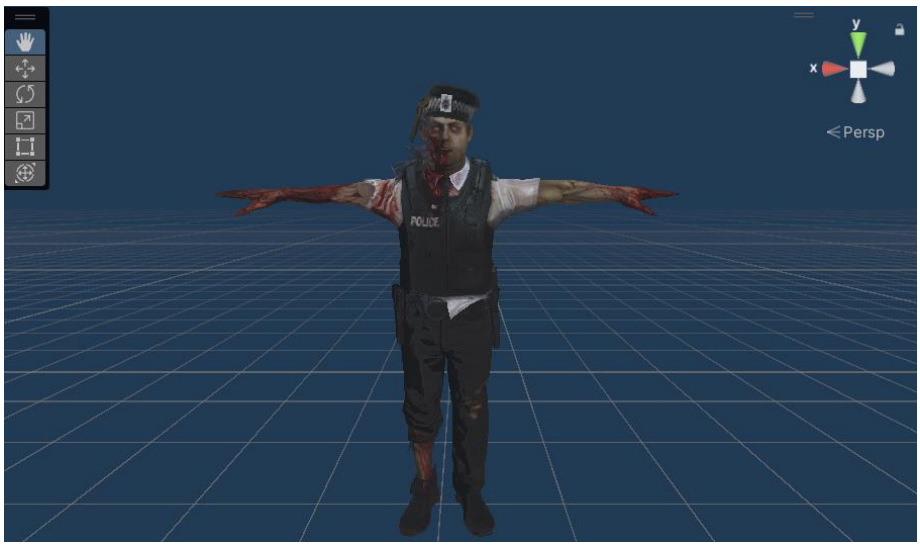
jangkauannya maka *Chaser* akan berlari dengan sangat cepat agar musuh bisa memukul *player*. Berikut “merupakan musuh *Chaser* pada gambar 3.3 dibawah ini”.



Gambar 3. 3 Musuh *Chaser*

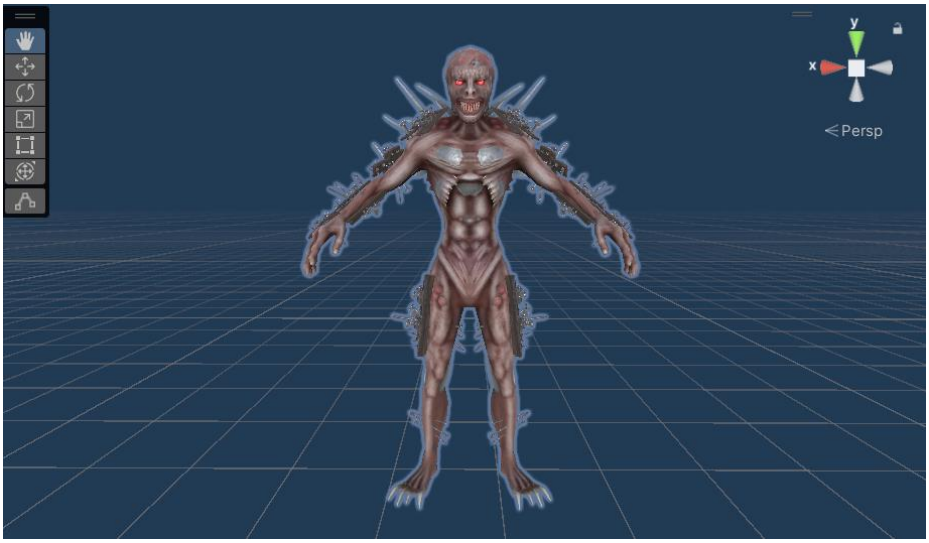
c. *Range*

Range mempunyai musuh yang menyerang *player* dengan jaraknya yang cukup jauh. Apabila *player* terkena dari *Range* maka musuh tersebut akan menyemberurkan cairan beracun dari mulutnya ke *player* dan ketika terkena racun darah dari *player* akan berkurang secara perlahan.



Gambar 3. 4 Karakter Musuh *Range*

d. *Charger*

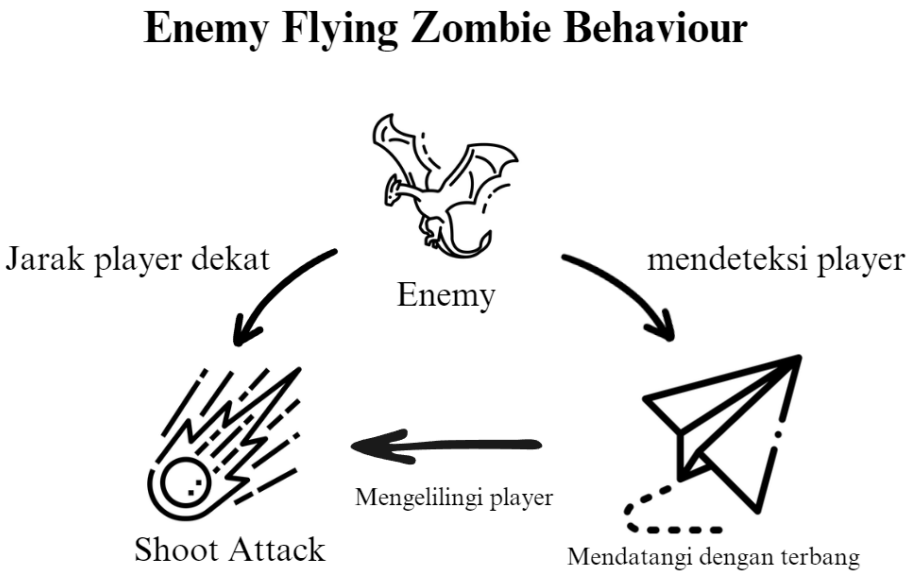


Gambar 3. 5 Karakter Musuh *Charger*

Charger yaitu musuh dengan skill menyerang *player* dengan jarak jangkauan yang sedang, dengan jarak sedang *Charger* akan mengejar *player* dan untuk jarak akurat pada *Charger* akan ancang-ancang sebelum lompat dan setelah itu *Charger* akan langsung melompat ke arah *player*.

3.4 Perancangan *Finite State Machine enemy*

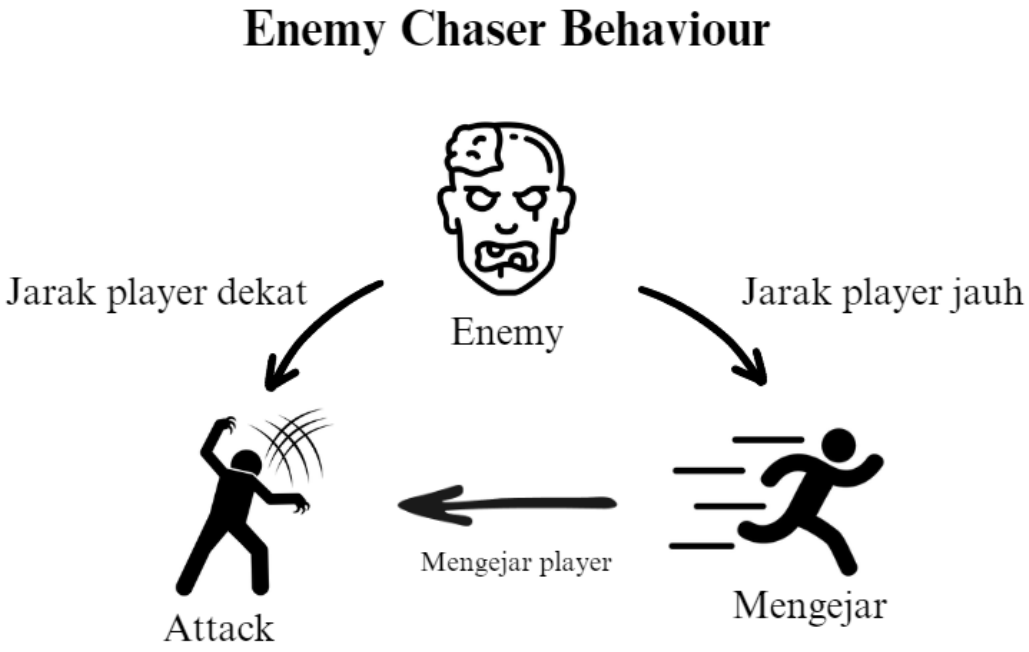
3.4.1 Finite State Machine Enemy Flying Zombie



Gambar 3. 6 *Melee Finite State Machine*

"Gambar 3.5 merupakan perilaku musuh dengan serangan jarak atas" Ketika musuh mendeteksi pemain, jika jarak pemain dekat, musuh akan melakukan serangan dengan menembakkan racun (*shoot attack*) sambal mengikuti pemain dengan mengelilinginya. Namun, jika jarak pemain jauh, musuh akan mendatangi pemain dengan terbang. Dengan demikian, perilaku musuh ini bergantung pada jarak antara pemain dan musuh, dengan dua kondisi utama: serangan jarak dekat dan pendekatan terbang dari jarak jauh.

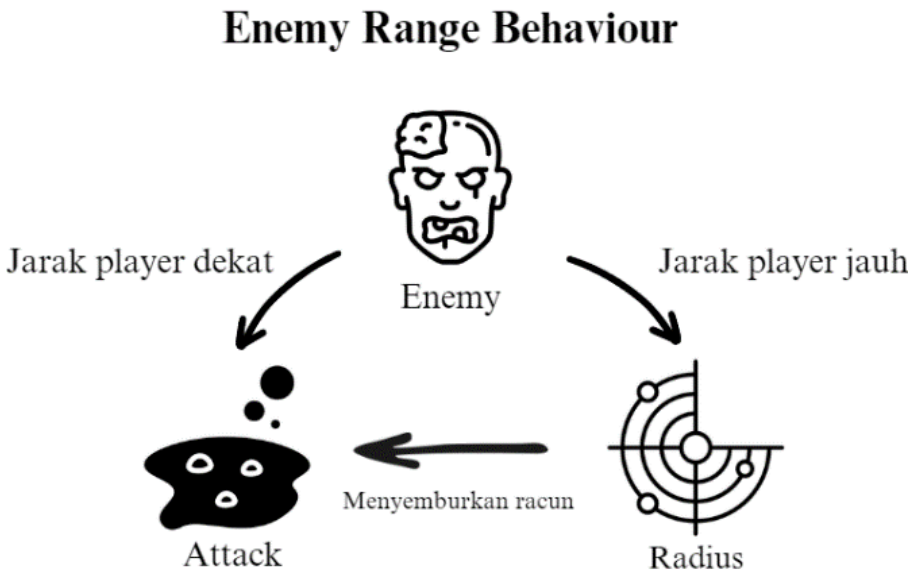
3.4.2 Finite State Machine Enemy Chaser



Gambar 3. 7 *Chaser Finite State Machine*

"Gambar 3.7 merupakan perilaku musuh mengejar" dalam permainan, dimana tindakan musuh dipengaruhi oleh jarak antara musuh dan pemain. Jika jarak antara musuh dan pemain dekat, musuh akan menyerang pemain sedangkan jika jaraknya jauh musuh akan mengejar pemain. Dengan demikian "gambar 3.7 menunjukkan reaksi musuh berdasarkan jarak pemain", menyerang ketika dekat dan mengejar ketika jauh.

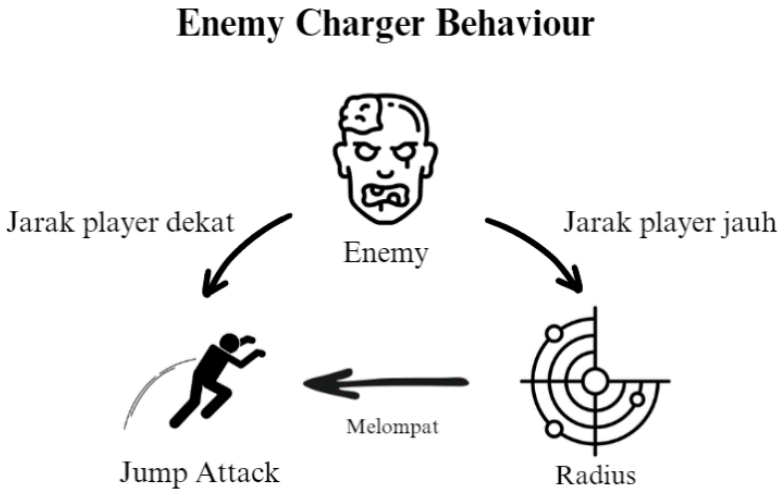
3.4.3 Finite State Machine Enemy Range



Gambar 3. 8 *Range Finite State Machine*

"Gambar 3.8 merupakan perilaku musuh dengan serangan jarak jauh" *enemy range behaviour* dalam permainan dimana tindakan musuh dipengaruhi oleh jarak antara musuh dan pemain. Jika jarak antara musuh dan pemain dekat, musuh akan menyerang pemain dengan menyemburkan racun. Jika jarak antara musuh dan pemain jauh, musuh akan melakukan serangan area atau radius. "Gambar 3.8 menunjukkan reaksi musuh berdasarkan jarak pemain", menyemburkan racun ketika pemain berada dekat dan menyerang dalam radius tertentu ketika pemain berada jauh.

3.4.4 Finite State Machine Enemy Charger



Gambar 3. 9 *Charger Finite State Machine*

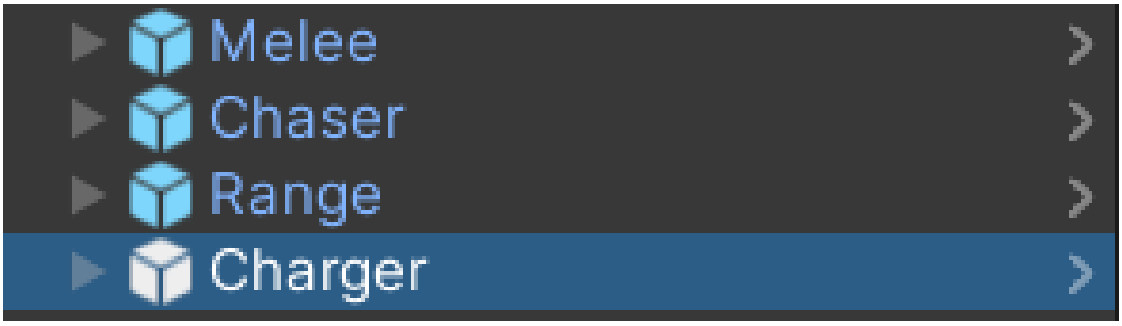
"Gambar 3.9 merupakan perilaku musuh bertipe *Charger* dalam permainan berdasarkan jarak pemain". Jika pemain berada dalam jarak dekat musuh akan melakukan serangan melompat *Jump Attack*. Sebaliknya jika pemain berada dalam jarak jauh, musuh akan bergerak mendekati pemain yang ditunjukkan dengan ikon radius. Perilaku ini menunjukkan bahwa musuh memiliki dua respons utama, melompat menyerang ketika pemain dekat dan bergerak mendekati pemain ketika jaraknya jauh.

BAB 4

IMPLEMENTASI DAN PENGUJIAN

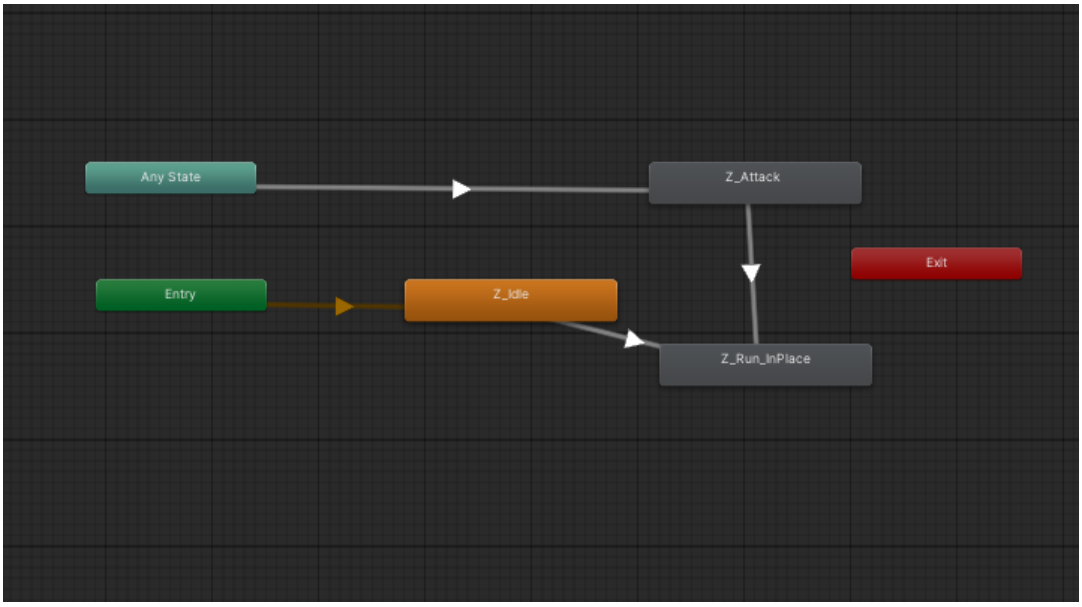
4.1 Implementasi Karakter Musuh

Tahap selanjutnya yaitu implementasi pembuatan karakter musuh. Tahap pertama dalam pembuatan yaitu import 3D dari empat musuh yang akan dibuat. Kemudian, "masukkan objek musuh yang dinamakan *Flying Zombie, Chaser, Range, Charger* seperti pada gambar 4.1 dibawah ini".



Gambar 4. 1 Objek Musuh

"Pada gambar 4.1 menunjukkan urutan setiap karakter musuh yang dibuat" selama pembuatan musuh dapat membedakan setiap musuhnya. Kemudian, setelah memasukkan setiap objek dari musuh tahap selanjutnya mengatur animasi yang dibutuhkan selama pembuatan karakter. Mengatur animasi musuh dapat dibuat di *Animator Controller*, Setiap animasi yang dibuat harus dihubungkan dengan *Transition*.



Gambar 4. 2 *Animator Controller Musuh*

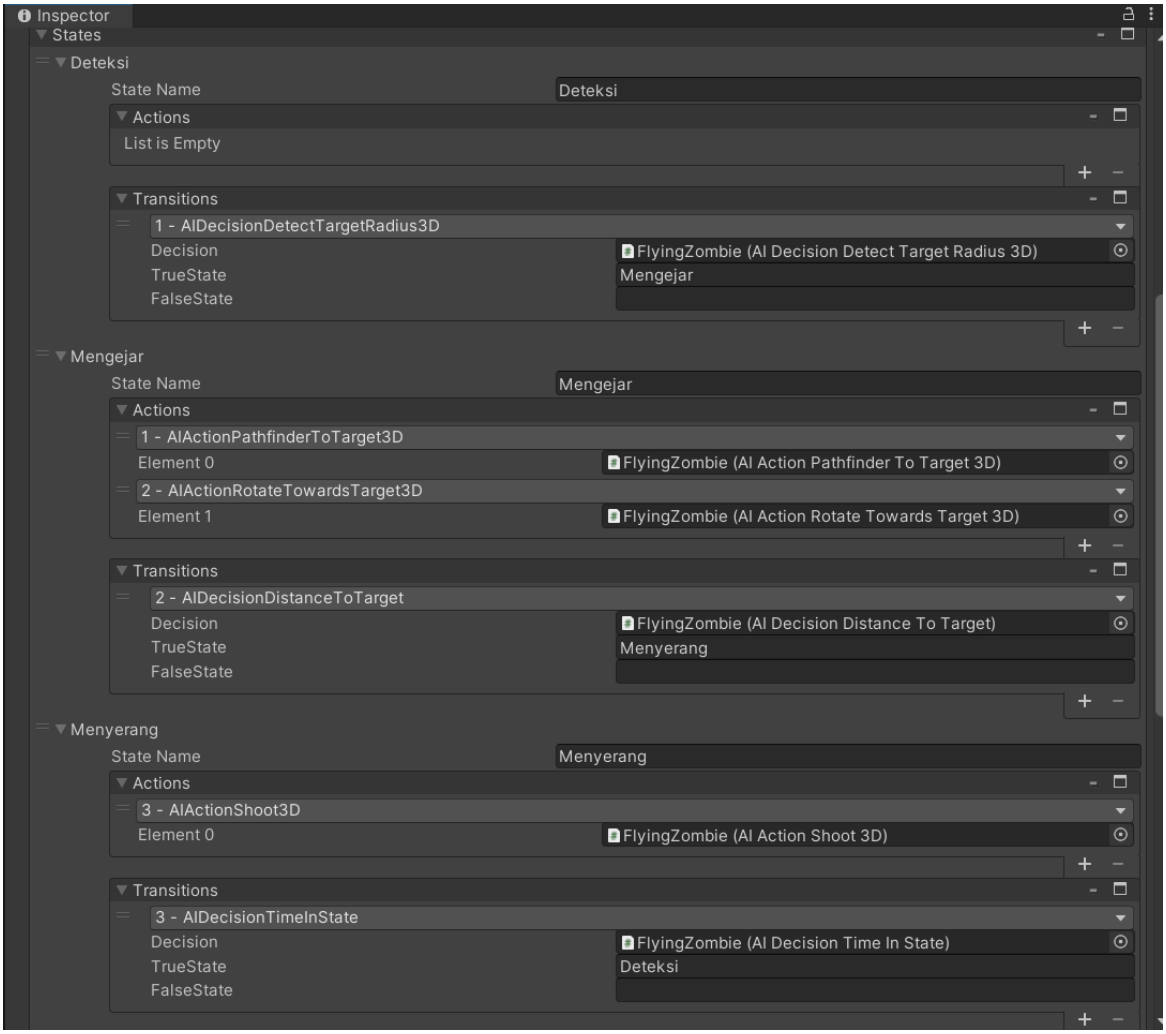
"Pada gambar 4.2 animasi yang dihubungkan oleh *Transition* yang bekerja sebagai transisi dari setiap kegiatan musuh", setiap *Attack* diawali dengan *Any state* yang berfungsi menjalankan musuh ketika *Attack* sedang berada di *State* apapun. Pada *State Entry* terhubung dengan *Idle* yang menandakan *State* tersebut *Default*.

4.2 Implementasi *Finite State Machine*

Dalam mengatur Kegiatan perilaku setiap karakter, terdapat kerangka kerja pada *TopDown Engine* menyediakan komponen yang bernama *AIBrain. AIBrain* dapat menggunakan cara kerja dari *Finite State Machine* dimana AIBrain tersebut terdiri dari kumpulan state yang terhubung dengan transisi.

4.2.1 Finite State Machine Flying Zombie

AIBrain pada *Flying Zombie* dapat ditambahkan beberapa state, yaitu deteksi, mengejar, menyerang. Tampilan pada komponen *AIBrain* untuk setiap musuh memiliki kesamaan beberapa komponen dan dapat dilihat pada "Gambar 4.3 *AIBrain* dari musuh *Flying Zombie*".



Gambar 4. 3 *AIBrain* dari *Flying Zombie*

"Terdapat gambar 4.3 dimana ketiga musuh memiliki *AIBrain* yang sama", kemudian setiap *state* masukkan *action* dan *transition* yaitu:

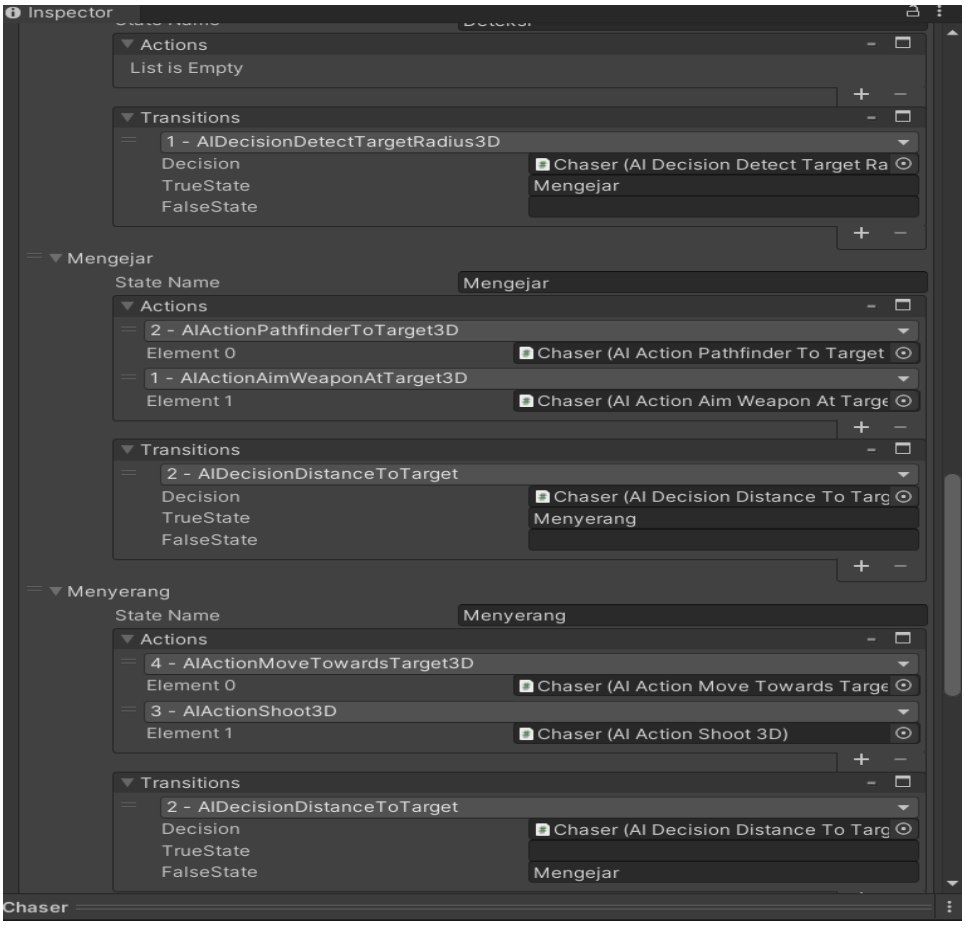
1. Deteksi
   - *Action* : Musuh mencari keberadaan *player* dengan ketika bermain.
   - *Transition* : Setelah *player* terdeteksi dengan radius maka musuh mulai terbang dengan radius 500 untuk mengejar.
2. Mengejar
   - *Action* : Ketika *player* terdeteksi oleh musuh maka musuh langsung mengejar.

- Transition : setelah musuh mendapat *player* maka musuh mulai menyerang dengan jarak *Distance* sebesar 1.5.

3. Menyerang
   - *Action* : Ketika *player* terdeteksi maka musuh langsung menyerang *player* dengan cara mencakarnya dengan terus menerus.
   - *Transision* : Lanjut ke deteksi untuk mengejar kembali.

4.2.2 Finite State Machine Chaser

AIBrain pada *Chaser* memiliki beberapa *state*, yaitu deteksi, mengejar dengan speed melebihi dari *Melee* dan menyerang. Tampilan pada komponen *AIBrain* untuk setiap musuh memiliki kesamaan komponen *Melee* dan dapat dilihat "pada Gambar 4. 4 dibawah ini".



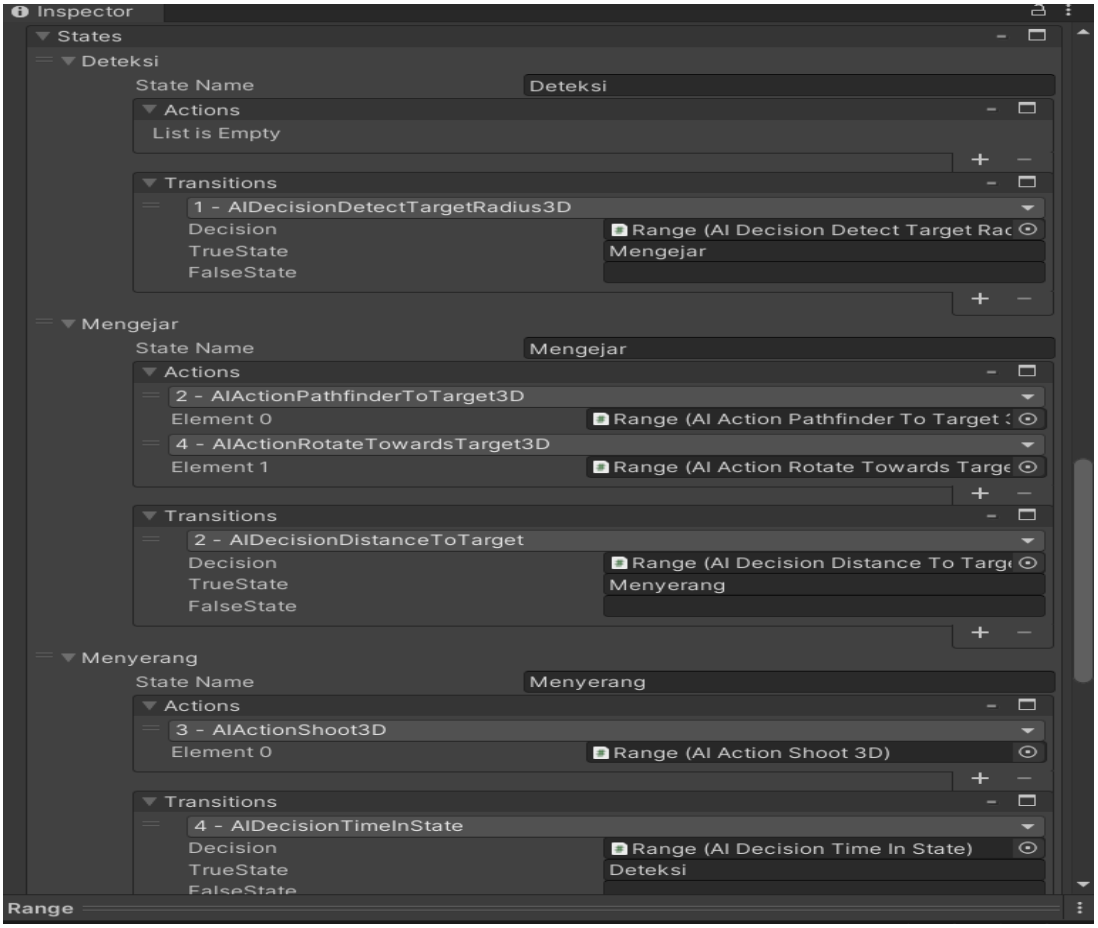
Gambar 4. 4 *AIBrain* dari *Chaser*

Pada tiap *state* kita masukkan *action* dan juga *transition*. Adapun detail *action* dan *transitions* sebagai berikut:

1. Deteksi
    - *Action* : Musuh mencari keberadaan *player* ketika bermain.
    - *Transition* : Setelah *player* terdeteksi maka musuh mulai mengejar.
2. Mengejar
    - *Action* : Ketika *player* terdeteksi oleh musuh maka musuh langsung mengejar dengan *Speed Walk 9*.
    - *Transition* : Setelah musuh mendapat *player* maka musuh mulai menyerang.
3. Menyerang
    - *Action* : Musuh menyerang *player* dengan mencakarnya sehingga dapat mengurangi Health Point *player*.
    - *Transition* : Musuh selalu mendeteksi *player* untuk selalu mengikutinya.

4.2.3 Finite State Machine Range

AIBrain pada *Range* memiliki beberapa *state*, yaitu deteksi, mengejar dan menyerang dengan menyemburkan racun. Tampilan pada komponen *AIBrain* untuk setiap musuh memiliki kesamaan pada musuh sebelumnya dapat dilihat pada “Gambar 4.5 *AIBrain* dari musuh *Range*”.



Gambar 4. 5 *AIBrain* dari *Range*

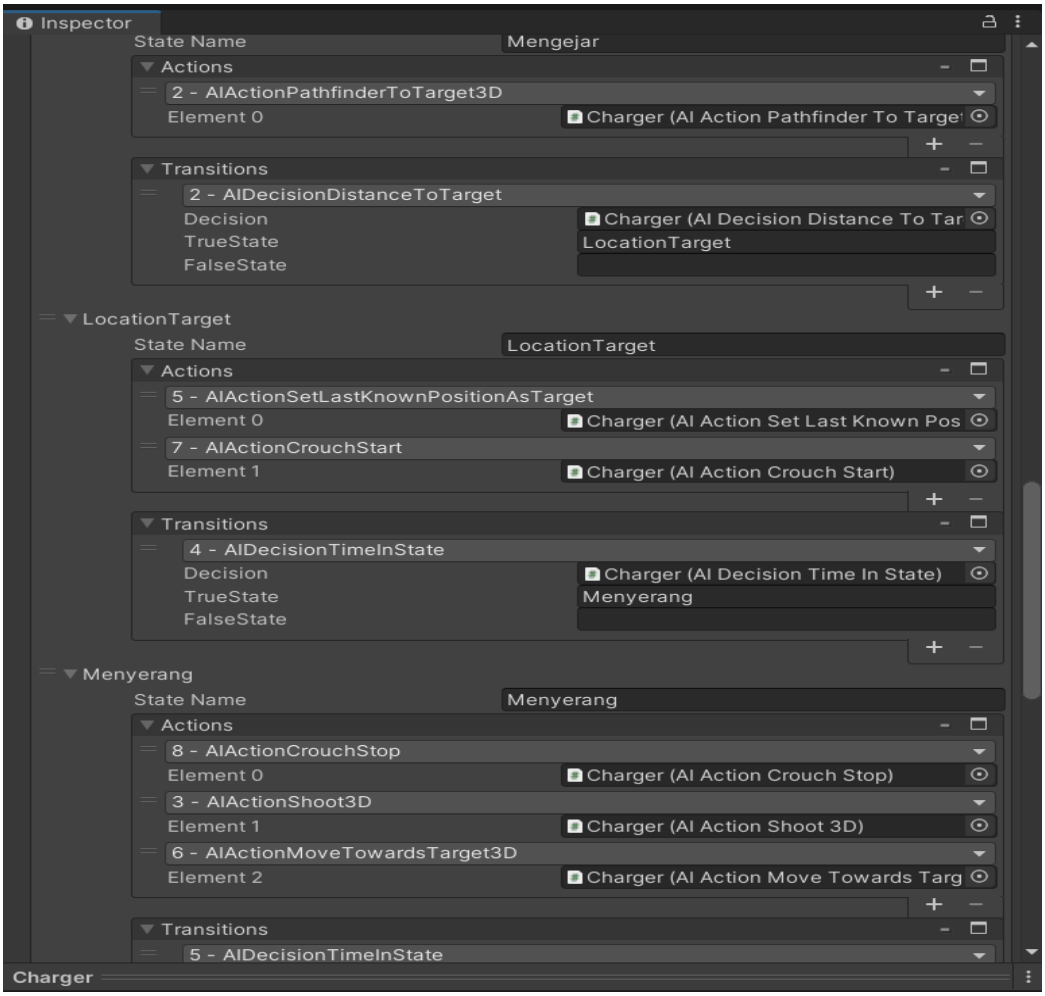
Pada tiap *state* kita masukkan *action* dan juga *transition*. Adapun detail *action* dan *transitions* sebagai berikut:

1. Deteksi
    - *Action* : Musuh mencari keberadaan *player* ketika bermain.
    - *Transition* : Setelah *player* terdeteksi dengan radius 500 maka musuh mulai mengejar.
2. Mengejar
    - *Action* : Ketika *player* terdeteksi oleh musuh maka musuh langsung mendatanginya.
    - *Transition* : Setelah musuh mendapat *player* dengan jarak <8 maka musuh mulai Menyerang dengan mengeluarkan racun.

3. Menyerang
    - *Action* : Musuh menyerang *player* dengan mengeluarkan racun sehingga dapat mengurangi *Health Point player*.
    - *Transition* : Musuh selalu mendeteksi *player* untuk selalu mengikutinya.

4.2.4 Finite State Machine Charger

AIBrain pada *Charger* dapat ditambahkan beberapa *state*, yaitu deteksi, mengejar, *Location Target* dan menyerang. Tampilan pada komponen *AIBrain* untuk musuh pada *Charger* sendiri memiliki komponen yang berbeda dari sebelumnya dan dapat dilihat pada "Gambar 4.6 *AIBrain* dari musuh *Charger*".



Gambar 4. 6 *AIBrain* dari *Charger*

Pada tiap *state* kita masukkan *action* dan juga *transition*. Adapun detail *action* dan *transitions* sebagai berikut:

1. Deteksi
    - *Action* : Musuh mencari keberadaan *player* dengan radius 500 ketika bermain.
    - *Transition* : Setelah *player* terdeteksi maka musuh mulai mengejar.
2. Mengejar
    - *Action* : Ketika *player* terdeteksi oleh musuh maka musuh langsung mengejar.
    - *Transition* : Setelah musuh mendapat *player* maka musuh mulai menandainya.
3. *Location Target*
    - *Action* : Ketika jarak Pemain < 5 maka musuh akan menandai lokasi agar musuh bisa menyerangnya dengan melompat.
    - *Transition* : Musuh akan bersiap siap untuk melompat.
4. Menyerang
    - *Action* : Musuh menyerang *player* dengan melompat kearahnya dan menimbulkan *Damage* cukup sakit sehingga dapat mengurangi *Health Point player*.
    - *Transition* : Musuh selalu mendeteksi *player* untuk selalu mengikutinya.

4.3 Pengujian

Pada tahap pengujian, sistem dapat diuji dengan memainkan game yang telah dirancang dan diimplementasikan sistem *Finite State Machine* di dalamnya. Untuk pertama dalam pengujian *player* dapat mengambil berbagai macam senjata untuk melawan para musuh agar pemain bisa memulai berperang melawan para *Zombie* Dan berikut ini adalah hasil pengujian yang telah dilakukan.

Gambar 4. 7 Tampilan awal *player*

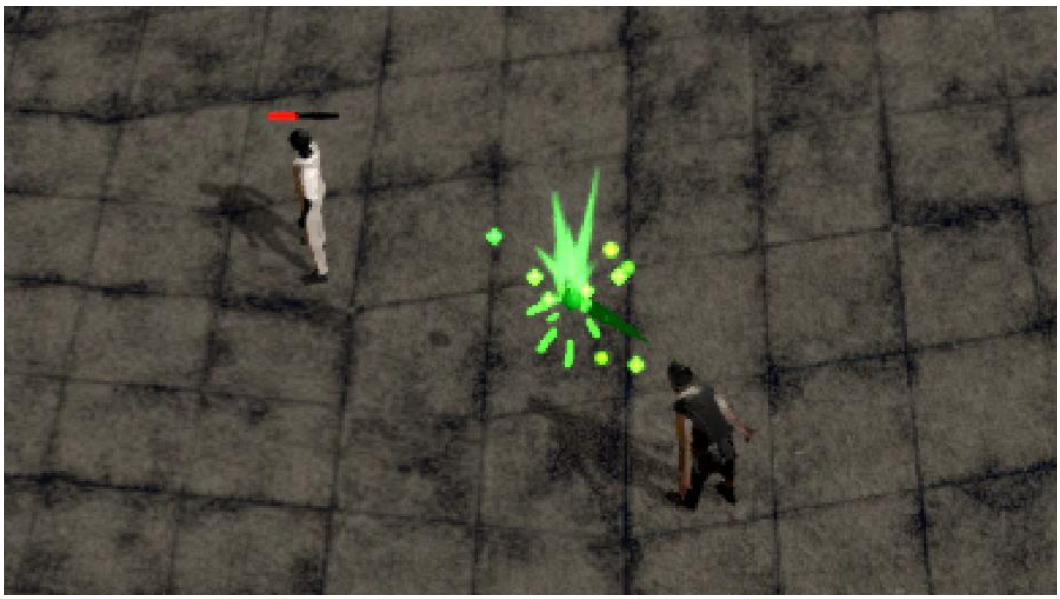
"Berikut gambar 4.7 merupakan gambar awal mulainya game *From The Downtown* dimana *player* menyiapkan persiapan untuk berperang".

Gambar 4. 8 Pengujian musuh *Flying Zombie*

Pada "gambar 4.8 ketika dimulainya permainan *player* akan menghadapi *Flying Zombie* dimana karakter tersebut mulai bekerja dengan *AIBrain*nya". Musuh mendatangi pemain kemudian setelah pemain masuk dalam radius musuh maka *Flying Zombie* akan memutari pemain dan menembakkan racun dari atas kearah pemain.

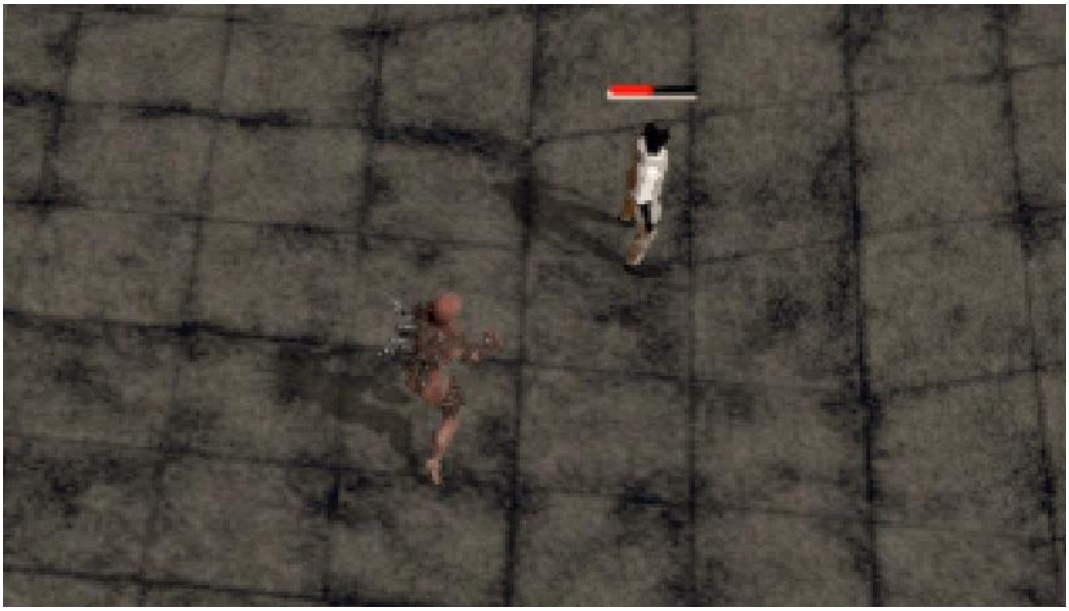
Gambar 4. 9 Pengujian musuh *Chaser*

Berikut pada "gambar 4.9 dimana *Chaser* mengejar *player* yang dapat selalu mengikuti *player* dengan skill larinya sehingga Chaser sangat mudah mendatangi para pemain".



Gambar 4. 10 Pengujian musuh *Range*

Pada pengujian ke-3 "gambar 4.10 musuh *Range* mendatangi *player* agar bisa menyerang *player* dengan mengeluarkan racun".



Gambar 4. 11 Pengujian musuh *Charger*

Berikut pengujian ke-4 pada "gambar 4.11 dimana *Charger* mengejar *player* agar dapat menyerang *player* dengan skill lompat tingginya sehingga *Charger* sangat suka melompat saat mendatangi para pemain".

Untuk *Finite State Machine* dalam permainan karakter musuh menunjukkan *Behavior* selama permainan. Perlunya melakukan beberapa tes pada musuh *Flying Zombie, Chaser, Range, Charger*. Hasil dari tes tersebut dapat melihat "Status beberapa musuh pada Tabel 4.1 berikut ini".

Tabel 4. 1 Hasil tes pada *Flying Zombie*

Status	Keadaan	Perilaku	Hasil
Deteksi	Musuh mencari keberadaan radius > 500	musuh akan mendatangi *player*	Berhasil
Mengejar	Musuh sudah mendeteksi keberadaan	Musuh terbang menuju *player*	Berhasil
Menyerang	Musuh memutari pemain	Melakukan penyerangan pada *player*	Berhasil

Tabel 4. 2 Hasil tes pada *Chaser*

Status	Keadaan	Perilaku	Hasil
Deteksi	Musuh mencari keberadaan radius > 500	musuh akan mengejar *player*	Berhasil
Mengejar	Musuh sudah mendeteksi keberadaan	Musuh berlari menuju *player* dengan kecepatan > 9	Berhasil
Menyerang	Musuh dekat dengan *player*	Melakukan penyerangan pada *player*	Berhasil

Tabel 4. 3 Hasil tes pada *Range*

Status	Keadaan	Perilaku	Hasil
Deteksi	Musuh mencari keberadaan dengan radius > 500	musuh akan mengejar *player*	Berhasil
Mengejar	Musuh sudah mendeteksi keberadaan	Musuh berlari menuju *player*	Berhasil
Menyerang	Musuh menjaga jarak dengan *player* < 8	Melakukan penyerangan pada *player* dengan mengeluarkan racun	Berhasil

Tabel 4. 4 Hasil tes pada *Charger*

Status	Keadaan	Perilaku	Hasil
Deteksi	Musuh mencari keberadaan	musuh akan mengejar *player*	Berhasil
Mengejar	Musuh sudah mendeteksi keberadaan	Musuh berlari menuju *player*	Berhasil
Location Target	jarak Pemain < 5 musuh akan menandai lokasi	Musuh akan besiap untuk melompat	Berhasil
Menyerang	Musuh melompat ke arah *player*	Melakukan penyerangan dengan lompatannya	Berhasil

KUISIONER ASPEK *GAME* YANG BERHUBUNGAN DENGAN *UJI COBA PADA FROM THE DOWNTOWN*

Sangat Buruk = 1, Buruk = 2, Cukup = 3, Baik = 4, Sangat Baik = 5

No.	Pertanyaan	1	2	3	4	5
1	Bagaimana penilaian anda terhadap tiap *musuh* dalam *game* "*From The Downtown".*			✓		
2	Bagaimana penilaian anda terhadap keseimbangan tingkat kemunculan *musuh* yang berhadapan dengan *player* "*From The Downtown"*.				✓	
3	Bagaimana penilaian anda terhadap keseimbangan tingkat kemunculan jumlah dan tipe musuh dalam *game* "*From The Downtown".*				✓	
4	Bagaimana pengalaman bermain anda terhadap game "*From The Downtown"*					✓

BAB 5

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil pengujian dapat ditarik kesimpulan sebagai berikut :

1. Terbukti bahwa metode *Finite State Machine* (FSM) dapat meningkatkan dinamika dan tantangan dalam permainan dengan mengubah perilaku musuh dalam game *top-down shooter*.
2. Musuh dengan perilaku seperti *Flying Zombie, Range, Chaser, dan Charger* menunjukkan kemampuan untuk beradaptasi dengan tindakan pemain.
3. Game *From The Downtown* menghasilkan pengalaman bermain yang lebih menarik dan menantang.
4. Dengan metode *Finite State Machine* (FSM) musuh dapat berperilaku lebih realistis dan responsif terhadap lingkungan serta tindakan pemain.
5. Berdasarkan hasil uji yang telah dilakukan dapat disimpulkan tingkat kepuasan bermain game "*From The Downtown"* mendapatkan tingkat kepuasan yang positif.

5.2 Saran

Saran yang diberikan untuk penelitian selanjutnya adalah sebagai berikut:

1. Menambahkan musuh yang lebih bervariasi yang membuat permainan lebih menarik.
2. Menambahkan mode *Multiplayer* dengan model *Model Peer-to-Peer* (P2P) agar dapat dimainkan lebih dari 1 pemain.
3. Menambahkan *Story Line* agar pemain dapat mengetahui tujuan dan cerita dari game *From The Downtown* sehingga lebih menarik.

DAFTAR PUSTAKA

Adams, E (2010). Fundamentals Of£ Game Design£, Second£ Edition£. New Riders.

Alaa A. Qaffas (2020). An Operational£ Study of£ Video Games£ Genres£. Paper—An Operational Study£ of Video Games£' Genres. 1-20.Russell, S., & Norvig, P. (2020). Artificial£ Intelligence: A Modern Approach (4th ed.). Pearson.

Andersen, J., Pragantha, J., & Haris, D.A. (2021). Perancangan Game£ Top Down Roguelike£ Shooter£ "Arcana Memories£" Pada£ Pc. Jurnal Ilmu Komputer dan Sistem Informasi.1-7.

Astuti, Y.W., Yunus, A., & Ahsan, M. (2019). Perilaku Non *Player*£ Character (NPC) Pada Game FPS "Zombie£ Colonial £Wars" Menggunakan£ Finite£ State Machine (FSM). Kurawal£ - Jurnal Teknologi, Informasi£ dan Industri. Vol. 2 No.1 Tahun 2019, 1-11

Fahmi Nurfauzi, Kusum, D. P., Ashri Dinimaharawati (2023). Perancangan Perilaku Eric Pada£ Permainan Pertualangan£ Adam£ dan Eric Menggunakan£ Metode£ Finite State Machine. e-Proceeding of Engineering 10(1) 685-691.

Laurentia, N., & Haryadi, T. £ (2021). Kajian£ Desain Kostum£ Game Lifeafter Sebagai Upaya Mengeksplorasi£ Gameplay yang £Imersif. CITRAKARA, 3(1), 69-83.

Liu, S., Claypool, M., Kuwahara, £ A., Sherman, J., & Scovell, J.J. (2021). Lower is Better? The Effects£ of£ Local Latencies£ on Competitive First-Person Shooter£ Game *Player*s£. Proceedings of the 2021 CHI £Conference on Human Factors in Computing£ Systems.

Natasha Laurentia, Toto Haryadi (2021). Kajian£ Desain£ Kostum£ Game Lifeafter Sebagai£ Upaya Mengeksporasi£ Gameplay Yang Imersif. Jurnal Citrakara, Vol. 3 No. 1, 69 – 83.

Rumakey, A.M., Irawan, J.D., & Wahid, A. (2020). Pembuatan£ Game£ 2D "Escape Plan" Dengan£ Metode Finite£ State Machine£. Jurnal Mahasiswa Teknik Informatika Vol. 2 No.2 Tahun 2022, 1-8.

Setiawan, M. L., Arbansyah, & Suryawan, S. H. (2023). Penerapan Algoritma A* Dan Behaviour£ Trees£ Untuk Perilaku£ Non-*Player* Character£ (NPC) Pada Game£ "The Last Hope" Berbasis£ Android Menggunakan£ Unity 2D. Jurnal Computer£

Science£ and Information Technology£ (CoSciTech). 4(2), 451-460.

Sofyan, £I. A., Muhammad Aminul Akbar, M. A., & Afirianto, T. (2019). Implementasi Dynamic£ Difficulty£ Adjustment£ pada Racing Game Menggunakan Metode£ Behaviour£ Tree. Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, 3(1), 129-132.

Vinsensa Audrey Roseline Waluyo, Asidigisianti Surya Patria (2022). Analisis Semiotika Desain Karakter Silverash£ Pada£ Game£ Arknights. Jurnal Barik, Vol. 3 No. 2, Tahun 2022, 78-88.

Wennedy Surya pratama, Amak Yumus, Wasum (2019). Turn Based Strategy Game "Perang £Komando" Dengan£ Metode Finite State£ Machine£ Pada Karakter Musuh. Jurnal Terapan£ Sains£ & Teknologi (RAINSTEK) 1(1) 1-6.

LAMPIRAN

Lampiran 1. Kode Program

```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Random = UnityEngine.Random;

namespace MoreMountains.Tools
{
  /// <summary>
  /// the AI brain is responsible from going from one state to the other based on the defined transitions. It's basically just a collection of states, and it's where you'll link all the actions, decisions, states and transitions together.
  /// </summary>
  [AddComponentMenu("More Mountains/Tools/AI/AIBrain")]
  public class AIBrain : MonoBehaviour
  {
        [Header("Debug")]
        /// the owner of that AI Brain, usually the associated character
        [MMReadOnly]
        public GameObject Owner;
        /// the collection of states
        public List<AIState> States;
        /// this brain's current state
        public virtual AIState CurrentState { get; protected set; }
        /// the time we've spent in the current state
```

```
        [MMReadOnly]
        public float TimeInThisState;
        /// the current target
        [MMReadOnly]
        public Transform Target;
        /// the last known world position of the target
        [MMReadOnly]
        public Vector3 _lastKnownTargetPosition = Vector3.zero;

        [Header("State")]
        /// whether or not this brain is active
        public bool BrainActive = true;
        public bool ResetBrainOnStart = true;
        public bool ResetBrainOnEnable = false;

        [Header("Frequencies")]
        /// the frequency (in seconds) at which to perform actions (lower values : higher frequency, high values : lower frequency but better performance)
        public float ActionsFrequency = 0f;
        /// the frequency (in seconds) at which to evaluate decisions
        public float DecisionFrequency = 0f;

        /// whether or not to randomize the action and decision frequencies
        public bool RandomizeFrequencies = false;
        /// the min and max values between which to randomize the action frequency
        [MMVector("min","max")]
        public Vector2 RandomActionFrequency = new Vector2(0.5f, 1f);
```

```
        /// the min and max values between which to
randomize the decision frequency
        [MMVector("min","max")]
        public Vector2 RandomDecisionFrequency = new
Vector2(0.5f, 1f);

        protected AIDecision[] _decisions;
        protected AIAction[] _actions;
        protected float _lastActionsUpdate = 0f;
        protected float _lastDecisionsUpdate = 0f;
        protected AIState _initialState;

        public virtual AIAction[] GetAttachedActions()
        {
            AIAction[] actions =
this.gameObject.GetComponentsInChildren<AIAction>();
            return actions;
        }

        public virtual AIDecision[]
GetAttachedDecisions()
        {
            AIDecision[] decisions =
this.gameObject.GetComponentsInChildren<AIDecision>();
            return decisions;
        }

        protected virtual void OnEnable()
        {
            if (ResetBrainOnEnable)
            {
                ResetBrain();
            }
```

```
        }

        /// <summary>
        /// On awake we set our brain for all states
        /// </summary>
        protected virtual void Awake()
        {
             foreach (AIState state in States)
             {
                  state.SetBrain(this);
             }
             _decisions = GetAttachedDecisions();
             _actions = GetAttachedActions();
             if (RandomizeFrequencies)
             {
                  ActionsFrequency                    =
Random.Range(RandomActionFrequency.x,
RandomActionFrequency.y);
                  DecisionFrequency                   =
Random.Range(RandomDecisionFrequency.x,
RandomDecisionFrequency.y);
             }
        }

        /// <summary>
        /// On Start we set our first state
        /// </summary>
        protected virtual void Start()
        {
             if (ResetBrainOnStart)
             {
                  ResetBrain();
             }
```

```
        }

        /// <summary>
        /// Every frame we update our current state
        /// </summary>
        protected virtual void Update()
        {
            if (!BrainActive || (CurrentState == null)
|| (Time.timeScale == 0f))
            {
                return;
            }

            if (Time.time - _lastActionsUpdate >
ActionsFrequency)
            {
                CurrentState.PerformActions();
                _lastActionsUpdate = Time.time;
            }

            if (!BrainActive)
            {
                return;
            }

            if (Time.time - _lastDecisionsUpdate >
DecisionFrequency)
            {
                CurrentState.EvaluateTransitions();
                _lastDecisionsUpdate = Time.time;
            }

            TimeInThisState += Time.deltaTime;
```

```
            StoreLastKnownPosition();
        }

        /// <summary>
        /// Transitions to the specified state, trigger
exit and enter states events
        /// </summary>
        /// <param name="newStateName"></param>
        public  virtual  void  TransitionToState(string
newStateName)
        {
            if (CurrentState == null)
            {
                CurrentState                          =
FindState(newStateName);
                if (CurrentState != null)
                {
                    CurrentState.EnterState();
                }
                return;
            }
            if            (newStateName            !=
CurrentState.StateName)
            {
                CurrentState.ExitState();
                OnExitState();

                CurrentState                          =
FindState(newStateName);
                if (CurrentState != null)
                {
                    CurrentState.EnterState();
```

```
                }
            }
        }

        /// <summary>
        ///  When exiting a state we reset our time
counter
        /// </summary>
        protected virtual void OnExitState()
        {
            TimeInThisState = 0f;
        }

        /// <summary>
        /// Initializes all decisions
        /// </summary>
        protected virtual void InitializeDecisions()
        {
            if (_decisions == null)
            {
                _decisions = GetAttachedDecisions();
            }
            foreach(AIDecision decision in _decisions)
            {
                decision.Initialization();
            }
        }

        /// <summary>
        /// Initializes all actions
        /// </summary>
        protected virtual void InitializeActions()
        {
```

```
            if (_actions == null)
            {
                _actions = GetAttachedActions();
            }
            foreach(AIAction action in _actions)
            {
                action.Initialization();
            }
        }

        /// <summary>
        /// Returns a state based on the specified state name
        /// </summary>
        /// <param name="stateName"></param>
        /// <returns></returns>
        protected AIState FindState(string stateName)
        {
            foreach (AIState state in States)
            {
                if (state.StateName == stateName)
                {
                    return state;
                }
            }
            if (stateName != "")
            {
                Debug.LogError("You're trying to transition to state '" + stateName + "' in " + this.gameObject.name + "'s AI Brain, but no state of this name exists. Make sure your states are named properly, and that your transitions states match existing states.");
```

```
            }
            return null;
        }

        /// <summary>
        ///  Stores the last known position of the
target
        /// </summary>
        protected virtual void StoreLastKnownPosition()
        {
            if (Target != null)
            {
                _lastKnownTargetPosition            =
Target.transform.position;
            }
        }

        /// <summary>
        /// Resets the brain, forcing it to enter its
first state
        /// </summary>
        public virtual void ResetBrain()
        {
            InitializeDecisions();
            InitializeActions();
            BrainActive = true;
            this.enabled = true;

            if (CurrentState != null)
            {
                CurrentState.ExitState();
                OnExitState();
            }
```

```
            if (States.Count > 0)
            {
                CurrentState = States[0];
                CurrentState?.EnterState();
            }
        }
    }
}
```

Lampiran 2. Biodata

Nama	: Muhammad Ridwan Lubis
Tempat, Tanggal Lahir	: Medan, 23 Maret 2002
Alamat	: Jln.Karya Jaya Ujung , Kota Medan
Telpon/HP	: 081362173448
Email	: muhammadone8@gmail.com
Agama	: Islam
Nama Orang Tua	
Ayah	: Muhammad Irwan
Ibu	: Julidar
Jumlah Saudara	: 4
Anak Ke	: 1
Riwayat Pendidikan	: SD Al Fithriah (2006 - 2012)
	SMP Harapan 3 Medan (2013 - 2016)
	SMA Harapan 3 Medan (2016 - 2019)
	Universitas Sumatera Utara (2020 - sekarang)