IMPLEMENTASI ALGORITMA A* DAN FINITE STATE MACHINE UNTUK SPECIAL CHARACTER (BOSS) PADA GAME TOP-DOWN SHOOTER "FROM THE DOWNTOWN"

SKRIPSI

NABIL RISNANDA AZMI SIREGAR
201401058

PROGRAM STUDI S1 ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA
MEDAN
2024

IMPLEMENTASI ALGORITMA A* DAN FINITE STATE MACHINE UNTUK SPECIAL CHARACTER (BOSS) PADA GAME TOP-DOWN SHOOTER "FROM THE DOWNTOWN"

SKRIPSI

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

NABIL RISNANDA AZMI SIREGAR
201401058

PROGRAM STUDI S1 ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA
MEDAN
2024

PERSETUJUAN

Judul	:	IMPLEMENTASI ALGORITMA A* DAN FINITE STATE MACHINE UNTUK SPECIAL CHARACTER (BOSS) PADA GAME TOP-DOWN SHOOTER "FROM THE DOWNTOWN"
Kategori	:	SKRIPSI
Nama	:	NABIL RISNANDA AZMI SIREGAR
Nomor Induk Mahasiswa	:	201401058
Program Studi	:	SARJANA (S-1) ILMU KOMPUTER
Fakultas	:	ILMU KOMPUTER DAN TEKNOLOGI INFORMASI UNIVERSITAS SUMATERA UTARA
Komisi Pembimbing	:	

Pembimbing 2	Pembimbing 1
Dewi Sartika Br Ginting, S.Kom., M.Kom.	Dian Rachmawati, S.Si., M.Kom.
NIP. 199005042019032023	NIP. 198307232009122004

Diketahui/Disetujui
Program Studi S-1 Ilmu Komputer
Ketua

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

PERNYATAAN

IMPLEMENTASI ALGORITMA A* DAN FINITE STATE MACHINE UNTUK SPECIAL CHARACTER (BOSS) 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, 08 Juli 2024

Nabil Risnanda Azmi Siregar

201401058

PENGHARGAAN

Puji syukur peneliti ucapkan kepada Allah SWT atas rahmat dan karunia-nya peneliti diberi kesempatan dan waktu untuk menyelesaikan skripsi yang berjudul "Implementasi Algoritma A* Dan Finite State Machine Untuk Special Character (Boss) Pada Game Top-Down Shooter “From The Downtown”". Pada kesempatan kali ini peneliti ingin mengucapkan rasa terima kasih kepada semua pihak yang ikut terlibat dan membantu dalam proses pembuatan skripsi ini.

Terima kasih kepada:

1. Bapak Dr. Muryanto Amin, S.Sos., M.Si, selaku Rektor Universitas Sumatera Utara.
2. Dr. Maya Silvi Lydia B.Sc., M.Sc Sebagai Dekan Fakultas Ilmu Komputer Universitas Sumatera Utara,
3. Dr. Amalia, ST., M.T., sebagai Ketua Prodi Program Studi Ilmu Komputer.
4. Ibu Dian Rachmawati, S.Si., M.Kom. sebagai Dosen Pembimbing I, yang memberikan arahan dan bimbingan yang sangat berharga dalam menyelesaikan skripsi ini.
5. Ibu Dewi Sartika Br Ginting, S.Kom., M.Kom. sebagai Dosen Pembimbung II, yang memberikan arahan dan bimbingan yang sangat berharga dalam menyelesaikan skripsi ini.
6. Bapak Dr. Jos Timanta Tarigan S.Kom., M.Sc., selaku Ketua Laboratorium Computer Vision dan Multimedia Program Studi Ilmu Komputer, yang memberikan arahan dan bimbingan yang sangat berharga dalam menyelesaikan skripsi ini.
7. Ayah saya Heri Indra Siregar, ST., MT. dan ibu saya Usiana yang selalu mensupport dari segi mental ataupun materi selama peneliti menjalani kehidupan perkuliahan ini.
8. Anggota grup GameDev, & IoT yang selalu membantu saya dalam mengerjakan skripsi hingga selesai.

9. Anggota grup Kaum Mujahiruddin yang selalu menyemangati dan menemani pengerjaan skripsi.
10. Teman-teman seperjuangan di program studi Ilmu Komputer yang telah memberikan dukungan dan bantuan yang berarti selama proses perkuliahan.

Dan juga banyak nama yang tidak bisa disebutkan dalam kesempatan ini.

Dengan rendah hati dan segala hormat, peneliti menyampaikan permohonan maaf atas segala kekurangan dan ketidaksempurnaan yang terdapat dalam penulisan ini. Semoga hasil dari penelitian ini memberikan efek yang positif terhadap perkembangan yang pengetahuan kedepannya Semoga hasil penelitian ini dapat memberikan kontribusi positif bagi perkembangan ilmu pengetahuan. Terima kasih atas kesempatan ini.

Medan, 08 Juli 2024

Peneliti,

Nabil Risnanda Azmi Siregar

NIM 201401058

ABSTRAK

Dalam industri game, pengembangan kecerdasan buatan (AI) sangat penting untuk meningkatkan imersi dan kesenangan bermain. Penelitian ini bertujuan untuk mengimplementasikan algoritma A* dan Finite State Machine (FSM) pada karakter boss dalam game top-down shooter "From the Downtown" untuk meningkatkan tantangan dan keterlibatan pemain. Algoritma A* digunakan untuk pathfinding, memungkinkan boss menemukan jalur optimal dalam mengejar pemain. FSM digunakan untuk mengatur perilaku boss berdasarkan berbagai keadaan permainan, seperti menyerang, mengejar, dan diam. Penelitian ini melibatkan pengembangan dan integrasi kedua algoritma ke dalam engine game, diikuti dengan pengujian untuk mengevaluasi kinerja dan responsivitas boss. Hasil penelitian menunjukkan bahwa algoritma A* berjalan dengan baik, memungkinkan boss menemukan jalur efektif menuju pemain. FSM berhasil menghasilkan perilaku dinamis dan responsif dari boss sesuai dengan perubahan keadaan selama permainan. Pengujian dilakukan dengan berbagai skenario untuk memastikan kehandalan dan keefektifan AI yang diterapkan. Penerapan kedua algoritma ini tidak hanya meningkatkan tingkat tantangan, tetapi juga memperkaya pengalaman bermain dengan menyediakan interaksi yang lebih alami dan responsif. Penelitian ini memberikan kontribusi dalam pengembangan AI dalam game, khususnya dalam menciptakan boss yang lebih cerdas dan menantang. Hasil implementasi menunjukkan bahwa kombinasi algoritma A*, dan FSM dapat diterapkan pada game lain untuk meningkatkan realitas dan keseruan permainan. Dengan demikian, penelitian ini memberikan wawasan baru dalam pengembangan AI untuk karakter boss dalam game, yang dapat bermanfaat bagi industri game secara keseluruhan.

Kata kunci: *Algoritma A*, Finite State Machine, Kecerdasan Buatan, Game Top-Down Shooter, pencarian jalur.*

IMPLEMENTATION OF A* ALGORITHM AND FINITE STATE MACHINE FOR SPECIAL CHARACTER (BOSS) IN TOP-DOWN SHOOTER GAME "FROM THE DOWNTOWN"

ABSTRACT

In the gaming industry, the development of artificial intelligence (AI) is essential to enhance immersion and fun. This research aims to implement the A* algorithm and Finite State Machine (FSM) on the boss character in the top-down shooter game "From the Downtown" to increase challenge and player engagement. The A* algorithm is used for pathfinding, allowing the boss to find the optimal path in chasing the player. FSM is used to manage the boss behavior based on various game states, such as attacking, chasing, and idle. This research involved the development and integration of both algorithms into the game engine, followed by testing to evaluate the performance and responsiveness of the boss. The results showed that the A* algorithm performed well, allowing the boss to find an effective path to the player. The FSM successfully generated dynamic and responsive behavior of the boss according to the changing circumstances during the game. Tests were conducted with various scenarios to ensure the reliability and effectiveness of the implemented AI. The application of these two algorithms not only increases the level of challenge, but also enriches the gaming experience by providing more natural and responsive interactions. This research contributes to the development of AI in games, especially in creating smarter and more challenging bosses. The implementation results show that the combination of the A* algorithm, and FSM can be applied to other games to enhance the reality and excitement of the game. Thus, this research provides new insights in the development of AI for boss characters in games, which can benefit the game industry as a whole.

***Keywords**: A* Algorithm, Finite State Machine, Artificial Intelligence, Top-Down Shooter Game, Pathfinding*

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I

PENDAHULUAN

1.1. Latar Belakang

Dalam industri game, pengembangan kecerdasan buatan (AI) dalam game memiliki potensi untuk meningkatkan imersi dan kesenangan saat bermain game, sebab game menjadi lebih responsif, dan menyediakan interaksi yang lebih alami (Nareyek, 2004). Salah satu tantangan dalam pengembangan kecerdasan buatan adalah non-player character (NPC) khusus atau boss yang memiliki tingkat kecerdasan yang memadai untuk memperkaya gameplay dan meningkatkan tingkat kesulitan.

Boss dalam game memiliki peran krusial dalam menghadirkan tantangan yang menantang bagi pemain (Siu, Butler, & Zook, 2016). Kehadiran boss tidak hanya membutuhkan kekuatan fisik untuk mengalahkannya, tetapi juga kecerdasan yang tepat dalam mengantisipasi gerakan dan serangan mereka. Untuk memberikan pengalaman bermain yang lebih dinamis dan menantang, pengembangan kecerdasan buatan untuk boss menjadi fokus penelitian yang penting.

Adapun algoritma yang akan diterapkan adalah algoritma A* yang akan digunakan boss untuk mencari rute dalam mengejar pemain. Algoritma A* adalah algoritma pencarian jalur yang digunakan untuk menemukan jalur terpendek atau jalur optimal antara dua titik. Algoritma A* memiliki kelebihan konsisten menghasilkan jalur terpendek atau jalur optimal, dan A* cenderung lebih efisien karena menggabungkan teknik pencarian heuristik dengan pencarian terinformasi (Simbolon, 2022).

Sementara itu, Finite State Machine (FSM) akan digunakan untuk menggambarkan dan mengatur perilaku boss dalam game. FSM adalah model matematis yang digunakan untuk menggambarkan sistem dengan jumlah keadaan terbatas dan transisi antara keadaan-keadaan tersebut. FSM memiliki kelebihan dengan memiliki struktur yang sederhana dan mudah dipahami, yang terdiri dari

sejumlah kecil keadaan (states) dan transisi antara keadaan-keadaan tersebut (Pukeng et al., 2019).

Pada penelitian yang dilakukan oleh Delima (2014), algoritma A* diterapkan pada karakter musuh dalam sebuah game 3D dengan tujuan untuk meningkatkan realitas permainan dan keterlibatan pemain. Percobaan dilakukan dengan memvariasikan situasi game untuk menguji kemampuan algoritma A* dalam menemukan jalur optimal untuk menangkap pemain. Hasilnya menunjukkan bahwa algoritma A* berhasil menemukan jalur yang optimal dalam sebagian besar kasus, dengan 73% dari seratus kali percobaan menunjukkan jalur optimal.

Selanjutnya pada penelitian Sanjaya, Pratiwi, & Adytia (2021), diterapkan algoritma Finite State Machine (FSM) pada game yang berjudul “Heroes of Dawn”. Penerapan algoritma ini bertujuan untuk menentukan respons dan perilaku musuh. Penerapan metode FSM pada musuh dapat membuat musuh berjalan, menyerang dan memiliki posisi diam.

Adapun algoritma yang sama diterapkan pada penelitian yang dilakukan Safitra, Faisol, & Wibowo (2020) untuk game yang berjudul “Ouroboros”. Penerapan FSM menghasilkan beberapa gerakan pada musuh, yaitu berkeliling, mencari player, mengejar, dan menembak.

Video game “From The Downtown” pada penelitian ini akan dikembangkan menggunakan perspektif top-down shooter. Top-down shooter adalah jenis permainan video di mana pemain mengendalikan karakter atau kendaraan dari perspektif pandangan yang "dari atas" atau "dari atas ke bawah", yang berarti bahwa adegan diwakili dalam sudut pandang bird's-eye view (Kho, Pragantha, & Kristyadi, 2018). Dalam top-down shooter, pemain melihat seluruh peta atau lingkungan permainan dari sudut pandang ini, memberikan pandangan yang luas atas aksi dan area sekitarnya.

Dalam fase pertama, boss mungkin lebih cenderung melakukan serangan yang pelan dan sederhana, sementara dalam fase kedua, serangan cenderung cepat, kompleks dan juga memberikan poin damage yang lebih banyak. Fase dari musuh akan ditentukan sesuai dengan poin darah musuh. Oleh karena itu fase kedua dari boss harus memberikan tantangan yang lebih sulit kepada player yang dikarenakan rendahnya poin darah boss.

Dengan memperkenalkan kecerdasan buatan pada boss dalam "From the Downtown", diharapkan dapat meningkatkan tingkat tantangan dan kesenangan bagi para pemain. Selain itu, penelitian ini juga berpotensi untuk memberikan wawasan baru dalam pengembangan kecerdasan buatan dalam konteks video games, yang dapat diterapkan pada judul-judul permainan lainnya di masa depan.

1.2. Rumusan Masalah

Dalam permainan, *special character* berupa *boss monster* memainkan peran penting dalam memberikan tantangan kepada pemain. Namun, pada umumnya, implementasi kecerdasan buatan pada *boss monster* dalam permainan video sering kali belum memadai, menyebabkan perilaku *boss* menjadi terlalu terprediksi atau kurang menantang bagi pemain. Oleh karena itu, penelitian ini bertujuan untuk mengatasi masalah tersebut dengan mengembangkan kecerdasan buatan yang lebih untuk *special character* atau *boss* dalam permainan "From the Downtown".

1.3. Tujuan Penelitian

Tujuan penelitian ini adalah untuk meningkatkan tingkat tantangan dan keseruan bagi para pemain dalam permainan "From the Downtown" dengan mengimplementasikan kecerdasan buatan pada *boss* dalam permainan, memperkaya pengalaman bermain melalui *boss* yang memiliki tingkat kecerdasan yang memadai, serta meningkatkan realitas permainan dan keterlibatan pemain dengan menyediakan *boss* yang responsif dan cerdas dalam menghadapi pemain.

1.4. Batasan Masalah

Berikut batasan masalah dalam penelitian ini.

1. Penelitian ini akan membuat 2 tipe *boss* yaitu *Melee*, dan *Range*.
2. Kecerdasan buatan yang akan diterapkan pada *boss* adalah Algoritma A* untuk mencari rute dalam mengejar pemain, dan *Finite State Machine* untuk menggambarkan dan mengatur perilaku *boss* dalam *game*.
3. *Game* "*From The Downtown*" dibangun dengan perspektif *top-down shooter*.

4. Pembuatan *game* pada penelitian ini akan dilakukan dengan menggunakan platform *game engine Unity* dan dengan menggunakan bahasa pemrograman C#.

1.5. Manfaat Penelitian

Manfaat dari dilakukannya penelitian ini adalah sebagai berikut:

1. Implementasi kecerdasan buatan pada *boss* dalam permainan "From the Downtown" akan meningkatkan pengalaman bermain dengan memberikan tantangan yang lebih dinamis dan menantang bagi para pemain, yang dapat meningkatkan kesenangan dan kepuasan dalam bermain *game*.
2. Menyediakan *boss* yang responsif dan cerdas, penelitian ini akan meningkatkan realitas permainan dan keterlibatan pemain, menciptakan pengalaman bermain yang lebih imersif dan menarik.
3. Memberikan kontribusi pada pengembangan kecerdasan buatan dalam konteks video games secara lebih luas, membuka jalan untuk penggunaan teknologi tersebut dalam judul-judul permainan lainnya di masa depan.

1.6. Metodologi Penelitian

1. Studi Pustaka

   Pada tahap ini penelitian 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, Infinite Wave, Infinite Level, Dynamic Difficulty, Top Down Shooter dan pengimplementasiannya ke dalam Unity.
2. Analisis dan Perancangan

   Pada tahap ini dilakukan sebuah perancangan dari game yang akan dibuat sesuai dengan permasalahan dan perancangan game. Perancangan game ini akan dirancang dengan pembuatan flowchart, usecase diagram, activity diagram, sequence diagram, dan ishikawa diagram.

3. Implementasi

   Pada tahap ini, membuat sebuah game dengan menggunakan game engine Unity dan menggunakan bahasa pemrograman C# sesuai dengan flowchart yang telah dirancang.

4. Pengujian

   Pada tahap ini, dilakukan uji coba implementasi kecerdasan buatan pada boss yang telah dibuat berhasil dijalankan, serta mampu menjalankan game tersebut dengan baik.

5. Dokumentasi

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

1.7. Penelitian Relevan

1. Penelitian yang dilakukan oleh Delima (2014) yang berjudul "Analisis Implementasi Algoritma A* (A-Star) Pada *Game* Rpg (*Role Playing Game*) 3D Sebagai Dasar Pergerakan Npc (*Non-Player Character*) Mendekati *Player* Untuk Meningkatkan Realitas *Game* World" algoritma A* diterapkan pada karakter musuh dalam sebuah *game* 3D dengan tujuan untuk meningkatkan realitas permainan dan keterlibatan pemain. Percobaan dilakukan dengan memvariasikan situasi *game* untuk menguji kemampuan algoritma A* dalam menemukan jalur optimal untuk menangkap pemain. Hasilnya menunjukkan bahwa algoritma A* berhasil menemukan jalur yang optimal dalam sebagian besar kasus, dengan 73% dari seratus kali percobaan menunjukkan jalur optimal.
2. Penelitian yang dilakukan oleh Sanjaya, Pratiwi, & Adytia (2021), yang berjudul "Application of the Finite State Machine Method in the Desktop-Based "Heroes of Dawn" RPG Turn-Based *Game*.", diterapkan algoritma *Finite State Machine* (FSM) pada *game* yang berjudul "Heroes of Dawn". Penerapan algoritma ini bertujuan untuk menentukan respons dan perilaku musuh. Penerapan metode FSM pada musuh dapat membuat musuh berjalan, menyerang dan memiliki posisi diam. Hasilnya menunjukkan musuh dapat dengan baik memberikan respons sesuai dengan kondisi yang diberikan.

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 *infinite wave*, Dynamic Difficulty Adjustment (DDA) 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 akan diuji dengan membuat kuisioner menggunakan *google form* dengan *target playtester* orang yang sering atau bisa bermain game dan memainkan game tersebut

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.

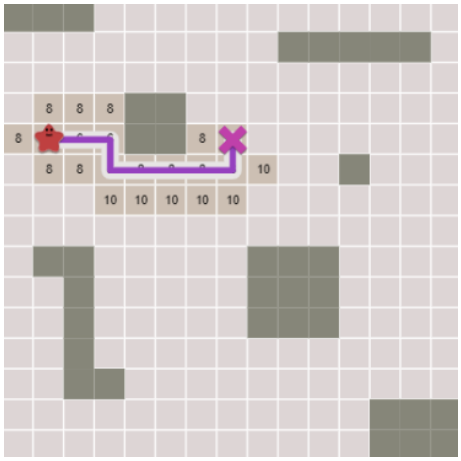
BAB II

LANDASAN TEORI

2.1. Algoritma A*

Algoritma A* adalah algoritma pencarian jalur yang digunakan untuk menemukan jalur terpendek antara dua titik dalam graf berbobot, dengan mempertimbangkan biaya yang terkait dengan setiap langkah (Zuhdi, Ahmad, & Putra, 2023). Algoritma ini merupakan gabungan dari pencarian terarah dan heuristik, di mana pencarian dilakukan dengan memperhitungkan biaya aktual dari simpul awal ke simpul saat ini (g(n)), serta estimasi biaya tersisa dari simpul saat ini ke tujuan (h(n)).

Algoritma A* adalah salah satu algoritma pencarian jalur yang banyak digunakan dalam pengembangan kecerdasan buatan untuk permainan video. Liu (2023) menjelaskan bahwa algoritma A* memiliki kelebihan konsisten menghasilkan jalur terpendek atau jalur optimal, serta cenderung lebih efisien karena menggabungkan teknik pencarian heuristik dengan pencarian terinformasi. Dengan demikian, algoritma A* menjadi salah satu pendekatan yang penting dalam pengembangan kecerdasan buatan untuk permainan video, termasuk dalam konteks permainan "From the Downtown".



Gambar 2. 1 Contoh A* Dalam Game

Pada Gambar 2.1 merupakan contoh dari penerapan dan perhitungan A* dalam *game*. Pada contoh diatas peta digambarkan menggunakan *grid system* dimana tiap *grid* yang dapat dilewati merupakan node dan dihubungkan dengan vertex. Rumus A*:

$$\boldsymbol{f(n) = g(n) + h(n)}$$

Keterangan:

- f(n) : biaya estimasi
- g(n) : biaya dari node awal ke node n
- h(n) : biaya dari node n ke tujuan

Pada *game* biaya yang dihitung adalah jarak, seperti pada gambar 2.1 tiap node memiliki angka di dalamnya yang merupakan nilai f(n) dari node tersebut, sedangkan node yang hitam gelap merupakan hambatan, pada *game* bisa berupa dinding ataupun rintangan lainnya.

Bisa dilihat pada gambar 2.1, jalur terpendek dapat ditemukan yang ditandai dengan garis ungu, dengan terus menerus memilih nilai f(n) terendah. Adapun langkah lengkap dari A* sebagai berikut:

1. Letakkan Node Awal ke dalam openSet
2. Selama openSet tidak kosong:
    a. Pilih Node Sekarang dari openSet yang memiliki nilai f(n) terendah.
    b. Jika Node Sekarang adalah Node Tujuan, maka telah ditenemukan jalur menuju tujuan. Rekonstruksi jalur dan kembalikan jalur tersebut.
    c. Pindahkan Node Sekarang dari openSet ke closedSet untuk menandai bahwa node tersebut telah dievaluasi.

3. Evaluasi setiap tetangga dari Node Sekarang:
    a. Hitung biaya g(n)Sementara untuk mencapai tetangga tersebut melalui Node Sekarang.
    b. Jika g(n)Sementara lebih kecil dari g(n) yang saat ini tercatat untuk tetangga tersebut, atau jika tetangga tersebut belum dievaluasi:
        i. Perbarui g(n), dan f(n) untuk tetangga tersebut.
        ii. Jika tetangga tersebut belum ada dalam openSet, tambahkan tetangga tersebut ke dalam openSet
4. Jika openSet kosong dan goalNode belum ditemukan, maka tidak ada jalur yang tersedia dari Node Awal ke Node Tujuan.
5. Rekonstruksi jalur dari Node Awal hingga Node Tujuan.

Keterangan:

- OpenSet: Himpunan yang berisi node-node yang akan dievaluasi.
- ClosedSet: Himpunan untuk menyimpan informasi node-node yang telah dievaluasi

2.2. *Finite State Machine (FSM)*

Finite State Machine (FSM) adalah model matematis yang digunakan untuk menggambarkan perilaku sistem berbasis keadaan dan terdiri dari serangkaian keadaan (state) diskret dan transisi antara keadaan-keadaan tersebut. Ben-Ari, & Mondada (2018) menjelaskan bahwa FSM terdiri dari sejumlah kecil keadaan (states) dan transisi antara keadaan-keadaan tersebut, serta memiliki keadaan, simbol masukan, simbol keluaran, dan fungsi transisi.

Dalam konteks pengembangan kecerdasan buatan untuk permainan video, FSM telah menjadi alat yang penting dalam mengatur perilaku karakter atau musuh dalam permainan. Sanjaya, Pratiwi, & Adytia (2021) menerapkan algoritma FSM pada *game* "Heroes of Dawn" untuk menentukan respons dan perilaku musuh dalam permainan. Hasilnya menunjukkan bahwa penerapan metode FSM pada musuh dapat memberikan variasi dalam perilaku mereka, seperti berjalan, menyerang, dan memiliki posisi diam. Oleh karena itu, FSM telah menjadi metode yang berguna untuk mengontrol bagaimana karakter atau musuh berperilaku dalam permainan video. Ini memberikan variasi dan dinamika dalam interaksi antara pemain dan lingkungan permainan.

2.3. *Top-Down Shooter Game*

2.3.1. Top-Down

Genre *game top-down* mengacu pada jenis permainan di mana pandangan kamera dari atas ke bawah, memberikan pemain pandangan yang luas atas lingkungan permainan (Holgersson, 2016). Dalam genre ini, pemain biasanya mengendalikan karakter atau entitas dalam lingkungan dua dimensi atau tiga dimensi dengan sudut pandang dari atas, yang memungkinkan mereka untuk melihat seluruh area permainan secara keseluruhan.

2.3.2. Shooter Game

Genre *game shooter* adalah salah satu genre permainan video di mana pemain mengendalikan karakter yang menggunakan senjata untuk menembak dan mengalahkan musuh-musuh dalam lingkungan permainan (Hitchens, Patrickson, & Young, 2014). Pemain biasanya melihat dunia permainan dari sudut pandang orang pertama (FPS, *First Person Shooter*) atau orang ketiga (TPS, *Third Person Shooter*).

Genre *game shooter* adalah salah satu genre permainan video yang paling populer dan berpengaruh dalam industri permainan. Dengan *gameplay* yang adiktif, pertempuran *real-time* yang intens, dan berbagai variasi sub-genre, *game shooter* terus menjadi favorit di antara pemain di seluruh dunia.

Gambar 2. 2 *Space Pioneer (Vivid Games., 2018)*

Gambar 2.2 merupakan contoh *game Top-Down Shooter* yang bernama “Space Pioneer” (Vivid Games., 2018). Dalam "Space Pioneer", pemain mengambil peran sebagai seorang penjelajah luar angkasa yang bertugas menjelajahi berbagai planet, mengalahkan musuh-musuh yang datang, mengumpulkan sumber daya, dan memperkuat karakter serta peralatannya.

BAB III

ANALISIS DAN PERANCANGAN SISTEM

3.1. Analisis

Tujuan dari bagian analisis adalah untuk mengidentifikasi dan memahami secara mendalam masalah-masalah yang dihadapi dalam pengembangan kecerdasan buatan (AI) untuk karakter boss dalam game "From the Downtown," serta menentukan kebutuhan fungsional dan non-fungsional yang harus dipenuhi oleh sistem. Analisis ini mencakup pemahaman mengenai navigasi dan perilaku adaptif boss menggunakan algoritma A* dan Finite State Machine (FSM), serta memastikan bahwa solusi yang diusulkan dapat diimplementasikan dengan efisien, andal, dan mudah digunakan. Dengan demikian, analisis ini menjadi dasar bagi perancangan dan implementasi sistem yang efektif untuk meningkatkan pengalaman bermain dan tantangan bagi pemain.

3.1.1. Analisis masalah

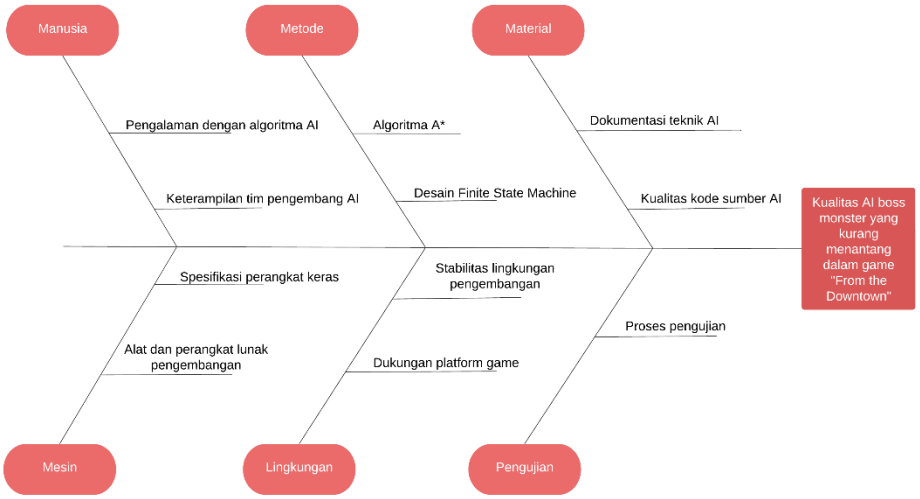
Dalam pengembangan video game, khususnya genre top-down shooter, kehadiran NPC (Non-Player Character) yang cerdas dapat meningkatkan kualitas gameplay secara signifikan. Salah satu NPC yang memiliki peran krusial adalah karakter boss. Boss harus menawarkan tantangan yang memadai, tidak hanya dari segi kekuatan fisik, tetapi juga kecerdasan dalam mengantisipasi gerakan dan serangan pemain. Implementasi kecerdasan buatan (AI) yang tepat untuk boss dalam game dapat memberikan pengalaman bermain yang lebih dinamis dan menantang.

3.1.2. Analisis kebutuhan

Untuk mengimplementasikan algoritma A* dan Finite State Machine (FSM) pada karakter boss dalam game top-down shooter "From the Downtown", diperlukan analisis yang mendetail tentang kebutuhan sistem. Kebutuhan ini meliputi aspek-aspek teknis, fungsional, dan non-fungsional yang harus dipenuhi agar implementasi AI pada karakter boss berjalan optimal.

1. Kebutuhan Fungsional
    a. Navigasi Boss dengan Algoritma A*
        - Boss harus mampu mencari rute terpendek untuk mengejar pemain di dalam peta permainan.
        - Boss harus dapat menghindari rintangan atau hambatan yang ada di peta selama pengejaran.
        - Algoritma A* harus dapat beradaptasi dengan perubahan posisi pemain secara real-time.
    b. Perilaku Boss dengan Finite State Machine
        - Boss harus memiliki beberapa state atau keadaan yang mencerminkan perilaku tertentu (e.g., Patrol, Chase, Attack).
        - Transisi antar state harus terjadi berdasarkan kondisi tertentu (e.g., jarak dengan pemain, jumlah poin darah boss).
        - Boss harus memiliki kemampuan untuk berubah fase serangan berdasarkan jumlah poin darah yang dimiliki.
2. Kebutuhan Non-Fungsional
    a. Kinerja
        - Sistem harus dapat bekerja secara real-time tanpa adanya lag yang signifikan.
        - Algoritma A* dan FSM harus efisien dalam penggunaan memori dan CPU agar tidak mengganggu performa permainan.
    b. Keandalan
        - Sistem harus mampu menangani berbagai skenario permainan tanpa mengalami crash atau bug.
        - Transisi antar state pada FSM harus terjadi dengan mulus tanpa gangguan.
    c. Kemudahan Penggunaan
        - Developer harus dapat dengan mudah menyesuaikan parameter dari A* dan FSM untuk tuning AI boss.
        - Dokumentasi yang jelas harus disediakan untuk membantu pengembangan dan debugging.

3.2. Diagram Ishikawa



Gambar 3. 1 Diagram Ishikawa

Berikut penjelasan Gambar 3.1:

1. Manusia (People)

   Faktor-faktor terkait dengan pengetahuan penulis.

   - Keterampilan penulis AI:

     Keterampilan teknis dan pemahaman mendalam tentang pengembangan AI yang dimiliki oleh penulis.
   - Pengalaman dengan algoritma AI:

     Pengalaman praktis dalam menerapkan algoritma seperti A* dan Finite State Machine dalam pengembangan game.

2. Metode (Methods)

   Pendekatan dan algoritma yang digunakan untuk pengembangan AI.

   - Pemilihan algoritma A*:

     Proses memilih dan menyesuaikan algoritma A* untuk navigasi dan perencanaan jalur boss monster.
   - Desain Finite State Machine:

     Cara merancang Finite State Machine (FSM) untuk mengatur perilaku boss monster agar lebih dinamis.

3. Material (Materials)

   Sumber daya yang digunakan dalam pengembangan AI.

   - Kualitas kode sumber AI:

     Kualitas dan kebersihan kode yang digunakan untuk mengimplementasikan AI, termasuk modularitas dan kemudahan pemeliharaan.

   - Dokumentasi teknik AI:

     Ketersediaan dokumentasi dan referensi yang memadai tentang teknik AI, yang dapat membantu tim dalam pengembangan dan pemecahan masalah.

4. Mesin (Machines)

   Perangkat keras dan perangkat lunak yang digunakan untuk pengembangan dan pengujian.

   - Spesifikasi perangkat keras:

     Kekuatan dan kemampuan perangkat keras yang digunakan untuk mengembangkan dan menguji AI, seperti komputer dengan prosesor, RAM, dan GPU yang memadai.

   - Alat dan perangkat lunak pengembangan:

     Perangkat lunak dan alat bantu seperti game engine, debugger, dan tools lain yang mendukung proses pengembangan AI.

5. Lingkungan (Environment)

   Lingkungan pengembangan dan pengujian, serta kondisi eksternal.

   - Stabilitas lingkungan pengembangan:

     Konsistensi dan stabilitas lingkungan pengembangan, termasuk sistem operasi, versi perangkat lunak, dan infrastruktur jaringan.

   - Dukungan platform game:

     Tingkat dukungan dan kompatibilitas dari platform game Unity, terhadap implementasi AI yang digunakan.

6. Pengujian

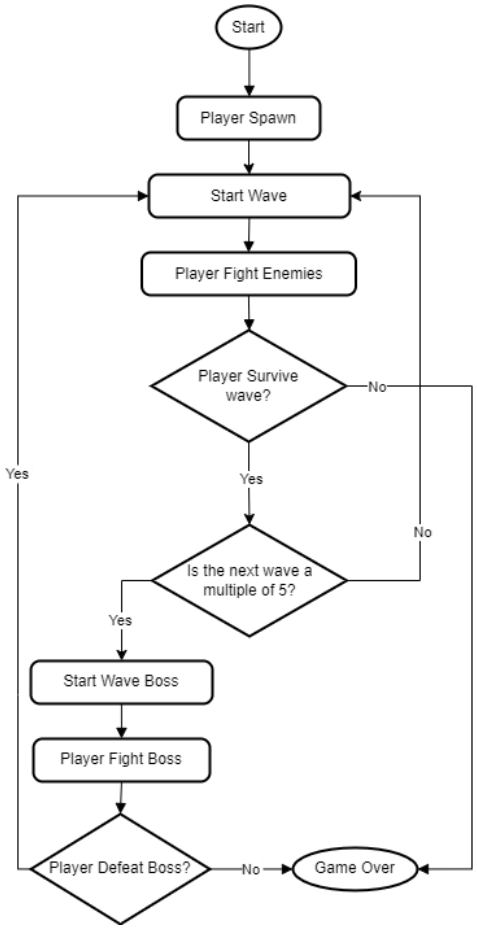
   Cara menguji kinerja AI.

   - Proses Pengujian:

     Menguji kinerja AI yang telah diterapkan pada karakter bos.

3.3. Alur *Game*

Alur dari permainan yang akan dibuat akan dimulai dengan pemain yang yang diberikan tugas untuk mengalahkan musuh. Berikut gambar alur *game*.



Gambar 3. 2 Alur *Game*

Pada Gambar 3.2 digambarkan alur dengan penjelasan:

1. Start: Permainan dimulai dari titik ini.
2. Player Spawn: Pemain muncul di dalam permainan.
3. Start Wave: *Wave* dimulai dan akan muncul musuh normal.
4. Player Fight Enemies: Pemain harus melawan musuh-musuh tersebut.

5. Player Survive Wave?: Kondisi untuk memeriksa apakah pemain berhasil bertahan.
    - Yes: Jika pemain berhasil bertahan, lanjut ke langkah berikutnya.
    - No: Jika pemain tidak berhasil bertahan, permainan berakhir (Game Over).
6. Is the next wave a multiple of 5?: Kondisi untuk memeriksa apakah wave selanjutnya adalah wave kelipatan 5.
    - Yes: Jika wave selanjutnya kelipatan 5, lanjut Start Wave Boss.
    - No: Jika tidak kembali Start Wave.
7. Start Wave Boss: *Wave* dimulai, musuh normal dan boss akan muncul.
8. Player Fight Boss: Pemain harus melawan boss.
9. Player Defeat Boss?: Kondisi untuk memeriksa apakah pemain berhasil mengalahkan boss.
    - Yes: Jika pemain berhasil mengalahkan boss, proses kembali ke langkah awal untuk memulai lagi dengan spawning musuh.
    - No: Jika pemain tidak berhasil mengalahkan boss, permainan berakhir (Game Over).

3.4. Perancangan Model Bos

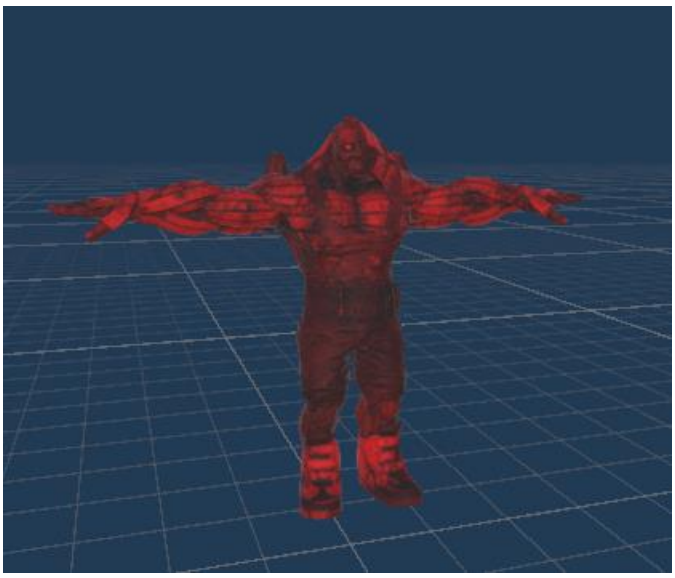
Pada game akan dibuat 2 jenis bos yaitu, *Melee Boss,* dan *Range Boss.* Setiap bos memiliki 2 model 3D yaitu normal dan berkobar. Mode normal adalah mode awal bos saat muncul, dan mode berkobar adalah mode saat bos memiliki darah 30% dari total darah yang dimiliki. Setiap mode akan memiliki 2 buah gerakan menyerang tergantung dari jarak pemain. Selain itu, diperlukan animasi untuk setiap gerakan dari bos. Tiap bos memiliki 6 gerakan yang terdiri dari diam, berjalan, dan 4 gerakan menyerang.

3.4.1. Model Melee Boss

Melee Boss merupakan boss yang menyerang pemain dengan jarak dekat. *Melee Boss* akan memiliki 4 gerakan yaitu, 2 di mode normal, dan 2 di mode berkobar. Pada mode normal boss memiliki combo attack dan jump attack. Sedangkan pada mode berkobar boss memiliki combo attack 2 dan swing attack. Adapun model 3D boss yang dirancang dapat dilihat pada Gambar 3.3.

Gambar 3. 3 Model 3D *Melee Boss* mode normal

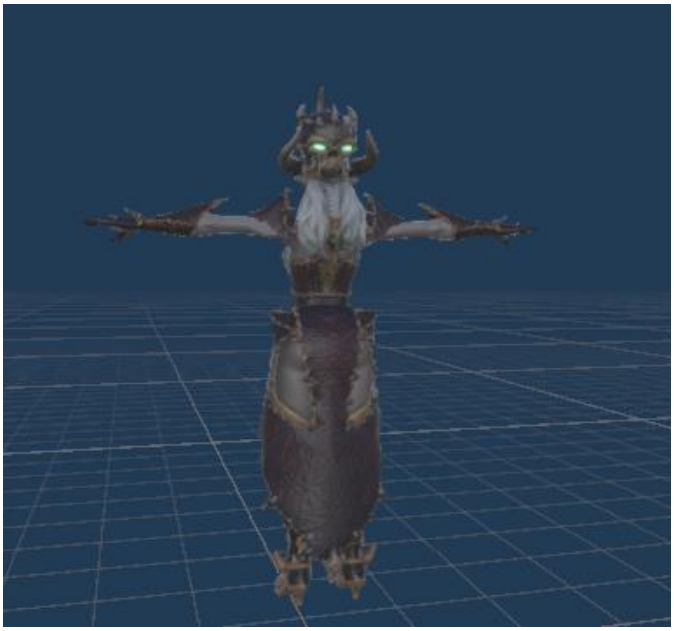
Untuk tampilan model 3D *melee boss* mode berkobar dilihat pada gambar 3.4.



Gambar 3. 4 Model 3D *Melee Boss* mode berkobar

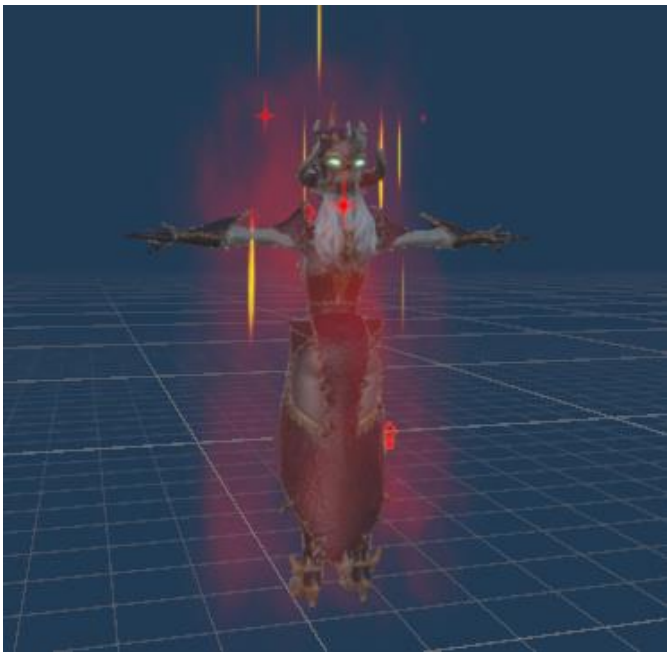
3.4.2. Model Range Boss

Range Boss merupakan boss yang menyerang pemain dari jarak jauh dengan menembak. *Range Boss* akan memiliki 4 gerakan yaitu, 2 di mode normal, dan 2 di mode berkobar. Pada mode normal boss memiliki Fireball dan Explosive. Sedangkan pada mode berkobar boss memiliki Fireball 2 dan Bigshoot. Adapun model boss yang dirancang dapat dilihat pada Gambar 3.5.



Gambar 3. 5 Model 3D *Range Boss* mode normal

Untuk tampilan model 3D *melee boss* mode berkobar dilihat pada Gambar 3.6.



Gambar 3. 6 Model 3D *Range Boss* mode berkobar

3.5. Perancangan A*

Algoritma A* (A-star) adalah salah satu algoritma pencarian jalur terpendek yang sering digunakan dalam pengembangan game untuk navigasi karakter, termasuk karakter boss. Untuk mengimplementasikan A* pada karakter boss dalam game "From the Downtown," langkah pertama adalah membuat navigasi mesh (navmesh) pada peta

permainan. Navmesh akan menjadi dasar bagi algoritma A* untuk menentukan jalur optimal yang harus dilalui oleh karakter boss.

3.5.1. Pembuatan Navmesh

Navmesh adalah representasi dari area navigasi yang dapat dilalui oleh karakter dalam bentuk mesh yang terdiri dari poligon-poligon yang saling terhubung (Zikky, 2016). Adapun hal yang dilakukan adalah mengidentifikasi area yang dapat dilalui dan yang tidak dapat dilalui pada peta permainan, seperti dinding, dan rintangan. Kemudian akan menandai lantai pada peta dengan layer *Ground*, dan benda-benda seperti bangunan, dan gedung sebagai *Obstacle*.

*3.5.2. Implementasi Pathfinding dengan Algoritma A**

Setelah navmesh selesai dibuat, langkah selanjutnya adalah mengimplementasikan algoritma A* untuk pathfinding pada karakter boss. Adapun langkah-langkah cara kerjanya sebagai berikut:

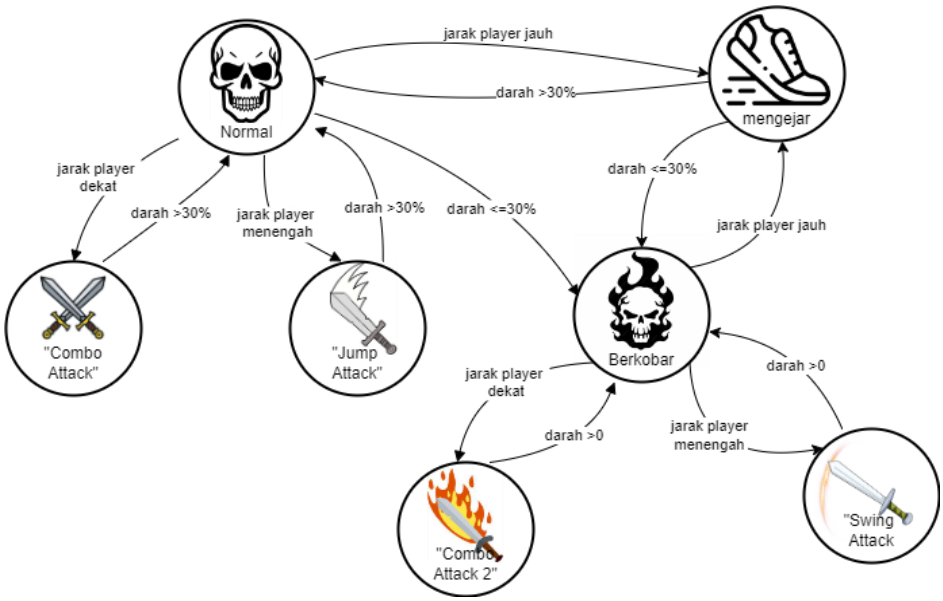
1. Inisialisasi Algoritma:
    a. Tentukan node awal (posisi awal boss) dan node tujuan (posisi pemain) di navmesh.
    b. Inisialisasi Open List (daftar node yang akan dievaluasi) dan Closed List (daftar node yang sudah dievaluasi).
2. Pencarian Jalur:
    a. Mulai dari node awal, tambahkan node ini ke Open List.
    b. Lakukan iterasi sebagai berikut hingga tujuan tercapai atau Open List kosong:
        - Pilih node dengan nilai f (g + h) terendah dari Open List. Nilai g adalah biaya dari awal ke node ini, dan nilai h adalah estimasi biaya dari node ini ke tujuan menggunakan heuristik jarak antara node.
        - Pindahkan node ini dari Open List ke Closed List.
        - Evaluasi semua tetangga dari node ini (poligon yang berbatasan langsung).
        - Jika tetangga adalah node tujuan, rekonstruksi jalur dengan melacak kembali node dari tujuan ke awal melalui node-node yang dipilih.

- Jika tetangga belum ada di Closed List atau Open List, hitung nilai g, h, dan f, lalu tambahkan tetangga ini ke Open List.
- Jika tetangga sudah ada di Open List, periksa apakah jalur baru ini lebih baik (nilai g lebih rendah). Jika ya, perbarui nilai g, h, dan f serta tetapkan parent node baru.

3. Integrasi dengan Karakter Boss:
   a. Setelah jalur optimal ditemukan, integrasikan jalur ini dengan sistem navigasi karakter boss.
   b. Buat karakter boss mengikuti jalur yang telah ditentukan dengan bergerak dari satu node ke node berikutnya hingga mencapai tujuan.
   c. Implementasikan penyesuaian jalur secara dinamis jika posisi pemain berubah selama pengejaran, dengan memulai kembali pencarian jalur menggunakan algoritma A*.

3.6. Perancangan Finite State Machine

3.6.1. FSM melee boss



Gambar 3. 7 *Melee Boss Finite State Machine*

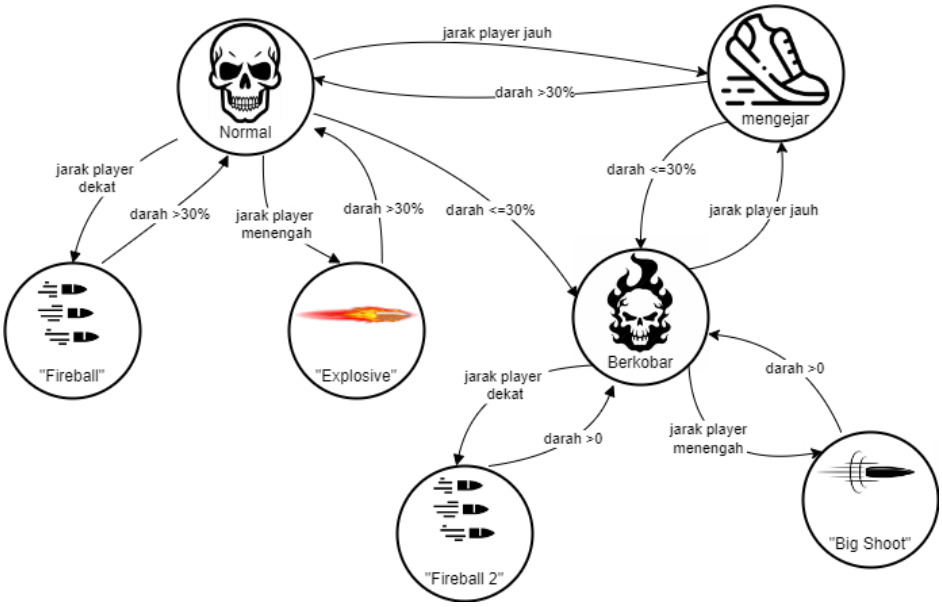
Pada Gambar 3.7 merupakan diagram yang menunjukkan state dari melee boss adapun state awal adalah “normal” dimana boss memiliki gerakan menyerang jika dalam kondisi darah diatas 30%, yang nantinya akan berpindah ke state “berkobar” jika darah boss dibawah 30% serta mengubah gerakan serangannya. Boss akan berpindah state

tergantung jaraknya dengan pemain. Jika pemain dalam jarak dekat maka akan menyerang dengan combo attack, jika jarak menengah akan menyerang dengan jump attack, dan jika jarak dengan pemain jauh maka akan mengejar pemain. Untuk dalam state berkobar serangannya akan berubah menjadi combo attack 2 untuk dekat, dan swing attack untuk menengah.

Adapaun penjelasan singkat untuk tiap serangan, yaitu:

1. Combo Attack: Boss melakukan serangkaian pukulan atau tusukan dengan cepat. Serangan ini memiliki kecepatan tinggi tetapi mungkin memiliki sedikit kerusakan.
2. Jump Attack: Boss melompat atau berlari ke arah pemain dan kemudian melakukan serangan kuat dari atas atau samping. Serangan ini memiliki kerusakan besar.
3. Combo Attack 2: Sama dengan serangan beruntun tetapi kerusakan yang diberikan boss akan lebih banyak.
4. Swing Attack: Boss mengejar pemain dengan kecepatan yang sangat tinggi dan melakukan serangan yang sangat kuat ketika berhasil mencapai pemain. Serangan ini mungkin sulit dihindari dan memiliki kerusakan besar.

3.6.2. FSM range boss

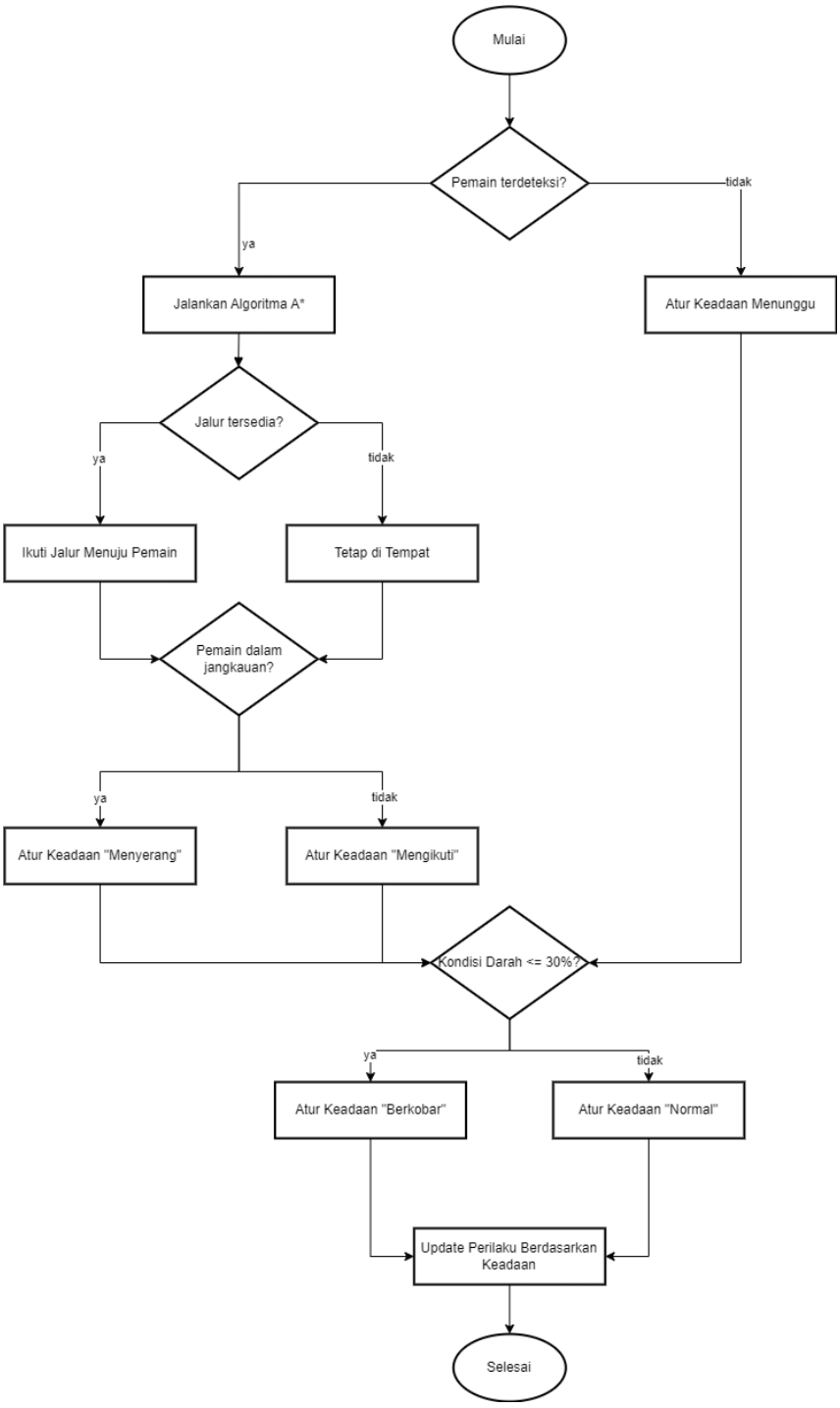


Gambar 3. 8 *Range Boss Finite State Machine*

Pada Gambar 3.8 terlihat state yang dimiliki oleh range boss. Diagram diatas mirip dengan diagram FSM untuk melee boss, perbedaan hanya terletak pada gerakan menyerangnya. Adapun penjelasan singkat untuk gerakan menyerang yang dimiliki, yaitu:

1. Fireball: Boss melepaskan serangan proyektil secara beruntun ke arah pemain dalam jumlah besar dengan kecepatan tinggi. Serangan ini mungkin memiliki akurasi tinggi tetapi kerusakannya mungkin lebih rendah.
2. Explosive: Boss menembakkan proyektil yang meledak di titik kerusakan, menciptakan area efek ledakan yang luas dan memberikan kerusakan besar kepada pemain yang terkena dampak langsung atau dampak ledakan.
3. Fireball 2: Serangan ini sama dengan “Tembakan Beruntun”, akan tetapi memiliki jumlah tembakan lebih banyak.
4. Big Shoot: Boss menembakkan proyektil yang memiliki diameter besar, memberikan kerusakan besar dan damage yang berlanjut kepada target yang terkena. Serangan ini mungkin sedikit lebih lambat dan memerlukan akurasi yang baik untuk mengenai target.

3.7. Flowchart Sistem Bos



Gambar 3. 9 *Flowchart* Sistem Bos

Gambar 3.9 merupakan flowchart dari sistem dari boss. Adapun penjelasan dari flowchart sebagai berikut:

1. Inisialisasi: Alur dimulai dengan inisialisasi, yang menandakan awal dari siklus perilaku bos.

2. Pemeriksaan Pemain: Langkah pertama dalam alur adalah memeriksa apakah pemain terdeteksi oleh bos. Jika pemain terdeteksi, bos akan beralih ke langkah selanjutnya untuk mengevaluasi dan mengambil tindakan. Jika tidak boss akan dalam keadaan menunggu.
3. Algoritma A* dan Penentuan Jalur: Jika pemain terdeteksi, boss akan menjalankan algoritma A* untuk menentukan jalur terpendek ke arah pemain. Ini berarti boss akan mencoba mendekati atau mengejar pemain untuk melakukan serangan.
4. Penentuan Keadaan: Berdasarkan hasil algoritma A* dan jarak antara bos dan pemain, bos akan menentukan keadaan yang sesuai. Dalam flowchart ini, ada dua kemungkinan keadaan: "Normal" dan "Berkobar".
5. Serangan Normal: Jika boss berada dalam keadaan "Normal", ia akan memiliki dua pilihan serangan: "Serang Beruntun" dan "Serang Gempur" untuk melee boss, “Tembakan Beruntun” dan “Tembakan Meledak” untuk range boss. Boss akan memilih untuk melakukan salah satu serangan ini sesuai dengan strategi yang diambil dalam keadaan normal.
6. Berkobar: Jika kondisi boss berubah menjadi "Berkobar", ia akan memiliki opsi serangan tambahan yang aktif: "Serang Berapi" dan "Serang Pengejar" untuk melee boss, “Tembakan Beruntun 2” dan “Tembakan Berat” untuk range boss. Kondisi "Berkobar" terjadi ketika darah bos turun di bawah 30%.
7. Pelepasan Serangan: Setelah memilih serangan yang sesuai, boss akan meluncurkan serangan yang dipilihnya.
8. Update Perilaku: Terakhir, boss akan memperbaharui perilakunya berdasarkan tindakan yang diambil dan hasil serangan. Ini mungkin melibatkan kembali ke keadaan normal setelah serangan selesai atau tetap berada dalam keadaan "Berkobar" jika kondisi masih terpenuhi.

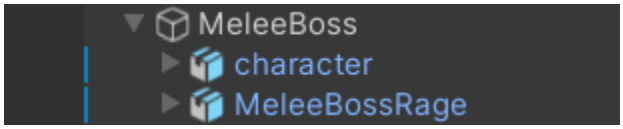
Dengan demikian, flowchart tersebut menggambarkan alur logika perilaku boss, termasuk penentuan jalur, serangan, dan respons terhadap perubahan kondisi dalam permainan. Ini membantu menjelaskan bagaimana bos akan berperilaku dan berinteraksi dengan pemain saat bermain game.

BAB IV

IMPLEMENTASI DAN PENGUJIAN SISTEM

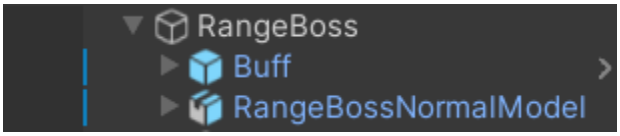
4.1. Implementasi Karakter Bos

Import model 3D dan animasi yang dibutuhkan kedalam projek. Buat 2 objek baru dan namakan sebagai MeleeBoss dan juga RangeBoss. Tarik model 3D kedalam scene menjadi child dari objek MeleeBoss dan RangeBoss.



Gambar 4. 1 Objek MeleeBoss

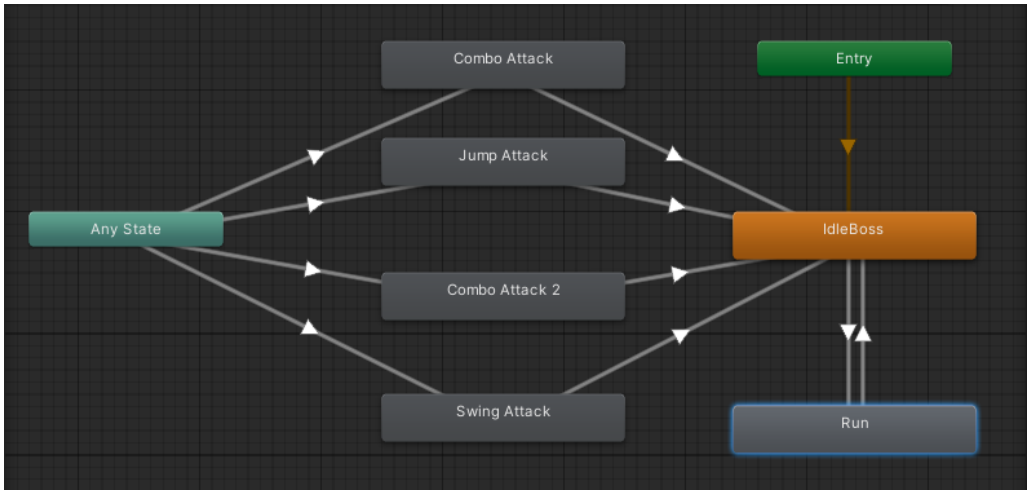
Pada Gambar 4.1 MeleeBoss merupakan objek *parent* dari objek character dan MeleeBossRage. Character adalah model 3D dari mode normal *melee boss* dan MeleeBossRage adalah model 3D dari mode berkobar *melee boss*.



Gambar 4. 2 Objek RangeBoss

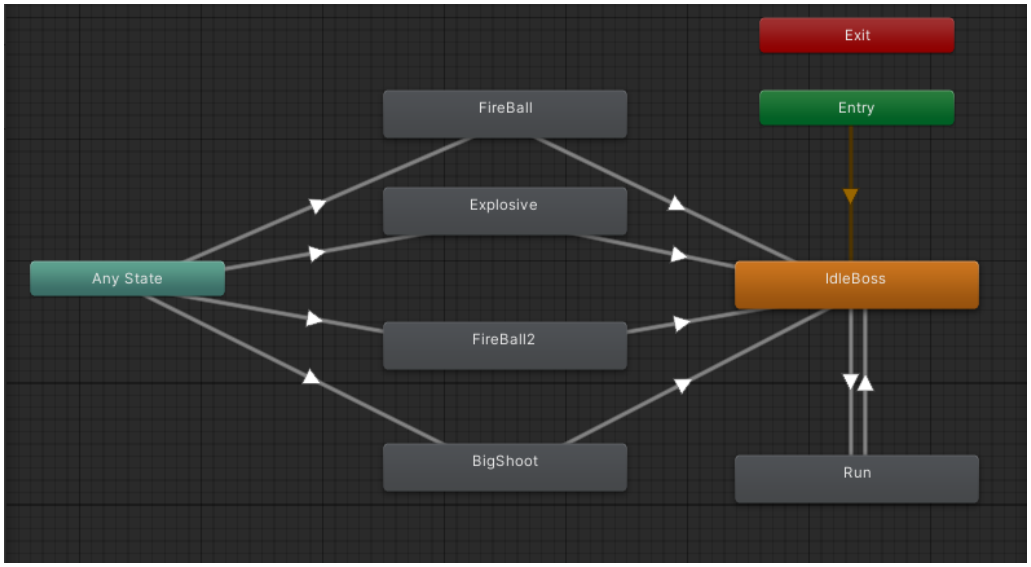
Pada Gambar 4.2 RangeBoss merupakan *parent* dari objek Buff dan RangeBossNormalModel. Seperti namanya RangeBossNormalModel adalah model 3D dari mode normal *range boss* dan Buff merupakan efek animasi saat bos memasuki mode berkobar.

Selanjutnya, atur animasi yang dibutuhkan kedalam *animator controller*. Pada *animator controller* tiap animasi dihubungkan dengan *transition*.



Gambar 4. 3 *Animator Controller Melee Boss*

Pada Gambar 4.3 terdapat animasi *idle*, *run,* dan 4 animasi *attack*. Seluruh animasi *attack* diawali dengan *any state* yang berarti animasi *attack* dapat dijalankan saat sedang berada di *state* apapun begitu bos menyerang. *State idle* terhubung dengan *entry* yang menandakan *idle* merupakan *state default.*

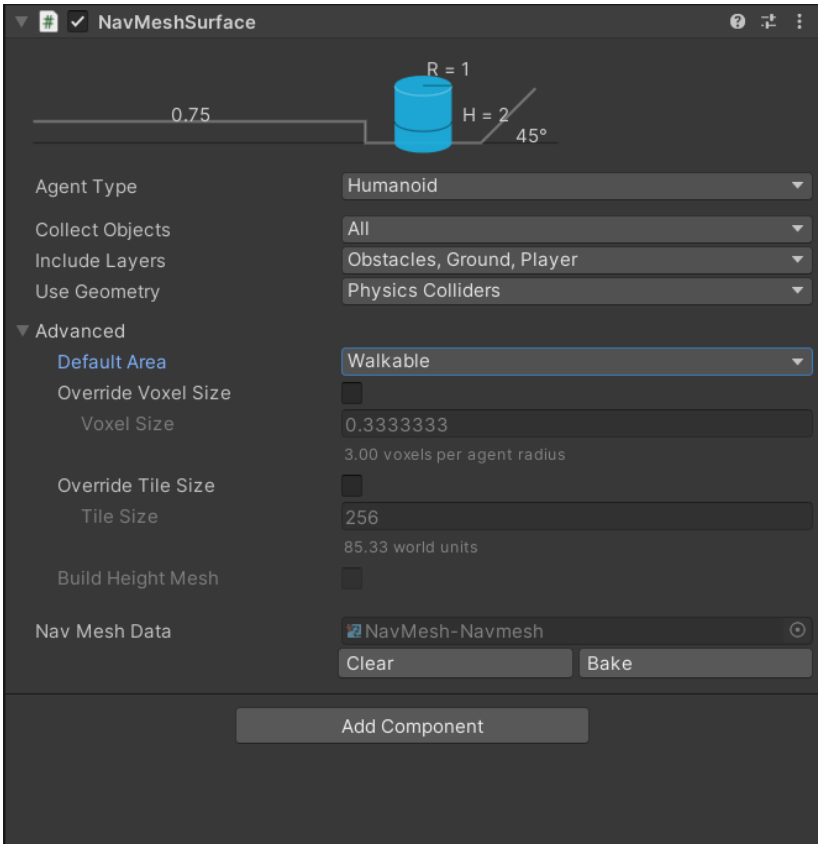


Gambar 4. 4 *Animator Controller Range Boss*

Sama halnya dengan *animator* untuk *melee boss,* Pada Gambar 4.4 terdapat animasi *idle*, *run,* dan 4 animasi *attack*. Desain *state* untuk *range boss* serupa dengan desain *state* untuk *melee boss.* Dimana Seluruh animasi *attack* diawali dengan *any state,* dan *state idle* terhubung dengan *entry* yang menandakan *idle* merupakan *state default.*

4.2. Implementasi NavMesh

Pada tahap ini dibuat sebuah objek NavMesh, pada objek ini dimasukkan component NavMeshSurface melalui *framework* TopDown Engine.

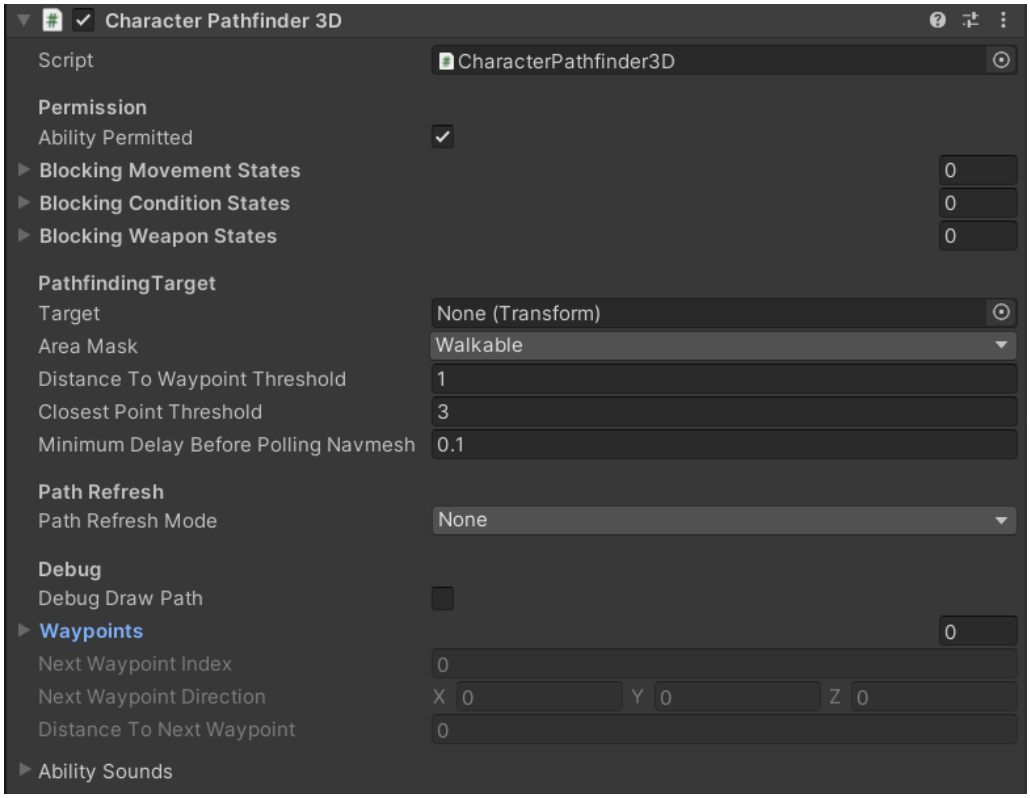


Gambar 4. 5 NavMeshSurface

Gambar 4.5 menunjukkan beberapa parameter yang dapat diatur, parameter yang diatur pertama adalah *layers* yang berfungsi untuk pemetaan area yang dapat dilalui dan yang tidak dapat dilalui pada peta permainan. Selanjutnya, menentukan lapisan mana saja yang akan disertakan dalam pembuatan NavMesh. Lapisan yang disertakan adalah *Obstacles, Ground, Player.* Pada *Default Area* tentukan area bawaan *NavMesh* untuk area yang dapat dilalui *(Walkable)*.

4.3. Implementasi A*

Untuk menerapkan algoritma A* pada karakter bos tambahkan *component* Character Pathfinder 3D yang disediakan *framework* TopDown Engine pada objek *parent* dari karakter bos. Komponen ini memungkinkan karakter mengelola navigasi menggunakan NavMesh yang telah dibuat sebelumnya. Parameter yang dapat diatur dalam komponen dapat dilihat pada Gambar 4.6.



Gambar 4. 6 *Component Character Pathfinder 3D*

Hal pertama yang perlu diatur adalah centang pada kotak *ablity permitted,* ini untuk mengizinkan komponen ini berjalan. Untuk poin target biarkan kosong karena poin ini akan terisi otomatis ke pemain dikarenakan karakter bos adalah musuh. Pada *area mask* pilih area yang dapat dilalui *(walkable)* agar karater bos hanya bisa melalui area yang dapat dilalui. Atur *distance to waypoint threshold* menjadi 1, yang artinya jarak karakter bos dianggap telah mencapai *waypoint* saat karakter bos dan pemain berjarak 1 meter. Selainnya biarkan tetap pada pengaturan bawaan.

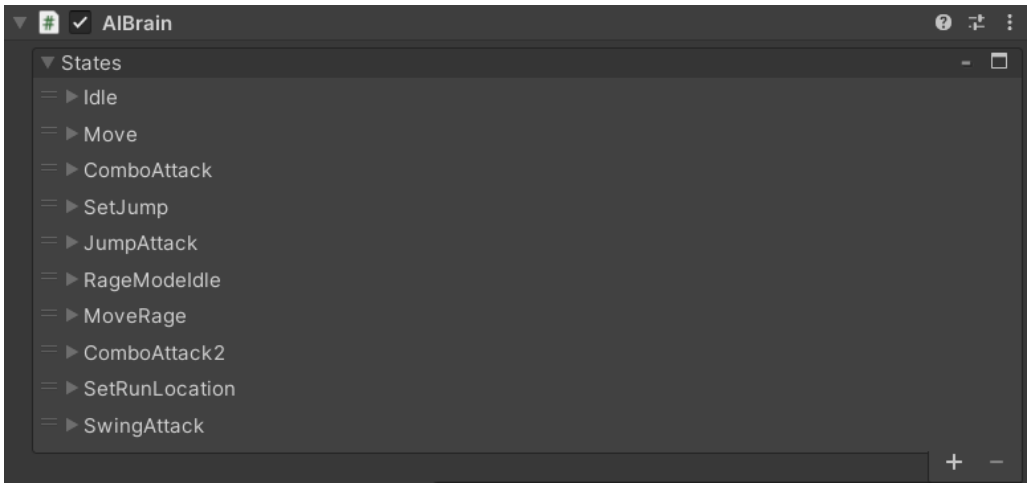
4.4. Implementasi Finite State Machine

Untuk mengatur perilaku karakter, *framework* TopDown Engine menyediakan sebuah komponen bernama AIBrain. AIBrain menggunakan cara kerja FSM (*Finite State Machine)* dimana terdiri dari kumpulan *state* yang terhubung dengan transisi.

4.4.1. FSM melee boss

Pada AIBrain *melee boss* tambahkan beberapa *state,* yaitu Idle, Move, ComboAttack, SetJump, JumpAttack, RageModeIdle, MoveRage, ComboAttack2, SetRunLocation,

SwingAttack. Tampilan komponen AIBrain untuk *melee boss* dapat dilihat pada Gambar 4.7.



Gambar 4. 7 *Melee Boss States*

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

1. Idle
   - Actions: Diam
   - Transitions: Deteksi pemain, lanjut *state* "Move" atau *state "RageModeIdle"* jika darah dibawah 30%.
2. Move
   - Actions: Bergerak menuju pemain
   - Transitions: Menghitung jarak antara bos dengan pemain, lanjut *state* "ComboAttack" jika jarak <= 4 atau *state* "SetJump" jika jarak <=8
3. ComboAttack
   - Actions: Serang
   - Transitions: kembali ke *state* "Idle"
4. SetJump
   - Actions: Menentukan titik lokasi pemain
   - Transitions: lanjut *state* "JumpAttack"
5. JumpAttack
   - Actions: Serang
   - Transitions: kembali ke *state* "Idle"

6. RageModeIdle
    - Actions: Diam
    - Transitions: Deteksi pemain, lanjut *state* “Move”
7. MoveRage
    - Actions: Bergerak menuju pemain
    - Transitions: Menghitung jarak antara bos dengan pemain, lanjut *state* “ComboAttack2” jika jarak <= 4 atau *state* “SetRunLocation” jika jarak <=8
8. ComboAttack2
    - Actions: Serang
    - Transitions: kembali ke *state* “RageModeIdle”
9. SetRunLocation
    - Actions: Menentukan titik lokasi pemain
    - Transitions: lanjut *state* “SwingAttack”
10. SwingAttack
    - Actions: Lari dan serang
    - Transitions: kembali ke *state* “RageModeIdle”

Selain itu kita perlu menuliskan kode baru untuk mengubah model 3D dari bos saat memiliki darah dibawah 30%. Cara kerjanya dengan membandingkan darah bos sekarang dengan 30%.

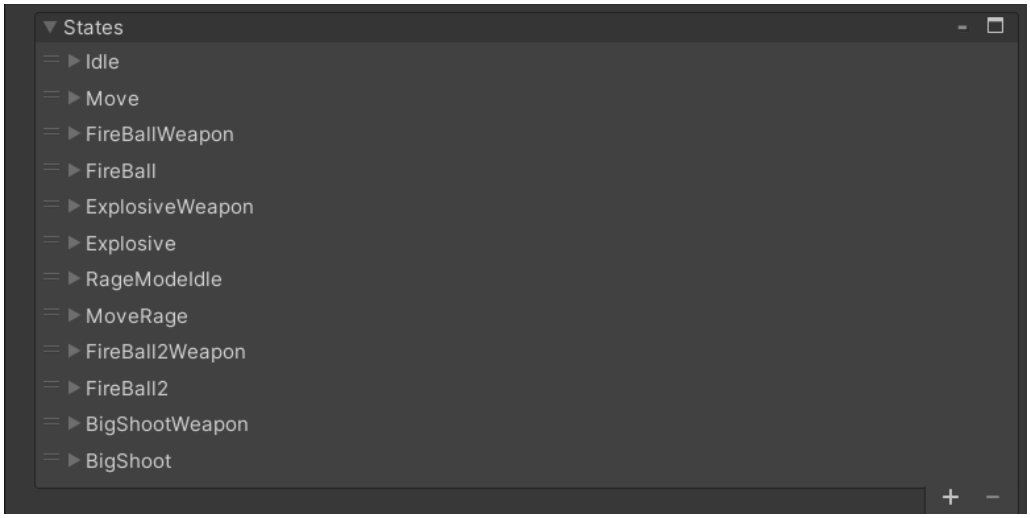
```
144     // Switch to the next model in line
145     protected void SwitchModel()
146     {
147         if (CharacterModels.Length <= 1)
148         {
149             return;
150         }
151 
152         CharacterModels[CurrentIndex].SetActive(false);
153 
154         // Determine the next index
155         if (NextCharacterChoice == NextModelChoices.Random)
156         {
157             CurrentIndex = Random.Range(0, CharacterModels.Length);
158         }
159         else
160         {
161             CurrentIndex = CurrentIndex + 1;
162             if (CurrentIndex >= CharacterModels.Length)
163             {
164                 CurrentIndex = 0;
165             }
166         }
167 
168         // Activate the new current model
169         CharacterModels[CurrentIndex].SetActive(true);
170         _character.CharacterModel = CharacterModels[CurrentIndex];
171 
172         // Bind animator if needed
173         if (AutoBindAnimator)
174         {
175             _character.CharacterAnimator = CharacterModels[CurrentIndex].GetComponent<Animator>();
176             _character.AssignAnimator(true);
177             SendMessage(_bindAnimatorMessage, SendMessageOptions.DontRequireReceiver);
178 
179             List<CharacterHandleWeapon> handleWeapons = _character.FindAbilities<CharacterHandleWeapon>();
180             foreach (CharacterHandleWeapon handleWeapon in handleWeapons)
181             {
182                 if ((handleWeapon.AutomaticallyBindAnimator) && (handleWeapon.CurrentWeapon != null))
183                 {
184                     handleWeapon.CharacterAnimator = _character.CharacterAnimator;
185                     handleWeapon.CurrentWeapon.SetOwner(_character, handleWeapon);
186                 }
187             }
188         }
```

Gambar 4. 8 Kode Switch Model

Dengan kode pada Gambar 4.8 model 3D dari bos akan berubah jika perbandingan sebelumnya bernilai *true*, dengan cara menonaktifkan model yang sedang aktif sekarang, kemudian mengaktifkan model selanjutnya.

4.4.2. FSM range boss

Pada AIBrain *range boss* tambahkan beberapa *state,* yaitu Idle, Move, ComboAttack, SetJump, JumpAttack, RageModeIdle, MoveRage, ComboAttack2, SetRunLocation, SwingAttack. Tampilan komponen AIBrain untuk *range boss* dapat dilihat pada Gambar 4.9.



Gambar 4. 9 *Range Boss States*

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

1. Idle
   - Actions: Diam
   - Transitions: Deteksi pemain, lanjut *state* "Move" atau *state "RageModeIdle"* jika darah dibawah 30%.
2. Move
   - Actions: Bergerak menuju pemain
   - Transitions: Menghitung jarak antara bos dengan pemain, lanjut *state* "FireBallWeapon" jika jarak <= 15 atau *state* "ExplosiveWeapon" jika jarak <=17
3. FireBallWeapon
   - Actions: Mengganti ke senjata FireBall
   - Transitions: lanjut ke *state* "FireBall"
4. FireBall
   - Actions: Serang
   - Transitions: kembali ke *state* "Idle"
5. ExplosiveWeapon
   - Actions: Mengganti ke senjata Explosive
   - Transitions: lanjut ke *state* "Explosive"

6. Explosive
    - Actions: Serang
    - Transitions: kembali ke *state* “Idle”
7. RageModeIdle
    - Actions: Diam
    - Transitions: Deteksi pemain, lanjut *state* “Move”
8. MoveRage
    - Actions: Bergerak menuju pemain
    - Transitions: Menghitung jarak antara bos dengan pemain, lanjut *state* “FireBall2Weapon” jika jarak <= 15 atau *state* “BigShootWeapon” jika jarak <=17
9. FireBall2Weapon
    - Actions: Mengganti ke senjata FireBall2
    - Transitions: lanjut *state* “FireBall2”
10. FireBall2
    - Actions: Serang
    - Transitions: kembali ke *state* “RageModeIdle”
11. BigShootWeapon
    - Actions: Mengganti ke senjata BigShoot
    - Transitions: lanjut *state* “SwingAttack”
12. BigShoot
    - Actions: Serang
    - Transitions: kembali ke *state* “RageModeIdle”

Kemudian buat kode baru untuk mengaktifkan aura effect saat darah bos dibawah 30%. Cara kerjanya sama saat mengganti model 3D untuk *melee boss*.

```
108     // Activate the effect object
109     protected void ActivateEffect()
110     {
111         if (EffectObject != null)
112         {
113             EffectObject.SetActive(true);
114         }
115     }
```

Gambar 4. 10 *Activate Effect Objek*

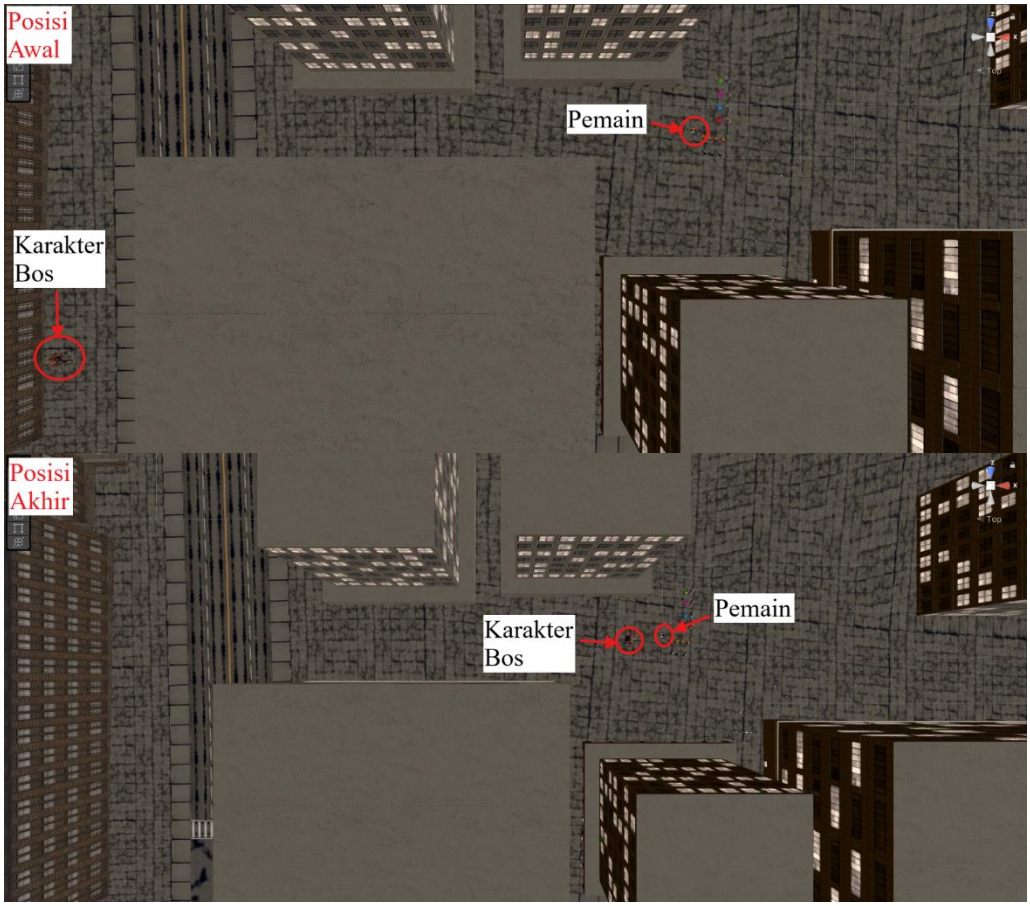
Cara kerja kode pada Gambar 4.10 hanya mengaktifkan objek *child* dari karakter bos yaitu aura efek bos. Pada bagian *melee boss* kita menonaktifkan objek *child* dari karakter *melee boss* yaitu model 3D mode normalnya, sedangkan pada *range boss* tidak ada objek yang dinonaktifkan.

4.5. Pengujian

Untuk pengujian karakter bos dilakukan dengan langsung dengan memainkan permainan.

Gambar 4. 11 Uji *Path Finding 1*

Gambar 4.11 adalah pengujian 1 yang dilakukan dengan menggerakkan pemain, terlihat karakter bos dapat bergerak mengikuti pemain.

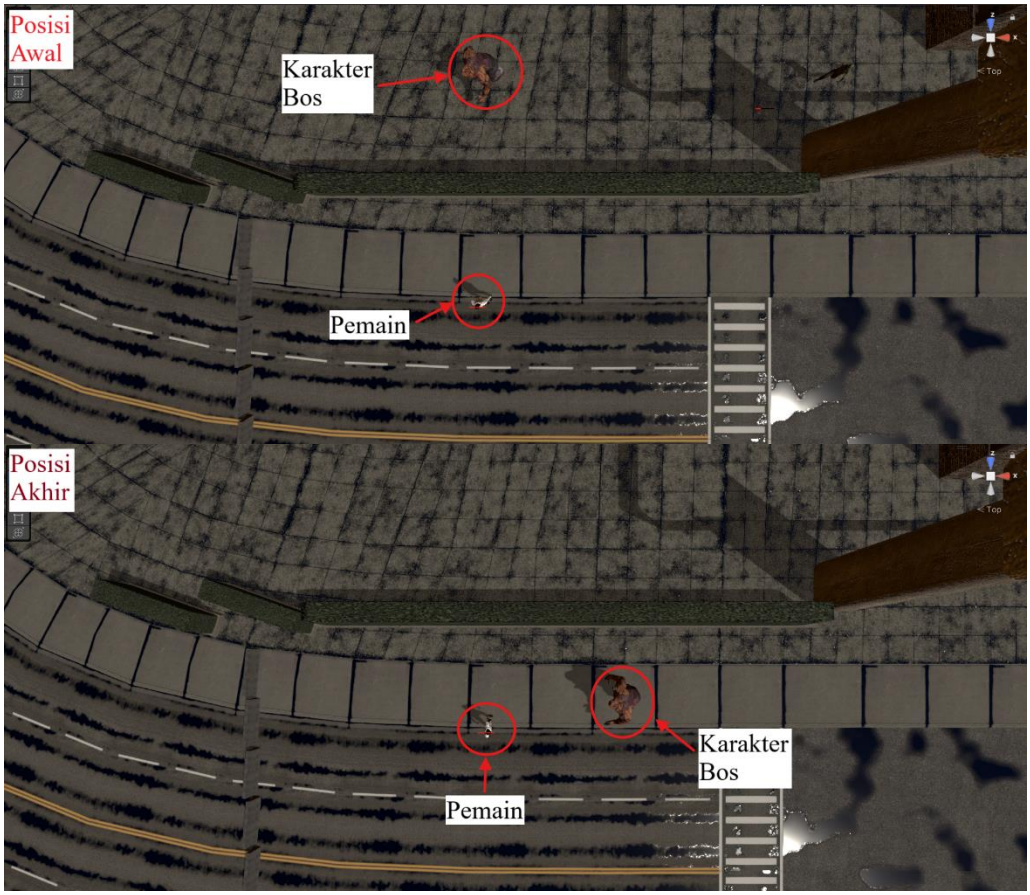


Gambar 4. 12 Uji *Path Finding 2*

Pada gambar 4.12 karakter bos dan pemain memiliki jarak yang jauh, dimana pada peta permainan terdapat jalan persimpangan. Terlihat karakter bos dapat mencari jalur untuk mendekati pemain.

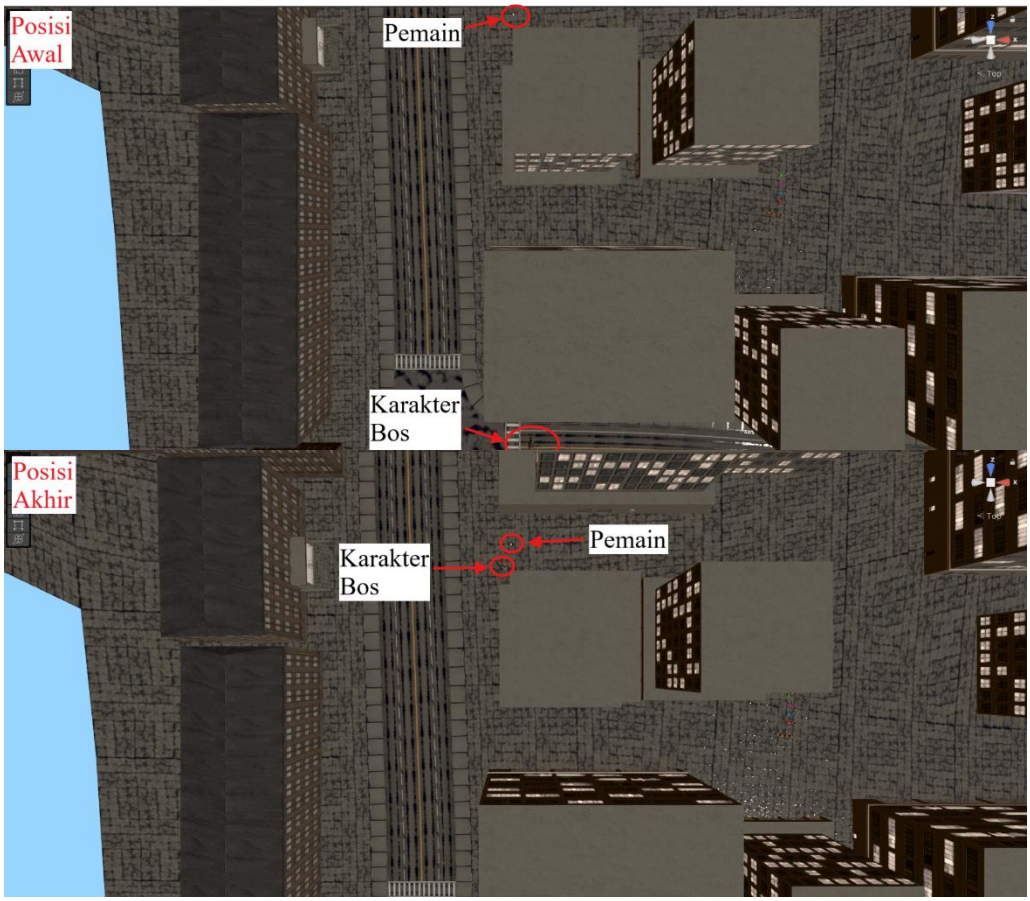
Gambar 4. 13 Uji *Path Finding 3*

Pada uji *path finding* ke-3 karakte bos dan pemain dipisahkan oleh sebuah gedung seperti pada Gambar 4.13, terlihat karakter bos berhasil menghampiri pemain.



Gambar 4. 14 Uji *Path Finding 4*

Pada uji *path finding* ke-4, karakter bos dan pemain dipisahkan dengan pagar seperi pada Gambar 4.14. Hasil menunjukkan karakter bos dapat menghampiri pemain.



Gambar 4. 15 Uji *Path Finding 5*

Pada uji *path finding* ke-5, pemain dan karakter bos diletakkan dengan jarak yang jauh seperti pada Gambar 4.14. Hasilnya karakter bos juga berhasil menghampiri pemain.

Untuk FSM *(Finite State Machine),* dalam permainan karakter bos menunjukkan perilaku yang sesuai di dalam permainan. Dilakukan beberapa tes dengan memberikan bos beberapa skenario, hasil dari tes *melee boss* dapat dilihat pada Tabel 4.1.

Tabel 4. 1 Hasil FSM *Melee Boss*

Status	Keadaan	Aksi	Hasil
Mengejar	Jika pemain berjarak > 8 meter	Karakter bos bergerak menuju pemain	Sesuai
Mode Berkobar	Jika karakter bos memiliki 30% poin darah dari total poin darah	Karakter bos memiliki model 3D yang berbeda.	Sesuai
Combo Attack	Jika pemain berjarak <= 4 meter	Karakter bos melancarkan Combo Attack	Sesuai
Jump Attack	Jika pemain berjarak <= 8 meter	Karakter bos melancarkan Jump Attack	Sesuai
Combo Attack 2	Jika pemain berjarak <= 4 meter dan karakter bos memiliki 30% poin darah dari total poin darah	Karakter bos melancarkan Combo Attack 2	Sesuai
Swing Attack	Jika pemain berjarak <= 4 meter dan karakter bos memiliki 30% poin darah dari total poin darah	Karakter bos melancarkan Swing Attack	Sesuai

Untuk hasil dari tes dari *range boss*, ditampilkan pada Tabel 4.2.

Tabel 4. 2 Hasil FSM *Range Boss*

Status	Keadaan	Aksi	Hasil
Mengejar	Jika pemain berjarak > 17 meter	Karakter bos bergerak menuju pemain	Sesuai
Mode Berkobar	Jika karakter bos memiliki 30% poin darah dari total poin darah	Karakter bos memiliki aura efek.	Sesuai
FireBall	Jika pemain berjarak <= 15 meter	Karakter bos melancarkan FireBall	Sesuai
Explosive	Jika pemain berjarak <= 17 meter	Karakter bos melancarkan Explosive	Sesuai
FireBall 2	Jika pemain berjarak <= 15 meter dan karakter bos memiliki 30% poin darah dari total poin darah	Karakter bos melancarkan FireBall 2	Sesuai
Big Shoot	Jika pemain berjarak <= 17 meter dan karakter bos memiliki 30% poin darah dari total poin darah	Karakter bos melancarkan Big Shoot	Sesuai

BAB V

KESIMPULAN DAN SARAN

5.1. Kesimpulan

Berdasarkan hasil implementasi dan pengujian Algoritma A* dan FSM (*fFinite State Machine)* ke dalam karakter bos pada game "From The Downtown", diperoleh kesimpulan sebagai berikut:

1. Dari 5 tes skenario yang diberikan, Algoritma A* *(pathfinding)* memberikan hasil sukses untuk semua skenario.
2. FSM *(Finite State Machine)* menghasilkan perilaku karakter bos yang sesuai dengan berbagai keadaan yaitu, mengejar, mode berkobar, dan 4 variasi gerakan menyerang.
3. Dengan kemampuan boss untuk mengejar pemain, dan memberikan variasi gerakan dalam menyerang, maka boss memberikan peningkatan tantangan dalam bermain.

5.2. Saran

Saran yang diberikan untuk penelitian selanjutnya adalah sebagai berikut:

1. Untuk FSM (*fFinite State Machine)* yang diterapkan pada karakter bos, dapat ditambahkan *state-state* lainnya, sehigga karakter bos memiliki aksi yang lebih variatif.
2. Menambahkan kecerdasan buatan lainnya, agar karakter bos dapat lebih adaptif dengan pemain.
3. Menambah karakter-karakter bos yang memiliki gerakan-gerakan yang unik.

DAFTAR PUSTAKA

Ben-Ari, M., and Mondada, F., 2018. Finite state machines. Elements of Robotics, pp.55-61.

Delima, R., 2014. Analisis Implementasi Algoritma a*(A-star) Pada Game Rpg (Role Playing Game) 3d Sebagai Dasar Pergerakan Npc (Non-player Character) Mendekati Player Untuk Meningkatkan Realitas Game World. Jurnal Informatika, 9(2).

Hitchens, M., Patrickson, B. and Young, S., 2014. Reality and terror, the first-person shooter in current day settings. Games and Culture, 9(1), pp.3-29.

Holgersson, J., 2016. Camera impact on social impact games: Top down, third person and immersion.

Jagdale, D., 2021. Finite state machine in game development. International Journal of Advanced Research in Science, Communication and Technology, 10(1).

Kho, D., Pragantha, J. and Kristyadi, R., 2018. PEMBUATAN GAME TOP DOWN SHOOTER" SURROUNDED" UNTUK PLATFORM ANDROID. Jurnal Ilmu Komputer dan Sistem Informasi, 6(2), pp.36-36.

Liu, D., 2023. Research of the Path Finding Algorithm A* in Video Games. Highlights in Science, Engineering and Technology, 39, pp.763-768.

Nareyek, A., 2004. AI in Computer Games: Smarter games are making for a better user experience. What does the future hold?. Queue, 1(10), pp.58-65.

Pukeng, A.F., Fauzi, R.R., Andrea, R., Yulsilviana, E. and Mallala, S., 2019, October. An intelligent agent of finite state machine in educational game “Flora the Explorer”. In Journal of Physics: Conference Series (Vol. 1341, No. 4, p. 042006). IOP Publishing.

Safitra, W., Faisol, A. and Wibowo, S.A., 2020. Application of the Finite State Machine Method to Non Player Character (NPC) Action Strategy Game'Ouroboros'. JATI (Jurnal Mahasiswa Teknik Informatika), 4(2), pp.292 297.

Sanjaya, M.F., Pratiwi, H. and Adytia, P., 2021. Application of the Finite State Machine Method in the Desktop-Based “Heroes of Dawn” RPG Turn-Based Game. Tepian, 2(2), pp.69-73.

Simbolon, O.J., 2022. Simulasi Pencarian Rute Terpendek dengan Metode Algoritma A*(A-Star). Login: Jurnal Teknologi Komputer, 16(1), pp.23-32.

Siu, K., Butler, E., and Zook, A., 2016. A programming model for boss encounters in 2d action games. In Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (Vol. 12, No. 2, pp. 86-92).

Vivid Games (2018). Space Pioneer

Zikky, M., 2016. Review of A*(A star) navigation mesh pathfinding as the alternative of artificial intelligent for ghosts agent on the Pacman game. EMITTER International journal of engineering technology, 4(1), pp.141-149.

Zuhdi, A., Ahmad, I., and Putra, A.D., 2023. Implementation Of A* Algorithm In A Great Elephant Game With Unity 2D. SINTECH (Science and Information Technology) Journal, 6(2), pp.118-123.