

**PENERAPAN ALGORITMA PENCARIAN A* PADA
NAVIGATION MESH SEBAGAI METODE *PATHFINDING*
UNTUK OBJEK DINAMIS PADA *GAME* PENCARIAN
TARGET BERBASIS 3 DIMENSI MENGGUNAKAN UNITY**

SKRIPSI



MUTAWALLY SYARAWY

H13116524

PROGRAM STUDI ILMU KOMPUTER DEPARTEMEN MATEMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS HASANUDDIN

MAKASSAR

AGUSTUS 2020

**PENERAPAN ALGORITMA PENCARIAN A* PADA
NAVIGATION MESH SEBAGAI METODE
PATHFINDING UNTUK OBJEK DINAMIS PADA
GAME PENCARIAN TARGET BERBASIS 3 DIMENSI
MENGUNAKAN UNITY**

SKRIPSI

**Diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana Sains
pada Program Studi Ilmu Komputer Departemen Matematika Fakultas
Matematika dan Ilmu Pengetahuan Alam Universitas Hasanuddin**

MUTAWALLY SYARAWY

H13116524

**PROGRAM STUDI ILMU KOMPUTER DEPARTEMEN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS HASANUDDIN**

MAKASSAR

AGUSTUS 2020

LEMBAR PERNYATAAN KEOTENTIKAN

Saya yang bertanda tangan di bawah ini menyatakan dengan sungguh-sungguh bahwa skripsi yang saya buat dengan judul:

**PENERAPAN ALGORITMA PENCARIAN A* PADA NAVIGATION
MESH SEBAGAI METODE PATHFINDING UNTUK OBJEK DINAMIS
PADA GAME PENCARIAN TARGET BERBASIS 3 DIMENSI
MENGUNAKAN UNITY**

adalah benar hasil karya sendiri, bukan hasil plagiat dan belum pernah dipublikasikan dalam bentuk apapun.

Makassar, 18 Agustus 2020



NIM. H13116524

**PENERAPAN ALGORITMA PENCARIAN A* PADA
NAVIGATION MESH SEBAGAI METODE PATHFINDING
UNTUK OBJEK DINAMIS PADA GAME PENCARIAN
TARGET BERBASIS 3 DIMENSI MENGGUNAKAN UNITY**

Disetujui oleh:

Pembimbing Utama



Dr. Hendra, S.Si., M.Kom.
NIP. 19760102 200212 1 001

Pembimbing Pertama



Edy Saputra R, S.Si., M.Si.
NIP. 19910410 202005 3 001

Pada 18 Agustus 2020

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh:

Nama : Mutawally Syarawy
NIM : H13116524
Program Studi : Ilmu Komputer
Judul Skripsi : Penerapan Algoritma Pencarian A* pada *Navigation Mesh*
Sebagai Metode *Pathfinding* Untuk Objek Dinamis Pada
Game Pencarian Target Berbasis 3 Dimensi
Menggunakan Unity

Telah berhasil mempertahankan di hadapan dewan penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Sains pada Program Studi Ilmu Komputer Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Hasanuddin.

DEWAN PENGUJI

Tanda Tangan

1. Ketua : Dr. Hendra, S.Si., M.Kom. (.....)
2. Sekretaris : Edy Saputra, S.Si., M.Si. (.....)
3. Anggota : Dr.Eng. Armin Lawi, S.Si., M.Eng. (.....)
4. Anggota : Supri Bin Hj. Amir, S.Si., M.Eng. (.....)

Ditetapkan di : Makassar

Tanggal : 18 Agustus 2020



KATA PENGANTAR

Segala puji bagi Allah *Subhanahu Wa ta'ala*, Tuhan alam semesta yang telah memberikan nikmat kesempatan, waktu, kesehatan dan kemampuan sehingga penulisan skripsi ini bisa selesai tepat waktu. Shalawat serta salam senantiasa tercurah kepada *Rasulullah Muhammad Shallallahu Alaihi Wasallam*, yang merupakan teladan mulia dalam menjalankan kehidupan di dunia ini.

Alhamdulillah, skripsi dengan Judul “Penerapan Algoritma Pencarian A* Pada *Navigation Mesh* Sebagai Metode *Pathfinding* Untuk Objek Dinamis Pada *Game* Pencarian Target Berbasis 3 Dimensi Menggunakan Unity” yang disusun sebagai salah satu syarat akademik untuk meraih gelar Sarjana Sains pada Program Studi Ilmu Komputer Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Hasanuddin ini dapat diselesaikan. Walaupun adanya kendala-kendala yang dihadapi khususnya pada wabah COVID-19 saat ini. Tetapi dalam penulisan skripsi ini, penulis mampu menyelesaikan tepat waktu berkat bantuan dan dukungan dari berbagai pihak. Oleh karena itu, ucapan terima kasih dan apresiasi yang tak terhingga kepada kedua orang tua penulis, Ayahanda **Agus Susanto Jamaluddin** dan Ibunda **Insyana Arsyad** yang tak kenal lelah dalam memanjatkan doa serta memberikan nasihat, dorongan dan motivasi kepada penulis. Serta adinda **Azizah Almi** yang telah memberi ketenangan hati kepada penulis. Tugas akhir ini hanya setitik kebahagiaan kecil yang bisa penulis persembahkan.

Terima kasih juga penulis ucapkan kepada:

1. Rektor Universitas Hasanuddin, Ibu **Prof. Dr. Dwia Aries Tina Pulubuhu** beserta jajarannya.
2. Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam (FMIPA), **Dr.Eng. Amiruddin** beserta jajarannya.
3. Bapak **Dr. Nurdin, S.Si., M.Si.** sebagai Ketua Departemen Matematika FMIPA Unhas, bapak **Dr. Diaraya, M.Ak.** sebagai Ketua Program Studi Ilmu

Komputer Unhas, dosen-dosen pengajar, dan staf Departemen Matematika dan Fakultas MIPA atas ilmu dan bantuan yang selama ini telah diberikan.

4. Bapak **Dr. Hendra, S.Si., M.Kom.** sebagai dosen pembimbing utama sekaligus ketua tim penguji atas semua ilmu yang telah diberikan selama proses perkuliahan dan senantiasa memberi bimbingan dan memotivasi penulis dalam penulisan skripsi ini.
5. Bapak **Edy Saputra, S.Si., M.Si.** sebagai dosen pembimbing pertama sekaligus sekteraris tim penguji atas ilmu yang diberikan selama proses perkuliahan dan bimbingan, serta segala bentuk bantuan yang telah diberikan, khususnya dalam penyusunan skripsi ini.
6. Bapak **Dr.Eng. Armin Lawi, S.Si., M.Eng.** sebagai anggota tim penguji atas segala ilmu yang telah diberikan selama proses perkuliahan serta berbagai masukan dan kritik yang membangun dalam proses penyusunan skripsi ini.
7. Bapak **Supri Bin Hj. Amir, S.Si., M.Eng.** sebagai anggota tim penguji atas segala kritikan dan masukan yang membangun dalam penyusunan skripsi ini..
8. Bapak **Dr. Diaraya, M.Ak.** sebagai dosen pembimbing akademik yang senantiasa memberikan motivasi, dorongan, dan masukan dalam hal akademik selama menjadi mahasiswa Ilmu Komputer Unhas.
9. Teman-teman **ILMU KOMPUTER UNHAS 2016** atas kebersamaan, kepedulian, suka duka, canda tawa yang telah dilalui selama ini. Semoga persahabatan dan kebersamaan kita tidak hilang ditelan waktu.
10. Teman-teman **SSC Squad** yang senantiasa ada untuk memberikan masukan, bantuan, kritik dan solusi terhadap masalah yang dihadapi penulis.
11. Keluarga besar **HIMATIKA FMIPA UNHAS** terkhusus **ALGORITMA 2016** atas segala bentuk dukungan dan bantuan selama menjalani kehidupan kampus. Semoga kesuksesan selalu kita dapatkan dalam setiap langkah kita.
12. Keluarga besar **KM FMIPA UNHAS** terkhusus kepada **MIPA 2016** atas persahabatan, kekerabatan, kerjasama, serta cerita-cerita lain yang telah kita ukur bersama.

13. Teman-teman **KKN International Jepang** yang telah Bersama-sama menjalani pengabdian untuk belajar, berbagi budaya dan berbagi pengalaman di Fukuoka, Jepang. Kebersamaan, suka duka, canda tawa dari kalian tidak akan pernah penulis lupakan.
14. Semua pihak yang tidak dapat disebutkan satu persatu, atas segala bentuk kontribusi, partisipasi, serta motivasi yang diberikan kepada penulis selama ini. Semoga apa yang telah diberikan akan dilipatgandakan oleh Allah.

Penulis menyadari bahwa masih banyak kekurangan dalam tugas akhir ini, untuk itu, dengan segala ketulusan dan kerendahan hati penulis mohon maaf sebesar-besarnya. Semoga tulisan ini memberikan manfaat dan ilmu untuk siapapun yang membacanya.

PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Universitas Hasanuddin, saya yang bertanda tangan di bawah ini:

Nama : Mutawally Syarawy
NIM : H13116524
Programa Studi : Ilmu Komputer
Departemen : Matematika
Fakultas : Matematika dan Ilmu Pengetahuan Alam
Jenis Karya : Skripsi

Demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Hasanuddin **Hak Prediktor Royalti Noneksklusif (*Non-exclusive Royalty-Free Right*)** atas tugas akhir saya yang berjudul:

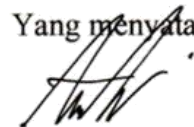
“Penerapan Algoritma Pencarian A* Pada Navigation Mesh Sebagai Metode Pathfinding Untuk Objek Dinamis Pada Game Pencarian Target Berbasis 3 Dimensi Menggunakan Unity”

beserta perangkat yang ada (jika diperlukan). Terkait dengan hal diatas, maka pihak universitas berhak menyimpan, mengalih-media/format-kan, mengelola dalam bentuk pangkalan data (database), merawat, dan memublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian surat pernyataan ini saya buat dengan sebenarnya.

Dibuat di Makassar pada 18 Agustus 2020

Yang menyatakan



(Mutawally Syarawy)

ABSTRAK

Pathfinding merupakan metode yang digunakan untuk mengetahui jalur dari suatu titik awal menuju titik akhir. Banyak aspek yang dapat mempengaruhi kinerja *pathfinding*. Salah satunya yaitu algoritma yang digunakan pada *pathfinding* itu sendiri. Terdapat banyak algoritma yang dapat digunakan pada *pathfinding* untuk mendapatkan hasil yang diinginkan. Tetapi tidak semua algoritma yang digunakan akan menghasilkan hasil yang optimal untuk tujuan yang sama. Salah satu penerapan *pathfinding* yaitu pada navigasi dalam *game*. Navigasi didalam video *game* merupakan salah satu metode dimana dibutuhkan waktu komputasi yang cepat dan penyimpanan data yang besar untuk menghasilkan jalur navigasi yang cepat dan tepat. Maka dari itu, penelitian ini menerapkan sebuah algoritma pencarian A* pada metode navigasi bernama *Navigation Mesh* dalam sebuah game pencarian target. Dua objek karakter 3 dimensi akan dibuat. Hasil karakter tersebut akan diterapkan pada *navigation mesh*. Satu karakter akan menjadi pengejar yang akan mengejar karakter lainnya yang dikendalikan oleh pengguna. Karakter pengejar bergerak berdasarkan jalur navigasi yang akan dibuat menggunakan metode pencarian A*. Setelah selesai maka akan menghasilkan sebuah program *pathfinding* yang dapat digunakan pada game dan dapat dikalkulasi waktu rata-rata komputasi pada program tersebut.

Kata Kunci: *Pathfinding*, *Navigation Mesh*, algoritma A*, *Game*.

ABSTRACT

Pathfinding is a method used to find a path from the start point to its finish point. Many aspects can affect the performance of pathfinding. One of them is the algorithm used in the pathfinding itself. Many algorithms that can be used in pathfinding to get the desired result. But not all algorithms used will produce an optimal result for the same purpose. One application of pathfinding is in-game navigation. Navigation in a video game is one method where fast computing and big data storage is needed to produce fast and precise navigation paths. Therefore, this study applies an A* search algorithm to the navigation method called Navigation Mesh in a target search game. Two 3-dimensional character objects will be created. The character results will be applied to the navigation mesh. One character will be a chaser who will chase another character that is controlled by the user. The chaser will move based on the navigation path that will be created using the A* search method. When finished, it will produce a pathfinding program that can be used in the game and can calculate the average computation time in the program.

Keywords: Pathfinding, Navigation Mesh, A* Algorithm, Game.

DAFTAR ISI

HALAMAN JUDUL.....	ii
LEMBAR PERNYATAAN KEOTENTIKAN.....	iii
PERSETUJUAN PEMBIMBING.....	iv
HALAMAN PENGESAHAN.....	v
KATA PENGANTAR	vi
PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS	ix
ABSTRAK	x
ABSTRACT	xi
DAFTAR ISI.....	xii
DAFTAR GAMBAR	xiv
DAFTAR TABEL.....	xvi
DAFTAR LAMPIRAN	xvii
BAB I 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
BAB II TINJAUAN PUSTAKA.....	5
2.1 <i>Game</i>	5
2.2 Objek 3D	9
2.3 Animasi	10
2.4 <i>Pathfinding</i>	13
2.5 <i>Navigation Mesh (Navmesh)</i>	16
2.6 Unity	17
BAB III METODE PENELITIAN.....	18

3.1	Tahapan Penelitian	18
3.1.1	Pembuatan objek dan animasi <i>game</i>	18
3.1.2	Pembuatan <i>scene</i> dan level <i>game</i>	18
3.1.3	Pembuatan dan penerapan program <i>Navmesh</i> pada <i>game</i>	19
3.1.4	<i>Preprocessing navmesh</i>	19
3.1.5	Uji coba <i>game</i>	20
3.2	Waktu dan Lokasi Penelitian.....	20
3.3	Instrumen Penelitian.....	20
3.3.1	Perangkat Lunak.....	20
3.3.2	Perangkat Keras	20
BAB IV HASIL DAN PEMBAHASAN		21
4.1	Pembuatan Objek dan Animasi	21
4.1.1	Pemodelan dan Animasi Karakter.....	21
4.1.2	Pemodelan Arena dan Rintangan	23
4.2	Pembuatan <i>Scene</i> dan Arena	24
4.3	Penerapan dan <i>Bake Navmesh</i>	26
4.4	Implementasi <i>Navmesh Agent</i> Pada Karakter NPC.....	28
4.5	Uji Coba	29
4.6	Analisis Waktu Komputasi.....	33
BAB V KESIMPULAN.....		36
5.1	Kesimpulan.....	36
5.2	Saran	36
DAFTAR PUSTAKA		37
LAMPIRAN.....		39

DAFTAR GAMBAR

Gambar 2.1. Video <i>game</i> Overwatch.....	5
Gambar 2.2. Perbandingan antara game 2d dan 3d.....	5
Gambar 2.3. Mesin Arcade <i>game</i>	6
Gambar 2.4. Jenis-jenis <i>game</i> pada PC.....	6
Gambar 2.5. Jenis-jenis konsol pada console <i>game</i>	7
Gambar 2.6. Nintendo Switch.....	7
Gambar 2.7. <i>Game-game</i> pada platform mobile.....	8
Gambar 2.8. Contoh Objek 3D	9
Gambar 2.9. Pemodelan kepala.....	10
Gambar 2.10. Penerapan <i>Armature</i> sebagai dasar animasi.....	11
Gambar 2.11. Pathfinding	13
Gambar 2.12. Penerapan algoritma A*	14
Gambar 2.13. Flowchart Algoritma A*	15
Gambar 2.14. Penerapan Navmesh pada area game	16
Gambar 2.15. Pengembangan game pada Unity	17
Gambar 3.1. Diagram alur penelitian.....	18
Gambar 4.1. Pemberian <i>texture</i> warna menggunakan UV <i>map</i> pada karakter <i>player</i>	21
Gambar 4.2. Pembuatan animasi bergerak pada karakter <i>player</i>	22
Gambar 4.3. Pemberian <i>texture</i> warna menggunakan UV <i>map</i> pada karakter NPC	22
Gambar 4.4. Pembuatan animasi bergerak pada karakter <i>player</i>	23
Gambar 4.5. Model arena dan rintangan.....	23
Gambar 4.6. <i>prefab</i> objek pada Unity.....	24
Gambar 4.7. Pengaturan arena pada <i>Complex Obstacle Scene</i>	24
Gambar 4.8. Inspektor pada karakter <i>player</i>	25
Gambar 4.9. <i>Finite-state machine</i> karakter <i>player</i>	26

Gambar 4.10. <i>Finite-state machine</i> karakter NPC	26
Gambar 4.11. <i>Window navigation</i> pada Unity	27
Gambar 4.12. Hasil <i>bake navmesh</i> pada arena <i>complex scene</i>	28
Gambar 4.13. Atribut <i>Nav Mesh Agent</i> pada karakter NPC	29
Gambar 4.14. Pengambilan rute NPC ketika program dijalankan	32
Gambar 4.15. Penempatan objek kubus pada koordinat <i>node</i> jalur NPC	32
Gambar 4.16. Menu <i>Console</i> pada program <i>complex scene</i>	33
Gambar 4.17. Skenario penempatan karakter	34
Gambar 4.18. Waktu komputasi <i>Observer Scene</i>	34
Gambar 4.19. Waktu komputasi <i>Simple Scene</i>	34
Gambar 4.20. Waktu komputasi <i>Complex Scene</i>	35
Gambar B.1. Model karakter <i>Player</i>	44
Gambar B.2. Model karakter NPC	44
Gambar B.3. Model arena dan rintangan	44
Gambar C.1. <i>Observer scene</i> beserta <i>navmeshnya</i>	45
Gambar C.2. <i>Simple scene</i> beserta <i>navmeshnya</i>	45
Gambar C.3. <i>Complex scene</i> beserta <i>navmeshnya</i>	45

DAFTAR TABEL

Tabel 4.1. Cara kerja <i>navmesh</i>	30
Tabel 4.2. Rata-rata waktu komputasi program <i>navmesh</i>	35

DAFTAR LAMPIRAN

Lampiran A. Source Code.....	39
Lampiran B. Model 3 Dimensi	44
Lampiran C. Scene Game	45

BAB I

PENDAHULUAN

1.1 Latar Belakang

Game merupakan salah satu media hiburan yang dapat dimainkan oleh setiap kalangan untuk menghilangkan rasa jenuh. *Game* juga merupakan media yang dapat digunakan untuk keperluan – keperluan tertentu dan untuk tujuan tertentu. *Game* juga memiliki beberapa genre, seperti *action*, *racing*, *sport*, *boardgame*, *adventure*, *simulation*, dan *strategy*. (Apperley, 2006).

Game sendiri telah berkembang pesat dari awal pembuatannya, dimulai dari tahun 1972. Ketika *game* pertama Pong dibuat, yang hanya berupa *game* 2D sejenis ping pong, hingga sekarang yang menghasilkan *game* yang dapat menyerupai dunia nyata bahkan dapat dibawa ke dunia nyata berupa *Virtual Reality* (VR).

Di Indonesia sendiri telah banyak studio *game* developer yang berkembang, pada Bekraf *Game* Prime 2018 setidaknya ada 50 developer dari 100 yang terpilih berpartisipasi pada event tersebut. Untuk potensi developer *game* di Indonesia itu sendiri juga tidak sedikit, berdasarkan hasil Lembaga survey *game* global, Newzoo, setidaknya ada 43,7 juta orang Indonesia adalah seorang pemain *game*. Pada tahun 2017, besarnya potensi *game* Indonesia bernilai USD 800.000.000 atau setara dengan Rp.11,9 triliun. Indonesia juga berhasil masuk sebagai negara dengan pasar *game* terbesar di dunia menempati posisi ke-16 (Sanny, 2018).

Pengembangan suatu *game* dapat dipengaruhi oleh banyak faktor, contohnya waktu, sumber daya, tenaga kerja, lingkungan kerja, metode yang digunakan, dan lainnya. Faktor-faktor pendukung lainnya bagi developer di Indonesia masih bisa dikatakan dibawah dari developer di luar Indonesia, dikarenakan kurangnya kesadaran pemerintah akan dampak pasar *game* Indonesia yang bila dimanfaatkan dapat menaikkan ekonomi Indonesia itu sendiri, dan menghasilkan developer-developer yang lokal dapat bersaing secara internasional. Pengembang *game* Indonesia dapat menghasilkan *game* yang memiliki daya tarik yang baik agar mengundang pasar internasional untuk berinvestasi pada perusahaan developer *game* di Indonesia.

Daya tarik dari suatu *game* bisa dilihat dari genre, cerita, instrumen, *arts* serta gameplay dari *game* itu sendiri. Suatu *game* harus memberikan hasil yang memuaskan agar dapat dinikmati oleh pemainnya, jika terdapat suatu aspek yang mempengaruhi suatu *game* yang tidak sesuai akan menurunkan daya tarik *game* tersebut.

Salah satu aspek yang dipakai dalam sebuah *game* adalah *Pathfinding* (pencari Jalur), *Pathfinding* digunakan untuk mencari jalur dari suatu titik ke titik yang lain, penggunaan *pathfinding* itu sendiri sudah banyak diterapkan pada *game-game* bergenre *role-playing Game*. Biasa digunakan oleh *Non Player Character* (NPC), pencari jalur pada peta, ataupun karakter pendamping *player* agar selalu mengikuti *player* tersebut. *Pathfinding* sendiri merupakan elemen penting dalam suatu *game* baik 2 dimensi maupun 3 dimensi, semakin rumit area atau *map* pada *game* maka semakin rumit pola metode *pathfinding* yang dihasilkan (Pramono, 2015).

Penentuan teknik *pathfinding* yang menghasilkan hasil yang baik sangatlah penting dalam keberhasilan suatu kecerdasan buatan pada komputer *game*, dikarenakan *pathfinding* adalah bagian dasar dalam bidang kecerdasan buatan pada *game*. Banyak teknik yang digunakan pada *pathfinding*, dan bisa saja satu teknik lebih baik dalam suatu kondisi tertentu, tetapi buruk pada kondisi yang lainnya. Tingkat keberhasilan yang didapatkan tergantung dengan persyaratan dan asumsi dari *game* itu sendiri. semakin banyak faktor yang diberikan pada *game*, semakin kompleks proses *pathfinding* yang dijalankan (Reese & Stout, 1999).

Terdapat banyak metode yang dapat digunakan pada *pathfinding*, salah satunya adalah algoritma A*. Algoritma A* sendiri banyak digunakan untuk mengambil jarak terdekat dari suatu node awal menuju node akhir dengan melihat nilai dari masing-masing node tersebut. Dengan metode ini, kita dapat gunakan pada *pathfinding* untuk mencari jalur dari titik awal ke titik akhir, mencari jalur terpendek dan mencari jalur secara cepat pada *game* 3D.

Penerapan algoritma A* sendiri telah umum digunakan pada *pathfinder game* 2D dikarenakan koordinat yang digunakan hanya berupa x dan y, penerapan *node* atau *mesh* dinilai mudah diterapkan dalam bidang 2D dan tidak memerlukan proses pencarian yang lama, dan tidak sedikit *game* telah menentukan sendiri *node-*

node tetap yang akan digunakan sehingga memudahkan proses pengolahan tanpa khawatir terjadi perubahan-perubahan *node-node* secara konstan. Tetapi dalam game 3D terdapat atribut-atribut yang tidak ada pada game 2D dan membuat proses pengolahan *pathfinding* lebih rumit dari *pathfinding game* 2D. Pada game 3D terdapat variabel z (atau y pada Unity) yang harus dipertimbangkan ketika membuat jalur yang akan digunakan. Penentuan jalur itu sendiri akan lebih rumit jika area yang digunakan berskala besar dan terdapat banyak rintangan.

menerapkan metode algoritma A* pada *pathfinding game*. Penelitian ini berjudul **“Penerapan Algoritma Pencarian A* pada Navigation Mesh Sebagai Metode Pathfinding untuk Objek Dinamis pada Game Pencarian Target Berbasis 3 Dimensi Menggunakan Unity”**.

1.2 Rumusan Masalah

Adapun rumusan masalah dalam penelitian ini yaitu:

1. Bagaimana penerapan algoritma pencarian A* pada *navigation mesh* yang akan digunakan karakter 3D dalam Unity?
2. Bagaimana waktu komputasi pencarian rute pada karakter NPC untuk sampai ke tujuan menggunakan algoritma A*?

1.3 Batasan Masalah

Berikut ini merupakan beberapa batasan dalam penelitian ini.

1. *Game* yang akan dibuat berupa *prototype game* 3D sederhana.
2. Jumlah karakter pada *game* ini hanya 2, karakter yang dikontrol oleh *user*, dan karakter yang dikontrol oleh program.
3. Jumlah *scene game* hanya 3.

1.4 Tujuan Penelitian

Tujuan dari penelitian ini yaitu:

1. Mengetahui penerapan metode algoritma pencarian A* pada *Navigation mesh* yang akan digunakan oleh karakter di Unity.
2. Menganalisis waktu komputasi dan ketepatan pengambilan rute tercepat pada karakter NPC untuk sampai ke tujuan.

1.5 Manfaat Penelitian

Penelitian ini dapat menghasilkan metode *pathfinding* yang bisa di implementasikan kemudian pada navigasi karakter. Selain itu, penelitian ini dapat memberikan pemahaman tentang cara menggunakan algoritma pencarian A* pada *game* di Unity.

BAB II

TINJAUAN PUSTAKA

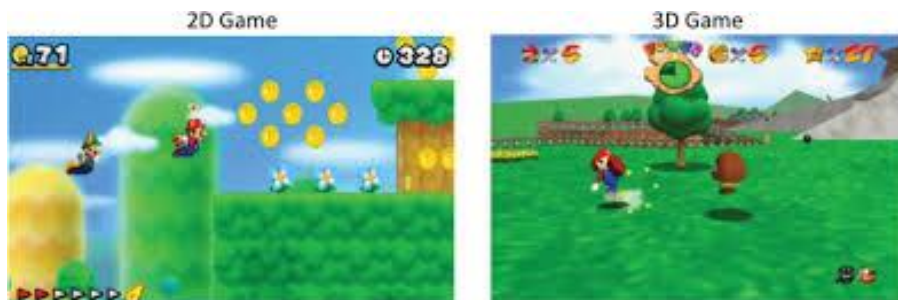
2.1 *Game*

Game jika diartikan dalam Bahasa Indonesia berarti permainan, *game* adalah permainan yang diprogram dalam suatu perangkat yang dapat dimainkan baik *offline* maupun *online*. Dalam sebuah *game* biasanya memiliki suatu aturan untuk menuntun arah permainan itu sendiri. Contoh dari *game* adalah *game* Overwatch dari Activiton Blizzard Entertainment pada gambar 2.1.



Gambar 2.1. Video *game* Overwatch

Dalam sebuah video *game* biasa menggunakan model 2D maupun 3D yang memiliki antarmuka permainan, *game play*, *arts style* maupun konsep yang berbeda seperti yang tertera pada gambar 2.2.



Gambar 2.2. Perbandingan antara game 2d dan 3d

Game sendiri dikategorikan menjadi 2 bagian, yaitu berdasarkan Platform (Media yang digunakan) dan Genre (Tema permainan).

Berdasarkan platform, *game* dibagi menjadi 5 bagian, yaitu :

- ***Arcade game***

Arcade games yaitu sebuah permainan yang menggunakan media mesin khusus pada tempat-tempat permainan anak, contohnya Timezone. Mesin yang digunakan didesain khusus untuk 1 permainan tertentu dengan kendali karakter, fitur dan tampilan yang diperuntukkan khusus pada *game* tersebut. Bentuk mesin *arcade* tertera pada gambar 2.3.



Gambar 2.3. Mesin Arcade *game*

- ***PC game***

PC games yaitu permainan yang dimainkan menggunakan PC (*Personal Computer*). Jenis platform ini yang sedang menguasai pasar *game* internasional dikarenakan dapat memainkan banyak jenis *game* dengan sistem kendali *game* yang sama, biasa berupa *Mouse* dan *Keyboard*. Telah banyak *game* PC yang dikembangkan seperti yang tertera pada gambar 2.4.



Gambar 2.4. Jenis-jenis *game* pada PC

- ***Console game***

Console games yaitu permainan yang dimainkan menggunakan suatu konsol tertentu, seperti Playstation, XBOX ataupun Nintendo.

Perbedaan *game* pada console dan arcade adalah pada Console dapat memainkan *game* yang berbeda menggunakan konsol yang sama. Arcade hanya dapat memainkan *game* yang telah diprogram khusus pada mesin tersebut. Contoh konsol-konsol yang digunakan tertera pada gambar 2.5.



Gambar 2.5. Jenis-jenis konsol pada console *game*

- ***Handheld game***

Handheld games yaitu permainan yang dimainkan pada konsol khusus yang dapat dibawa kemana-mana seperti Sony PSP, atau seperti yang tertera pada gambar 2.6, yaitu Nintendo Switch.



Gambar 2.6. Nintendo Switch.

- ***Mobile game***

Mobile games yaitu permainan yang dapat dimainkan pada perangkat *mobile* seluler atau *Smartphone*. Sekarang *Mobile game* telah banyak dipasarkan dikarenakan pemain dapat memainkannya dimana saja. Contoh *mobile game* tertera pada gambar 2.7.



Gambar 2.7. *Game-game* pada platform mobile

Berdasarkan Genre, *game* dibagi menjadi 10 bagian, yaitu :

- ***Action***

Genre *action* banyak digunakan pada video *game*, genre ini memerlukan kecepatan refleks, koordinasi mata dan tangan, dan juga . Jenis permainan yang masuk dalam genre *action* bisa berupa tembak-tembakan atau perkelahian.

- ***Race***

Genre *race* digunakan pada *game* balap, bisa berupa balap mobil, motor dan lainnya. Salah satu *game* yang menggunakan genre ini yaitu F1, MotoGP, MarioKart dan lainnya.

- ***Role play***

Genre *role play* bisa disebut RPG (*Role Playing Game*) adalah *game* dimana pemainnya menjalankan peran dari suatu karakter. Genre ini banyak digunakan pada permainan yang bertema fantasi atau kerajaan. Salah satu *game* yang menggunakan genre ini yaitu World of Warcraft.

- ***Sport***

Genre *Sport* digunakan pada *game* bertema olahraga, atau mensimulasikan sebuah olahraga pada dunia nyata dalam sebuah *game*.

- ***Board***

Genre *Board* digunakan pada *game* bertema papan seperti catur, Monopoly dan lainnya.

- ***Adventure***

Genre *Adventure* adalah permainan bertema petualangan, dimana pemain dituntut untuk memainkan berdasarkan alur dan narasi yang telah ditetapkan.

- ***Action – Adventure***

Genre *Action - Adventure* tidak jauh beda dengan *Adventure* biasa. bedanya pada *Action – Adventure* terdapat aksi ketika sedang memainkan permainan.

- ***Simulation***

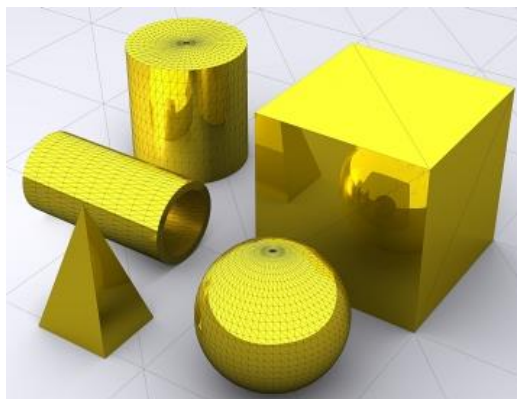
Genre *Simulation* digunakan pada *game* simulasi, bisa berupa simulasi pengaturan kota, mengoperasikan kendaraan dan lainnya.

- ***Strategy***

Genre *Strategy* digunakan pada *game* bertema strategi. Bisa berupa Tower Defense, catur, *Turn-Based game* dan lainnya. *Game* genre ini memerlukan kalkulasi dalam memainkannya.

2.2 Objek 3D

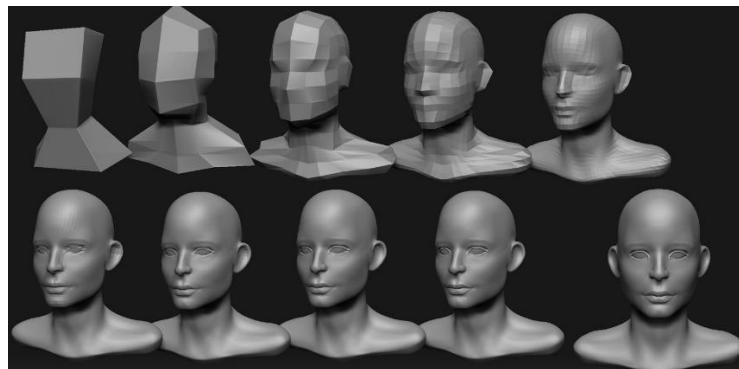
Objek 3D atau 3 dimensi adalah objek yang memiliki lebar, tinggi, dan kedalaman (*width,height,depth*). Dalam grafik, objek 3D adalah objek yang dapat ditampilkan dalam bentuk 3 dimensi pada koordinat x, y, dan z pada grafik. Objek 3D dibuat dari rangkaian *polygon* dimana *polygon* adalah segmen garis yang dihubungkan antara sudut garis dengan lainnya membentuk suatu bidang tertutup.



Gambar 2.8. Contoh Objek 3D

Adapun pada gambar 2.8 merupakan contoh-contoh objek 3D yang banyak ditemukan. Bisa berupa kubus, balok, bola, tabung, prisma, dan lainnya. Terdapat juga metode untuk membuat objek 3D yaitu dengan pemodelan objek 3D. Pemodelan objek 3D atau biasa disebut *3D modeling* adalah proses pembuatan representasi matematis dari segala permukaan suatu objek pada objek 3D menggunakan aplikasi khusus. Objek 3D yang digunakan disebut dengan model. Model dapat dibuat secara otomatis maupun manual. Proses pengerjaan suatu model bisa dibilang sama dengan memahat patung atau membentuk tanah liat. 3D model bisa digunakan untuk membuat animasi atau dapat dicetak menggunakan *3D printer* dengan bentuk dan ukuran sesuai dengan model yang dibuat (Van Gumster, 2020).

Model 3D yang digunakan pada *film* animasi biasa dibuat dengan metode *3D modeling*. Hasil yang dibuat bisa berupa karakter, binatang, kendaraan, dan lainnya. Pada gambar 2.9, diberikan contoh pemodelan objek 3D untuk membentuk kepala dari karakter.



Gambar 2.9. Pemodelan kepala

2.3 Animasi

Animasi Adalah gambar bergerak berbentuk dari kumpulan objek yang disusun secara berurutan mengikuti alur pergerakan yang telah ditentukan pada setiap *frame* waktu yang terjadi. Animasi berasal dari Bahasa Inggris yaitu *Animate* yang berarti menghidupkan. Animasi bisa dikatakan proses membuat objek yang asalnya mati, kemudian disusun dalam posisi yang berbeda seolah menjadi hidup. Pada animasi terdapat 2 objek penting, yaitu objek atau gambar dan alur gerak.

Objek disini merupakan suatu *entity* yang akan ditampilkan dan digerakkan sesuai dengan alur gerak yang diberikan, dapat berupa gambar 2D atau objek 3D yang ditampilkan dalam bentuk video. Alur gerak sendiri merupakan rangkaian gerak yang akan diterapkan pada suatu objek agar objek tersebut mengikuti alur gerak yang diberikan. Urutan alur gerak itu sendiri dapat ditentukan. Salah satu metode yang digunakan yaitu penerapan *Armature* (tulang) pada suatu objek 3D yang nantinya dapat digerakkan untuk membuat animasi seperti yang tertera pada gambar 2.10.



Gambar 2.10. Penerapan *Armature* sebagai dasar animasi

Dalam animasi ada 12 prinsip (Lasseter, 1987) yang harus diikuti untuk menghasilkan animasi yang baik, yaitu :

- ***Squash & stretch***

Semakin lumat dan renggang objek yang ditampilkan, objek tersebut akan dinilai sebagai objek lunak, begitupun sebaliknya.

- ***Anticipation***

Memberikan objek suatu pergerakan antisipasi sebelum melakukan pergerakan inti.

- ***Staging***

Merepresentasikan suatu ide dari sebuah animasi agar dapat diterima secara lengkap dan jelas.

- ***Straight ahead & pose to pose***

2 proses dalam menggambar animasi, yaitu *straight ahead* yang membuat sebuah gambar secara berurutan dari awal hingga akhir, dan *pose to pose* yang menggambar dari awal, pertengahan, akhir lalu menggambar diantaranya.

- ***Follow through & Overlapping action***

Membuat objek lain yang ada pada objek inti dari animasi mengikuti pergerakan objek inti secara nyata.

- ***Slow in & slow out***

Membuat pergerakan pada animasi secara nyata dengan membuat awal pergerakan melambat dan akhir pergerakan melambat juga.

- ***Arcs***

Membuat pergerakan suatu objek mengikuti busur pergerakan (*arcs*) secara nyata.

- ***Secondary action***

Membuat pergerakan tambahan setelah pergerakan inti untuk mendukung pergerakan inti.

- ***Timing***

Tampilan dan sifat animasi berpengaruh besar terhadap jumlah *frame* yang diberikan antara pergerakan inti. Semakin banyak gambar yang diberikan maka tampilan akan terlihat lambat, begitupun sebaliknya.

- ***Exaggeration***

Melebih-lebihkan suatu pergerakan inti agar terlihat nyata dan tidak kaku.

- ***Solid drawing***

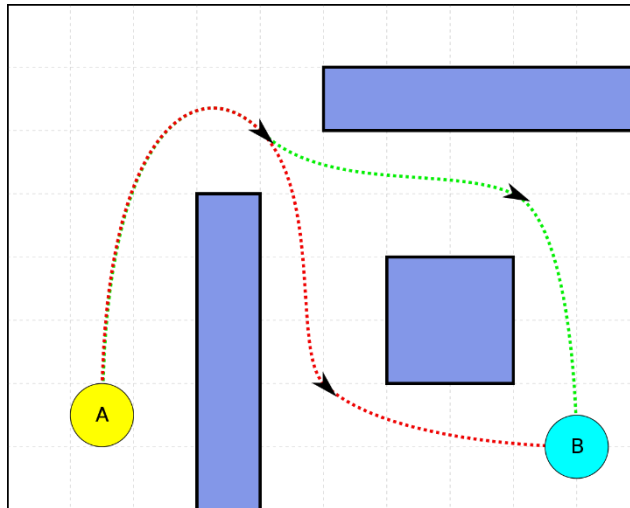
Membuat animasi dengan menampilkan objek yang dibuat mengikuti ruang, berat, dan keseimbangan yang sesuai.

- ***Appeal***

Membuat objek animasi memiliki karakteristik dan karisma dengan menampilkan atribut-atribut yang mendukung hal tersebut.

2.4 Pathfinding

Pathfinding adalah suatu metode yang digunakan untuk mencari jalur dari suatu titik awal menuju titik akhir seperti yang tertera pada gambar 2.11. *Pathfinding* telah banyak digunakan dalam kehidupan sehari-hari, contohnya menentukan jalur transportasi, navigasi labirin maupun penentuan jalur robot industri (Shi & Hao, 2011).

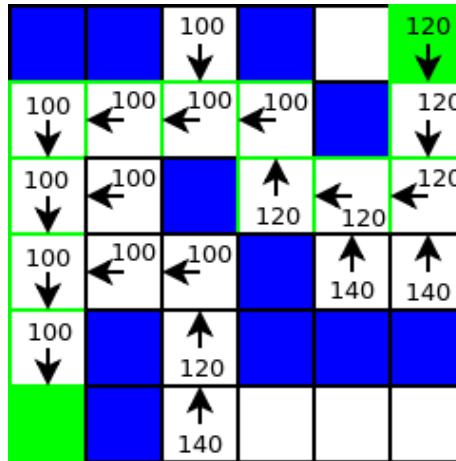


Gambar 2.11. Pathfinding

Untuk setiap masalah pada kecerdasan buatan (AI) pada *game*, *pathfinding* mungkin yang paling umum untuk mencari jalur untuk pergerakan suatu *entity* dari titik awal ke titik yang lain. *Entity* yang digunakan bisa berupa orang, kendaraan atau lawan pada *game*. Genre yang biasa menggunakan *pathfinding* adalah *action*, *simulation*, *role-playing game*, atau strategi. Tetapi setiap *game* yang dimana komputer yang bertanggung jawab untuk menggerakkan suatu *entity* harus menyelesaikan masalah *pathfinding* (Stout, 1996).

Masalah yang biasa dihadapi pada *pathfinding* adalah menghindari rintangan, bergerak secara acak, melacak lajur antar rintangan, dan lain-lain. Pengambilan keputusan pada *pathfinding* bisa terbilang rumit dikarenakan proses harus dijalankan secara *real-time* dan perubahan lokasi tujuan atau pergerakan *entity* secara konstan membuat perhitungan algoritma harus tepat dan cepat. Semua masalah *game* AI membutuhkan *processor* komputer yang baik dan *memory* komputer yang besar agar proses kerja AI berjalan dengan baik dan cepat. Salah satu metode dalam *pathfinding* yaitu menggunakan algoritma A*. Algoritma A* adalah algoritma dasar yang dapat digunakan untuk menemukan solusi berbagai masalah,

pathfinding salah satunya. A* memproses area yang belum dijelejah yang dinilai bagus. Algoritma akan berhenti jika lokasi tersebut adalah lokasi tujuan. Jika tidak, maka algoritma akan menghitung lokasi-lokasi sekitar lokasi tersebut untuk menjelajah lebih lanjut. Algoritma A* adalah salah satu algoritma yang paling banyak digunakan sebagai *pathfiding* pada *game* AI (Shi & Hao, 2011).

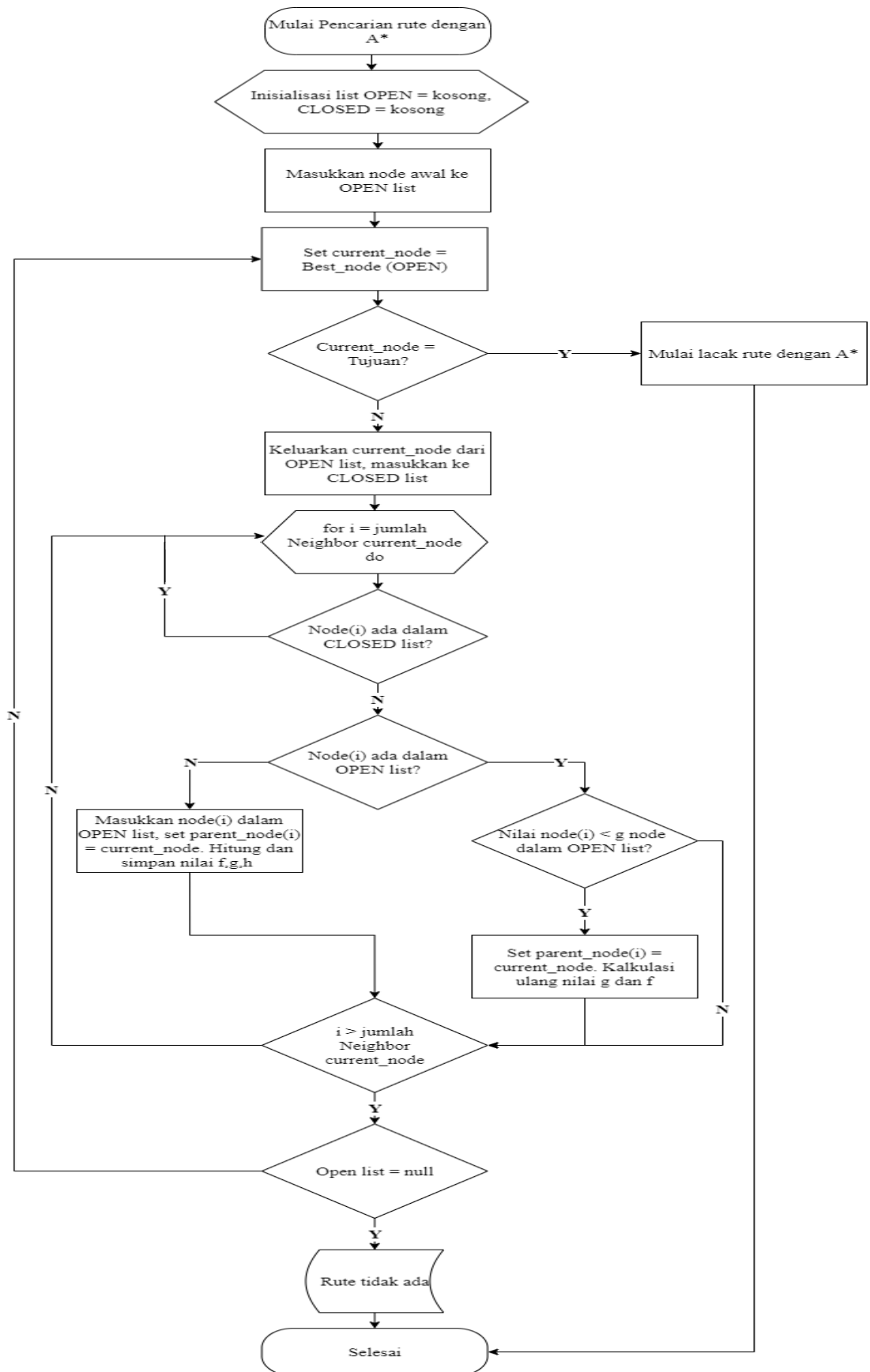


Gambar 2.12. Penerapan algoritma A*

Pada gambar 2.12, adalah contoh penerapan algoritma A*. Algoritma A* memiliki 3 variabel, $g(n)$ sebagai *cost* dari *starting point* ke sebarang *point n*. $h(n)$ sebagai estimasi *cost* dari *point n* ke *point* tujuan, dan $f(n)$ adalah *cost* total dari *current point*. Jadi penghitungan proses algoritma A* adalah :

$$f(n) = g(n) + h(n)$$

Sebelum Algoritma A* dijalankan, area yang digunakan harus telah disediakan atau telah diproses sebelumnya. Pada proses ini, area akan dibagi menjadi beberapa bagian atau lokasi yang dinamakan *nodes*. Proses pengerjaan algoritma tertera seperti pada gambar 2.13.

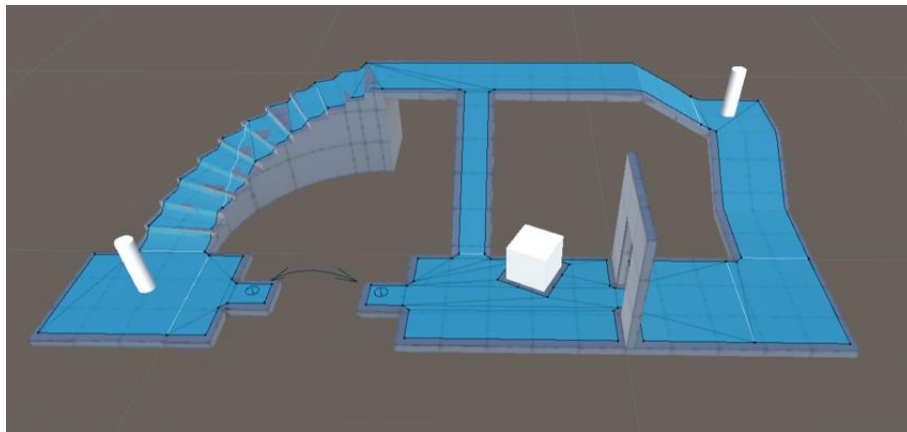


Gambar 2.13. Flowchart Algoritma A*

Algoritma A* dinilai lebih baik dalam penerapan *pathfinding* pada *game* dibanding dengan algoritma *Dijkstra*. A* adalah pengembangan dari algoritma *Dijkstra* untuk mendapatkan hasil algoritma yang lebih cepat, dimana A* hanya mencari jalur berdasarkan nilai terbaik yang ada sedangkan *Dijkstra* akan terus memproses jalur yang ada secara keseluruhan tanpa melihat jalur terpendek antara *point* awal ke *point* tujuan sehingga membuang-buang waktu dalam proses pengerjaannya.

2.5 Navigation Mesh (Navmesh)

Navigation Mesh (Navmesh) adalah sebuah struktur data yang digunakan oleh *game* AI untuk membantu dalam *pathfinding* dengan menanda area *mesh* yang dapat dilalui (*walkable*) oleh *agent* (*entity pathfinding*) untuk mencari jalur berdasarkan area *mesh* tersebut. Pembuatan *mesh* itu sendiri dihitung berdasarkan luas dan ukuran arena sebenarnya, ukuran *agent*, rintangan yang ada, dan *jump-link* yang ada. *Jump-link* merupakan jalur yang dapat dilewati *agent* antara 1 *mesh* dengan *mesh* yang lain tanpa harus dihubungkan oleh *mesh* lain. *Navmesh* sendiri dapat dibuat pada aplikasi Unity dan telah disediakan komponen khusus untuk pembuatan *Navmesh* itu sendiri, tetapi *physics* dan *logic* dari *Navmesh* itu sendiri dibuat oleh *user*. Pada gambar 2.14 adalah contoh penerapan *Navmesh* pada Unity.



Gambar 2.14. Penerapan Navmesh pada area game

2.6 Unity

Unity adalah aplikasi yang digunakan untuk membuat game multi platform yang didesain untuk mudah digunakan. Unity telah banyak digunakan sebagai aplikasi bagi pengembang game dikarenakan menu yang mudah, dan dibekali dengan *Assets store* dimana pengembang dapat mengunduh langsung aset-aset game dan dapat langsung diterapkan pada game yang sedang dikembangkan di Unity. Unity juga dapat digunakan untuk membuat film, pembuatan aplikasi *Virtual Reality* dan *Augmented Reality*, dan lain-lain. Contoh tampilan Unity tertera pada Gambar 2.15.



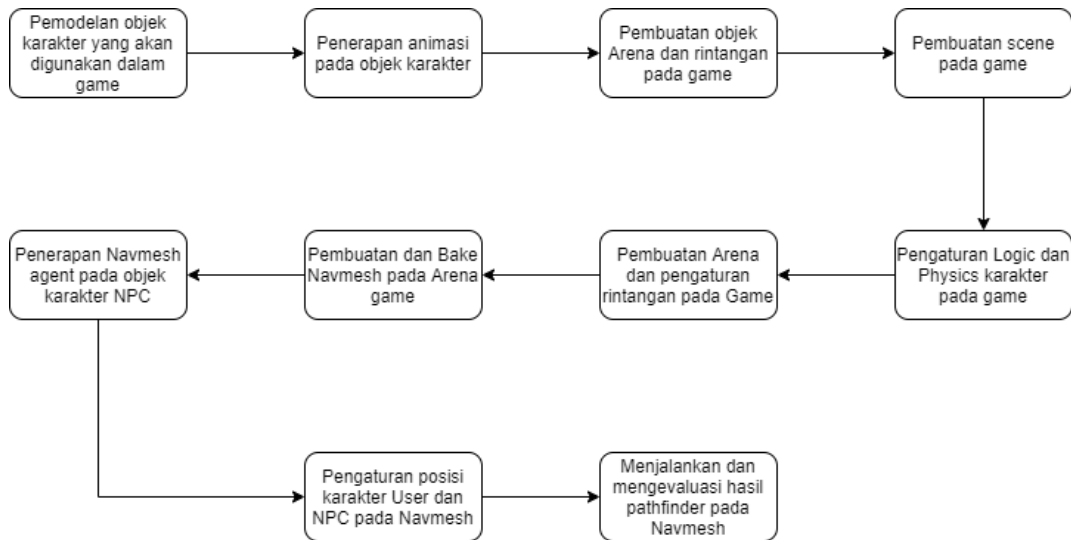
Gambar 2.15. Pengembangan game pada Unity

BAB III

METODE PENELITIAN

3.1 Tahapan Penelitian

Tahapan pada penelitian ini tertera pada gambar 3.1.



Gambar 3.1. Diagram alur penelitian

3.1.1 Pembuatan objek dan animasi *game*

Langkah awal pada penelitian ini yaitu membuat pembuatan objek yang akan digunakan pada *game*. Objek yang akan dibuat berupa karakter orang beserta animasinya, arena dan juga rintangan yang akan digunakan pada *game*. Jumlah yang akan dibuat berupa 2 objek karakter, 1 objek arena dan 5 objek rintangan. Untuk pemakaian pada *game*, objek dapat disalin jika dibutuhkan lebih banyak pada *game*. Pada pembuatan objek akan dibuat pada aplikasi Blender.

3.1.2 Pembuatan *scene* dan level *game*

Setelah pembuatan objek, maka selanjutnya yaitu pembuatan *scene* dan level pada *game*. Aplikasi yang digunakan yaitu Unity. Jumlah *scene* yang akan dibuat berupa 3 *scene*:

- *Scene observer*, berupa arena kosong tanpa rintangan.
- *Scene simple obstacle*, berupa arena dengan rintangan yang mudah.
- *Scene complex obstacle*, berupa arena dengan rintangan yang rumit.

Pada *scene observer*, level yang akan dibuat hanya berupa arena kosong dengan dinding penghalang pada sekitar arena agar objek karakter tidak melewati batas arena. Pada *scene simple obstacle*, level yang dibuat akan diberi beberapa objek rintangan yang akan menjadi penghalang bagi karakter *user* dan NPC untuk menguji *pathfinding* pada NPC. Pada *scene complex obstacle*, jumlah objek rintangan akan dibuat seperti labirin untuk menguji tingkat kecepatan *rendering* dari *navmesh pathfinding* dan ketepatan NPC dalam mencari jalur.

3.1.3 Pembuatan dan penerapan program *Navmesh* pada *game*

Selanjutnya dari penelitian ini yaitu pembuatan program *Navmesh* pada *game*. Program *navmesh* yang akan dibuat ada 2, yaitu:

- *Navmesh area*
- *Navmesh agent*

Navmesh area digunakan pada arena level untuk menghitung dan menetapkan area yang dapat digunakan sebagai jalur yang dapat dilalui oleh karakter (*walkable*). Penghitungan *navmesh* akan membuat objek rintangan menjadi penghalang dan tidak dapat dijadikan *mesh* oleh *navmesh*.

Navmesh agent sendiri akan digunakan pada NPC sebagai *logic pathfinding*, *navmesh agent* akan menggunakan *navmesh area* yang telah dibuat sebelumnya sebagai acuan jalur. Pada program *navmesh agent* informasi posisi karakter *user* akan dijadikan sebagai *point* untuk digunakan sebagai titik akhir pada *pathfinding*. *Navmesh agent* akan mencari jalur tercepat dan terpendek dari titik awalnya menuju titik akhir lalu menggunakan jalur tersebut untuk menuju titik akhir.

3.1.4 *Preprocessing navmesh*

Sebelum program dijalankan, *Navmesh area* akan memproses area untuk menerapkan *mesh* pada area yang akan dilalui oleh *agent (walkable)*, kemudian akan mengambil informasi lokasi *agent NPC* dan *user* pada *mesh* untuk diubah sebagai *point*. Kemudian setelah informasi tersebut akan diberikan ke *agent NPC* untuk dilakukan proses *pathfinding* secara *real-time* antara *agent NPC* sebagai *starting point* dan *agent user* sebagai *finish point*.

3.1.5 Uji coba *game*

Selanjutnya uji coba akan dilakukan untuk melihat keberhasilan NPC dalam mencari jalur dengan program *Navmesh* dengan menjalankan program dan melihat kinerja dari *agent* NPC dalam mencari jalur menuju *agent user*. Kemudian melakukan beberapa kali uji coba untuk melihat tingkat keberhasilan *agent* dalam ketepatan dan kecepatan mencari jalur.

3.2 Waktu dan Lokasi Penelitian

Penelitian ini dilakukan pada Juni 2020 hingga Juli 2020 di Laboratorium Rekayasa Perangkat Lunak, Program Studi Ilmu Komputer, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Hasanuddin.

3.3 Instrumen Penelitian

3.3.1 Perangkat Lunak

- Unity version: 2018.2.3f1
- Blender 2.81a
- Visual Studio 2017

3.3.2 Perangkat Keras

Perangkat keras yang digunakan penelitian ini yaitu Laptop ASUS ROG G551VW dengan Processor Intel® Core™ i7-6700HQ CPU @ 2.60GHz (8 CPUs), ~2.6GHz 64-bit, Ram 8 GB Menggunakan Sistem Operasi Windows 10 Home Single Language.

BAB IV

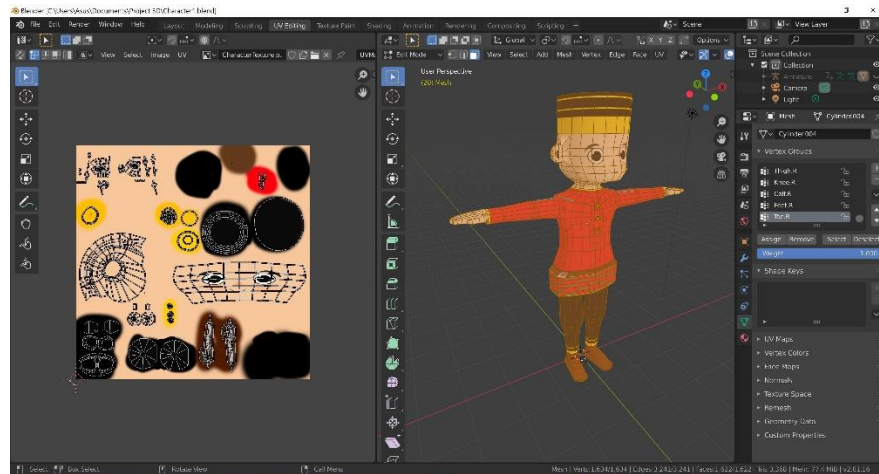
HASIL DAN PEMBAHASAN

4.1 Pembuatan Objek dan Animasi

Pembuatan objek dilakukan pada aplikasi Blender. Dalam aplikasi Blender juga dapat membuat animasi dari objek yang kita buat dan dapat diekspor secara satu ekstensi FBX (Filmbox).

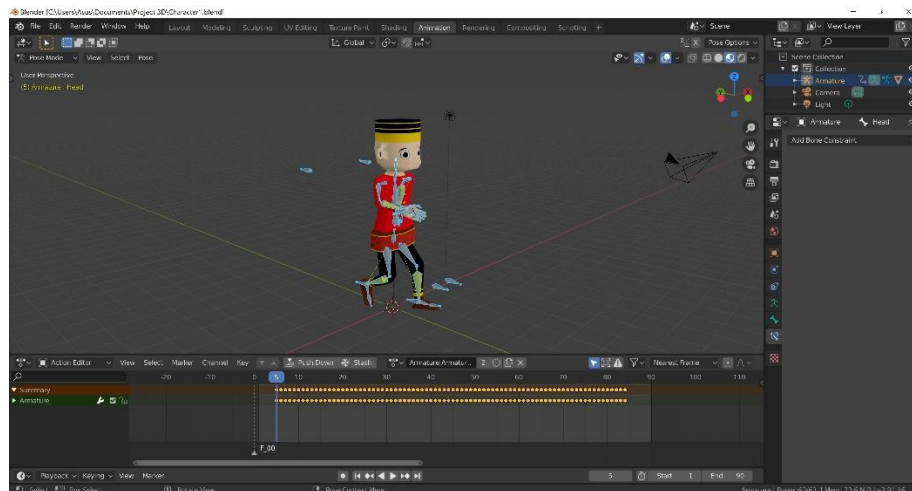
4.1.1 Pemodelan dan Animasi Karakter

Untuk pemodelan karakter *player* dibuat dari sebuah objek silinder lalu dimodel hingga menyerupai seorang karakter. Lalu diberikan *texture* warna dan gambar pada permukaan objek tersebut seperti yang tertera pada gambar 4.1.



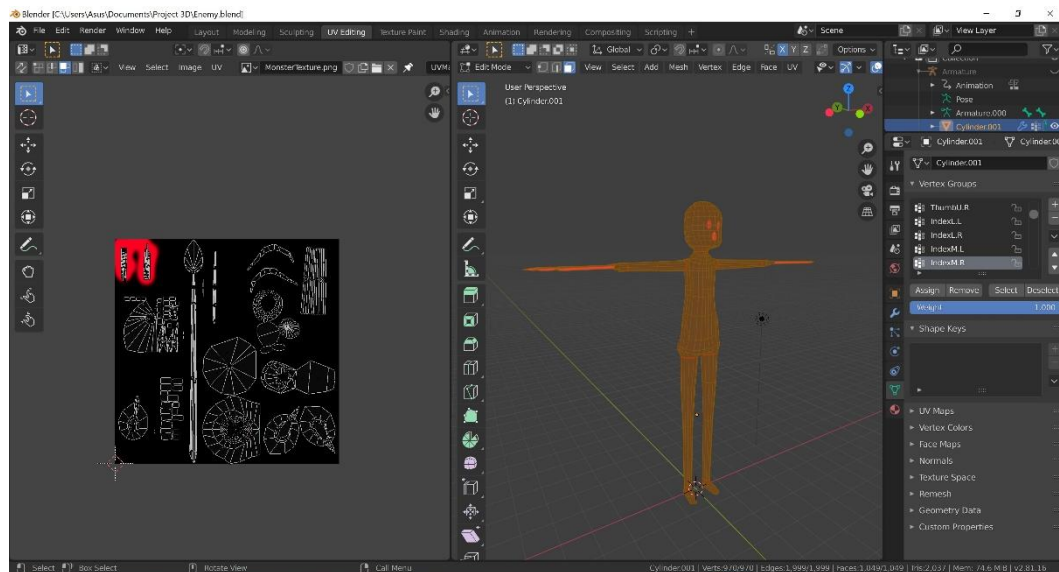
Gambar 4.1. Pemberian *texture* warna menggunakan *UV map* pada karakter *player*

Setelah objek karakter dimodel maka dibuatkan kerangka tulang atau bisa disebut *Armature*. Pembuatan tulang mengikuti struktur tulang pada manusia, seperti kaki, jari, leher, tangan, dan sebagainya. Untuk membuat struktur gerak yang sama seperti manusia, digunakan metode *Inverse Kinematics* pada bagian siku dan lutut karakter sehingga dapat digerakkan secara satu arah tanpa harus diatur secara manual. Selanjutnya dibuatkan animasi karakter saat diam dan bergerak dengan cara membuat pergerakan yang diinginkan *frame per frame* seperti yang tertera pada gambar 4.2.



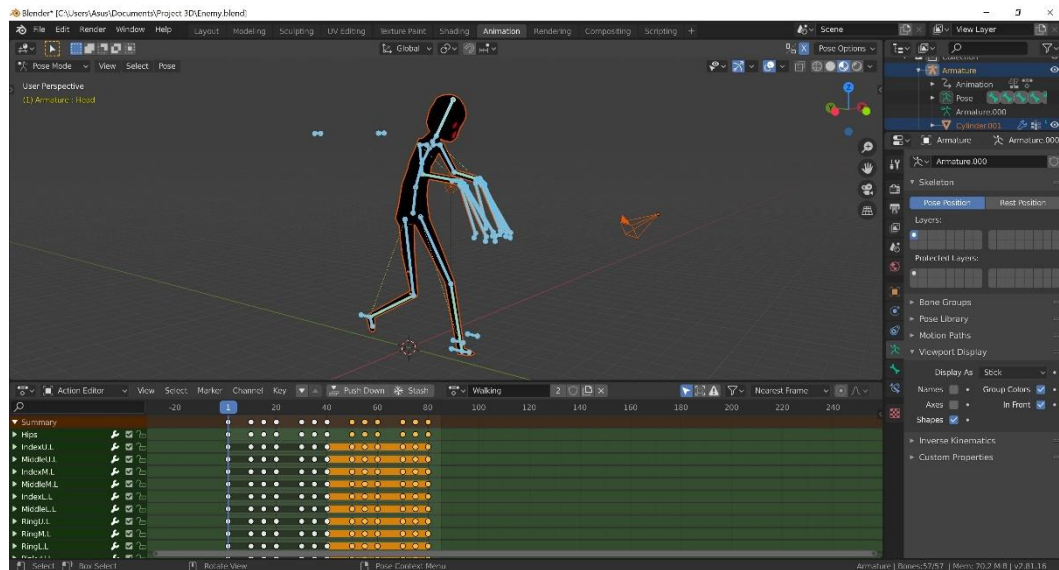
Gambar 4.2. Pembuatan animasi bergerak pada karakter *player*

Untuk pembuatan karakter NPC, metode yang digunakan sama dengan pembuatan karakter *player*, tetapi model karakter dan *texture* warna dibuat berbeda agar dapat dibedakan satu sama lain. Model dan *texture* warna karakter NPC dapat dilihat pada gambar 4.3.



Gambar 4.3. Pemberian *texture* warna menggunakan *UV map* pada karakter NPC

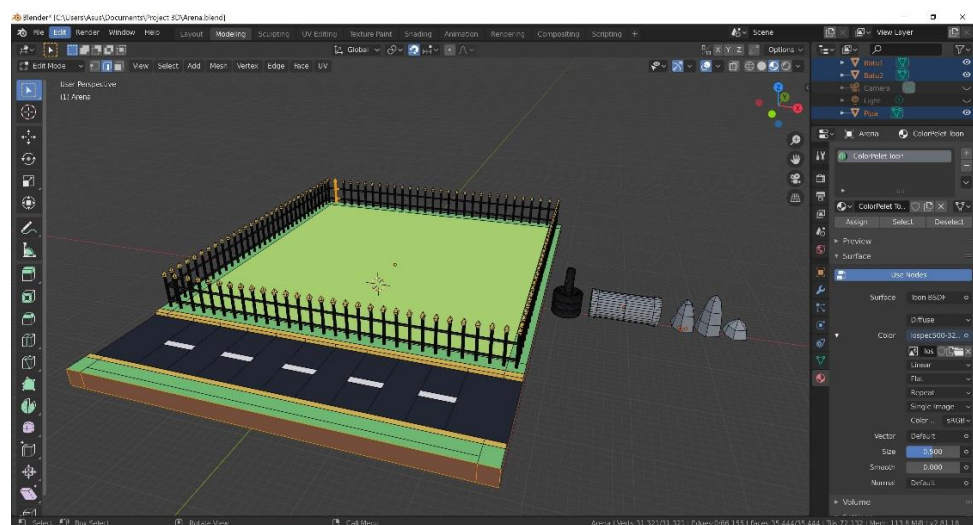
Selanjutnya untuk pembuatan animasi karakter NPC juga menggunakan metode yang sama seperti pembuatan animasi karakter *player* yang dapat dilihat pada gambar 4.4.



Gambar 4.4. Pembuatan animasi bergerak pada karakter *player*

4.1.2 Pemodelan Arena dan Rintangan

Untuk pembuatan Arena sendiri dimulai dari objek kubus dan dimodel membentuk sebuah lapangan kosong di samping jalan. Kemudian untuk pembuatan *texture* warna pada arena metode yang digunakan sama seperti pada pembuatan karakter dengan menggunakan *UV map* pada Blender. Untuk pembuatan rintangan juga menggunakan metode yang sama. Model rintangan yang dibuat berupa tumpukan ban, pipa beton dan bebatuan. Model dan *texture* warna arena dan rintangan tertera pada gambar 4.5.



Gambar 4.5. Model arena dan rintangan

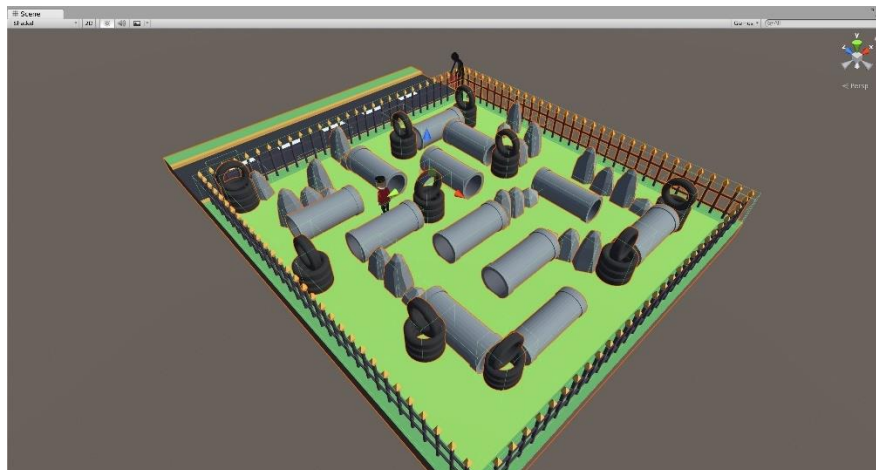
4.2 Pembuatan Scene dan Arena

Selanjutnya objek yang telah dibuat diimport ke dalam aplikasi Unity. Format file FBX dari Blender dapat langsung digunakan pada Unity dengan berbentuk sebuah *prefab*. Hasil *prefab* objek tertera pada gambar 4.6 yang didalamnya terdapat objek, *armature*, material *texture*, dan *avatar* yang akan digunakan pada penganimasian di Unity.



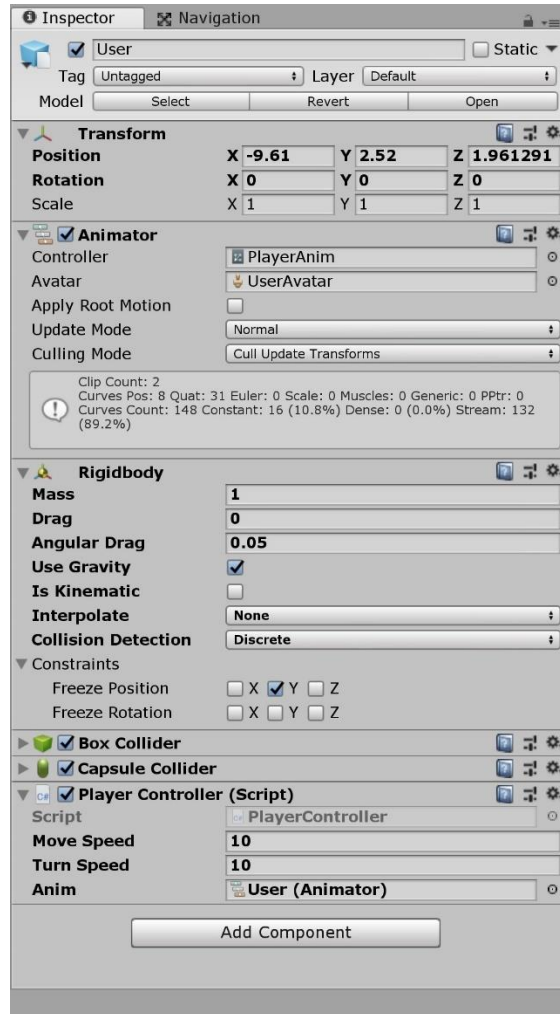
Gambar 4.6. *prefab* objek pada Unity

Selanjutnya dibuat tiga *scene* dimana masing-masing *scene* memiliki struktur penyusunan arena yang berbeda. Pada *observer scene*, pengaturan arenanya tidak diberikan rintangan apapun. Pada *simple obstacle scene*, pengaturan arena dibuat dengan diberikan objek rintangan yang mudah. Pada *complex obstacle scene*, pengaturan arena dan rintangannya berbentuk seperti labirin seperti tertera pada gambar 4.7.



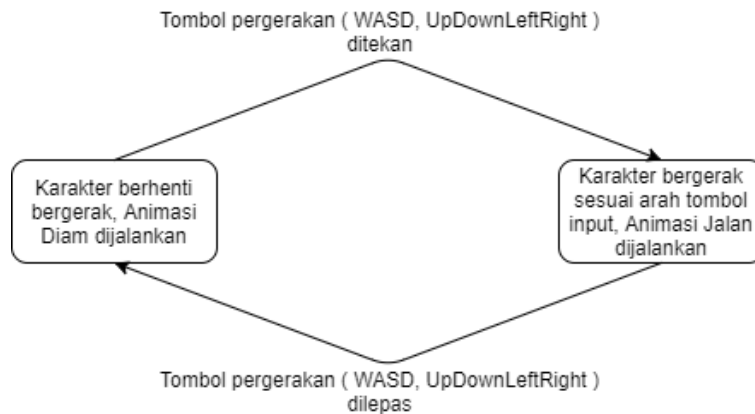
Gambar 4.7. Pengaturan arena pada *Complex Obstacle Scene*

Kemudian dimasukkan karakter player dan NPC pada masing-masing scene dengan diberikan skrip kontrol untuk mengontrol karakter, *animator* untuk mengontrol animasi dari karakter, *physics* seperti *collision*, *rigidbody* dan sebagainya. Contohnya adalah *physics* pada karakter *player* yang tertera pada gambar 4.8.

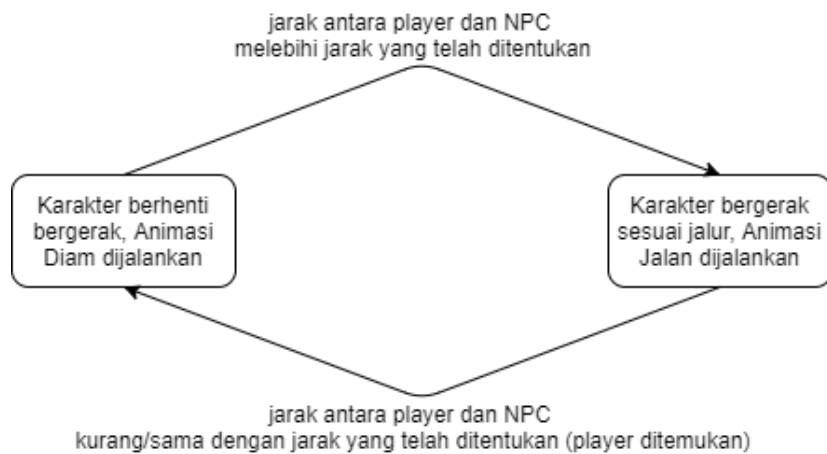


Gambar 4.8. Inspektor pada karakter player

Untuk pergerakan karakter dan animasi akan ditampilkan dalam bentuk *Finite-state machine* (Mesin Keadaan Hingga) agar mudah dipahami. *Finite-state machine* pada karakter tertera pada gambar 4.9 dan gambar 4.10.



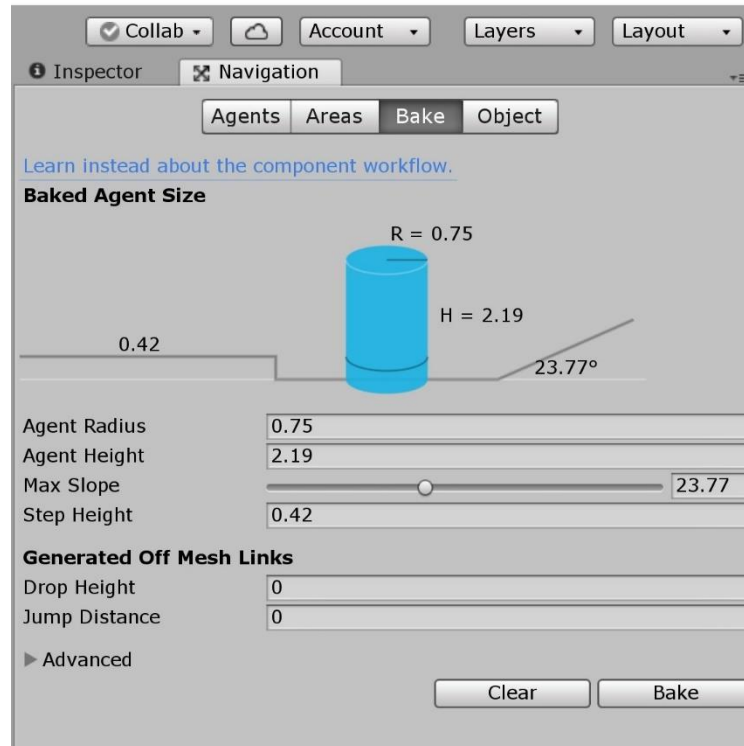
Gambar 4.9. *Finite-state machine* karakter *player*



Gambar 4.10. *Finite-state machine* karakter NPC

4.3 Penerapan dan *Bake Navmesh*

Kemudian dibuatkan *navmesh* pada arena di masing-masing *scene*. Unity telah menyediakan *tool* khusus yang memudahkan untuk pembuatan *navmesh*. *Tool* yang digunakan yaitu *Navigation*. Saat memilih objek arena kita dapat membuka *tool navigation* tersebut seperti yang tertera pada gambar 4.11.



Gambar 4.11. *Window navigation* pada Unity

Pada *tool navigation* terdapat empat menu, yaitu:

- *Agents*

Agents berguna untuk membuat agent dan mengatur ukuran agent tersebut. Atribut-atribut pada *agent* berupa lebar *agent*, tinggi *agent*, maksimal kemiringan yang dapat dilalui *agent* dan ketinggian dari langkah kaki *agent*.

- *Areas*

Areas berguna untuk mengkategorikan area-area pada *navmesh* dan memberikan *cost* yang mempengaruhi metode pencarian jalur *agent* berdasarkan *cost* dari area tersebut.

- *Bake*

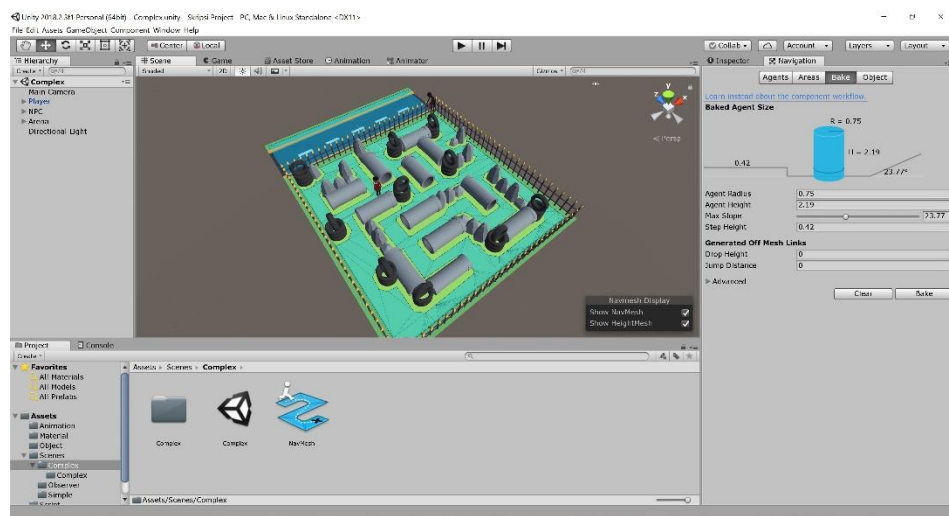
Bake berguna untuk membuat *navmesh* berdasarkan objek yang kita pilih berdasarkan ukuran agent yang telah kita buat.

- *Object*

Object berguna untuk menseleksi objek-objek yang nantinya akan diperhitungkan ketika pembuatan *navmesh*. Terdapat tiga kategori yang

dapat diterapkan pada objek, yaitu *walkable*, *non walkable*, dan *jump*. Dimana *walkable* berarti objek dapat dilalui oleh *agent*, *non walkable* tidak dapat dilalui oleh *agent*, dan *jump* untuk membuat objek dapat dilompati oleh *agent*.

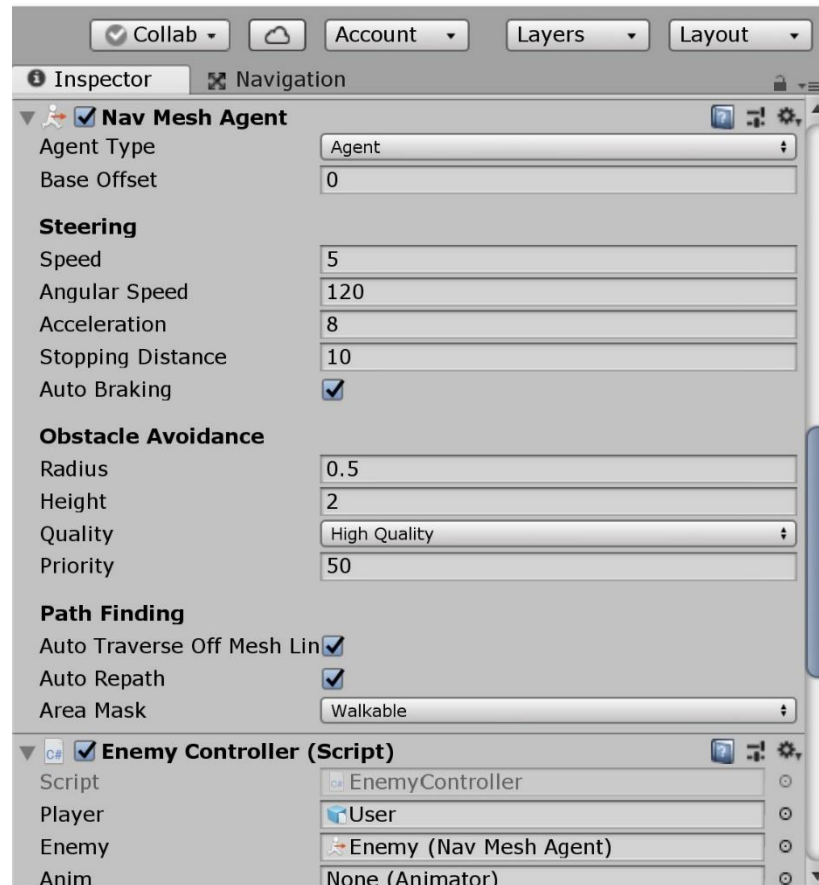
Sebelum menerapkan navmesh pada arena, objek karakter, rintangan dan dinding pada arena akan dikategorikan sebagai non walkable agar navigation dapat membuat navmesh sesuai dengan yang diinginkan. Hasil bake dari navmesh tertera pada gambar 4.12. Metode ini juga diterapkan pada scene yang lain.



Gambar 4.12. Hasil *bake navmesh* pada arena *complex scene*

4.4 Implementasi *Navmesh Agent* Pada Karakter NPC

Setelah navmesh telah dibuat maka akan dibuatkan agent yang akan menggunakan navmesh tersebut. Unity juga telah menyediakan atribut tersendiri untuk pembuatan *agent* pada objek karakter yaitu *Nav Mesh Agent*. Pada karakter NPC ditambahkan *physics Nav Mesh Agent* seperti yang tertera pada gambar 4.13.



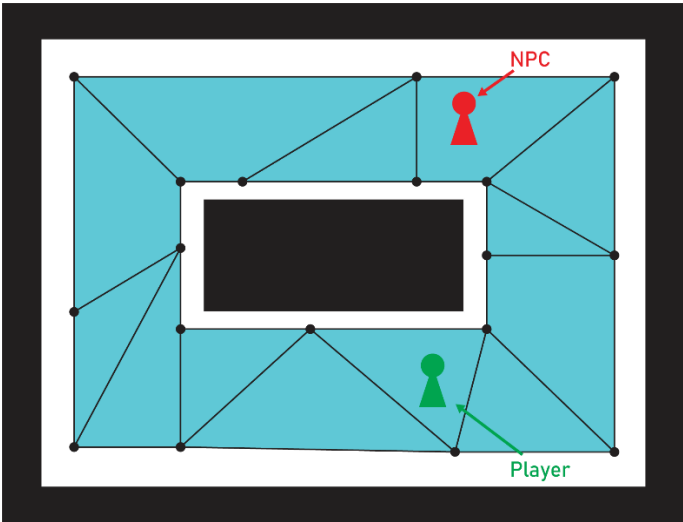
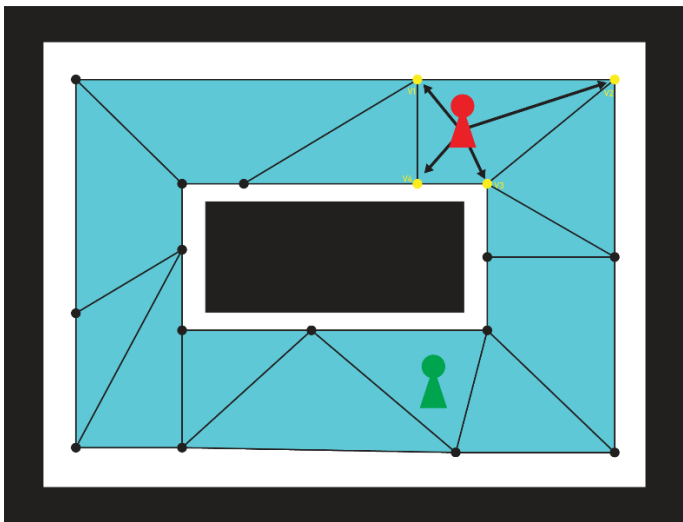
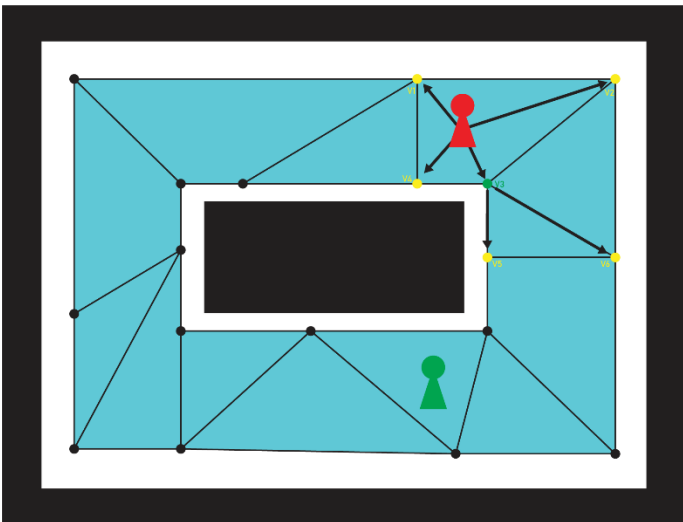
Gambar 4.13. Atribut *Nav Mesh Agent* pada karakter NPC

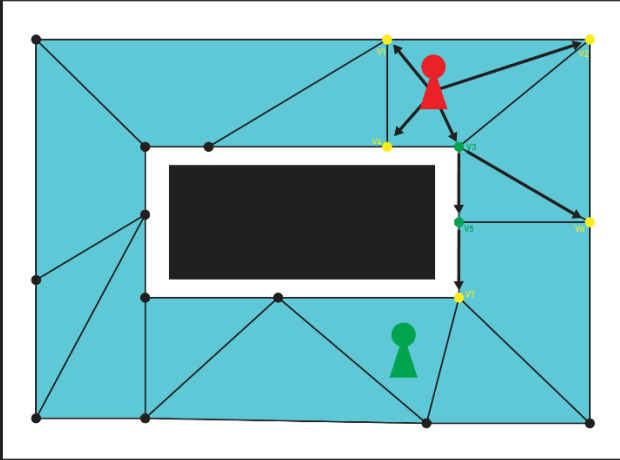
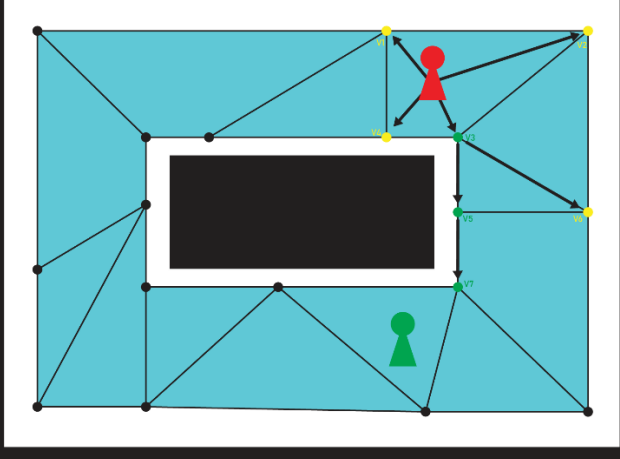
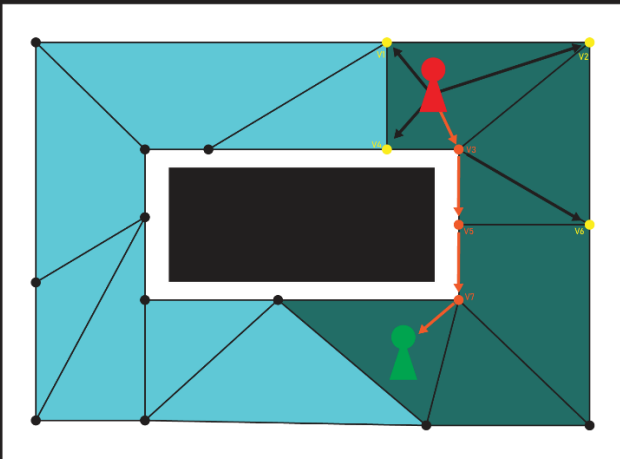
Pada atribut ini dapat diatur kecepatan agent, akselerasi, jarak berhenti, kecepatan rotasi agent dan lainnya. Kemudian dibuatkan skrip untuk mengatur pergerakan NPC berdasarkan navmesh, atribut-atribut yang telah diberikan.

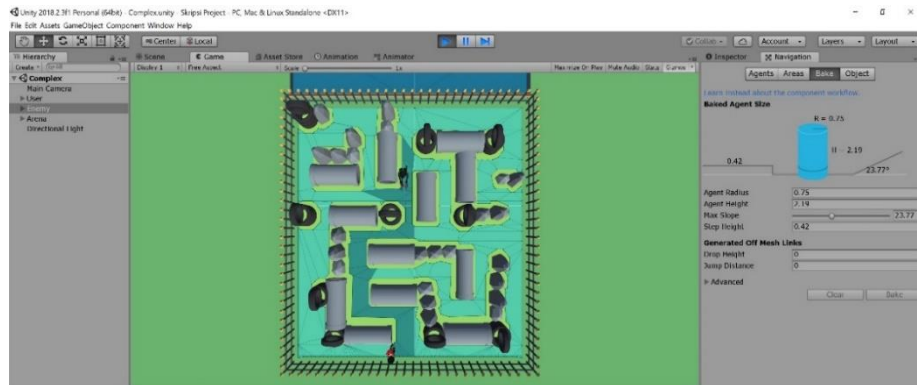
4.5 Uji Coba

Sebelum program dijalankan *navmesh* harus *dibake* dahulu jika terjadi perubahan pada struktur arena yang dapat mempengaruhi bentuk *navmesh* itu sendiri. Kemudian ketika program dijalankan, skrip pada karakter NPC akan mengambil informasi *navmesh*, informasi *agent*, lokasi NPC dan lokasi *player*. Cara kerja *navmesh* tersebut tertera pada table 4.1.

Tabel 4.1. Cara kerja *navmesh*

Gambar	Penjelasan
	<p><i>Navmesh</i> mengambil data <i>mesh</i> yang telah dibuat sebelumnya beserta lokasi <i>NPC (agent)</i> dan <i>player</i></p>
	<p>Lalu kemudian akan diambil sudut (<i>point/vertices</i>) pada segitiga (<i>mesh</i>) lokasi NPC.</p>
	<p>Pada setiap sudut dilakukan kalkulasi menggunakan algoritma pencarian A* yang telah dijelaskan sebelumnya. Setelah itu akan diambil sudut dengan nilai terendah dan dilakukan kalkulasi yang sama pada tetangga (<i>neighbor</i>) dari sudut tersebut.</p>

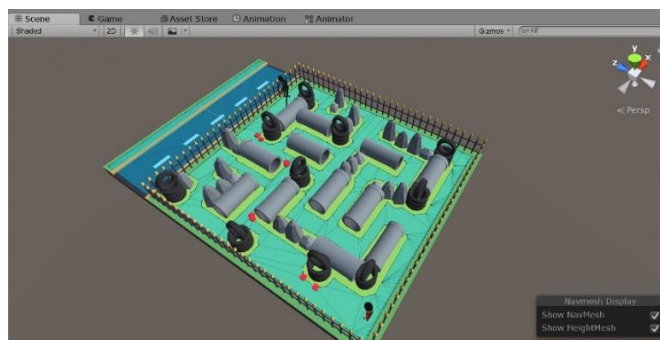
	<p>Lalu sudut dengan nilai terkecil akan diambil dan disimpan jalurnya hingga sampai pada sudut segitiga lokasi <i>player</i></p>
	<p>Ketika sudut segitiga lokasi <i>player</i> ditemukan maka dapat diberikan garis lurus antara sudut dan lokasi <i>player</i></p>
	<p>Setelah mendapatkan lokasi <i>player</i>, lacak ulang jalur kembali ke NPC dan NPC dapat menggunakan jalur yang telah didapatkan, dan juga navmesh akan merubah warna segitiga yang digunakan pada jalur NPC.</p>



Gambar 4.14. Pengambilan rute NPC ketika program dijalankan

Untuk melihat hasil pada Unity, pilih objek NPC dan pilih *window navigation*, Rute yang diambil ditandai oleh warna biru gelap seperti yang tertera pada gambar 4.14. Proses penghitungan dan pengambilan jalur ini dijalankan secara terus menerus saat program dijalankan. Jika terjadi perubahan lokasi NPC atau player maka akan dilakukan penghitungan ulang secara cepat dan pengambilan jalur yang optimal berdasarkan perubahan data tadi secara langsung tanpa dibutuhkan penghentian program.

Bentuk dari node pada *navmesh* Unity yaitu *Vector3* atau variabel x,y, dan z, dimana merupakan koordinat posisi pada unity. Pada saat jalur dibuat, program akan menyusun node dari posisi NPC menuju posisi *player*. Kemudian node-node tersebut akan terhubung dan menghasilkan jalur bagi NPC. Untuk menampilkan node tersebut, dibuatkan *method pathViewer()* pada *script* NPC untuk memunculkan sebuah objek kubus merah dan menempatkannya sesuai dengan koordinat-koordinat node jalur. Hasil *script* tertera pada gambar 4.15.

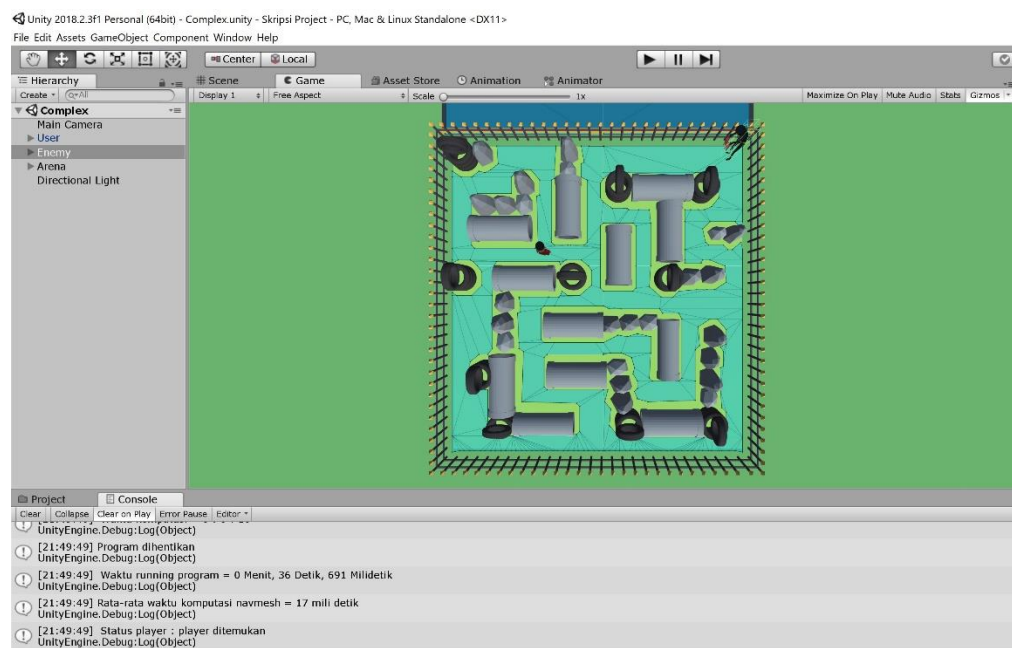


Gambar 4.15. Penempatan objek kubus pada koordinat *node* jalur NPC

Hal ini dilakukan untuk membuktikan apakah program navmesh yang digunakan untuk membuat *node-node* koordinat objek kubus dan program navmesh yang digunakan untuk membuat jalur NPC menghasilkan hasil jalur yang sama, metode pencarian yang sama, dan juga untuk mengetahui lokasi *node-node* pada jalur NPC.

4.6 Analisis Waktu Komputasi

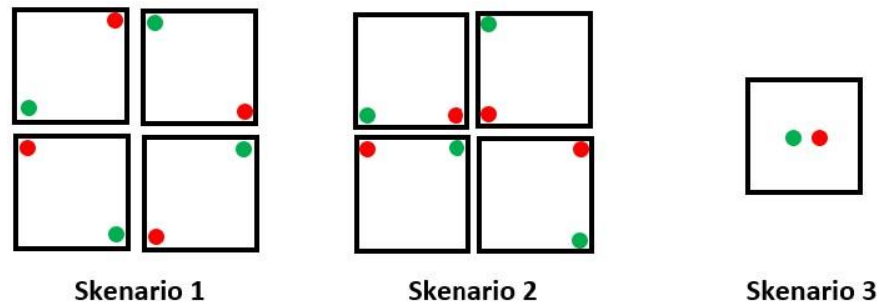
Pada program diberikan variabel waktu untuk menghitung waktu komputasi *navmesh* menghitung, memilih jalur, dan status player. Untuk menganalisis waktu komputasi, masing-masing *scene* dijalankan 10 kali dengan waktu sekitar 30 detik. Karena program *navmesh* dijalankan setiap frame pada Unity dalam kasus ini 30 *frame* per detik, maka setiap program dijalankan akan dihitung waktu rata-rata komputasi *navmesh* dibagi dengan berapa kali komputasi *navmesh* dijalankan seperti yang tertera pada gambar 4.16.



Gambar 4.16. Menu *Console* pada program complex scene

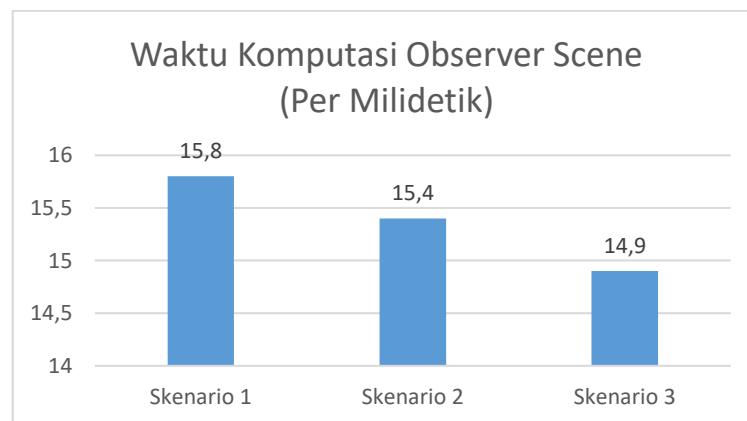
Pada Masing-masing *scene* akan dilakukan 3 skenario penentuan lokasi karakter *player* dan NPC. Skenario 1 yaitu dengan menempatkan karakter NPC dan *player* berlawanan arah secara diagonal, skenario 2 yaitu menempatkan karakter NPC dan *player* berlawanan arah tetapi pada sisi yang sama, terakhir skenario 3

yaitu menempatkan NPC dan *player* dengan jarak yang cukup dekat. Contoh scenario tertera pada gambar 4.17 dimana lingkaran hijau melambangkan karakter *player* dan lingkaran merah melambangkan karakter NPC.

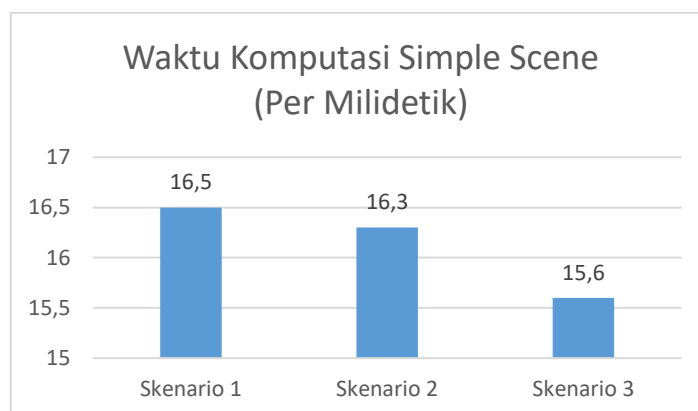


Gambar 4.17. Skenario penempatan karakter

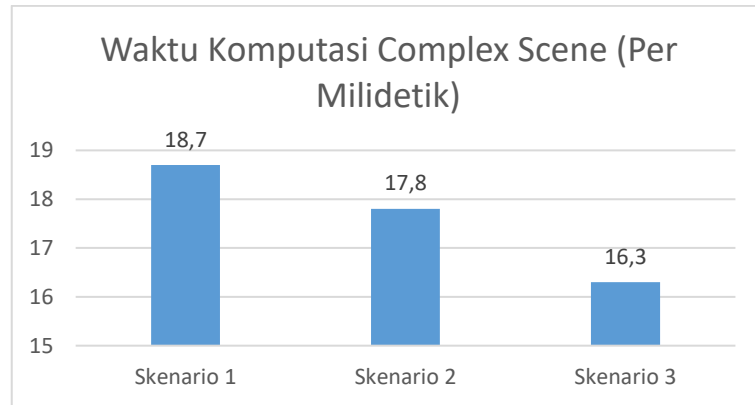
Maka hasil penghitungan waktu masing-masing *scene* sebagai berikut:



Gambar 4.18. Waktu komputasi *Observer Scene*



Gambar 4.19. Waktu komputasi Simple Scene



Gambar 4.20. Waktu komputasi *Complex Scene*

Tabel 4.2. Rata-rata waktu komputasi program navmesh

Scene	Skenario 1	Skenario 2	Skenario 3	Total waktu rata-rata
Observer Scene	15.8	15.4	14.9	15.4
Simple Scene	16.5	16.3	15.6	16.1
Complex Scene	18.7	17.8	16.3	17.6
Total keseluruhan				16.366

Pada hasil komputasi *observer scene* pada gambar 4.15, dilakukan percobaan dan penghitungan waktu komputasi *navmesh*. Pada sepuluh kali percobaan pada masing-masing skenario didapatkan hasil berupa waktu komputasi yang berbeda-beda. Lalu dihitung jumlah rata-rata waktu komputasi berdasarkan hasil sepuluh kali percobaan. Begitupula pada hasil komputasi *simple scene* dan *complex scene* pada gambar 4.16 dan gambar 4.17. Berdasarkan data tersebut, maka dihitung waktu rata-rata komputasi *navmesh* keseluruhan berdasarkan waktu rata-rata komputasi *navmesh* ketiga *scene* tersebut pada table 4.2 yaitu 16.366 milidetik.

BAB V

KESIMPULAN

5.1 Kesimpulan

Berdasarkan hasil komputasi dan uji coba program navigasi game menggunakan algoritma pencarian A*, maka dapat disimpulkan bahwa:

1. Penerapan algoritma A* dapat langsung diterapkan berkat *library* pada *navmesh* yang telah disediakan pada Unity, tetapi pengaturan *agent*, ukuran, dan rintangan *navmesh* itu sendiri masih harus diatur secara manual.
2. Hasil komputasi *navmesh* selalu mendapatkan jalur dari posisi awal karakter pengejar menuju posisi karakter pemain. Waktu komputasi dari *navmesh* dinilai cukup cepat yaitu sebesar 0.016 detik atau 16.366 milidetik per *frame* berdasarkan perhitungan 30 frame per detik pada *Unity* yaitu 0,033 detik atau 33.333 milidetik per *frame*. Beberapa faktor yang dapat mempengaruhi kecepatan waktu komputasi adalah ukuran arena, jumlah rintangan dan perangkat yang dipakai.

5.2 Saran

Setelah melakukan penelitian ini, penulis menyarankan:

1. Pada penelitian ini program *navmesh* yang digunakan merupakan *library* yang telah disediakan oleh Unity. Untuk kedepannya bisa dilakukan percobaan menggunakan program *navmesh* yang dibuat sendiri sehingga memungkinkan untuk dilakukan pengubahan yang diinginkan.
2. Pergerakan animasi objek telah diatur secara tetap pada Blender dan tidak bisa diubah sesuai dengan kecepatan pada Unity. Untuk kedepannya dapat dibuat animasi pergerakan yang sesuai dengan kecepatan yang ditentukan pada Unity.
3. Menggunakan perangkat yang memiliki spesifikasi yang lebih baik sehingga menghasilkan waktu komputasi yang lebih cepat, dan dapat mengelola area yang lebih besar dengan cepat.

DAFTAR PUSTAKA

- Achmad, M. H., Findari, W. S., Ann, N. Q., Pebrianti, D., & Daud, M. R. (2016). Stereo camera—based 3d object reconstruction utilizing semi-global matching algorithm. *2016 2nd International Conference on Science and Technology-Computer (ICST)*.
- Apperley, T. H. (2006). Genres and Game Studies: Towards a Critical Approach to Video Game Genres. *Simulation & Gaming* 37.1, 6-23.
- Barnouti, N. H., Al-Dabbagh, S. S., & Naser, M. A. (2016). Pathfinding in Strategy Games and Maze Solving Using A* Search Algorithm. *Journal of Computer and Communications*, 2016, 4, 15-25.
- Cahyadi, M. A., Widhiarso, W., & & Yohannes, Y. (2017). *Perbandingan Algoritma A*, Dijkstra dan Floyd Warshall Untuk Menentukan Jalur Terpendek Pada Permainan "Bacteria Defense"*. Palembang: Jurusan Teknik Informatika, STMIK GI MDP.
- Fajri, C. (2012). Tantangan Industri Kreatif-Game Online di Indonesia. *Jurnal ASPIKOM* 1.5, 443-454.
- Funge, J. D. (2004). Artificial intelligence for computer games: an introduction. *CRC Press*.
- Graham, R., McCabe, H., & Sheridan, S. (2003). Pathfinding in Computer Game. *The ITB Journal*.
- Guruji, A. K., Agarwal, H., & & Parsediya, D. K. (2016). Time-efficient A* algorithm for robot path planning. *Procedia Technology*, 144-149.
- Harsadi, P. (2016). Pathfinding pada Lingkungan Statis Berdasarkan Artificial Potential Field Dengan Flocking Behavior Untuk Non-Player Character Follower Pada Game. *Jurnal Ilmiah SINUS*.
- Lasseter, J. (1987). Principle of Traditional Animation Applied To 3D Computer Animation. *Computer Graphics, Volume 21, Number 4*.
- Mathew, G. E. (2015). Direction Based Heuristic For Pathfinding In Video Game. *Procedia Computer Science*.
- Millington, I., & Funge, J. (2009). *Artificial Intelligence For Games, Second Edition*. Burlington, USA: Morgan Kaufmann Publishers.

- Pramono, A. (2015). Algoritma Pathfinding A* pada Game RPG Tanaman Higienis. *Jurnal Edukasi dan Penelitian Informatika (JEPIN) Vol. 1, No. 2, (2015)*, 76.
- Reese, B., & Stout, B. (1999). Finding a Pathfinder. *AAAI 99 Spring Symposium on Artificial Intelligence and Computer Games*.
- Rifai, W. A. (2015). *Pengembangan Game Edukasi Lingkungan Berbasis Android*. Yogyakarta: Fakultas Teknik Universitas Negeri Yogyakarta.
- Sanny, L. (2018). *Potensi Industri Game Indonesia*. Jakarta: Binus University Business School.
- Setiawan, A., Harsadi, P., & Siswanti, S. (2019). Penerapan Pathfinding Menggunakan Algoritma A* pada Non Player Character (NPC) Di Game. *Jurnal Ilmiah Sinus (JIS) Vol : 17, No. 2*.
- Sharlin, E., Watson, B., Kitamura, Y., Rorabeck, D., Lederer, R., Sutphen, S., & Kishino, F. (2004). The Tangible Pathfinder Design of a Wayfinding Trainer for the Visually Impaired. *Proc. Graphics Interface*.
- Shi, X. C., & Hao. (2011). A*-based Pathfinding in Modern Computer Games. *IJCSNS International Journal of Computer Science and Network Security, VOL.11 No.1, January 2011*.
- Silver, D. (2005). Cooperative Pathfinding. *AIIDE 1*, 117-122.
- Stout, B. (1996). Smart moves: Intelligent pathfinding. *Game developer magazine*, 28-35.
- Van Gumster, J. (2020). *Blender for dummies*. John Wiley & Sons.
- Wafiqurrahman, N. (2015). Penerapan Algoritma a* (A-Star) Untuk Penentuan Rute Terpendek Game Pramuka Berbasis Android. *Doctoral dissertation, Universitas Islam Negeri Maulana Malik Ibrahim*.
- Yannakakis, G. N., & Togelius, J. (2014). A panorama of artificial and computational intelligence in games. *IEEE Transactions on Computational Intelligence and AI in Games* 7.4.

LAMPIRAN

Lampiran A. Source Code

EnemyController.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Diagnostics;
using UnityEngine;
using UnityEngine.AI;
using Debug = UnityEngine.Debug;

public class EnemyController : MonoBehaviour {

    //Deklarasi variabel player
    public GameObject player;

    //Deklarasi navmesh agent bagi karakter NPC
    public NavMeshAgent enemy;

    //Deklarasi animasi bagi karakter NPC
    public Animator anim;

    //Deklarasi variabel untuk penghitungan waktu komputasi, waktu running, dan status
    player
    public Stopwatch computingTimer, runningTimer;
    public TimeSpan computing_timeSpan, running_timeSpan;
    public int mean, tick, minute, second, milisecond;
    public String status;

    //Deklarasi variabel untuk menampilkan node jalur
    private Vector3 path1;
    public GameObject cube, cubePath;
    public int pathDifference = 0;

    //Inisiasi awal program
    void Start () {

        //Pemanggilan variabel navmesh agent
        enemy = GetComponent<NavMeshAgent>();
        enemy.updateRotation = false;

        //Pemanggilan animator untuk animasi karakter NPC
        anim = GetComponent<Animator>();

        //Inisiasi waktu mulai penghitungan komputasi navmesh dan running program
        Debug.Log("Waktu komputasi mulai");
        computingTimer = Stopwatch.StartNew();
        runningTimer = Stopwatch.StartNew();

        //Inisiasi penghitungan waktu rata-rata komputasi navmesh
        mean = 0;
        tick = 0;
    }
}
```

```

//Update dijalankan setiap frame pada game (30 frame per detik)
void Update () {

    //Pengkondisian jika karakter player ditemukan
    if (player != null)
    {
        status = ("player ditemukan");

        //Pemanggilan proses penghitungan jalur berdasarkan posisi NPC, posisi
        player dan struktur navmesh yang telah dipanggil
        enemy.SetDestination(player.transform.position);

        //Pengkondisian untuk mengatur rotasi NPC sesuai dengan jalur
        if (enemy.velocity.sqrMagnitude > Mathf.Epsilon)
        {
            transform.rotation =
Quaternion.LookRotation(enemy.velocity.normalized);
        }

        // Pengkondisian untuk menjalankan animasi jalan jika jarak NPC dan jarak
        player dibawah dengan jarak yang ditentukan
        if (enemy.remainingDistance < 8)
        {
            anim.SetBool("isWalking", false);
        }
        else
        {
            anim.SetBool("isWalking", true);
        }
    }

    else
    {
        status = ("player tidak ditemukan");
    }

    //Penambahan waktu tick/update setiap kelas update dijalankan
    tick++;

    //Menginisiasi method untuk menampilkan node dalam bentuk kubus
    pathViewer();

    //Penghentian waktu komputasi navmesh
    computingTimer.Stop();
    computing_timeSpan = computingTimer.Elapsed;
    Debug.Log(" Waktu komputasi = " + computing_timeSpan.Minutes + " : " +
    computing_timeSpan.Seconds + " : " + computing_timeSpan.Milliseconds);
    //Penambahan waktu komputasi pada variabel mean untuk penghitungan waktu rata-
    rata komputasi
    computingTimer = Stopwatch.StartNew();
    mean += computing_timeSpan.Milliseconds;
}

//Method untuk menampilkan node jalur NPC dan menampilkan koorninat node tersebut
dalam bentuk kubus
void pathViewer()
{

    //Mengambil panjang array node jalur
    int pathLegth = enemy.path.corners.Length;

    //Mendeteksi jika jumlah array jalur berubah
    if (pathDifference != pathLegth)
    {

        //Perungalan untuk memunculkan dan menempatkan kubus sesuai koordinat node
        for (int j = 0; j < pathLegth; j++)
        {
            cubePath = Instantiate(cube) as GameObject;
            cubePath.transform.position = enemy.path.corners[j];
        }
    }
}

```

```

/*Penghapusan objek kubus pada node awal dan node akhir agar tidak terjadi pemunculan
kubus secara
    berkelanjutan ketika karakter NPC dan player bergerak*/
    if (j==0 || j==pathLegth-1)
    {
        Destroy(cubePath);
    }

    //Debug.log untuk menampilkan koordinat node
    /*Vector3 path1 = enemy.path.corners[j];
    Debug.Log(path1);*/
    }
}
else
{
    }
}

//Kelas dijalankan ketika program dihentikan
private void OnApplicationQuit()
{
    //Penghentian waktu running program
    runningTimer.Stop();
    running_timeSpan = runningTimer.Elapsed;

    //Penghitungan waktu rata-rata komputasi navmesh
    mean /= tick;

    Debug.Log("Program dihentikan");
    Debug.Log(" Waktu running program = " + running_timeSpan.Minutes + " Menit, "
+ running_timeSpan.Seconds + " Detik, " + running_timeSpan.Milliseconds + " Milidetik
");
    Debug.Log("Rata-rata waktu komputasi navmesh = " + mean + " mili detik");
    Debug.Log(" Status player : " + status);
}
}

```

PlayerController.cs

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

public class PlayerController : MonoBehaviour {

    //Deklarasi variabel kecepatan jalan dan rotasi player
    public float moveSpeed;
    public float TurnSpeed;

    //Deklarasi button movement bagi player
    Vector2 input;

    //Deklarasi sudut rotasi player
    Quaternion targetRotation;
    float angle;

    //Deklarasi animasi bagi karakter player
    public Animator anim;

    //Inisiasi awal program
    void Start ()
    {
        //Pemanggilan animator untuk animasi karakter player
        anim = GetComponent<Animator>();
    }

    //Kelas untuk mengambil nilai x dan y
    void GetInput()
    {
        //Pengambilan nilai x dan y sesuai dengan tombol yang ditekan oleh user (W, A, S, D) atau (Up, Down, Left, Right)
        input.x = Input.GetAxisRaw("Horizontal");
        input.y = Input.GetAxisRaw("Vertical"); }

    //Kelas untuk menghitung arah pergerakan player sesuai dengan nilai x dan y
    void CalculateDirection()
    {
        angle = Mathf.Atan2(input.x, input.y);
        angle = Mathf.Rad2Deg * angle;
    }

    //Kelas untuk menghitung arah rotasi dan kecepatan rotasi
    void Rotate()
    {
        targetRotation = Quaternion.Euler(0, angle, 0);
        transform.rotation = Quaternion.Slerp(transform.rotation, targetRotation,
        TurnSpeed * Time.deltaTime);

        //Inisiasi animasi jalan pada karakter player
        anim.SetBool("isWalking", true);
    }

    //Kelas untuk menghitung pergerakan karakter player
    void Move()
    {
        transform.position += transform.forward * moveSpeed * Time.deltaTime;

        //Inisiasi animasi jalan pada karakter player
        anim.SetBool("isWalking", true);
    }
}
```

```

//Update dijalankan setiap frame pada game (30 frame per detik)
void Update ()
{
    //Pemanggilan kelas GetInput
    GetInput();

    //Pengkondisian jika karakter diam maka animasi jalan akan dihentikan
    if (Mathf.Abs(input.x) < 1 && Mathf.Abs(input.y) < 1)
    {
        anim.SetBool("isWalking", false);
        return;
    }

    //Pemanggilan kelas CalculateDirection, Rotate dan Move
    CalculateDirection();
    Rotate();
    Move();
}
}

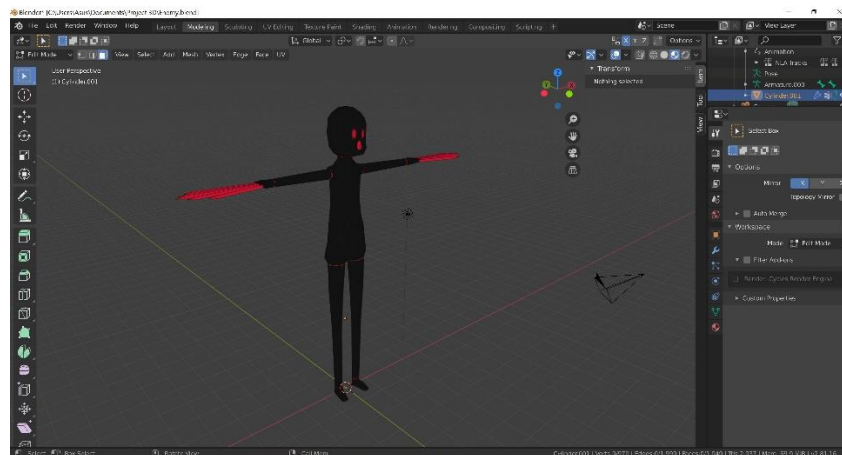
```


Lampiran B. Model 3 Dimensi Karakter *Player*



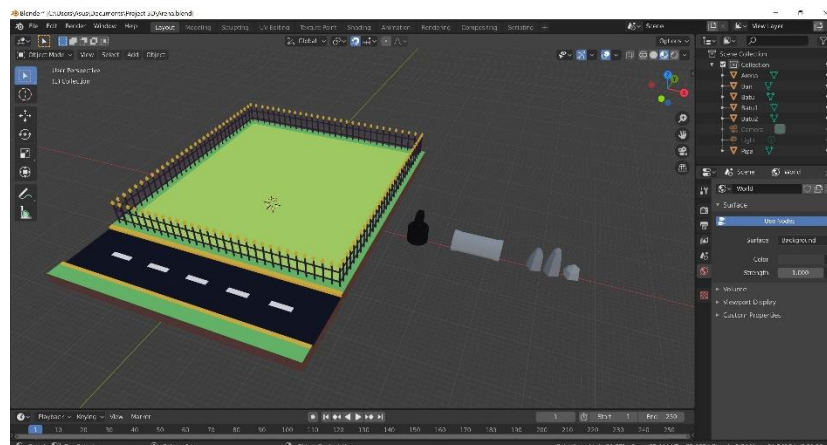
Gambar B.1. Model karakter *Player*

Karakter NPC



Gambar B.2. Model karakter NPC

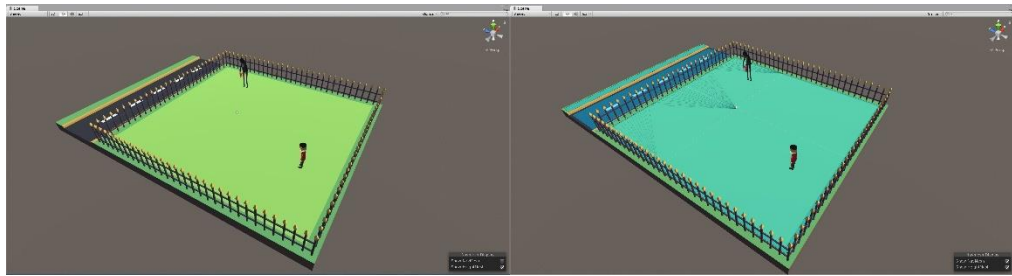
Arena dan Rintangan



Gambar B.3. Model arena dan rintangan

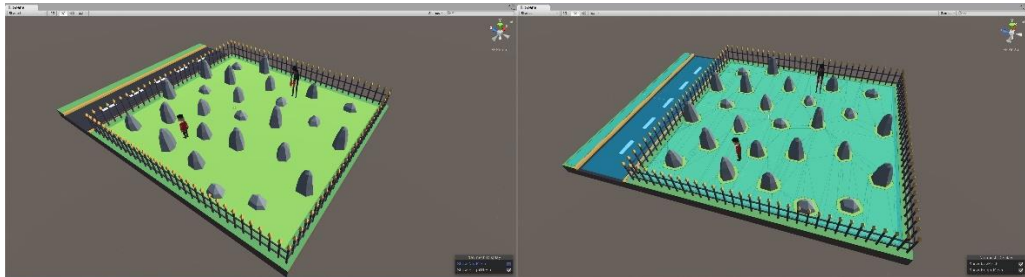
Lampiran C. Scene Game

Observer Scene



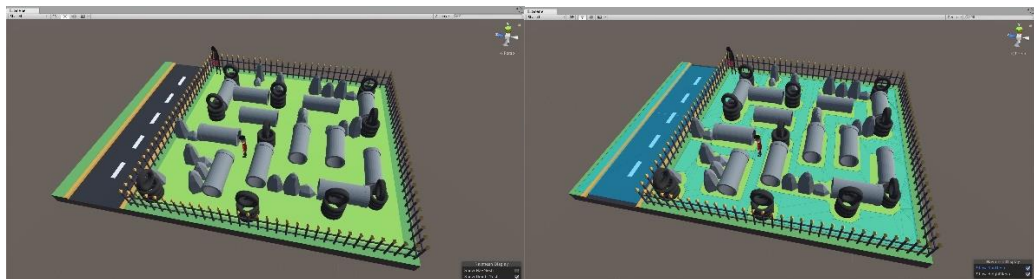
Gambar C.4. *Observer scene* beserta *navmeshnya*

Simple Scene



Gambar C.5. *Simple scene* beserta *navmeshnya*

Complex Scene



Gambar C.6. *Complex scene* beserta *navmeshnya*