

SKRIPSI

**PENERAPAN ALGORITMA A^* *PATHFINDING* DALAM
PEMBENTUKAN *ARTIFICIAL INTELLIGENCE ENEMY* PADA
GAME GHOST ADVENTURE 3D
(STUDI KASUS : PENCARIAN FAKULTAS UHO)**

**Diajukan Untuk Memenuhi
Salah Satu Syarat memperoleh derajat Sarjana Teknik**



ADNAN HIDAYAT

E1E1 15 001

**PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS HALU OLEO
KENDARI**

2020

HALAMAN PENGESAHAN

Skripsi Sarjana

**Penerapan Algoritma *A* Pathfinding* Dalam Pembentukan *AI Enemy* Pada
Game Ghost Adventure 3D (Studi kasus : Pencarian Fakultas UHO)**

Adalah benar dibuat oleh saya sendiri dan belum pernah dibuat dan diserahkan sebelumnya baik sebagian maupun seluruhnya, baik oleh saya ataupun orang lain, baik di Universitas Halu Oleo ataupun institusi pendidikan lainnya.

Kendari, Januari 2020



Adnan Hidayat
E1E1 15 001

Pembimbing I

Sutardi, S.Kom., MT
NIP. 19760222 201012 1 001

Pembimbing II

Jumadil Nangi, S.Kom., MT
NIP. 19870206 201504 1 003

Mengetahui,

Ketua Jurusan Teknik Informatika
Fakultas Teknik Universitas Halu Oleo



Sutardi, S.Kom., MT
NIP: 19760222 201012 1 001

HALAMAN PENGESAHAN

Skripsi Sarjana

**Penerapan Algoritma A* Pathfinding Dalam Pembentukan AI Enemy Pada
Game Ghost Adventure 3D (Studi kasus : Pencarian Fakultas UHO)**

Telah diuji dan dipertahankan dihadapan sidang penguji skripsi Jurusan Teknik Informatika Fakultas Teknik Universitas Halu Oleo dan dinyatakan memenuhi syarat guna memperoleh gelar Sarjana Teknik.

Kendari, Januari 2020

Tim Penguji:

1. Sutardi, S.Kom., MT

Pembimbing I

2. Jumadil Nangi, S.Kom., MT

Pembimbing II

3. Bambang Pramono, S.Si., MT

Penguji I (Ketua Sidang)

4. LM Tajidun, ST., M.Eng

Penguji II (Sekretaris Sidang)

5. Subardin, ST., MT

Penguji III (Anggota)

Tanda Tangan:

1.

2.

3.


4.

5.

Mengesahkan,

Dekan Fakultas Teknik

Universitas Halu Oleo,


Dr. Edward Ngii, ST., MT
NIP: 19720212 199802 1 001

Ketua Jurusan Teknik Informatika

Fakultas Teknik Universitas Halu Oleo,


Sutardi, S.Kom., MT
NIP: 19760222 201012 1 001

PERNYATAAN

Dengan ini saya menyatakan bahwa Laporan Skripsi ini tidak terdapat karya yang pernah diajukan untuk memperoleh gelar kesarjanaan di suatu perguruan tinggi dan sejauh yang penulis ketahui bahwa tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis diacu dalam naskah ini dan disebutkan dalam daftar pustaka.

Kendari, Januari 2020



Adnan Hidayat

INTISARI

Adnan Hidayat, E1E115001

PENERAPAN ALGORITMA A^* PATHFINDING DALAM PEMBENTUKAN ARTIFICIAL INTELLIGENCE ENEMY PADA GAME GHOST ADVENTURE 3D (STUDI KASUS : PENCARIAN FAKULTAS UHO)

Skripsi, Fakultas Teknik, 2019

Kata Kunci: Pencarian rute, *Artificial intelligence*, Algoritma A^*

Pencarian jalur biasanya sering digunakan oleh orang-orang yang ingin menuju tempat yang dia inginkan, awalnya orang akan bertanya kepada orang lain di jalur mana yang paling cepat untuk sampai ketujuan. Seiring perkembangan teknologi banyak algoritma atau metode pencarian rute terpendek diimplementasikan kedalam aplikasi seperti Maps. Algoritma A^* (*All star*) merupakan salah satu algoritma pencarian rute yang menggunakan fungsi *heuristic* nya dan mengambil nilai *cost* terkecil sebagai rute terpendek. Algoritma ini biasa digunakan dalam web GIS. Pada penulisan skripsi ini penulis merancang Algoritma A^* dalam *game 3D* pada permasalahan pencarian rute dari posisi pemain menuju ke fakultas yang dituju oleh pemain sebagai bantuan dalam mencari informasi yang ada dalam *game* ini.

Adapun tahapan dari algoritma A^* adalah mencari jarak rute tercepat yang akan ditempuh suatu *start node* sampai ke *node* target dengan memerhatikan penghalang yang ada, Teknik pencarian yang digunakan dalam penelitian ini adalah menggunakan algoritma A^* dengan fungsi *heuristic manhattan distance*.

Hasil yang didapatkan dari penelitian ini bahwa algoritma A^* pemain algoritma A^* dapat menunjukkan jalur tercepat yang harus ditempuh pemain agar bisa mencapai fakultas yang ada di Universitas Halu Oleo sebagai tujuan yang diinginkan.

ABSTRACT

Adnan Hidayat, E1E115001

PENERAPAN ALGORITMA A* PATHFINDING DALAM PEMBENTUKAN ARTIFICIAL INTELLIGENCE ENEMY PADA GAME GHOST ADVENTURE 3D (STUDI KASUS : PENCARIAN FAKULTAS UHO)

Mini Thesis, Faculty of Engineering, 2019

Keywords : *Route search, Artificial intelligence, A* algorithm*

*Path search is usually often used by people who want to go to the place he wants, initially people will ask others on which path is the fastest to get to their destination. As technology develops, many algorithms or shortest route search methods are implemented in applications such as Maps. The A * (All star) algorithm is one of the route search algorithms that uses its heuristic function and takes the smallest cost value for the shortest route. This algorithm is commonly used on GIS websites. In writing this thesis the author designed the A * Algorithm in 3D games on the problem of finding the route from the player's position to the faculty that the player is aiming for as an aid in finding information in this game.*

*The stages of the A * algorithm are looking for the distance of the fastest route that will be taken by a start node to the target node by observing the obstacles that exist, the search technique used in this study is to use the A * algorithm with the heuristic manhattan distance function.*

*The results obtained from this study that the A * algorithm A * player algorithm can show the fastest path that must be taken by players in order to reach the faculty at Halu Oleo University as the desired goal.*

HALAMAN PERSEMBAHAN



Assalamu'alaikum Warrohmatullahi Wabarokatuh

**Puji syukur selalu terpanjatkan kehadiran Allah Subhana Wa
Ta'ala dan salam semoga selalu tercurahkan kepada Rosulullah
Sholallahu 'Alaihi Wassalam**

Kumpersembahkan skripsi ini untuk:

Untuk kedua orang tuaku, ayahanda **Sukman S** dan ibunda **Hasrawati**. Sebagai tanda bakti, hormat dan rasa terima kasih atas seluruh dukungan serta doa yang tidak pernah putus kalian berikan padaku selama 4 tahun ini. Ini juga sebagai wujud buktiku menjalankan amanah yang telah kalian berikan padaku sebagai anak.

Untuk semua orang yang sungguh saya mohonkan maaf karena tidak dapat saya sebutkan satu per satu.

KATA PENGANTAR



Puji syukur kami panjatkan kehadirat Allah SWT. Karena atas rahmat dan ridho-Nya lah penulis dapat menyelesaikan skripsi ini sebagai persyaratan dalam menyelesaikan studi S-1 pada Jurusan Teknik Informatika Fakultas Teknik Universitas Halu Oleo dengan judul **“PENERAPAN ALGORITMA A* PATHFINDING DALAM PEMBENTUKAN ARTIFICIAL INTELLIGENCE ENEMY PADA GAME GHOST ADVENTURE 3D (STUDI KASUS : PENCARIAN FAKULTAS UHO)”**.

Dalam penelitian dan penyusunan skripsi, penulis mendapatkan bantuan baik secara teknis maupun non teknis berupa bimbingan, arahan maupun bantuan lainnya dari berbagai pihak. Untuk itu, penulis ingin mengucapkan terima kasih kepada :

1. Bapak **Dr. Muhammad Zamrun Firihi, S.Si. M.Si., M.Sc.** selaku Rektor Universitas Halu Oleo.
2. Bapak **Dr. Edward Ngii, S.T., M.T.** selaku Dekan Fakultas Teknik Universitas Halu Oleo.
3. Bapak **Sutardi, S.Kom., M.T.** selaku Pembimbing 1 & Ketua Jurusan S-1 Teknik Informatika Fakultas Teknik Universitas Halu Oleo.
4. Bapak **Jumadil Nangi, S.Kom., M.T.** sebagai Pembimbing II yang telah banyak memberikan saran, petunjuk, ilmu pengetahuan, dukungan dan motivasi sehingga skripsi ini dapat terselesaikan.
5. Bapak **Bambang Pramono, S.Si., M.T.** selaku Dosen Penguji atas koreksi dan sarannya mulai dari ujian proposal, ujian seminar hasil dan ujian akhir serta bapak **L.M. Tajidun, S.T., M.Eng.** selaku Dosen Penguji atas koreksi dan sarannya saat seminar proposal hingga ujian skripsi dan bapak **Subardin, S.T., M.T.** selaku Dosen Penguji atas koreksi dan sarannya mulai dari ujian seminar hasil sampai ujian akhir (skripsi).
6. Dosen serta para staf Fakultas Teknik Universitas Halu Oleo, khususnya pada Jurusan Teknik Informatika atas bimbingan dan bantuannya.
7. Kedua orang tuaku tercinta **Sukman S** dan **Hasrawati** yang selalu memberikan dukungan, doa, dan kasih sayang tiada tara.

8. Saudaraku yang selalu memberikan saya semangat untuk mengerjakan skripsi ini.
9. Teman - teman Teknik Informatika angkatan 2015 yang telah memberikan semangat, motivasi, bantuan dan doanya.
10. Serta kakak – kakak angkatan 2014, 2013 dan adik – adik 2016, 2017 ,2018 dan 2019 yang telah memberikan bantuan dan doanya
11. Terima kasih kepada sodara Bahamut Cyber yang telah menemani saya selama ini dalam mengerjakan skripsi saya, dan terkhusus uti, edi, amran, acrul, upik, ilong, ashar dan fahro.
12. Terima kasih buat ifa dan aisyah yang banyak membantu juga dalam hal print
13. Terima kasih juga kepada saudara anan, hafid, joe, iksan, ilam, dan orang-orang yang sudah membantu mengajar mendesain game ku dengan sangat sabar bagaikan seorang guru yang bersayap.
14. Big thanks buat kak demon, orewa sensei :v
15. Terima kasih kepada **Waode Astuti Has** yang sudah menjadi motivasi, penyemangat, moodbooster, dan selalu menemani sampai saat ini, thanks very much yaw love you all.

Penulis memohon maaf atas segala kekurangan yang terdapat di dalam skripsi ini. Semoga penulisan skripsi ini bermanfaat untuk pengembangan ilmu pengetahuan dan untuk semua pihak yang bersangkutan.

Kendari, Januari 2020

Penulis,

Adnan Hidayat

DAFTAR ISI

HALAMAN JUDUL.....	i
HALAMAN PENGESAHAN	ii
HALAMAN PERNYATAAN.....	iv
INTISARI.....	v
ABSTRACT.....	vi
HALAMAN PERSEMBAHAN	vii
KATA PENGANTAR	viii
DAFTAR ISI	x
DAFTAR TABEL.....	xiii
DAFTAR GAMBAR	xiv
BAB I PENDAHULUAN.....	1
1.1 Latar Belakang	2
1.2 Rumusan Masalah	3
1.3 Batasan Masalah	3
1.4 Tujuan Penelitian	4
1.5 Manfaat Penelitian	4
1.7 Tinjauan Pustaka	5
BAB II LANDASAN TEORI	7
2.1 Sejarah Perkembangan <i>Game</i>	7
2.2 Konsep Dasar <i>Game</i>	7
2.3 <i>Game</i>	8
2.4 Karakteristik <i>Game</i>	10
2.4.1 <i>Rules</i> (Aturan-aturan)	10
2.4.2 <i>Victory condition</i> dan <i>lose condition</i>	10
2.4.3 <i>Seting</i> (letak)	11
2.4.4 <i>Interaction Model</i> (model interaksi).....	11
2.4.5 <i>Perspective</i> (sudut pandang)	11
2.4.6 <i>Role</i> (peran)	11
2.4.7 <i>Mode</i>	13
2.4.8 <i>Story</i> (Cerita)	13
2.5 <i>Type Game</i>	13

2.5.1	Game dengan informasi lengkap (<i>Perfect Information Game</i>)	13
2.5.2	Game dengan informasi tak lengkap (<i>Imperfect Information Game</i>)	14
2.6	Jenis <i>Game</i>	16
2.7	Game Edukasi	20
2.8	Algoritma A Star (<i>A*</i>)	20
2.9	Unity 3D.....	23
2.10	AI (<i>Artificial Intelligence</i>).....	24
2.11	Flowchart	24
2.12	Unified Modelling Language (UML).....	25
2.12.1	Use case Diagram	26
2.12.2	Activity Diagram	26
2.12.3	Sequence Diagram	27
BAB III METODOLOGI PENELITIAN		29
3.1	Waktu dan Tempat Penelitian	29
3.1.1	Waktu	29
3.1.2	Tempat Penelitian.....	29
3.2	Metode Pengumpulan Data	29
3.3	Metode Pengembangan Sistem	30
3.3.1	Inception	30
3.3.2	Elaboration.....	30
3.3.3	Construction	30
3.3.4	Transition	30
BAB IV ANALISI DAN PERANCANGAN SISTEM		31
4.1	Analisis Sistem yang Direncanakan.....	31
4.1.1	Perancangan Flowchart.....	31
4.1.1.1	Flowchart Sistem	31
4.1.1.2	Flowchart Algoritma A* (<i>All Star</i>)	32
4.1.2	Unified Modeling Language (UML)	33
4.1.2.1	Use Case Diagram	34
4.1.2.2	Activity Diagram	35
4.1.3.2	Sequence Diagram	39
4.3	Penerapan Algoritma All star (<i>A*</i>).....	41

4.3 Perancangan Antarmuka (<i>Interface</i>)	41
4.3.1 Tampilan <i>menu game</i>	42
4.3.2 Tampilan <i>gameplay</i>	43
4.3.3 Tampilan <i>form instruction</i>	44
4.3.4 Tampilan <i>Form Credit</i>	45
4.3.5 Tampilan <i>Form Exit</i>	46
4.3.6 Tampilan <i>Settings</i>	46
BAB V IMPLEMENTASI DAN PENGUJIAN SISTEM	48
5.1 Implementasi antarmuka sistem	48
5.1.1. Tampilan Menu Game	48
5.1.2. Tampilan <i>Play Game</i>	49
5.1.3. Tampilan <i>Instruction</i>	50
5.1.4. Tampilan <i>Credit</i>	51
5.1.5. Tampilan <i>Setting</i>	51
5.1.6. Tampilan <i>Exit</i>	52
5.1.7. Tampilan Game Over	52
5.1.8. Tampilan <i>Item</i> Informasi	53
5.1.9. Tampilan Musuh	59
5.1.10. Tampilan pertanyaan	59
5.1.11. Tampilan Pencarian Fakultas	60
5.1.12. Tampilan <i>Inventory</i>	61
5.2 Pengujian Sistem	63
5.2.1 Pengujian Algoritma A*	63
5.3 Implementasi <i>Artificial Intelligence</i> pada <i>enemy</i>	76
BAB VI PENUTUP	79
6.1 Kesimpulan	79
6.2 Saran	79
DAFTAR PUSTAKA	
LAMPIRAN	

DAFTAR TABEL

Tabel 2.1 Simbol-simbol dalam <i>flowchart</i>	25
Tabel 2.2 Simbol <i>use case diagram</i>	26
Tabel 2.3 Simbol activity diagram	27
Tabel 2.4 Simbol <i>sequence diagram</i>	28
Tabel 3.1 <i>Gant Chart</i> Waktu Penelitian	29
Tabel 4.2 Keterangan <i>Use Case Diagram</i>	35
Tabel 5.1 Hasil perhitungan rute pertama	68
Tabel 5.2 Hasil perhitungan rute kedua	71
Tabel 5.3 Hasil perhitungan rute ketiga	74

DAFTAR GAMBAR

Gambar 2.1 Perancangan <i>Game</i>	9
Gambar 2.2 Algoritma A* pada <i>grid</i>	22
Gambar 4.1 Flowchart Sistem	32
Gambar 4.2 <i>Flowchart</i> Algoritma A* pada <i>NPC (Non Player Character)</i>	33
Gambar 4.3 <i>Use Case Diagram</i> Aplikasi	34
Gambar 4.4 <i>Activity Diagram</i> <i>play game</i>	36
Gambar 4.5 <i>Activity Diagram</i> <i>Instruction</i>	37
Gambar 4.6 <i>Activity Diagram</i> <i>Credit</i>	38
Gambar 4.7 <i>Activity Diagram</i> <i>Settings</i>	39
Gambar 4.8 <i>Sequence Diagram</i> <i>Game</i>	40
Gambar 4.9 <i>Menu Game</i>	42
Gambar 4.10 Tampilan <i>gameplay</i>	43
Gambar 4.11 <i>Form Instruction</i>	44
Gambar 4.12 Tampilan <i>Form Credit</i>	45
Gambar 4.13 Tampilan <i>Form Exit</i>	46
Gambar 4.14 Tampilan <i>Form Settings</i>	47
Gambar 5.1 Tampilan <i>menu game</i>	48
Gambar 5.2 Tampilan <i>Play Game</i>	49
Gambar 5.3 Tampilan <i>Gameplay</i> pertama	49
Gambar 5.4 Tampilan <i>Gameplay</i> kedua.....	50
Gambar 5.5 Tampilan <i>Instruction</i>	50
Gambar 5.6 Tampilan <i>Credit</i>	51
Gambar 5.7 Tampilan <i>Setting</i>	51
Gambar 5.8 Tampilan <i>Exit</i>	52
Gambar 5.9 Tampilan <i>Game Over</i>	52
Gambar 5.10 Tampilan <i>item</i> informasi Daftar Fakultas.....	53
Gambar 5.11 Tampilan <i>item</i> informasi Pendiri UHO	54
Gambar 5.12 Tampilan <i>item</i> informasi Rektor UHO	55
Gambar 5.13 Tampilan <i>item</i> Sejarah UHO	56
Gambar 5.14 Tampilan <i>item</i> sejarah Fakultas Teknik.....	57
Gambar 5.15 Tampilan <i>item</i> sejarah Teknik Informatika	58

Gambar 5.16 Tampilan <i>body</i> musuh	59
Gambar 5.17 Tampilan pertanyaan	60
Gambar 5.18 Tampilan pencarian rute terpendek pada <i>gameplay</i>	60
Gambar 5.19 Tampilan pencarian rute terpendek pada <i>scene</i>	61
Gambar 5.20 Tampilan <i>Inventory</i>	61
Gambar 5.21 Tampilan salah satu informasi dari <i>inventory</i>	62
Gambar 5.22 Contoh <i>grid</i> 10x10 pada <i>game</i>	63
Gambar 5.23 Hasil perhitungan rute pertama	68
Gambar 5.24 Hasil perhitungan rute kedua.....	71
Gambar 5.25 Hasil perhitungan rute ketiga.....	74
Gambar 5.26 Node yang didapatkan sebagai rute terpendek	75

BAB I

PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi yang semakin maju dan peralatan teknologi yang pesat, tuntutan dalam berbagai aspek dalam kehidupan dirasakan semakin berat oleh banyak orang. Untuk menghindari stress yang berlebih membutuhkan waktu untuk beristirahat atau *refreshing* seperti berjalan-jalan, membaca buku, bermain, dll. Seiring berkembangnya teknologi, permainan yang kita mainkan semakin beragam. Dari permainan yang dulunya petak umpet, kucing-kucingan, bermain gasing dan lain-lain. Sekarang permainan sudah berubah menjadi permainan digital yang biasa disebut *game*. Bermain *games* menjadi sarana populer oleh mereka dalam melarikan diri dari beratnya beban dalam hidup ini.

Universitas Halu Oleo adalah perguruan tinggi negeri di Kendari, Sulawesi tenggara, Indonesia, yang berdiri pada 19 agustus 1981. Setelah dua tahun diresmikan, dimulailah pembangunan kampus Hijau Bumi Tridharma Anduonohu yang menempati areal 250 Ha, yang ketika itu berada di pinggiran Kota Kendari, berjarak 14 kilometer dari pelabuhan laut Teluk Kendari. Setelah perluasan Kota Kendari, kampus Anduonohu saat ini berada di jantung kota. Terdapat 14 fakultas yang ada di Universitas Halu Oleo, dengan banyaknya fakultas yang ada di Universitas Halu Oleo terkadang orang – orang atau mahasiswa baru masih bingung jika ingin menuju ke fakultas yang ingin dituju, terkadang bertanya kepada orang lain untuk menunjukkan jalan menuju ke fakultas dengan rute terdekat.

Permainan atau yang sering kita sebut dengan *game* adalah sesuatu kegiatan yang dilakukan untuk menghibur diri kita dari rasa jenuh. Seiring berkembangnya IPTEK pada era modern ini membuat banyak beredarnya berbagai jenis *games*. Tak hanya untuk sekedar *refreshing* namun aplikasi *games* telah menjadi sebuah bisnis industri yang sangat besar. *Games* yang dahulu sering dikaitkan dengan masa kanak-kanak ternyata telah menyedot perhatian remaja dan bahkan orang-orang dewasa sekarang ini.

Di zaman sekarang banyak sekali *game* yang di kembangkan oleh perusahaan-perusahaan ternama, tentu dengan kualitas grafis yang sangat nyata dengan

gameplay yang sangat menarik dan juga jalan cerita yang menarik pula. Tapi yang kurang dalam industri pembuatan *game* adalah perusahaan besar pengembang *game* tersebut jarang mengembangkan *game* yang bersifat edukasi yang tentunya sangat bermanfaat bagi pemainnya, selain menambah wawasan, *game* edukasi juga merupakan salah metode pembelajaran yang banyak di sukai anak-anak dan mudah diterima oleh anak-anak atau pemain lainnya.

Algoritma A^* (*All Star*) adalah algoritma pencarian rute terpendek yang merupakan perbaikan dari algoritma *Best First Search (BFS)* dengan memodifikasi fungsi heuristiknya. Seperti halnya pada *BFS*, untuk menemukan solusi, A^* juga dituntun oleh fungsi heuristik, yang menentukan urutan titik mana yang akan dikunjungi terlebih dahulu. Heuristik merupakan penilai yang memberi harga pada tiap titik yang memandu A^* mendapatkan solusi yang diinginkan. (Sunarto dan Krisdiawan, 2015).

Selain itu ada juga metode pencarian rute terpendek yang hampir mirip dengan A^* (*All Star*) yaitu *Dijkstra*. Algoritma *Dijkstra* dinamai sesuai dengan nama penemunya yaitu Edsger Dijkstra. Algoritma *Dijkstra* menggunakan konsep *greedy*, dimana pada setiap langkah dipilih sisi dengan bobot minimum yang menghubungkan sebuah simpul yang sudah terpilih dengan simpul lain yang belum terpilih. *Input* algoritma ini adalah sebuah graf berarah dan berbobot. (Pugas, 2011).

Algoritma *Greedy* merupakan algoritma yang memecahkan masalah langkah demi langkah, pada setiap langkah akan mengambil pilihan yang terbaik diperoleh pada saat itu, Berharap bahwa dengan memilih optimum lokal pada setiap langkah akan mencapai optimum global. Algoritma *greedy* mengasumsikan bahwa optimum lokal merupakan bagian dari optimum global.

Berdasarkan latar belakang tersebut, penulis mengangkat judul penerapan algoritma *A* pathfinding* dalam pembentukan *AI enemy* pada game *Ghost Adventure 3D* menggunakan *unity* studi kasus pencarian fakultas Universitas Halu Oleo.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan sebelumnya, dapat dirumuskan permasalahan dalam penelitian ini yaitu bagaimana membuat game dengan genre *adventure* yang dapat mengedukasi pemain, membuat *AI (Artificial intelligence)* pada *enemy* dengan menggunakan algoritma *A* pathfinding* dan pencarian rute terpendek untuk menuju fakultas Universitas Halu Oleo yang ingin dituju oleh pemain.

1.3 Batasan Masalah

Adapun masalah dalam penelitian ini yaitu :

1. *Game* ini hanya bisa dimainkan pada PC atau Laptop.
2. *Game* ini hanya memiliki satu level saja pada *gameplay* nya.
3. Musuh pada *game* ini akan mencari pemain dan akan menyerangnya.
4. Lokasi dalam *game* ini adalah Universitas Halu Oleo versi *developer*.
5. Pemain hanya bisa lari dari musuh dan apabila tertangkap maka game akan di ulang.
6. Informasi yang didapatkan harus dapat di ingat karena merupakan kunci utama di dalam *game* ini.
7. Pertanyaan akan muncul jika pemain sudah mengumpulkan enam kertas yang berisi informasi.
8. Ketika pemain berhasil menjawab pertanyaan nya, maka pintu keluarnya akan terbuka, pemain harus mencari pintu keluar tersebut dan ketika salah maka *game* akan di ulang.
9. Musuh yang terdapat di dalam *game* ini ada 3.
10. Algoritma *A** digunakan sebagai pencarian fakultas dan pada musuh.
11. Rute yang digunakan pada game ini hanya jalan raya.

1.4 Tujuan Penelitian

Tujuan dari penelitian ini yaitu bagaimana cara membuat *AI enemy* pada *game* dengan menggunakan algoritma pencarian rute terpendek, membuat *game adventure* yang mengedukasi dan membuat pencarian rute terpendek untuk menuju ke tempat yang ingin dituju.

1.5 Manfaat Penelitian

Manfaat dari penelitian ini yaitu algoritma A^* diharapkan menciptakan sifat dari musuh yang dapat mengejar pemain pada *game* ini agar dapat menambah keseruan dalam bermain *game*, membuat pemain dapat mencari rute terpendek untuk menuju ke suatu tempat dalam *game* ini dan *game* ini berisi informasi tentang Teknik Informatika Universitas Halu Oleo.

1.6 Sistematika Penulisan

BAB I PENDAHULUAN

Bab ini berisi tentang latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, sistematika penulisan dan tinjauan pustaka.

BAB II LANDASAN TEORI

Bab ini berisi tentang sejarah perkembangan *game*, konsep dasar *game*, pengertian *game*, karakteristik *game*, *type game*, jenis *game*, pengertian *game* edukasi, algoritma A^* , penjelasan unity 3D, pengertian *AI* dan *flowchart*.

BAB III METODOLOGI PENELITIAN

Bab ini memuat metodologi penelitian yang meliputi metode pengumpulan data, uraian metode *Rational Unified Process* (RUP) untuk pengembangan sistem dan uraian waktu penelitian.

BAB IV ANALISIS DAN PERANCANGAN SISTEM

Bab ini memuat analisis serta rancangan sistem yang akan dibuat. Adapun rancangan sistem meliputi rancangan rancangan proses, rancangan UML dan rancangan antarmuka sistem.

BAB V IMPLEMENTASI DAN PENGUJIAN

Bab ini menyajikan implementasi dan pengujian dari sistem yang dibangun. Bagian implementasi menguraikan tentang implementasi secara detail dan runtut dari sistem yang dibangun berdasarkan hasil analisis dan rancangan pada bab sebelumnya sedangkan bagian pengujian menguraikan pengujian sistem serta pembahasan hasil pengujian sistem tersebut.

BAB VI PENUTUP

Bab ini memuat kesimpulan dari penelitian yang telah dilakukan serta saran untuk penelitian selanjutnya.

1.7 Tinjauan Pustaka

Penelitian ini dibuat berdasarkan penelitian yang dilakukan oleh Andy Pramono (2015) yang meneliti “**Algortima Pathfinding A* pada Game RPG Tanaman Higienis**”. Sistem ini dirancang untuk permainan Tanaman higienis dimana karakter di dalam game tersebut ketika ingin menuju ke suatu tempat maka akan di carikan rute terdekat nya dengan menggunakan algoritma A* dengan tampilan 2D.

Penelitian ini dibuat berdasarkan penelitian yang dilakukan oleh Ahmad Soleh Siregar (2017) yang meneliti “**Penerapan Algoritma A* Untuk Pencarian Solusi Terbaik Pada Game Congklak**”. *Game* ini dirancang untuk menentukan solusi terbaik untuk komputer dan komputer harus otomatis menghitung kemungkinan untuk menang.

Penelitian ini dibuat berdasarkan penelitian yang dilakukan oleh Latius Hermawan dan R. Kristoforus Jawa Bendi (2016) yang meneliti “**Penerapan Algoritma A* pada Aplikasi Puzzle**”. Pada puzzle ini algoritma A* berperan sebagai cara cepat untuk menyelesaikan puzzle 3x3 yang berisi 8 ubin dimana 1 ubin kosong berguna sebagai penggeser ubin lain nya, angka pada ubin tersebut bersifat random dan pemain harus menyusun ubin tersebut seperti semula menjadi angka yang berurutan, dengan menghitung nilai heuristic dari ubin tersebut dalam mencari solusi yang benar.

Penelitian ini dibuat berdasarkan penelitian yang dilakukan oleh Yenie Syukriah, Falahah dan Hermi Solihin (2016) yang meneliti “**Penerapan Algoritma A^* (*Star*) untuk Mencari Rute Tercepat dengan Hambatan**”. Pada penerapan algoritma A^* di aplikasi ini hanya diterapkan pada 2 dimensi saja dengan 3 jalur sebagai *alternative*, setiap jalur akan di hitung menggunakan algoritma A^* , dan akan didapatkan jalur terpendek.

BAB II

LANDASAN TEORI

2.1 Sejarah Perkembangan *Game*

Game merupakan kata dalam bahasa Inggris yang berarti permainan. Permainan adalah sesuatu yang dapat dimainkan dengan aturan tertentu sehingga ada yang menang dan ada yang kalah, biasanya dalam konteks tidak serius atau dengan tujuan *refreshing*. Dalam penggunaannya kata *game* sering digunakan untuk menyebutkan *video game*. *Video game* adalah *game* yang berbasis elektronik dan *visual*. *Video game* dimainkan dengan memanfaatkan media *visual* elektronik (Susilawati, 2014).

Pada tahun 1952, seorang mahasiswa Universitas Cambridge bernama A.S. Gouglas membuat permainan OXO (tic-tac-toe) dalam versi grafik. Permainan ini di kembangkan ketika hendak mendemonstrasikan tesisnya tentang interaksi antara manusia dan komputer. Pada tahun 1958 William Higin Botham mendesain sebuah *game* dengan judul *Tennis For Two* yang dimainkan dalam *oscilloscope*, dan kemudian ada pula Steve Russel pada tahun 1961 dengan *game* berjudul *Spacewar* yang dibuat dalam komputer mainframe DEC PDP-1 saat mereka menjalani studi di MIT (Syuhada, 2014).

2.2 Konsep Dasar *Game*

Game adalah sebuah permainan komputer interaktif yang di kendalikan mikroprosesor. Komputer dapat menciptakan bahan-bahan maya untuk digunakan dalam sebuah permainan seperti kartu dan dadu. Sebuah permainan komputer atau *video game* menggunakan satu atau lebih alat *input*, biasanya sebuah tombol atau kombinasi dari *joystick*, sebuah *keyboard* dan *mouse* dan *trackball* atau sebuah *controller* ataupun sebuah alat yang mempunyai sensor gerak. Sebuah *video game* adalah permainan yang biasanya melibatkan *player* berinteraksi dengan alat pengendali atau *controller* untuk menghasilkan umpan balik secara *visual* dalam sebuah layar video (Susilawati, 2014).

2.3 Game

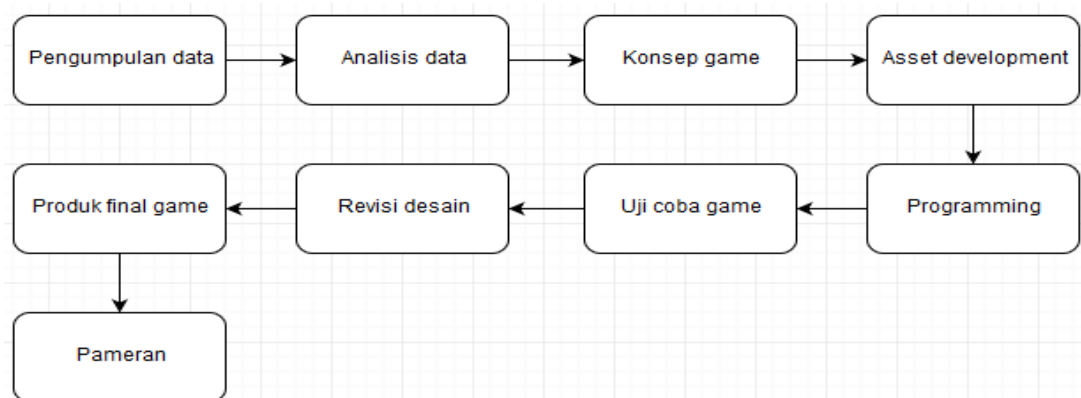
Game merupakan kata dalam Bahasa Inggris yang memiliki arti permainan. Pada dasarnya *game* merupakan sebuah media hiburan yang bertujuan untuk mengisi waktu luang dan menghilangkan rasa jenuh orang yang memainkannya. *Game* itu sendiri di dalamnya memiliki suatu aturan, alur atau tujuan tertentu untuk menentukan kondisi menang dan kalah dari *game* tersebut. *Game* saat ini dapat dimainkan oleh *single player* dimana hanya satu orang yang bermain pada arena permainan tersebut ataupun *multi player* dimana dalam satu arena permainan dan waktu yang sama terdapat lebih dari satu pemain. *Game multi player* memungkinkan pemain untuk berinteraksi dengan pemain lainnya baik itu dalam bentuk bekerja sama dalam tim yang sama atau menjadi lawan tanding, hingga bentuk komunikasi sosial yang hampir tidak ditemukan pada *game* dengan orientasi *single player* (Ayu., dkk., 2017)

Game adalah sebuah sistem yang terencana dan secara subyektif yang menggambarkan bagian dari kenyataan. Serangkaian bagian yang berinteraksi satu sama lain. Menciptakan gambaran kehidupan emosional yang subyektif dan mudah untuk dimainkan oleh para pemain. Sebuah *game* harus memiliki elemen-elemen dasar, diantaranya:

- a) Elemen *visual*, meliputi karakter tokoh, yang terbagi menjadi *Player Character*, *Non-player Character (NPC)*, *Enemies*; *Environment*; *Icon*; *Interface* (antar muka).
- b) Elemen audio, meliputi *Background Music (BGM)* yaitu suara latar dan *Sound Effect (SFX)*.
- c) *Gameplay*, meliputi sistem permainan yang mengatur alur cerita serta serangkaian program dan seperangkat aturan yang mendasari dinamika *game* tersebut.

Game dapat diklasifikasikan berdasarkan *gameplay* (cara bermain *game*) menjadi *board game*, *skill and Action Game*, *simulation*, *Real Time Strategy*, *Role Playing Game*.

Media interaktif kini banyak menjamur tidak hanya seluruh dunia, ini akibat kecanggihan teknologi informasi yang kian mempercantik diri untuk memenuhi kebutuhan konsumen. Baik dari *company profile* hingga media pembelajaran untuk sekolah dan perguruan tinggi. Menurut Tay Vaughan (2004:3) multimedia adalah kombinasi dari teks, foto, seni grafis, suara, animasi, dan elemen-elemen video yang dimanipulasi secara digital. Menurut Sutopo (2002:251) multimedia berhubungan dengan perangkat lunak yang digunakan dalam pengembangan dengan lebih dari satu cara untuk menyampaikan informasi kepada pengguna. Vektor adalah besaran yang memiliki nilai dan arah. Kecepatan, percepatan, gaya, tekanan, momentum dan sebagainya adalah contoh-contoh besaran vektor. Penulisan vektor dengan vektor satuan mempermudah pengertian tentang arah vektor itu. Beberapa vektor dapat dijumlahkan maupun dikalikan. Meliputi vektor dua dimensi dan vektor tiga dimensi. Menurut Murbandono (2002:3) mengartikan bahwa multimedia adalah media massa berdasar computer. Hal ini lah yang menjadi karakter khas dan muatan filosofis dari multimedia di bandingkan dengan semua model media massa lainnya. Secara filosofis dapat disimpulkan multimedia adalah *medium* berdasar *computer* yang lebih mengedepankan eksplorasi berkarakter riset dibandingkan *performance* teknologi demi bisnis (Pramono, 2015).



Gambar 2.1 Perancangan Game

2.4 Karakteristik *Game*

Didalam sebuah *game* terdapat beberapa elemen-elemen yang akan membentuk sebuah *game* itu sendiri yaitu:

2.4.1 *Rules* (Aturan-aturan)

Sebuah *game* mengambil tempat atau *setting* di dunia buatan yang di atur oleh aturan-aturan (*rules*). *Rules* istilah yang menentukan aksi dan gerakan pemain dalam sebuah *game*. Pada *game* komputer, kebanyakan *rules* ini tersembunyi. Karena pemain berinteraksi dengan *game* hanya melalui alat masukan atau yang biasanya disebut dengan *input device* dan mesin mengabaikan *input* yang tidak sesuai tanpa harus memberitahu aturan kepada pemain.

Beberapa *game* besar juga membuat aturan didalam game buatannya, aturan ini bertujuan untuk membatasi pemain agar tidak keluar dari lingkup permainan, aturan yang di buat tentu nya tidak mengurangi nilai dari *game* tersebut, misalnya aturan yang dibuat berupa pemain tidak bisa membunuh *NPC* (*non player character*) atau jika pemain membunuh *NPC* maka permainan akan selesai dan harus mengulangi nya lagi.

2.4.2 *Victory condition* (kondisi menang) dan *lose condition* (kondisi kalah)

Dalam sebuah *game* harus ada yang di capai oleh karena itu ada kondisi kemenangan, maka *game* juga memiliki kondisi kekalahan yang dapat membuat pemain merasa senang dan juga mendapat tantangan dari *game* yang dimainkan.

Game yang tidak memiliki elemen ini terkesan sangat membosankan dan tidak ada nilai hiburan yang terdapat didalamnya, kebanyakan pemain ingin mencapai suatu kemenangan dalam sebuah *game*, tentunya untuk mencapai kepuasan dalam bermain *game* dan ditambah dengan adanya tantangan yang membuat para pemain merasa sangat ingin untuk menang.

2.4.3 Seting (letak)

Sebuah *game* mengambil tempat pada suatu dunia atau tempat tertentu. Contohnya pada permainan bola yang berseting pada lapangan dengan Batasan-batasan tertentu.

Pembuatan suatu lokasi didalam *game* tentunya sangat berpengaruh dengan pengalaman pemain dalam menjelajahi dunia yang dibuat oleh pihak pengembang *game*, mulai dari lokasi yang ada di dunia nyata yang dibuat didalam *game* dan ada juga dunia yang bersifat *fantasy* atau dunia yang dibuat berdasarkan imajinasi pengembangnya.

2.4.4 Interaction Model (model interaksi)

Cara pemain berinteraksi dengan *game* dan melakukan aksi untuk menghadapi tantangan dari *game* tersebut. Model interaksi yang biasa di pakai pada *game* komputer ada jenis. Yang pertama, jika pemain mengendalikan tokoh tunggal yang mempresentasikan dirinya dalam *game* dan tokoh tersebut dapat mempengaruhi dunia sekitarnya, maka tokoh tersebut dapat disebut sebagai *avatar*-nya. Sedangkan yang kedua adalah jika pemain memiliki kemampuan untuk melihat bagian dari dunia *game* dan melakukan aksi pada banyak tempat maka pemain tersebut sebagai *omnipresent*. Namun hal ini dapat diterapkan pada tokoh atau unit yang menjadi miliknya, dengan memberikan perintah kepada mereka satu persatu. Salah contoh *game* yang termasuk jenis ini adalah catur.

2.4.5 Perspective (sudut pandang)

Perspective Menjelaskan bagaimana pemain melihat *game world* dari suatu *game* pada layer permainan, untuk *game* zaman sekarang kebanyakan menggunakan sudut pandang orang pertama dan sudut pandang orang ketiga, tentunya ada perbedaan dari sudut pandang tersebut, dan juga pemain tentunya akan terasa nyaman jika bermain *game* dengan konsep sudut pandang yang sesuai dengan kenyamanan mereka.

Dalam sebuah *game* biasanya sudut pandang karakter akan berpengaruh pada jalan cerita dan tingkat keseruan yang didapat oleh para pemain, untuk *game* yang bertipe *horror* lebih bagus menggunakan sudut pandang orang pertama karena sensasi *horror* nya akan terasa karena seolah-olah *player* memainkan karakter didalam *game* terasa seperti memasuki dunia *game* tersebut. Untuk *game* yang ber *genre adventure* lebih bagus menggunakan sudut pandang orang ketiga karena dapat melihat karakternya dengan lebih luas.

2.4.6 Role (peran)

Tokoh yang dimainkan oleh pemain dalam suatu *game*. Dengan adanya peran ini maka pemain akan lebih mudah untuk memahami apa yang sebenarnya ingin dicapai dan aturan apa yang dimainkan. Sebagai contoh, pada permainan monopoli pemain berperan sebagai pialang perumahan. Pada *game championship* manager pemain berperan sebagai manager sebuah tim sepak bola.

Setiap karakter memiliki peran nya masing-masing, didalam *game* selalu ada karakter dengan peran protagonis dan antagonis, peran ini sangat penting untuk keseruan didalam *game* dimana pemain biasanya akan mendapat peran sebagai karakter protagonis dan akan mengalahkan karakter yang memiliki peran antagonis.

2.4.7 Mode

Beberapa *game*, seperti catur, berlaku sama dari awal sampai akhir. Pemain selalu ingin mencapai atau menyelesaikan hal yang sama dengan cara yang berbeda pula. Namun ada juga *game* yang memiliki mode nyata, yaitu dimana *gameplay* nya berubah dari satu mode ke mode lainnya. Contoh nya seperti pada *game* perang, yaitu sebelum pemain turun ke medan perang biasanya pemain di beri *briefing* terlebih dahulu, lalu pindah ke mode pemilihan senjata, dan terakhir adalah perang itu sendiri.

Dalam sebuah *game* banyak mode yang biasanya di sediakan oleh pihak pengembang, mode tersebut biasanya merupakan tingkat kesulitan, tingkat

kesulitan pun merupakan sebuah mode yang dapat mengubah alur dalam *game* yang dimainkan, contoh nya mode *easy*, *normal* dan *hard*.

2.4.8 Story (Cerita)

Game computer merupakan perpaduan antara media pasif, pasif seperti televisi dan film dan media aktif, *non* pasif seperti permainan *poker* dan domino. Beberapa *game* komputer, seperti tetris tidaklah mempunyai cerita. Lain halnya dengan seri *Metal Gear Solid*, dan *Final Fantasy*. Beberapa *game* memiliki alur cerita yang linear atau hanya satu jalan cerita dan bersifat *non* interaktif, namun ada juga yang bersifat interaktif dimana cerita akan berbeda atau bercabang tergantung pada pilihan atau tindakan yang diambil pemain. Sehingga cerita akhir (*ending*) yang didapat akan berbeda- beda pula, atau biasanya disebut dengan *multiple ending*.

Penentuan cerita yang menarik dalam sebuah *game* akan menarik minat pemain, cerita yang menarik biasanya memiliki seri yang berkepanjangan atau memiliki season pada setiap ceritanya yang sangat membuat pemain penasaran dengan lanjutan ceritanya, namun *game* dengan alur cerita yang biasanya cuman sekali dimainkan tidak menutup kemungkinan cerita tersebut menarik, jadi biasanya penulis cerita biasa membuat cerita apa yang sesuai imajinasi atau keadaan yang benar-benar terjadi (Zaka, 2014).

2.5 Type Game

Menurut T. Sutojo, Edy Mulyanto, Vincent Suhartono (2010:436), berdasarkan tipe nya, *game* dibagi menjadi dua, yaitu *game* dengan informasi lengkap dan *game* dengan informasi tak lengkap.

2.5.1 Game dengan informasi lengkap (Perfect Information Game)

Game dengan informasi lengkap adalah suatu *game* di mana pemain mengetahui semua langkah yang mungkin terjadi dari dirinya sendiri dan dari lawan dan hasil akhir dari permainan mereka. Contoh *game* yang termasuk dalam tipe ini adalah catur dan *tic tac toe*.

2.5.2 *Game* dengan informasi tak lengkap (*Imperfect Information Game*)

Game ini adalah *game* di mana pemain tidak tahu semua kemungkinan langkah kawan. Contoh yang termasuk *game* ini adalah permainan kartu poker dan *brigde* , karena tidak semua kartu diketahui oleh para pemain.

2.6 Jenis *Game*

Menurut Ivan C. Sibero dalam bukunya “Langkah Mudah Membuat *Game 3D*” (2009) bahwa *game* atau permainan dapat diklasifikasikan menjadi dua bagian besar yaitu *game* fisik dan *game* elektronik. *Game* fisik mungkin sudah sering dilakukan dalam kehidupan sehari-hari sewaktu masih anak-anak, seperti lompat tali, petak umpet, dan sebagainya. Permainan disini adalah berhubungan dengan gerak fisik. *Game* elektronik selanjutnya dapat disebut sebagai video *game* yang pertama kali ditemukan oleh Thomas T. Goldsmith Jr dan Estle Ray Mann yang mana penemuan ini dipatenkan pada Januari tahun 1947 (Arief, 2014).

Menurut Andi Taru Nugroho NW (2012: 14), video *game* secara umum dikategori - kategorikan kedalam *genre*. Karena kurangnya penyetujuan syarat atas *genre* atau kriteria untuk definisi sebuah *genre*, klasifikasi dari *game* menjadi tidak konsisten atau sistematis. Di bawah ini merupakan daftar *genre-genre* yang sering dimainkan orang. Banyak dari kategori *game* dibawah ini bertabrakan satu sama lainnya, contohnya pada *game Legend of Zelda* yang mempunyai elemen *action*, *adventure*, dan *role-playing* (Susilawati, 2014).

Berdasarkan platformnya *game* dapat dibagi menjadi beberapa jenis, diantaranya adalah:

1. *Arcade game*, yaitu jenis *game* yang biasa di sebut dengan “*Ding- Dong*”. *Game* jenis ini juga memiliki banyak jenisnya, seperti halnya yang biasa ditemui di *Time Zone*.
2. *PC games*, yaitu jenis *game* yang hanya bisa dimainkan di dalam PC atau komputer yang biasanya di *install* dahulu.

3. *Console games*, yaitu jenis *game* yang dimainkan di alat atau *console* tertentu, seperti *PlayStation*, *XBOX*, *Wii*, *Sega* dll
4. *Handled games*, yaitu jenis *game* yang bisa dimainkan di *console* khusus *game* yang bisa di bawa kemana-mana, seperti *PSP*, *Tamagochi*, *nitendo DS* dll
5. *Mobile games*, yaitu jenis *game* yang bisa dimainkan di *mobile phone*, *smartphone*, atau *PDA*

Selain jenis platformnya *game* juga dibagi menjadi beberapa *type* atau *genre*, yang terdiri dari :

1. *Action Shooting*, (tembak-tembakan, atau hajar-hajaran bisa juga *hack and slash*, tergantung cerita dan tokoh di dalamnya), *video game* jenis ini sangat memerlukan kecepatan refleks, koordinasi mata-tangan, juga *timing*, inti dari *game* jenis ini adalah tembak, tembak dan tembak. Termasuk didalamnya:
 - a. *First person shooting (FPS)* seperti *Counter Strike* dan *Call of Duty*.
 - b. *Drive and shoot*, menggunakan unsur simulasi kendaraan tetapi tetap dengan tujuan utama menembak dan menghancurkan lawan, contoh: *Spy Hunter*, *Rock and Roll Racing*, *Road Rash*.
 - c. *Shoot up*, seperti *Raiden*, *1942*, dan *Gradius*.
 - d. *Beat up* (tonjok hajar) seperti *Double Dragon* dan *Final Fight*, lalu *hack and slash* (tusuk tebas) seperti *Shinobi* dan *Legend of Kage*.
 - e. *Light gun shooting*, yang menggunakan alat yang umumnya berbentuk seperti senjata, seperti *Virtual Cop* dan *Time Crisis*.
2. *Fighting* (pertarungan) Ada yang mengelompokan *video game fighting* di bagian Aksi, namun penulis berpendapat berbeda, jenis ini memang memerlukan kecepatan refleks dan koordinasi mata-tangan, tetapi inti dari *game* ini adalah penguasaan jurus (hafal caranya dan lancar mengeksekusinya), pengenalan karakter dan *timing* sangatlah penting, *combo* menjadi esensial untuk mengalahkan lawan secepat mungkin. dan berbeda seperti *game* aksi pada umumnya yang umumnya hanya melawan

Artificial Intelligence atau istilah umumnya melawan komputer saja, pemain jenis *fighting game* ini baru teruji kemampuan sesungguhnya dengan melawan pemain lainnya. Contohnya *Street Fighter*, *Tekken*, *Mortal Kombat*, *Soul Calibur* dan *King of Fighter*.

3. *Action – Adventure*. Memasuki gua bawah tanah, melompati bebatuan di antara lahar, bergelayutan dari pohon satu ke pohon lainnya, bergulat dengan ular sambil mencari kunci untuk membuka pintu kuil legendaris, atau sekedar mencari telepon umum untuk mendapatkan misi berikutnya, itulah beberapa dari banyak hal yang karakter pemain harus lakukan dan lalui dalam video game jenis ini. Menurut penulis, game jenis ini sudah berkembang jauh hingga menjadi *genre* campuran *action beat-em up* juga, dan sekarang, di tahun 2000 an, jenis ini cenderung untuk memiliki *visual 3D* dan sudut pandang orang ke-tiga. *Tomb Rider*, *Grand Theft Auto* dan *Prince of Persia* termasuk didalamnya.
4. *Adventure*. Bedanya dengan jenis video game *action-adventure*, refleks dan kelihaian pemain dalam bergerak, berlari, melompat hingga memecut atau menembak tidak diperlukan di sini. Video game murni petualangan lebih menekankan pada jalan cerita dan kemampuan berpikir pemain dalam menganalisa tempat secara visual, memecahkan teka-teki maupun menyimpulkan rangkaian peristiwa dan percakapan karakter hingga penggunaan benda-benda tepat pada tempat yang tepat.
5. Simulasi, Konstruksi dan manajemen. Video game jenis ini seringkali menggambarkan dunia di dalamnya sedekat mungkin dengan dunia nyata dan memperhatikan dengan detil berbagai faktor. Dari mencari jodoh dan pekerjaan, membangun rumah, gedung hingga kota, mengatur pajak dan dana kota hingga keputusan memecat atau menambah karyawan. Dunia kehidupan rumah tangga sampai bisnis membangun konglomerasi, dari jualan limun pinggir jalan hingga membangun laboratorium *cloning*. Video game jenis ini membuat pemain harus berpikir untuk mendirikan, membangun dan mengatasi masalah dengan menggunakan dana yang terbatas. Contoh: *Sim City*, *The Sims*, *Tamagotchi*.

6. *Role Playing*. Video *game* jenis ini sesuai dengan terjemahannya, bermain peran, memiliki penekanan pada tokoh/peran perwakilan pemain di dalam permainan, yang biasanya adalah tokoh utamanya, dimana sering dimainkan, karakter tersebut dapat berubah dan berkembang ke arah yang diinginkan pemain (biasanya menjadi semakin hebat, semakin kuat, semakin berpengaruh, dll) dalam berbagai parameter yang biasanya ditentukan dengan naiknya level, baik dari status kepintaran, kecepatan dan kekuatan karakter, senjata yang semakin sakti, ataupun jumlah teman maupun makhluk peliharaan. Secara kebudayaan, pengembang *game* Jepang biasanya membuat *Role Playing Game (RPG)* ke arah cerita linear yang diarahkan seolah karakternya adalah tokoh dalam cerita itu, seperti *Final Fantasy*, *Dragon Quest* dan *Xenogears*. Sedangkan pengembang *game RPG* Eropa, cenderung membuat karakter bebas memilih jalan cerita sendiri secara non-linear, seperti *Ultima*, *Never Winter Nights*, *Baldur's Gate*, *Elder Scroll*, dan *Fallout*.
2. 7. Strategi. Kebalikan dari video *game* jenis action yang berjalan cepat dan perlu refleksi secepat kilat, video *game* jenis strategi, layaknya bermain catur, justru lebih memerlukan keahlian berpikir dan memutuskan setiap gerakan secara hati-hati dan terencana. Video *game* strategi biasanya memberikan pemain atas kendali tidak hanya satu orang tapi minimal sekelompok orang dengan berbagai jenis tipe kemampuan, sampai kendaraan, bahkan hingga pembangunan berbagai bangunan, pabrik dan pusat pelatihan tempur, tergantung dari tema ceritanya. Pemain *game* strategi melihat dari sudut pandang lebih meluas dan lebih kedepan dengan waktu permainan yang biasanya lebih lama dan santai dibandingkan *game action*. Unsur-unsur permainannya biasanya berkisar sekitar, prioritas pembangunan, peletakan pasukan, mencari dan memanfaatkan sumberdaya (uang, besi, kayu, minyak, dll), hingga ke pembelian dan *upgrade* pasukan atau teknologi. *Game* jenis ini terbagi atas:
 - a. *Real time Strategy*, *game* berjalan dalam waktu sebenarnya dan serentak antara semua pihak dan pemain harus memutuskan setiap

langkah yang diambil saat itu juga berbarengan mungkin saat itu pihak lawan juga sedang mengeksekusi strateginya. Contoh: *Starcraft*, *Warcraft*, dan *Command and Conquer*.

- b. *Turn based Strategy*, *game* yang berjalan secara bergiliran, saat mengambil keputusan dan menggerakkan pasukan, saat itu pihak lawan menunggu, begitu pula sebaliknya, layaknya catur.

Sebenarnya ada yang memilah lagi menjadi jenis *tactical* dan *strategi*, namun penulis cenderung untuk menggabungkannya karena perbedaannya hanya ada di masalah skala dan kekompleksan dalam manajemen sumber daya-nya saja.

8. *Puzzle*. Video *game* jenis ini sesuai namanya berintikan mengenai pemecahan teka-teki, baik itu menyusun balok, menyamakan warna bola, memecahkan perhitungan matematika, melewati labirin, sampai mendorong-dorong kota masuk ke tempat yang seharusnya, itu semua termasuk dalam jenis ini. Sering pula permainan jenis ini adalah juga unsur permainan dalam video *game* petualangan maupun *game* edukasi. Contohnya *Tetris*, *Minesweeper*, *Bejeweled*, *Sokoban* dan *Bomberman*.
9. Simulasi kendaraan. Video *game* jenis ini memberikan pengalaman atau interaktifitas sedekat mungkin dengan kendaraan yang aslinya, meskipun terkadang kendaraan tersebut masih eksperimen atau bahkan fiktif, tapi ada penekanan khusus pada detail dan pengalaman *realistic* menggunakan kendaraan tersebut. Terbagi atas beberapa jenis:
 - a. Perang. Video *game* simulasi kendaraan yang sempat tenar di tahun 90-an ini mengajak pemain untuk menaiki kendaraan dan berperang melawan kendaraan lainnya. Dan kebanyakan diantaranya memiliki

- b. judul sama dengan nama kendaraannya. Contoh : *Apache 64*, *Comanche*, *Abrams*, *YF-23*, *F-16 fighting eagle*. Tetapi *game* kehidupan bajak laut pun dapat dikategorikan disini.
 - c. Balapan. Dari namanya sudah jelas, siapa sampai duluan di garis finish dialah pemenangnya. Terkadang malah pemain dapat memilih kendaraan, mendandani, *upgrade* mesin bahkan mengecatnya. Contoh: *Top Gear*, *Test Drive*, *Sega Rally Championship*, *Daytona*, *Grand Turismo*, *Need For Speed*, *Mario Kart*, *ManXTT*.
 - d. Luar Angkasa. Walau masih dapat dikategorikan simulasi kendaraan perang, tetapi segala unsur fiksi ilmiah dan banyaknya judul yang beredar membuat *subgenre* ini pantas dikategorikan diluar simulasi kendaraan perang. Jenis ini memungkinkan pemain untuk menjelajah luar angkasa, berperang dengan makhluk alien, mendarat di planet antah berantah atau sekedar ingin merasakan bagaimana menjadi kapten di film fiksi ilmiah kesayangan kamu. Contoh: *Wing Commander*, *Freelancer*, *Star Wars X-Wing*, *Star Wars Tie Fighter*.
 - e. Mecha. Pendapat bahwa hampir tidak ada orang yang terekspos oleh film robot jepang saat kecilnya tidak memimpikan ingin mengendalikan robot, memang sulit dibantah. Dipopulerkan oleh serial *Mechwarrior* oleh *Activision*, *subgenre* simulasi *mecha* ini memungkinkan pemainnya untuk mengendalikan robot dan menggunakannya untuk menghancurkan gedung, helikopter dan tentu saja robot lainnya. Contoh: *Mechwarrior*, *Gundam Last war Chronicles*, dan *Armored Core*.
10. Olahraga. Singkat padat jelas, bermain *sport* di PC atau konsol anda. Biasanya permainannya diusahakan serealistik mungkin walau kadang ada yang menambah unsur fiksi seperti *NBA JAM*. Contohnya pun jelas, Seri

Winning Eleven, seri *NBA*, seri *FIFA*, John Madden NFL, Lakers vs Celtics, *Tony hawk pro skater* (Zaka, 2014).

2.7 Game Edukasi

Game Edukasi adalah *game* yang dirancang untuk pendidikan dengan cara menyisipkan materi-materi pembelajaran tertentu pada permainan sehingga *user* atau Pemain tidak tertekan dengan belajar terlalu serius. *Game* edukasi adalah *game* yang khusus dirancang untuk mengajarkan *user* suatu pembelajaran tertentu, pengembangan konsep dan pemahaman dan membimbing mereka dalam melatih kemampuan mereka, serta memotivasi mereka untuk memainkannya.

Game edukasi mengajarkan anak-anak atau pun orang dewasa beberapa bentuk pembelajaran dengan cepat dan mudah diserap. *Game* jenis ini mengajarkan anak - anak ataupun orang dewasa beberapa bentuk pembelajaran dengan cepat dan mudah diserap, sementara pembelajaran di sekolah konvensional dirasa lambat dan cepat membosankan (Marwan., dkk., 2015).

2.8 Algoritma A Star (A^*)

Algoritma A^* (A star) dikenal sebagai salah satu algoritma yang paling sering digunakan untuk pencarian jalur (*path finding*) dan penerusan grafis (*graph traversal*), yaitu proses *plotting* jalur yang paling efisien antar titik, yang disebut dengan *nodes*.

Algoritma ini menggunakan fungsi *distance – plus – cost* (biasanya dinotasikan dengan $F(n)$) untuk menentukan urutan kunjungan pencarian *node* di dalam *tree*. Gabungan jarak – plus – biaya merupakan penjumlahan dari dua fungsi, yaitu fungsi *path – cost* (selalu dinotasikan dengan $G(n)$, dimungkinkan bernilai heuristik ataupun tidak), dan sebuah kemungkinan penerimaan atas “perkiraan heuristik” jarak ke titik tujuan (dinotasikan dengan $H(n)$). Fungsi *path - cost* $G(n)$ adalah jumlah biaya yang harus dikeluarkan dari *node* awal menuju *node* tujuan. Beberapa terminologi dasar yang terdapat pada algoritma ini adalah *starting node*, Target *node*, simpul (*nodes*), A^* , *open list*, *closed list*, harga (*cost*), halangan (*unwalkable*).

Algoritma ini akan mencari jalur terpendek dari sebuah node awal menuju node tujuan dengan memperhatikan nilai $F(n)$ terkecil. Tahapan algoritma A^* yaitu:

1. Menentukan *node* awal dan *node* tujuan
2. membuat *openset* untuk menyimpan nilai node yang sedang dihitung, dan *closedset* untuk menyimpan nilai node yang sudah dihitung.
3. menjadikan node awal sebagai *current node*, kemudian masukkan *current node* ke *openset*.
4. mengecek apakah *current node* sama dengan *target node*.
5. jika sama maka pencarian dihentikan, rute ditemukan, jika tidak maka *node neighbour* dari *current node* akan dihitung.
6. keluarkan *current node* dari *openset*, masukkan ke *closedset*.
7. mengecek *openlist* jika tidak mengandung *node neighbour*, maka *node neighbour* akan di masukkan kedalam *openlist*.
8. mendapatkan nilai $G(n)$ dan $H(n)$ menggunakan rumus *manhattan distance*.
9. mengecek semua nilai *node neighbour* yang ada di *openlist* dengan *current node*, dan mengambil nilai $F(n)$ terkecil. *Node* dengan nilai $F(n)$ terkecil akan menjadi *current node* berikutnya.
10. mengulangi proses sampai kondisi *current node* sama dengan *target node* bernilai benar. (Syukriah dkk., 2016).

Rumus pencarian Algoritma A^* adalah $F(n) = H(n) + G(n)$ dimana: $G(n)$ adalah *movecost*, dikarenakan simulasi berbentuk *grid* persegi, tiap koordinat antara titik koordinat berikutnya sama bernilai satu. Lalu $H(n)$ dapat dicari dengan rumus sebagai berikut :

```
function heuristic(node) = dx = abs(node.x - goal.x)
                        dy = abs(node.y - goal.y)
                        return dx + dy
```

Rumus ini digunakan dikarenakan pengembangan aplikasi simulasi ini menggunakan fungsi *heuristic manhattan distance* (Yenie, 2016).

G = 14 H = 60 F = 74	G = 10 H = 50 F = 60	G = 14 H = 50 F = 54				
G = 10 H = 50 F = 60		G = 10 H = 30 F = 40		G = 52 H = 10 F = 62		
G = 14 H = 60 F = 74	G = 10 H = 50 F = 60	G = 14 H = 40 F = 54		G = 42 H = 20 F = 62	G = 52 H = 10 F = 62	
	G = 28 H = 60 F = 88	G = 24 H = 50 F = 74	G = 28 H = 40 F = 68	G = 38 H = 30 F = 68		

Gambar 2.2 Algoritma A* pada grid

Fungsi merupakan ukuran biaya yang dikeluarkan dari keadaan awal sampai ke *node* n. Nilai yang diperoleh merupakan jumlah biaya penerapan setiap aturan yang dilakukan pada sepanjang lintasan terbaik menuju suatu simpul dan bukan merupakan hasil estimasi. Adapun fungsi merupakan pengukur biaya tambahan yang harus dikeluarkan dari *node* n sampai mendapatkan tujuan. Perlu diketahui bahwa tidak negatif karena bila negatif, maka lintasan yang membalik siklus pada graf akan tampak lebih baik dengan semakin panjangnya lintasan.

Prinsip dari algoritma ini adalah melakukan traversal satu per satu pada tiap simpul untuk memperoleh lintasan terpendek pada suatu graf. Algoritma A* akan menghitung jarak salah satu lintasan, lalu menyimpannya dan kemudian menghitung jarak lintasan lainnya. Ketika seluruh lintasan telah selesai dihitung, algoritma A* akan memilih lintasan yang paling pendek. Algoritma A* menyelesaikan masalah yang menggunakan graf untuk perluasan ruang statusnya.

Algoritma A* merupakan algoritma *best first search* yang melakukan traversal satu per satu pada tiap simpul untuk memperoleh lintasan terpendek pada suatu graf. Ketika seluruh lintasan telah selesai dihitung, algoritma A* akan memilih lintasan yang paling pendek. Biaya yang diperhitungkan didapat dari biaya sebenarnya ditambah dengan biaya perkiraan. Dalam notasi seperti ini, algoritma A* adalah *complete* dan optimal. Sama dengan algoritma dasar, algoritma A* ini juga menggunakan dua senarai *OPEN* dan *CLOSED*.

Terdapat tiga kondisi bagi setiap suksesor yang dibangkitkan, yaitu: sudah berada di *OPEN* sudah berada di, dan tidak berada di *OPEN* maupun *CLOSED*. Pada ketiga kondisi tersebut diberikan penanganan yang berbeda-beda. Jika suksesor sudah pernah berada di *OPEN*, maka dilakukan pengecekan apakah perlu pengubahan parent atau tidak tergantung pada nilai *g* nya melalui *parent* lama atau *parent* baru. Jika melalui *parent* baru memberikan nilai *g* yang lebih kecil, maka dilakukan pengubahan *parent*. Jika pengubahan *parent* dilakukan, maka dilakukan pula perbaruan (*update*) nilai *g* dan *f* pada suksesor tersebut. Dengan perbaruan ini, suksesor tersebut memiliki kesempatan yang lebih besar untuk terpilih sebagai simpul terbaik (*best node*). Jika suksesor sudah pernah berada di *CLOSED*, maka dilakukan perbaruan nilai *g* dan *f* pada suksesor tersebut serta pada semua “anak cucunya” yang sudah pernah berada di *BEST NODE*. Dengan perbaruan ini, maka semua anak cucunya tersebut memiliki kesempatan lebih besar untuk terpilih sebagai simpul terbaik (*best node*) (Pamungkas, 2011).

2.9 Unity 3D

Unity adalah alat *authoring* permainan 3D untuk Mac dan PC. *Game Engines* adalah mur dan baut yang duduk di balik layar dari setiap video *game*. Dari karya seni sampai ke matematika yang menentukan setiap frame di layar, "engine" membuat keputusan. Dimulai dengan render -metode menampilkan grafis pada layar, dan mengintegrasikan metode kontrol dan satu set aturan untuk permainan untuk mengikuti-mesin adalah apa yang pengembang membangun untuk "house" permainan.

Unity tidak dirancang untuk proses desain atau modelling, dikarenakan unity bukan *tool* untuk mendesain, untuk mendesain 3D *modelling* bisa menggunakan blender. Unity juga mendukung tiga bahasa pemrograman yaitu C#, javascript dan Boo (Pamungkas, 2011).

2.10 *AI (Artificial Intelligence)*

Kecerdasan buatan memang kerap diidentikkan dengan kemampuan robot yang dapat berperilaku seperti manusia. Berbagai definisi diungkapkan oleh para ahli untuk dapat memberi gambaran mengenai kecerdasan buatan beberapa diantaranya:

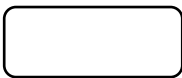
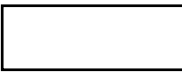

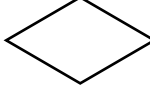

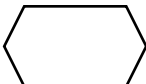
- a. Kecerdasan Buatan (*AI*) merupakan kawasan penelitian, aplikasi dan instruksi yang terkait dengan pemrograman komputer untuk melakukan sesuatu hal yang dalam pandangan manusia adalah cerdas.
- b. Kecerdasan Buatan (*AI*) merupakan sebuah studi tentang bagaimana membuat komputer melakukan hal-hal yang pada saat ini dapat dilakukan lebih baik oleh manusia.
- c. Kecerdasan Buatan (*AI*) merupakan cabang dari ilmu komputer yang dalam merepresentasi pengetahuan lebih banyak menggunakan bentuk simbol-simbol daripada bilangan, dan memproses informasi berdasarkan metode heuristik atau dengan berdasarkan sejumlah aturan (Pamungkas, 2011).

2.11 *Flowchart*

Ada dua *tool* yang sering digunakan untuk membantu menyusun dokumen logika pemrograman, yaitu *flowchart* dan *pseudocode* (kode semu). *Flowchart* adalah simbol-simbol pekerjaan yang menunjukkan bagan aliran proses yang sering terhubung. Jadi, setiap simbol *flowchart* melambangkan pekerjaan dan instruksinya. Simbol-simbol *flowchart* adalah standar yang ditentukan oleh American National Standard Institute Inc (Community, 2012).

Simbol–simbol yang digunakan dalam *flowchart* ditunjukkan pada Tabel 2.1 berikut:

Tabel 2.1 Simbol-simbol dalam *flowchart*

No.	Simbol	Keterangan
1.		Simbol <i>Start</i> atau <i>End</i> yang mendefinisikan awal atau akhir dari sebuah <i>flowchart</i> .
2.		Simbol pemrosesan yang terjadi pada sebuah alur kerja.
3.		Simbol <i>Input/Output</i> yang mendefinisikan masukan dan keluaran proses.
4.		Simbol untuk memutuskan proses lanjutan dari kondisi tertentu.
5.		<i>Storage</i> menyatakan <i>input</i> berasal dari <i>storage</i> atau <i>output</i> di simpan ke <i>storage</i> .
6.		<i>Preparation</i> merupakan simbol persiapan yang digunakan untuk memberi nilai awal suatu besaran.


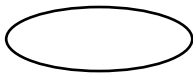
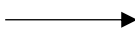
2.12 Unified Modelling Language (UML)

Unified Modelling Language (UML) adalah sebuah "bahasa" yg telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem. Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan class dan operation dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasabahasa berorientasi objek seperti C++, Java, C# atau VB.NET. Walaupun demikian, UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C (Sri dan Romi, 2003).

2.12.1 Use case Diagram

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”. Sebuah use case merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, membuat sebuah daftar belanja, dan sebagainya. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu (Sri dan Romi, 2003).



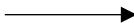

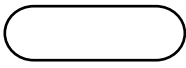
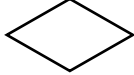
Tabel 2.2 Simbol use case diagram

No.	Simbol	Keterangan
1.		Aktor, Menunjukkan <i>user</i> yang akan menggunakan sistem.
2.		<i>Usecase</i> , Menunjukkan proses yang terjadi pada sistem.
3.		<i>Undirectional Association</i> , Menunjukkan hubungan antara aktor dengan dan <i>usecase</i> atau antar <i>usecase</i> .

2.12.2 Activity Diagram

Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi (Sri dan Romi, 2003).

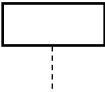
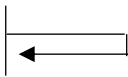

Tabel 2.3 Simbol activity diagram

No.	Simbol	Keterangan
1.	Start 	Kondisi Awal, Menunjukkan awal dari suatu diagram aktivitas.
2.	End 	Kondisi Akhir, Menunjukkan akhir dari suatu diagram aktivitas.
3.		Kondisi transisi, Menunjukkan kondisi transisi antar aktivitas.
4.		Swimlane, Menunjukkan aktor dari diagram aktivitas yang dibuat.
5.		Aktivitas, Menunjukkan aktivitas-aktivitas yang terdapat pada diagram aktivitas.
6.		Pengecekan kondisi, Menunjukkan pengecekan terhadap suatu kondisi.

2.12.3 Sequence Diagram

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa message yang digambarkan terhadap waktu. Sequence diagram terdiri atas dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). (Sri dan Romi, 2003).

Tabel 2.4 Simbol *sequence diagram*

No.	Simbol	Keterangan
1.		Objek, Menunjukkan objek yang terdapat di diagram <i>sequence</i> .
2.		Pesan ke objek sendiri, Menunjukkan pesan yang diproses pada objek itu sendiri.
s3.		Pesan objek, Menunjukkan pesan yang disampaikan ke objek lain dalam diagram <i>sequence</i> .

BAB III

METODOLOGI PENELITIAN

3.1 Waktu dan Tempat Penelitian

3.1.1 Waktu

Waktu pelaksanaan penelitian tugas akhir dilaksanakan mulai dari bulan September 2019 sampai dengan Desember 2019. Rincian kegiatan dapat dilihat pada Tabel 3.1 berikut:

Tabel 3.1 Gant Chart Waktu Penelitian

No	Uraian	Waktu (2019)															
		September				Oktober				November				Desember			
		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	<i>Inception</i>																
2	<i>Elaboration</i>																
3	<i>Construction</i>																
4	<i>Transition</i>																

3.1.2 Tempat Penelitian

Adapun tempat penelitian tugas akhir yang akan dilakukan tidak mengacu pada tempat tertentu.

3.2 Metode Pengumpulan Data

Dalam penelitian ini metode pengumpulan data yang digunakan pada perancangan aplikasi tersebut adalah studi *literatur*. Metode ini dilaksanakan dengan melakukan studi kepustakaan yang relevan. Metode ini dilakukan untuk mencari sumber pelengkap yang berhubungan dengan aplikasi yang akan dibangun, yaitu dengan mencari referensi yang berkaitan dengan kata serapan, sehingga dapat

di implementasikan dalam aplikasi kamus tersebut, mulai dari buku-buku, jurnal maupun artikel dan sumber-sumber lain di *internet*.

3.3 Metode Pengembangan Sistem

Metode pengembangan sistem yang digunakan dalam sistem ini adalah metode *Rational Unified Process* (RUP). Dalam metode ini, terdapat empat tahap pengembangan perangkat lunak, yaitu:

3.3.1 Inception

Pada *fase* ini dilakukan proses pengidentifikasian sistem, dilakukan dengan analisis kebutuhan akan aplikasi, melakukan kajian terhadap penelitian yang terkait dengan algoritma A*.

3.3.2 Elaboration

Setelah menentukan ruang lingkup penelitian, tahap ini akan dilakukan perancangan dan analisis sistem menggunakan *flowchart* meliputi *flowchart* algoritma A*. Pada perancangan ini, digunakan juga UML (*Unified Modelling Language*) yang meliputi *activity diagram*, *sequence diagram* dan *use case diagram*.

3.3.3 Construction

Proses yang dilakukan pada tahap ini yaitu membangun aplikasi dengan perancangan yang telah dilakukan sebelumnya, mulai dari tampilan *interface* menu *game* sampai *exit game*. Proses yang juga dilakukan pada tahap ini yaitu penerapan *coding* algoritma A* pada *game* sebagai metode untuk musuh melakukan pencarian rute terdekat menuju ke posisi *player*.

3.3.4 Transition

Pada tahap *Transition* difokuskan untuk melakukan proses pengujian terhadap aplikasi. Dalam penelitian ini, dilakukan pengujian menggunakan *black box* terhadap aplikasi yang meliputi pengujian pencarian rute terdekat. Selain menggunakan pengujian *black box*, pada pengujian ini juga dilakukan menggunakan *white box* pada *coding* aplikasi sehingga tidak ada kesalahan penulisan pada *coding* dan memastikan sistem sudah bekerja dengan baik.

BAB IV

ANALISIS DAN PERANCANGAN SISTEM

4.1 Analisis Sistem yang Direncanakan

Penerapan algoritma *all star (A*) pathfinding* dalam pembentukan *AI enemy* pada game *Ghost Adventure 3D*. Pada game ini algoritma *A** yang diterapkan pada *enemy* atau musuh dalam game ini akan mencari rute terpendek untuk mengejar *player* dan ditambahkan *AI* apabila *enemy* mendapat penghalang maka posisi *enemy* akan berubah sesuai arah jalur nya, algoritma *A** juga diterapkan pada *player* untuk memudahkan mencari informasi yang ada didalam game ini, informasinya tidak menutup kemungkinan berada didalam atau disekitar Gedung fakultas yang ada di UHO.

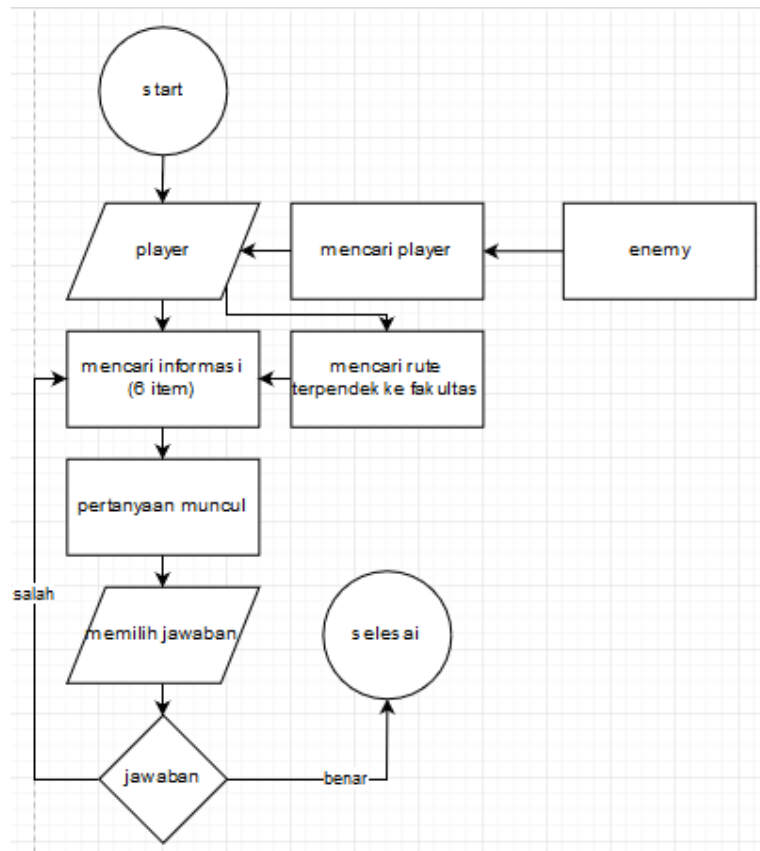
4.1.1 Perancangan *Flowchart*

Flowchart adalah simbol-simbol pekerjaan yang menunjukkan bagan aliran proses yang saling terhubung. Perancangan *flowchart* yang akan dibangun terdiri atas *flowchart* sistem dan *flowchart* algoritma *All star (A*)*.

4.1.1.1 *Flowchart* Sistem

Setelah menganalisis sistem, maka didapatkan *flowchart diagram game Ghost Adventure* yang ditunjukkan oleh Gambar 4.1. Adapun alur kerja *flowchart diagram game* adalah sebagai berikut:

- a. *Player* akan mencari kertas *item* yang berisi informasi sebanyak 6.
- b. *Player* mencari rute terpendek ke fakultas untuk mencari *item* informasi.
- c. Pertanyaan akan muncul jika *player* berhasil mengumpulkan 6 *item*.
- d. Musuh pada game ini akan terus menyerang *player* sebanyak 3x.
- e. *Player* harus menjawab pertanyaan yang muncul.

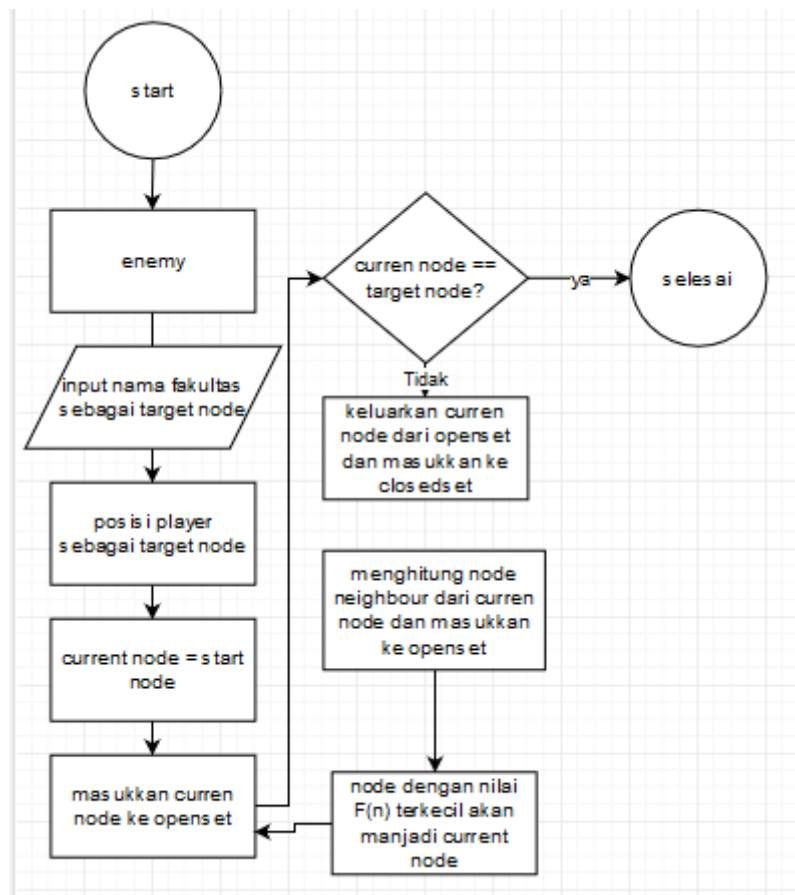


Gambar 4.1 Flowchart Sistem

4.1.1.2 Flowchart Algoritma A* (All Star)

Setelah menganalisis sistem, maka didapatkan *flowchart diagram* Algoritma A* pada Aplikasi *game Ghost Adventure* yang ditunjukkan oleh Gambar 4.2. Adapun alur kerja *flowchart diagram* algoritma A* pada NPC (*Non Player Character*) adalah sebagai berikut:

- Menentukan *start node* dan *target node*.
- Player akan menginput *target node* berupa nama fakultas yang akan dituju.
- Posisi *player* sebagai *start node* dan fakultas yang dituju sebagai *target node*.
- Algoritma A* akan menghitung *node* dengan nilai *F cost* terkecil sebagai rute.
- Apabila *current node* sama dengan *target node*, maka pencarian akan berhenti.



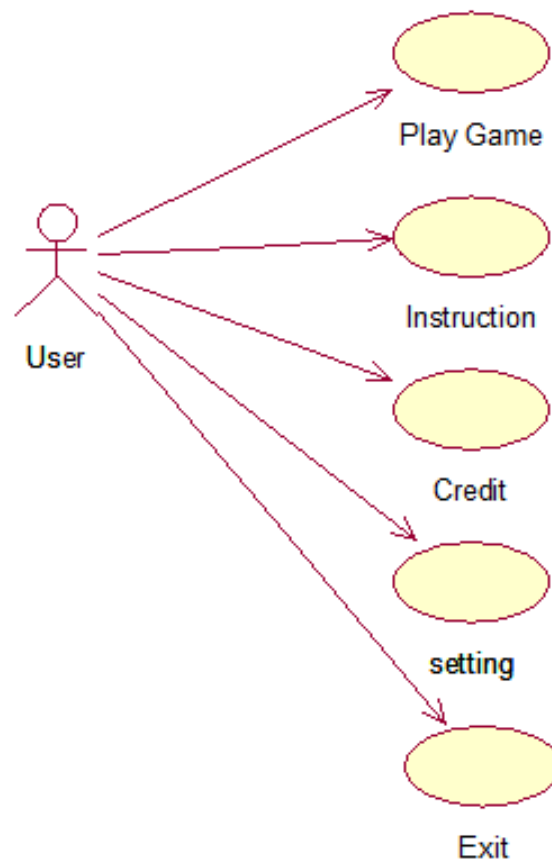
Gambar 4.2 Flowchart Algoritma A* pada pencarian rute

4.1.2 Unified Modeling Language (UML)

Aplikasi dibangun dengan menggunakan *Unified Modeling Language* (UML). UML merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram yang terdiri dari *Use Case Diagram*, *Activity Diagram*, dan *Sequence Diagram*.

4.1.2.1 Use Case Diagram

Use Case Diagram adalah sebuah diagram yang dapat merepresentasikan interaksi yang terjadi antara user dengan sistem. *Use Case Diagram* ini mendeskripsikan siapa saja yang menggunakan sistem dan bagaimana cara mereka berinteraksi dengan sistem. *Use Case Diagram* dari sistem yang akan dibangun dapat ditunjukkan pada Gambar 4.3.



Gambar 4.3 Use Case Diagram Aplikasi

Tabel 4.2 Keterangan *Use Case Diagram*

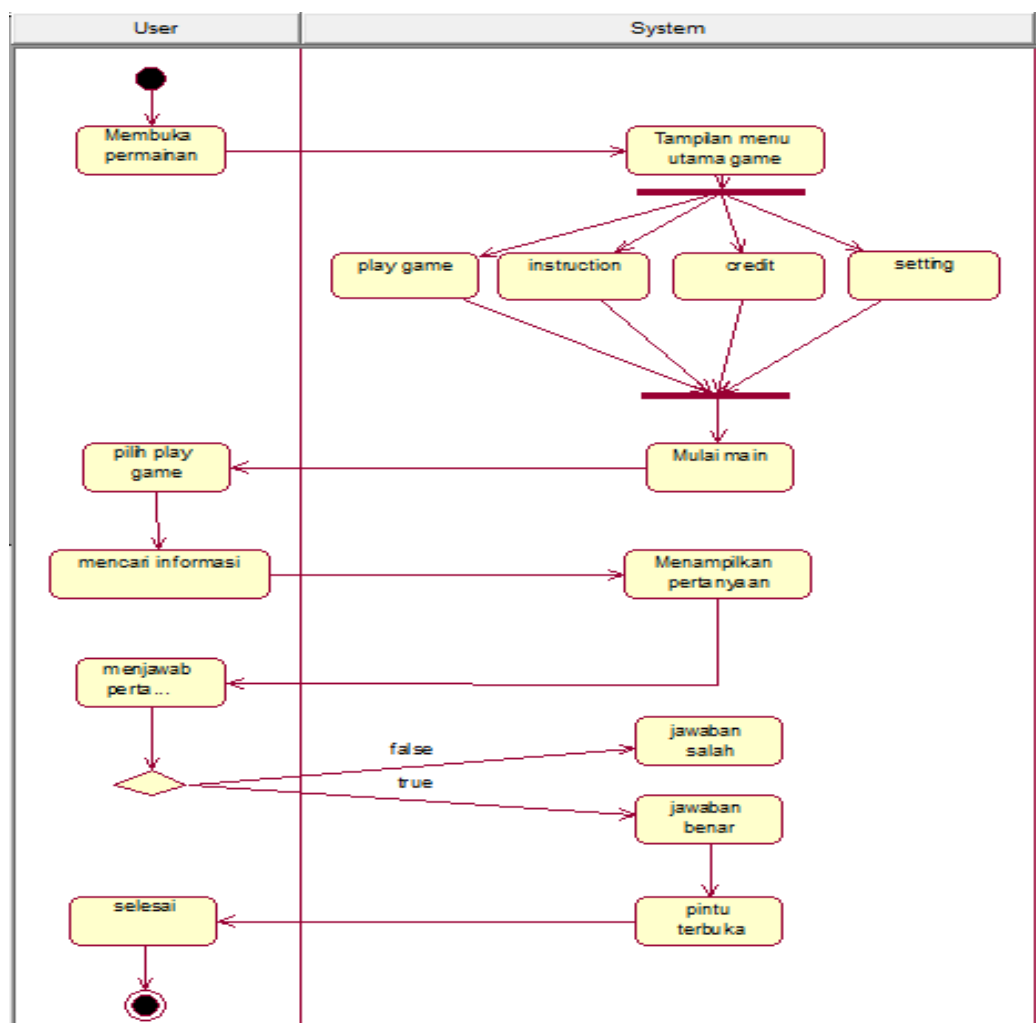
Aktor	Sistem
<i>User</i> memilih menu <i>Play Game</i>	Sistem akan menampilkan <i>room game</i> .
<i>User</i> memilih cari <i>Instruction</i>	Sistem akan menampilkan tampilan form petunjuk permainan.
<i>User</i> memilih menu <i>credit</i>	Sistem akan menampilkan informasi tentang pembuatan <i>game</i> tersebut
<i>User</i> memilih tombol <i>setting</i>	Sistem akan menampilkan pengaturan dari <i>game</i> tersebut.
<i>User</i> memilih menu <i>exit</i>	Sistem akan <i>close game</i> .

4.1.2.2 Activity Diagram

Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. Berikut ini adalah *activity diagram* yang akan menggambarkan alir aktivitas sistem.

1. Activity Diagram play game

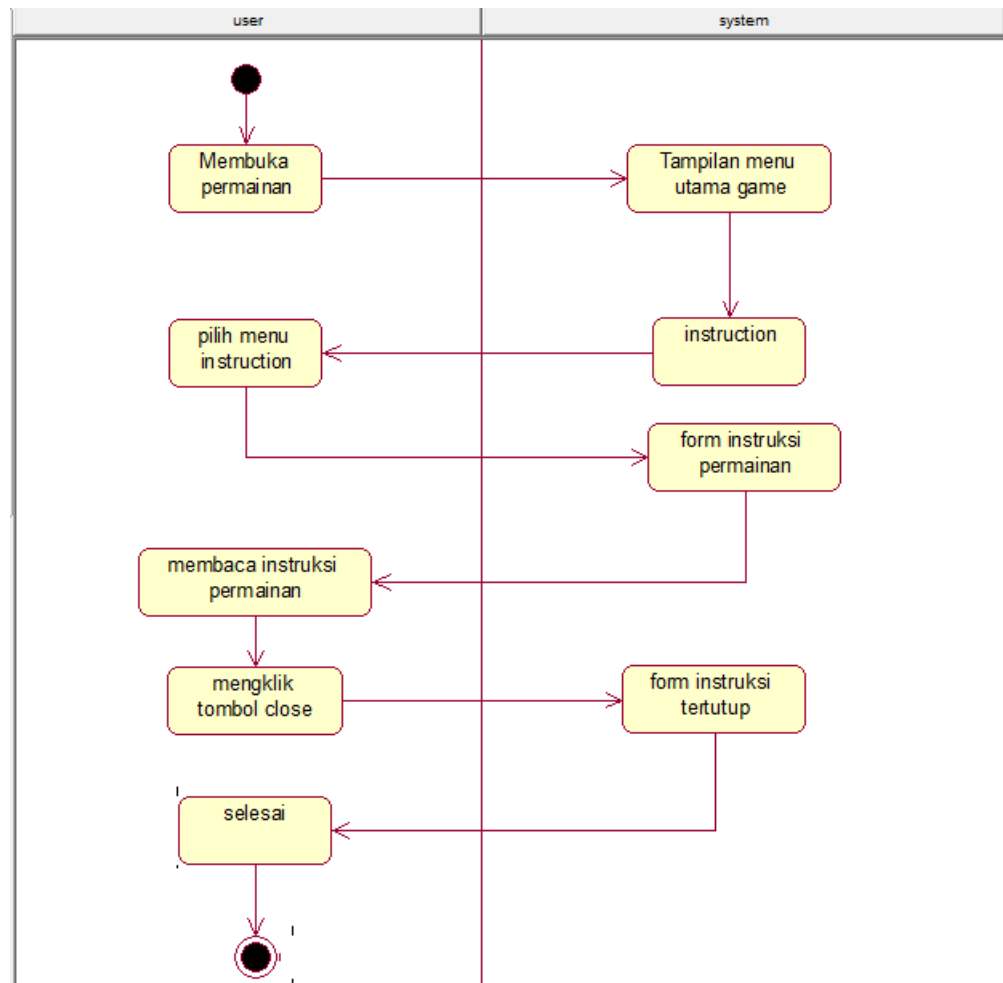
Gambar 4.4 merupakan diagram aktivitas yang menunjukkan aktivitas *user* ketika memilih menu *play game*, lalu sistem akan menampilkan *room game* dan *player* siap bermain.



Gambar 4.4 Activity Diagram play game

2. Activity Diagram Instruction

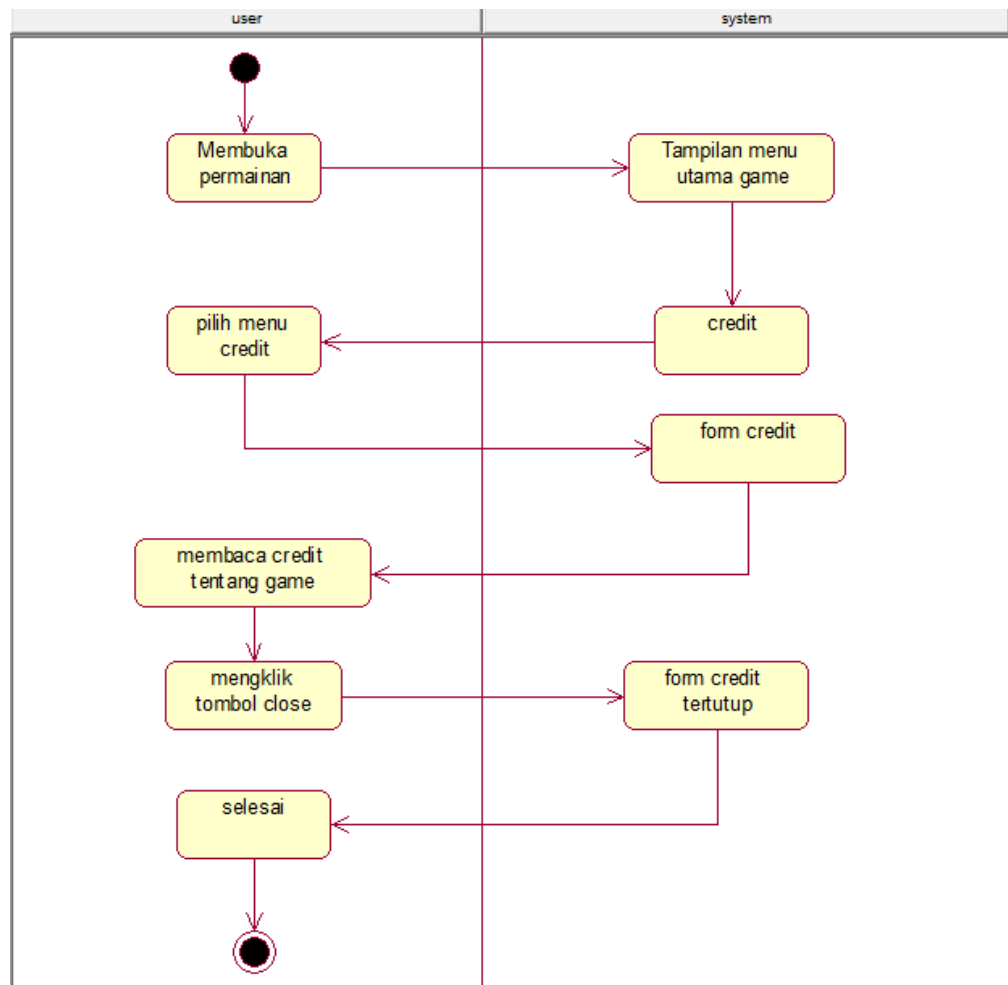
Gambar 4.5 tersebut merupakan diagram aktivitas yang menunjukkan aktivitas *user* ketika telah memilih menu *instruction*, lalu sistem akan menampilkan form yang berisi instruksi cara bermain game ini dengan jelas.



Gambar 4.5 Activity Diagram Instruction

3. Activity Diagram Credit

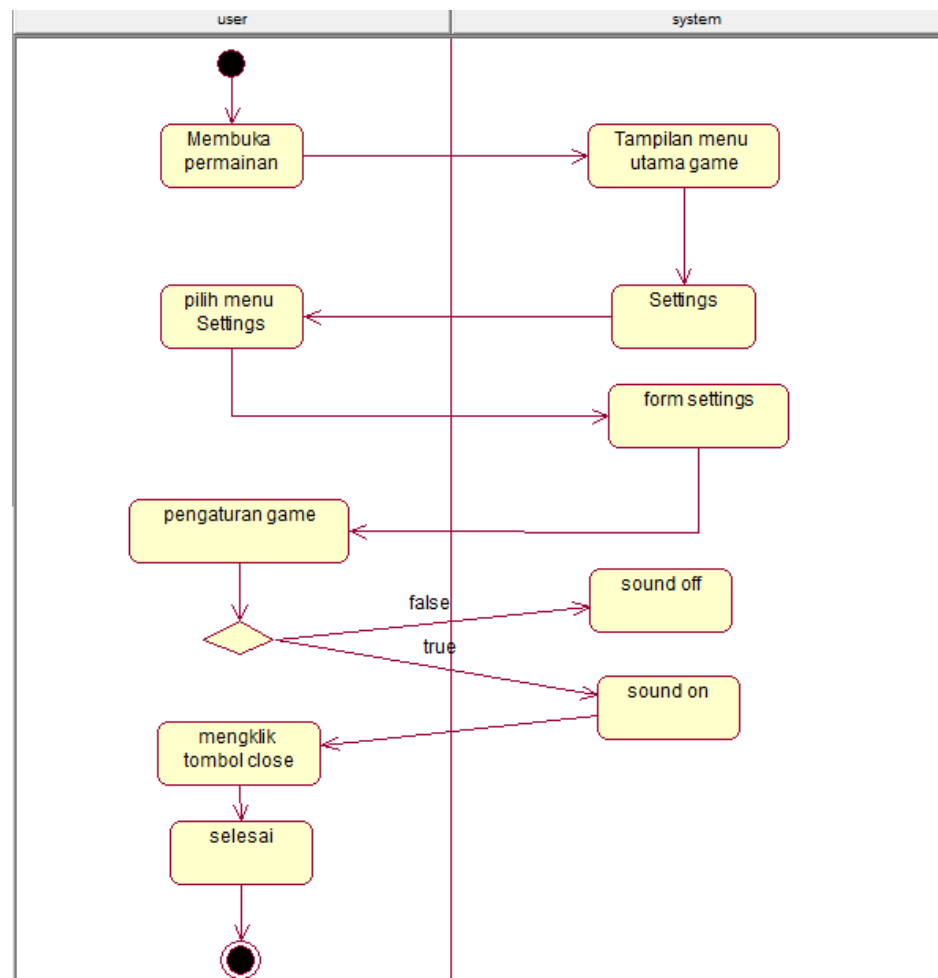
Gambar 4.6 merupakan gambar *Activity Diagram* yang menunjukkan aktivitas *user* memilih menu *credit*, maka sistem akan menampilkan form yang berisi informasi tentang game ini.



Gambar 4.6 Activity Diagram Credit

4. Activity Diagram Settings

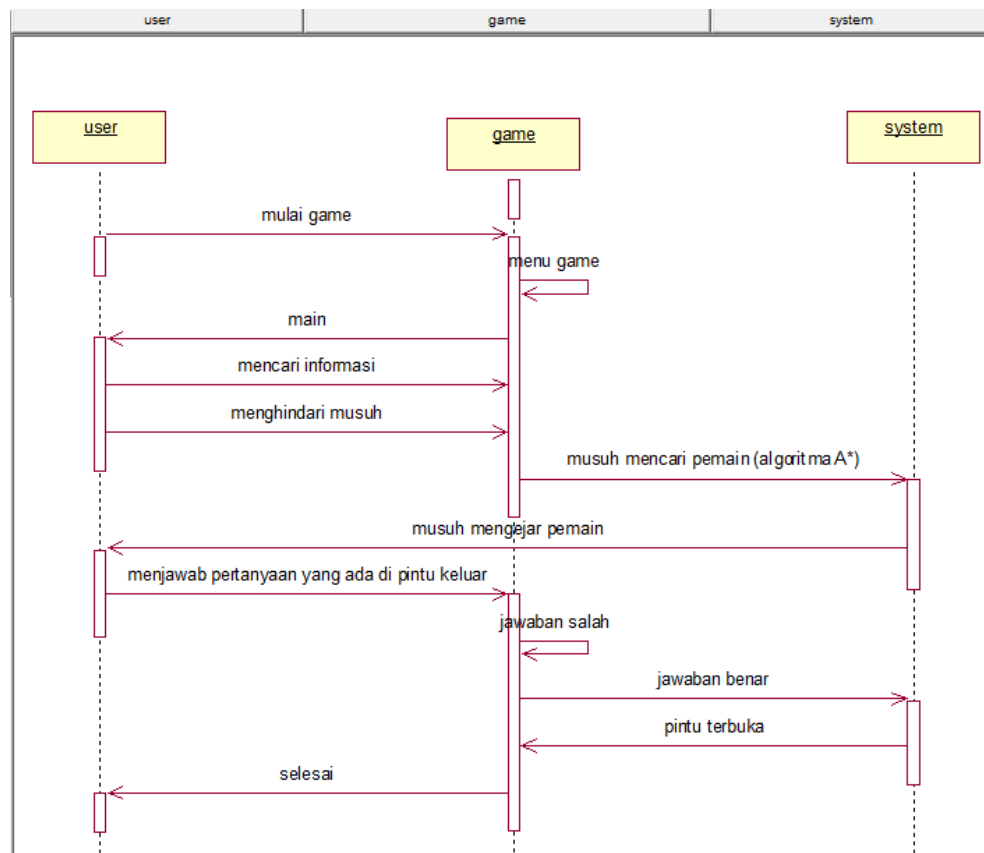
Gambar 4.7 merupakan gambar *Activity Diagram* yang menunjukkan aktivitas *user* memilih menu *settings*, maka sistem akan menampilkan menu pengaturan. Setelah itu, menu pengaturan tersebut akan menampilkan *checkbox* yaitu *sound off* atau *sound on*.



Gambar 4.7 Activity Diagram Settings

4.1.2.3 Sequence Diagram

Sequence Diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem yang digambarkan terhadap waktu. Berikut ini adalah *Sequence Diagram* yang akan menggambarkan interaksi antar objek dan sistem.



Gambar 4.8 Sequence Diagram Game

Gambar 4.8 adalah diagram *sequence* yang menunjukkan hubungan dari sistem terhadap *user*, dimana sistem akan menampilkan tampilan menu, lalu *user* memilih menu *play game* maka player akan dialihkan menuju *room game*, pemain harus mencari informasi terlebih dahulu dan menghindari musuh yang akan mengganggu pemain, musuh akan mencari lokasi pemain dengan menggunakan algoritma A* kemudian akan menyerang pemain, jika pemain sudah cukup dalam mengumpulkan informasi, pemain harus ke pintu keluar dan akan diberikan pertanyaan, apa bila pemain menjawab salah maka pertanyaan akan berganti, dan apabila benar maka pemain telah berhasil menyelesaikan *game* tersebut.

4.2 Penerapan Algoritma *All star* (A^*)

Algoritma A^* pada game ini digunakan sebagai pencarian rute terpendek dari musuh menuju ke posisi *player* ketika jarak sudah dekat dari *player* maka musuh akan menyerang. Untuk *player* akan menuju ke posisi fakultas yang akan di tuju dengan algoritma A^* sebagai penunjuk jalan. Pada musuh akan ditambahkan AI apabila *player* disernag oleh musuh maka nyawa *player* akan berkurang.

Player bisa menggunakan fitur pencarian fakultas universitas halu oleo yang menggunakan algoritma A^* sebagai pencarian rute terpendek untuk menuju ke fakultas yang inputkan oleh *player* karena *player* harus menuju ke setiap fakultas yang ada di universitas halu oleo karena beberapa informasi berada didalam atau diluar sekitar fakultas tersebut.

Untuk menggambarkan algoritma A^* , pertama akan membuat *grid* terlebih dahulu dan menentukan ukuran dari *map* pada *game* tersebut kemudian A^* akan menghitung *node* yang ada didalam *grid* dan akan menghasilkan *node* yang di beri kode warna sebagai rute terpendek dari posisi *start* ke posisi *target* dengan memperhatikan *node* yang bersifat sebagai penghalang.

4.3 Perancangan Antarmuka (*Interface*)

Perancangan *user interface* adalah tahapan pembuatan antarmuka yang akan digunakan pada pembangunan aplikasi *game* yang dibagi menjadi lima bagian yaitu perancangan tampilan *menu game*, *form instruction*, *form credit*, *menu settings* dan *form exit game*.

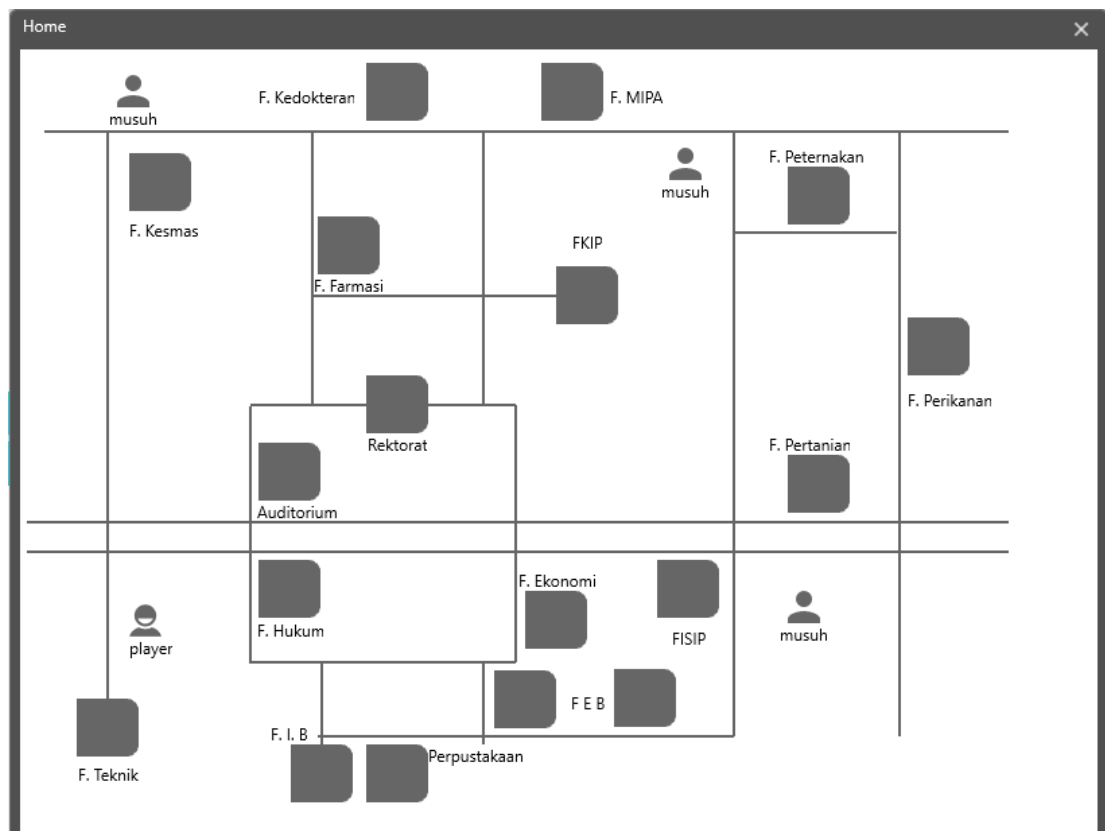
4.3.1 Tampilan *menu game*

Menu game adalah tampilan pertama yang akan ditampilkan ketika *user* membuka *game*. Pada *form menu game* berisi tombol *play game*, *instruction*, *credit*, *exit* dan icon *gear* untuk *settings*.



Gambar 4.9 Menu Game

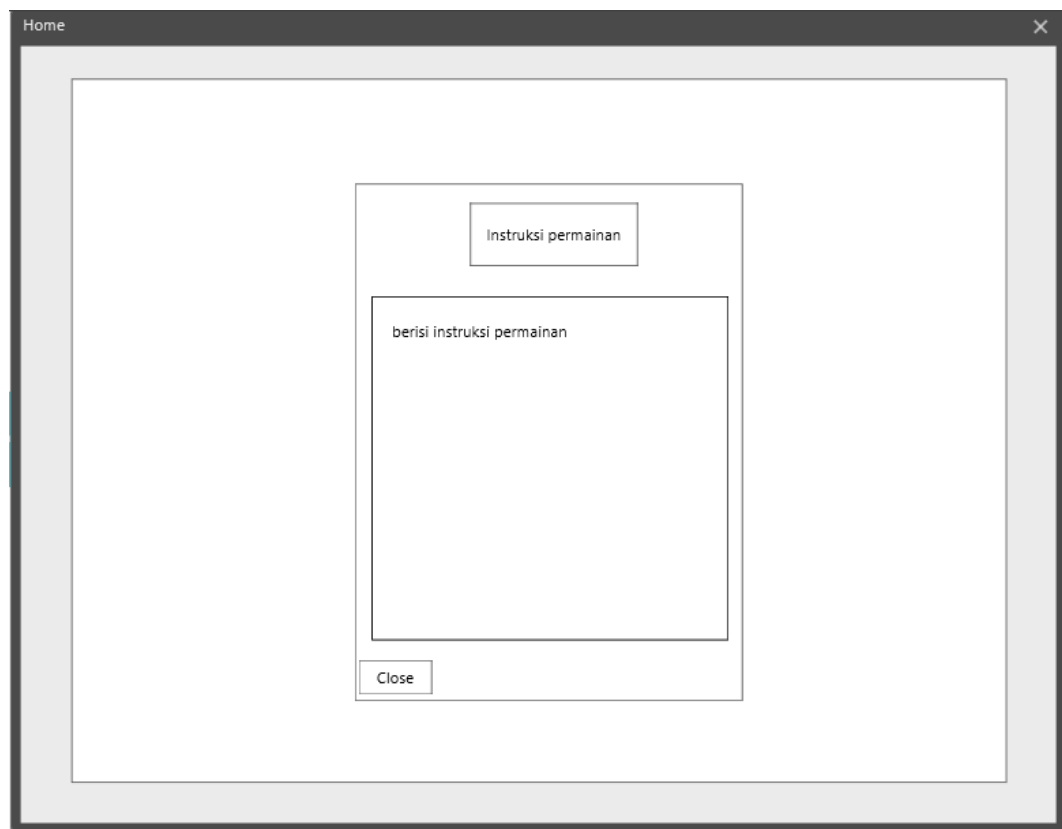
4.3.2 Tampilan *gameplay*



Gambar 4.10 Tampilan *gameplay*

Pada saat *user* menekan tombol *play game* maka sistem akan langsung menuju ke *room game*, di *room game* terdapat *player* dan juga musuh, musuh akan terus mengejar *player* dan *player* akan mencari informasi yang dibutuhkan untuk menjawab pertanyaan yang ada di pintu keluar, *player* harus mengeksplor *map*, sedangkan musuh akan mencari rute terdekat untuk mencari *player* dengan menggunakan algoritma A* dan akan menyerangnya.

4.3.3 Tampilan *form instruction*

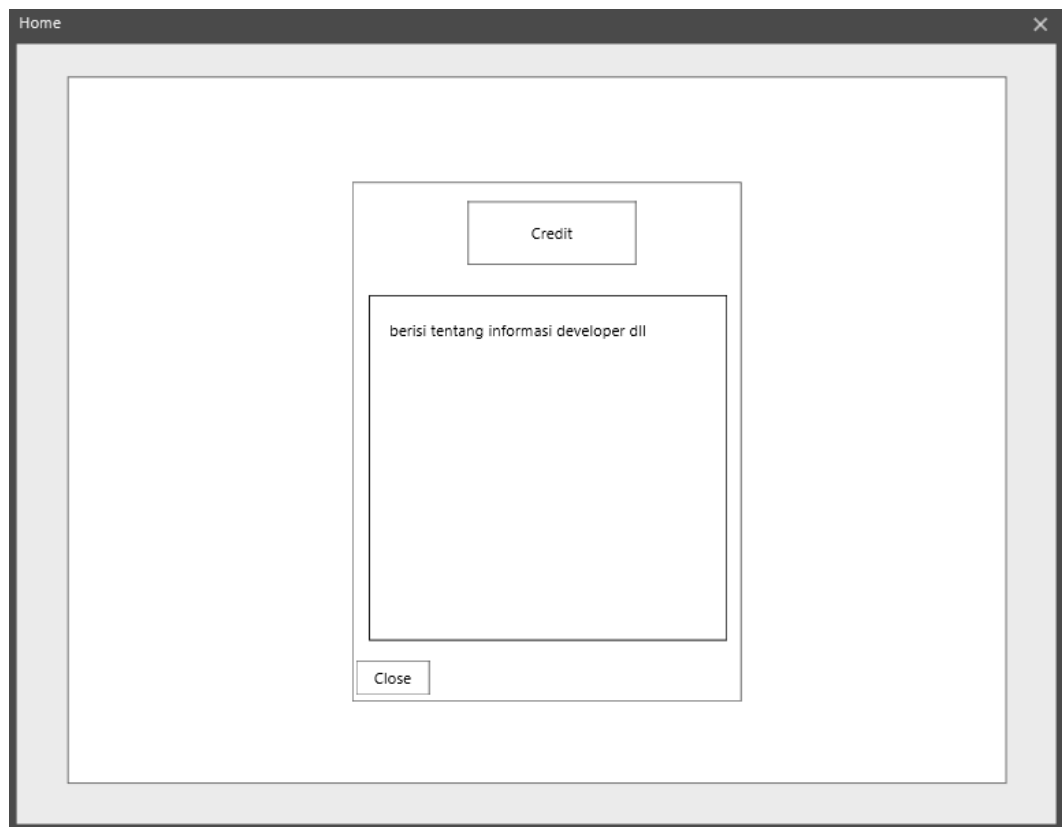


Gambar 4.11 *Form Instruction*

Pada *form* daftar menu berisi tips dan trik untuk menyelesaikan *game* ini dengan mudah.

4.3.4 Tampilan *Form Credit*

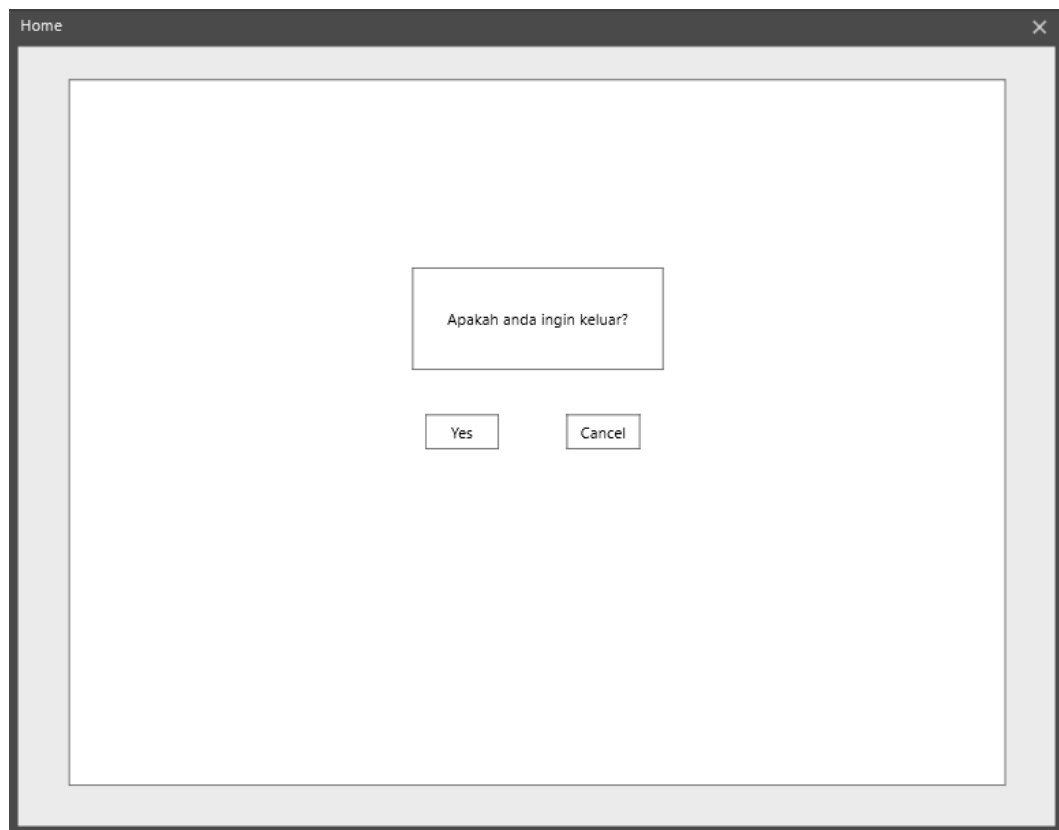
Pada *form credit* ini berisi informasi tentang pengembang dari *game* dan semua orang-orang yang terlibat dalam pembuatan *game* ini.



Gambar 4.12 Tampilan *Form Credit*

4.3.5 Tampilan *Form Exit*

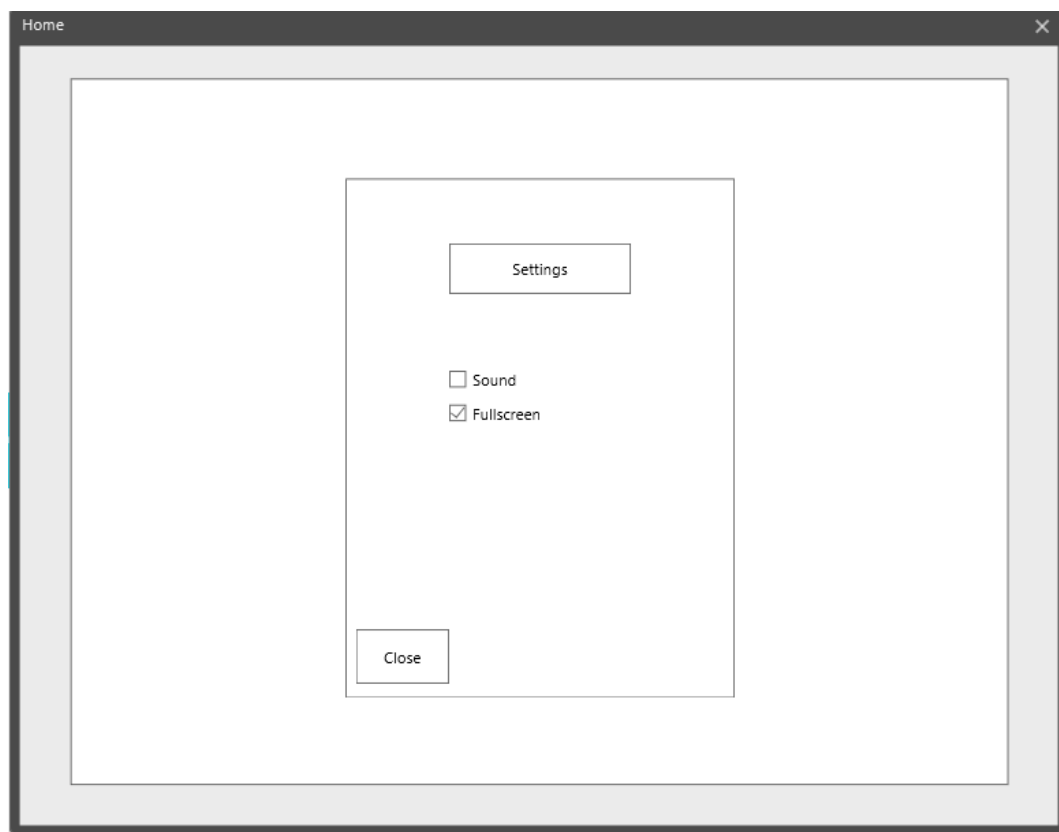
Pada saat *user* memilih tombol *exit*, sistem akan menampilkan pilihan *yes* atau *cancel*, apabila *user* memilih *cancel* maka akan kembali ke menu utama dan apabila *user* memilih *yes* maka *game* akan *close*.



Gambar 4.13 Tampilan *Form Exit*

4.3.6 Tampilan *Settings*

Pada saat *user* mengklik *icon gear* yang ada di sudut kiri atas menu utama, sistem akan menampilkan *form setting game*, yang berisi *checkbox sound* dan *fullscreen* yang berguna untuk mengaktif dan mematikan musik di dalam *game* dan juga mengubah layar menjadi *fullscreen* atau *windowed*.



Gambar 4.14 Tampilan *Form Settings*

BAB V

IMPLEMENTASI DAN PENGUJIAN SISTEM

5.1 Implementasi antarmuka sistem

Implementasi Algoritma *All Star (A*)* pada *Game Ghost Adventure* untuk pencarian rute terpendek pada *player* dan *target* (fakultas). Dalam penelitian ini penulis melakukan implementasi pada perangkat keras yang di miliki dengan spesifikasi sebagai berikut:

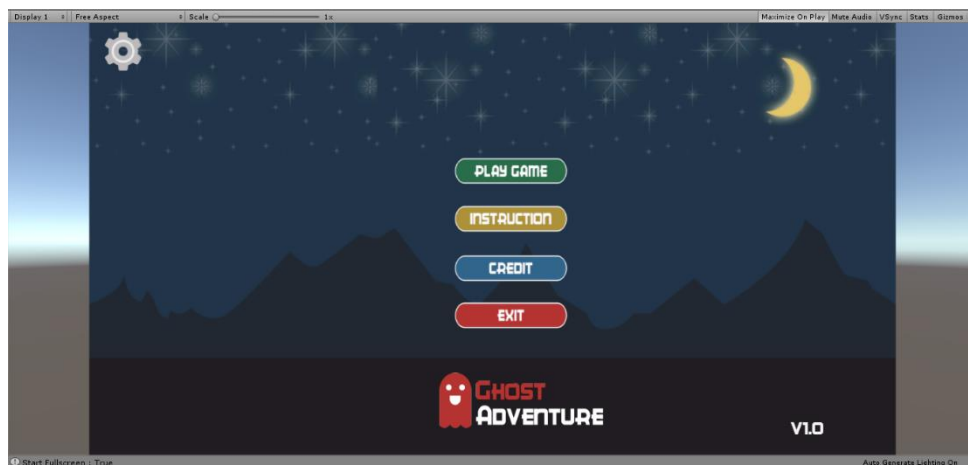
1. Sistem Operasi : Windows 10
2. *Builder* : Microsoft Visual Studio
3. *Compiler* : Unity 3D

Adapun spesifikasi perangkat keras yang digunakan sebagai berikut :

1. Processor Intel Core i5 7TH GEN
2. HDD 1000GB, RAM 8GB
3. VGA : Nvidia GEFORCE 940mx 2GB

5.1.1. Tampilan Menu *Game*

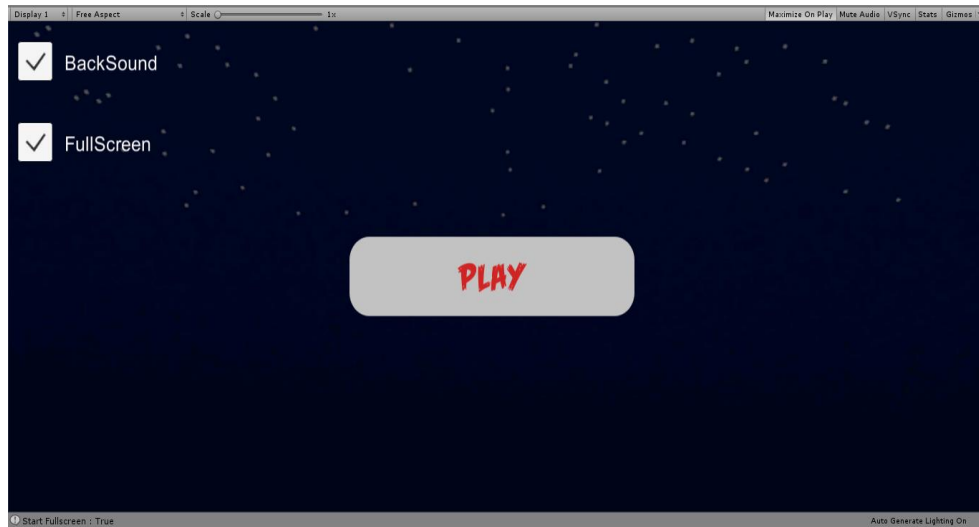
Tampilan halaman utama merupakan tampilan utama dari aplikasi *game Ghost Adventure* , didalam halaman utama terdapat fungsionalitas yang dapat diakses oleh pemain yaitu *menu Play Game*, *menu instruction*, *menu settings*, *menu credit* dan *menu exit*.



Gambar 5.1 Tampilan *menu game*

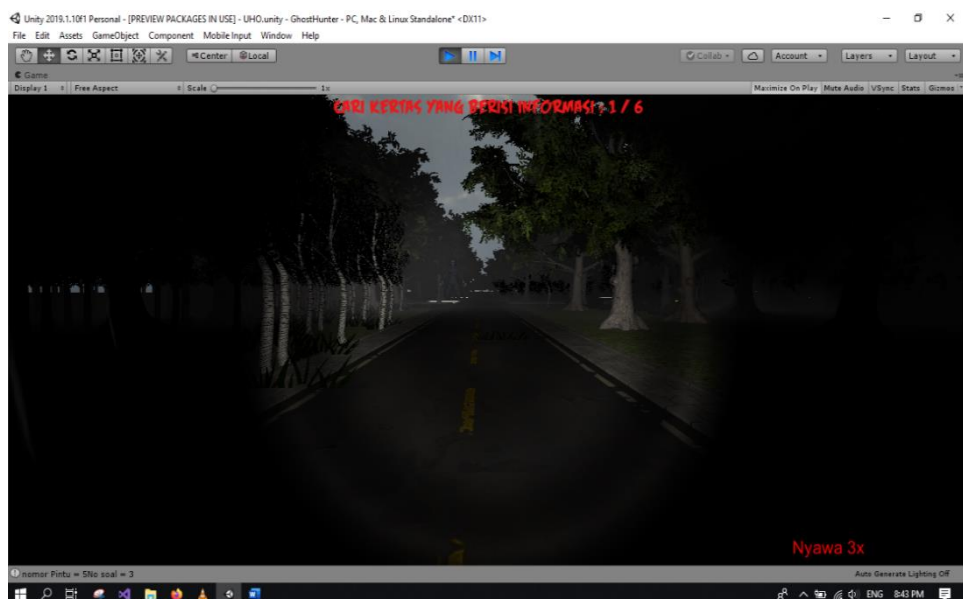
5.1.2. Tampilan *Play Game*

Tampilan *Gameplay* merupakan tampilan *user* yang siap memainkan *game ghost adventure*, ketika *user* mengklik tombol *play game* maka *user* akan di arahkan menuju *room* yang berisi tombol *play* dan pengaturan.

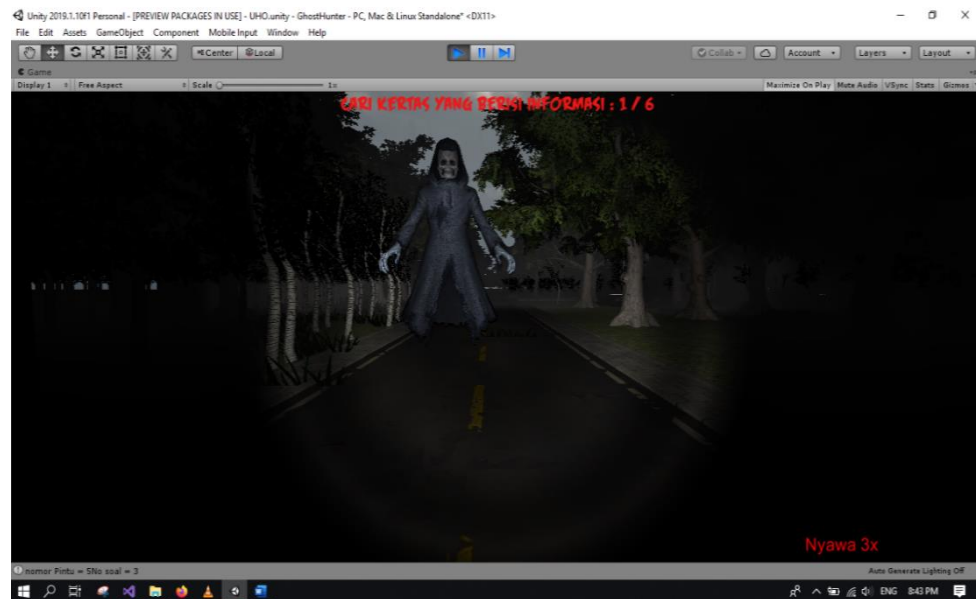


Gambar 5.2 Tampilan *Play Game*.

Ketika *user* menekan tombol *play* maka sistem akan mengarahkan *user* ke dalam *room gameplay ghost adventure*.



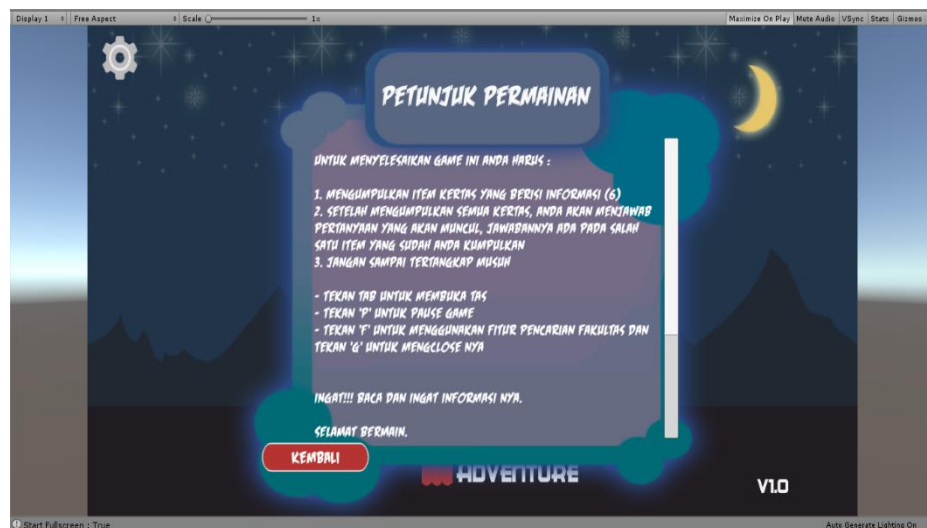
Gambar 5.3 Tampilan *Gameplay* pertama



Gambar 5.4 Tampilan *Gameplay* kedua

5.1.3. Tampilan *Instruction*

Ketika *player* memilih menu *instruction* maka akan di tampilkan menu intruksi permainan yang berisi tentang cara dan aturan bermain *game* ini seperti pada gambar 5.5.



Gambar 5.5 Tampilan *Instruction*

5.1.4. Tampilan *Credit*

Pada menu *credits* ini akan menampilkan informasi tentang pembuat *game* dan lain nya *Ghost Adventure* pada Gambar 5.6.



Gambar 5.6 Tampilan *Credit*

5.1.5. Tampilan *Setting*

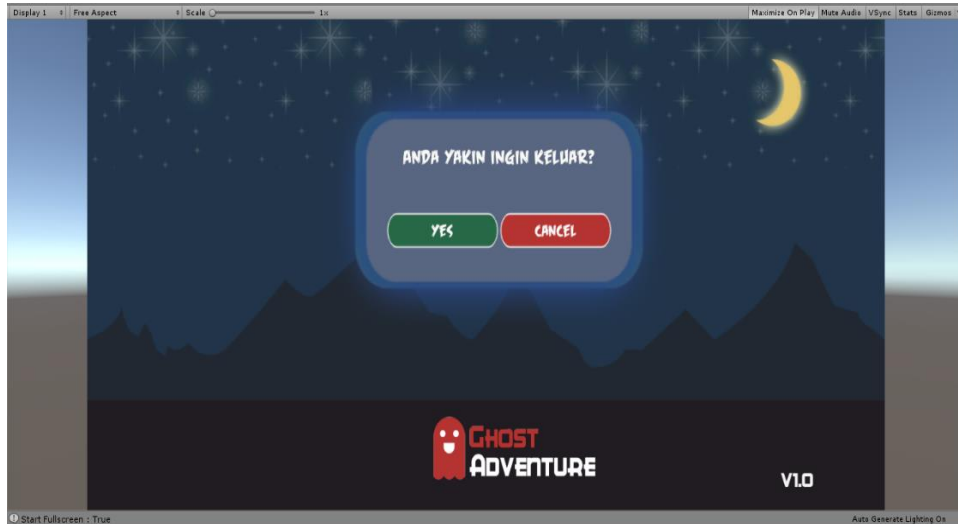
Pada tampilan ini merupakan tampilan pengaturan, dimana player dapat mengatur apakah ingin mematikan *background* atau membuat layar menjadi *fullscreen* atau tidak.



Gambar 5.7 Tampilan *Setting*

5.1.6. Tampilan *Exit*

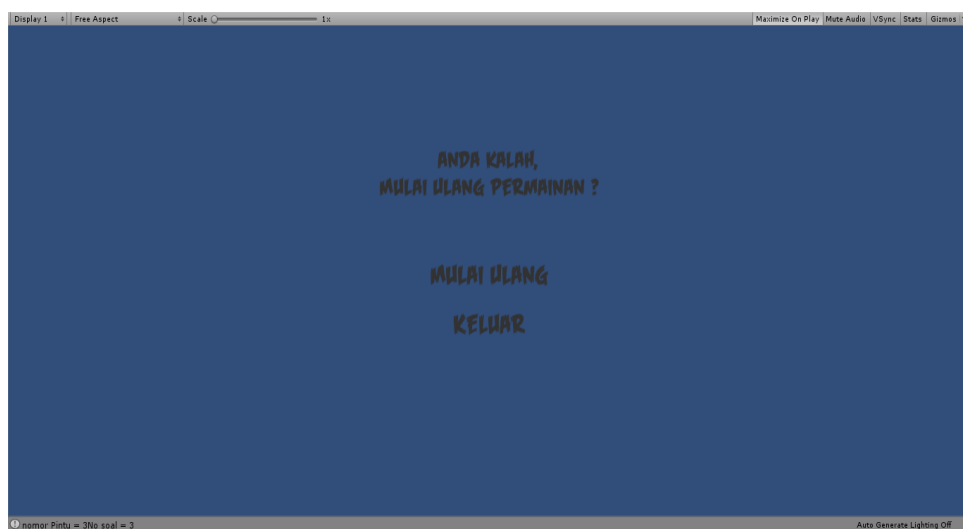
Pada tampilan ini berisi sebuah pesan konfirmasi ketika anda ingin keluar dari *game* tersebut.



Gambar 5.8 Tampilan *Exit*

5.1.7. Tampilan *Game Over*

Pada tampilan berisi sebuah pesan konfirmasi ketika anda gagal dalam menjawab pertanyaan atau *player* tertangkap oleh musuh yang ada di *game* ini, *player* akan memilih mau bermain lagi atau keluar dari permainan seperti pada gambar 5.9.



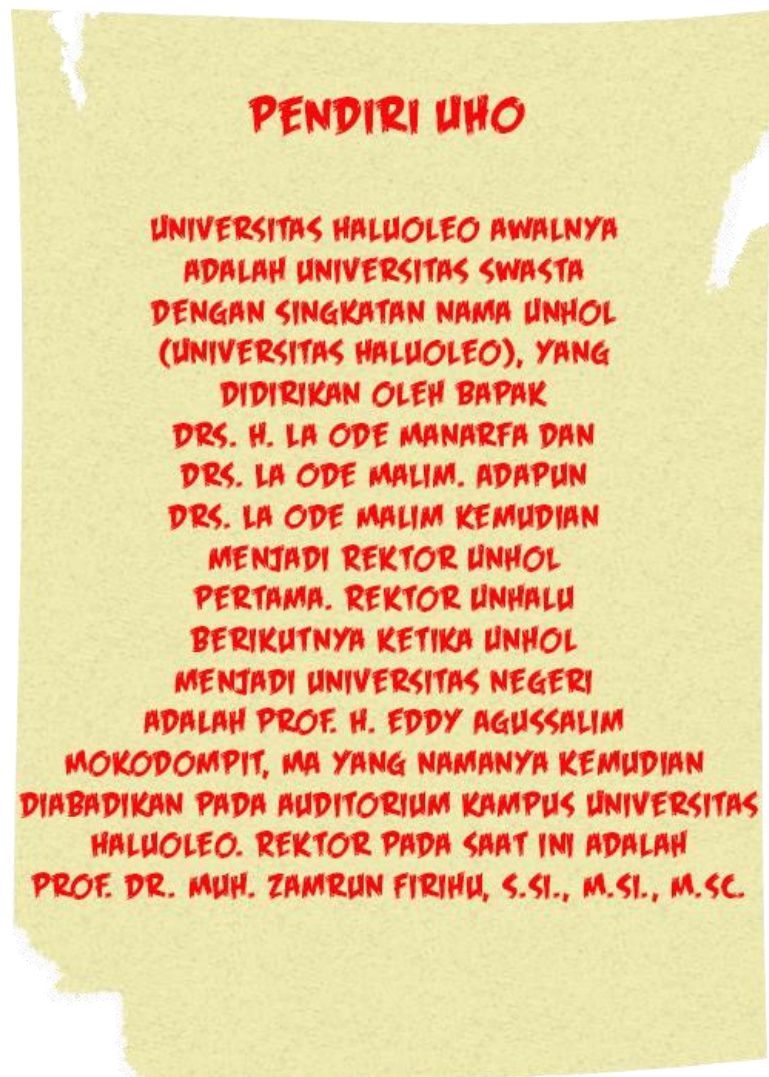
Gambar 5.9 Tampilan *Game Over*

5.1.8. Tampilan *Item* Informasi

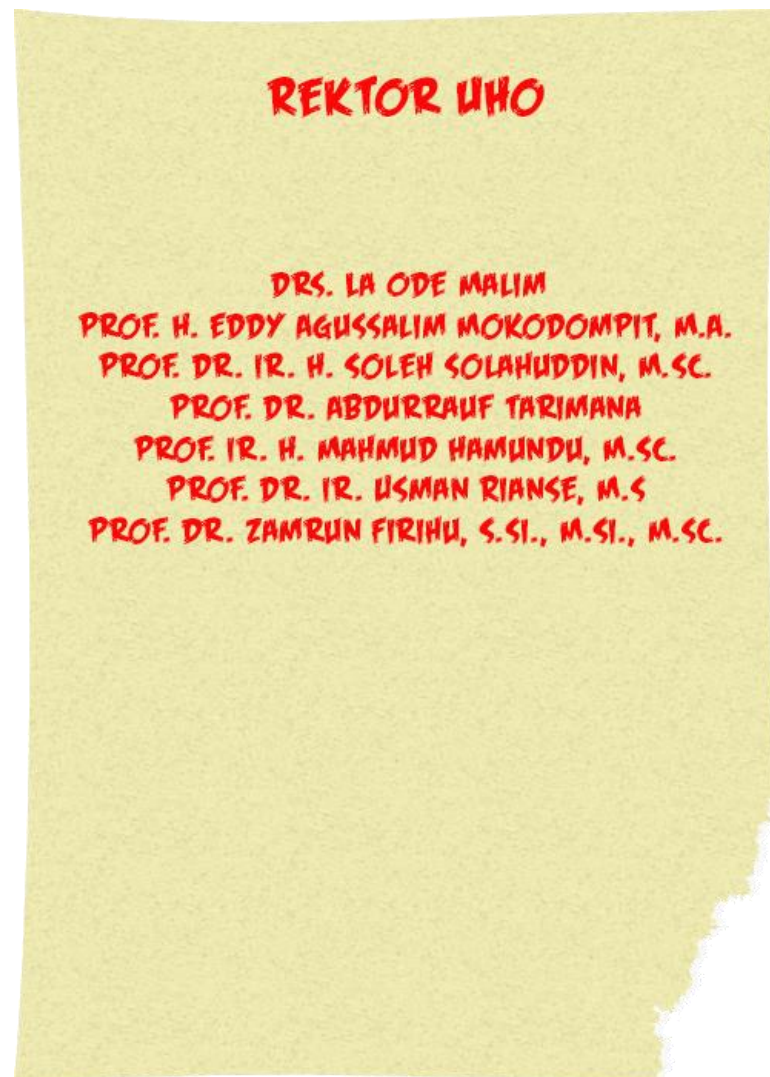
Tampilan ini berisi semua *item* yang harus dikumpulkan oleh *player* dan setiap *item* bisa menjadi jawaban dari soal yang akan muncul untuk menjadi jalan keluar dari *game* ini seperti pada gambar.



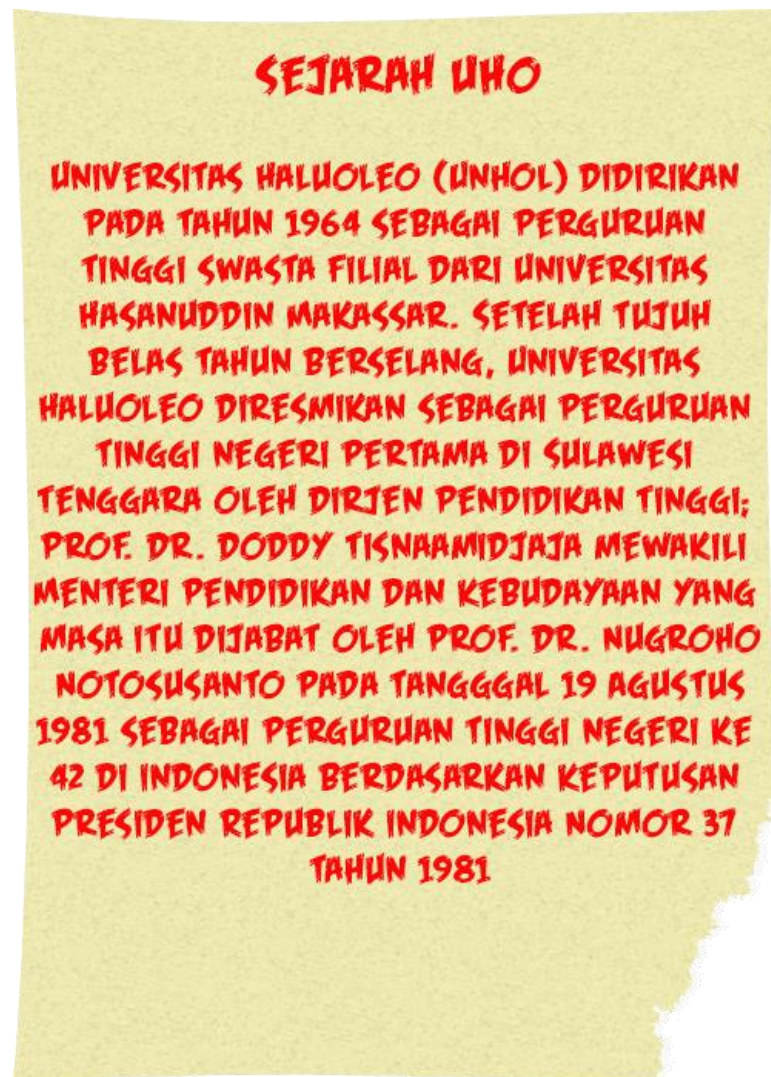
Gambar 5.10 Tampilan *item* informasi Daftar Fakultas



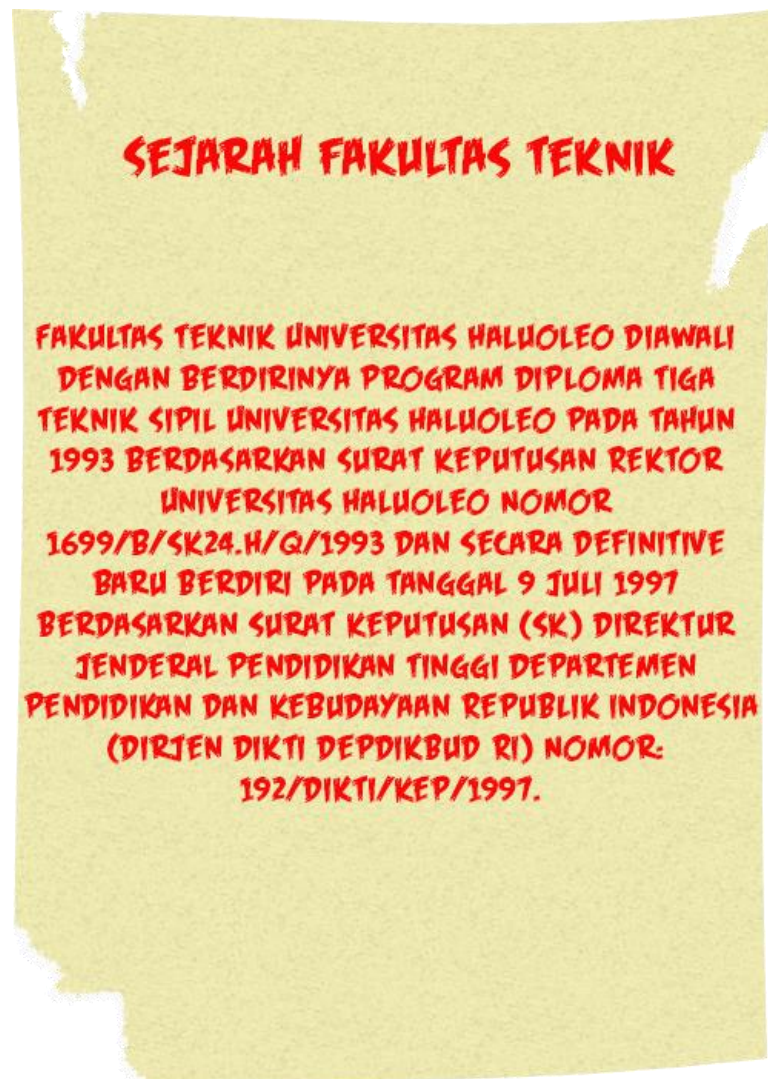
Gambar 5.11 Tampilan *item* informasi Pendiri UHO



Gambar 5.12 Tampilan *item* informasi Rektor UHO



Gambar 5.13 Tampilan *item* Sejarah UHO



Gambar 5.14 Tampilan *item* sejarah Fakultas Teknik

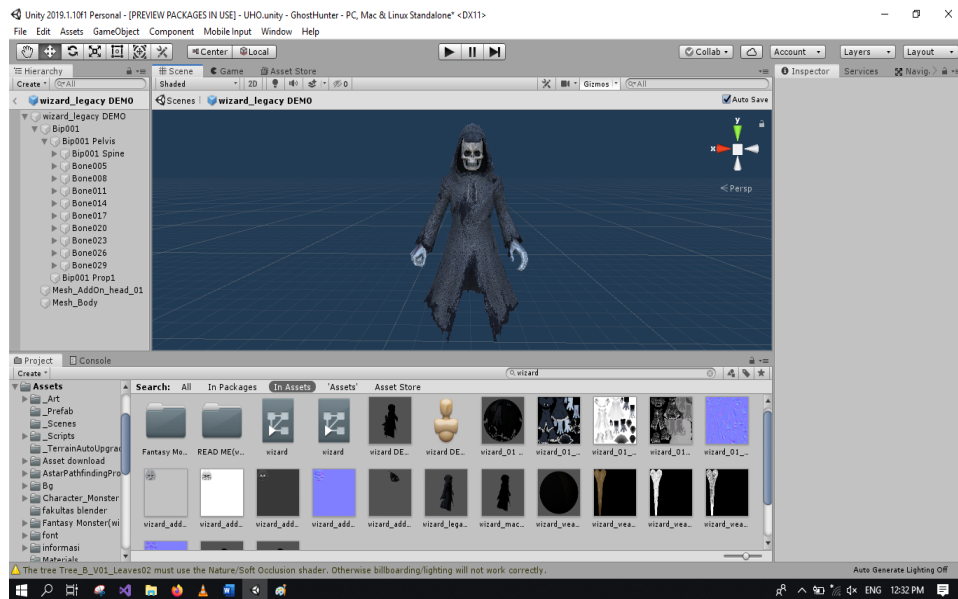
SEJARAH TEKNIK INFORMATIKA

TEKNIK INFORMATIKA DIDIRIKAN PADA TANGGAL 22 JUNI 2007 DENGAN NAMA TEKNIK PERANGKAT LUNAK DENGAN NOMOR SK PENDIRIAN PS 1502/D/T/2007. PENDIRIAN PROGRAM STUDI INI DIDASARI OLEH KEINGINAN UNIVERSITAS HALUOLEO UNTUK MENGAMBIL PERAN DALAM PENGEMBANGAN SDM DENGAN KUALIFIKASI KHUSUS DIBIDANG IT. PADA BULAN AGUSTUS 2007 PROGRAM STUDI TEKNIK INFORMATIKA TELAH RESMI DIBUKA. HINGGA SAAT INI, S1 TEKNIK INFORMATIKA TELAH TERAKREDITASI C.

Gambar 5.15 Tampilan *item* sejarah Teknik Informatika

5.1.9. Tampilan Musuh

Tampilan ini berisi gambar dari *body* musuh yang akan menjadi lawan *player* saat didalam *room game*, musuh akan mengejar *player* dan akan menyerangnya.



Gambar 5.16 Tampilan *body* musuh

5.1.10. Tampilan pertanyaan

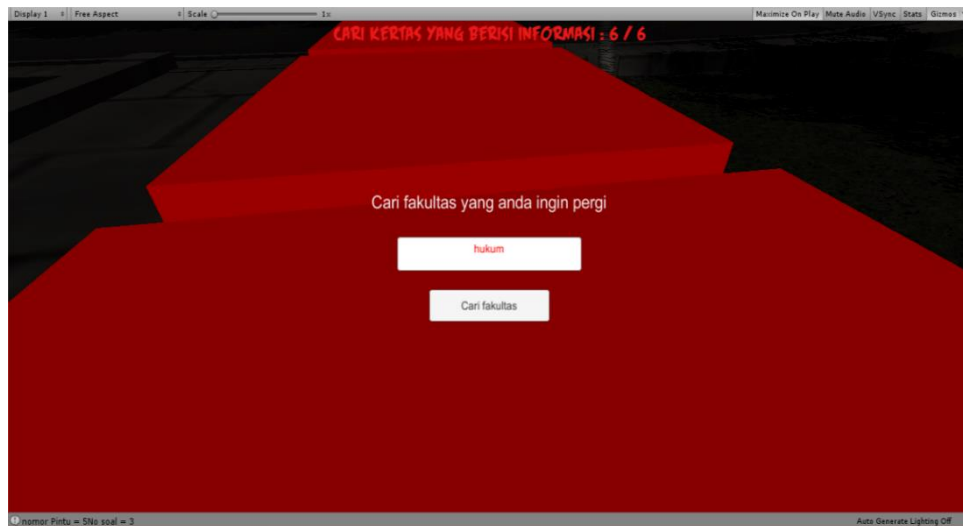
Tampilan ini berisi semua pertanyaan yang akan muncul pada layar *game* ketika *player* sudah berhasil mengumpulkan enam item yang berisi informasi, pertanyaan yang muncul akan bersifat *random* dan jawaban nya ada di antara salah satu dari enam *item* informasi seperti pada gambar 5.17.



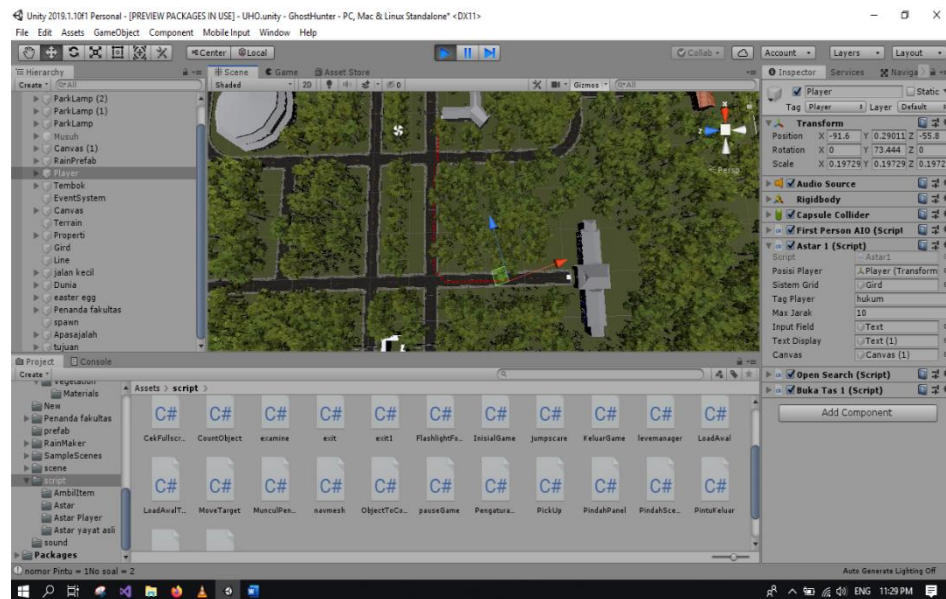
Gambar 5.17 Tampilan pertanyaan

5.1.11. Tampilan Pencarian Fakultas

Pada tampilan ini *player* akan mengisi *form* tujuan yaitu ke gedung yang ingin di tuju, ketika *player* sudah mengisi *form* tersebut maka sistem akan memunculkan rute terpendek dari A* dan *player* bisa mengikuti rute tersebut untuk sampai di gedung tujuan seperti pada gambar 5.18.



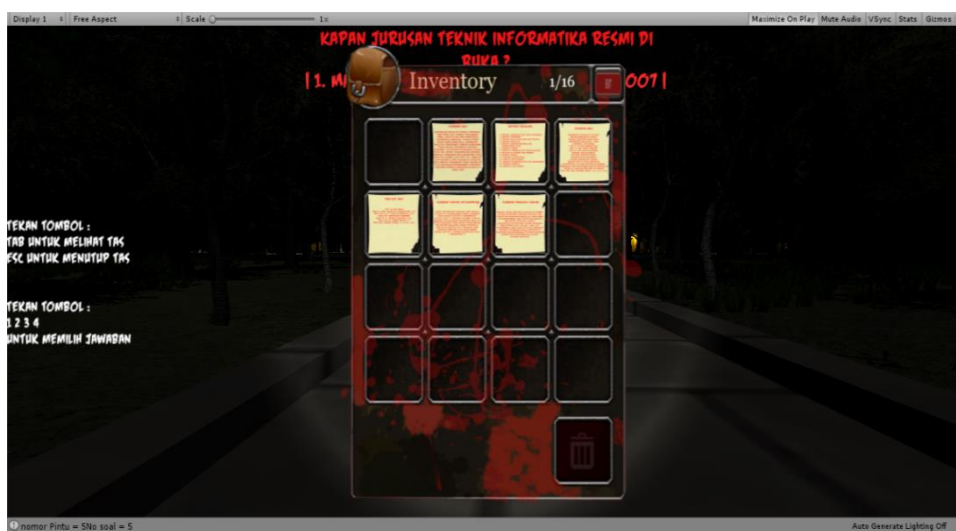
Gambar 5.18 Tampilan pencarian rute terpendek pada gameplay



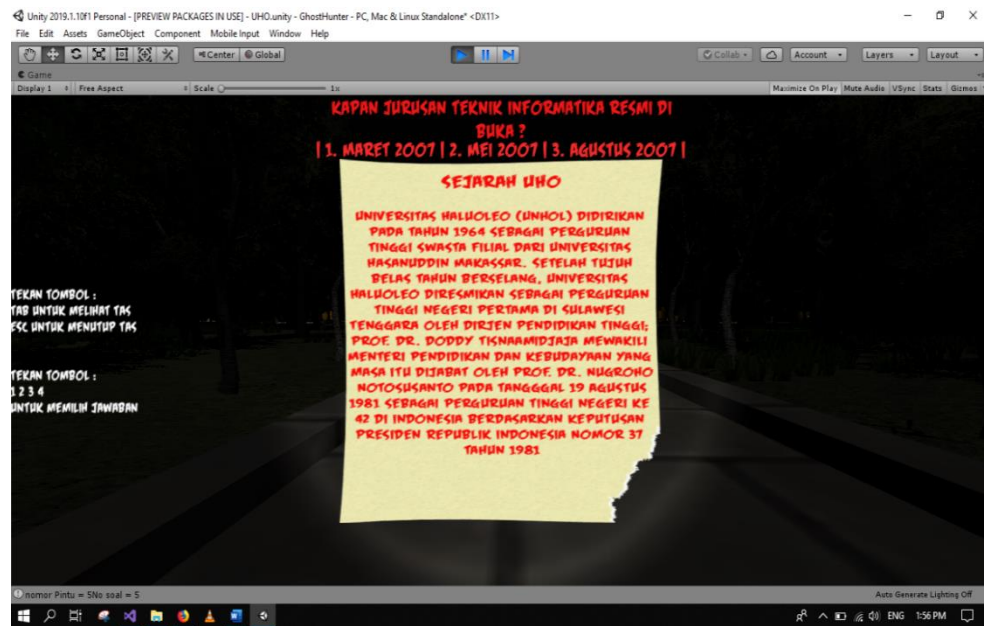
Gambar 5.19 Tampilan pencarian rute terpendek pada scene

5.1.12. Tampilan *Inventory*

Tampilan ini berisi *item* yang berisi informasi yang sudah *player* kumpulkan dan *player* ingin membacanya lagi untuk menjawab pertanyaan yang muncul seperti pada gambar 5.20.



Gambar 5.20 Tampilan *Inventory*



Gambar 5.21 Tampilan salah satu informasi dari *inventory*

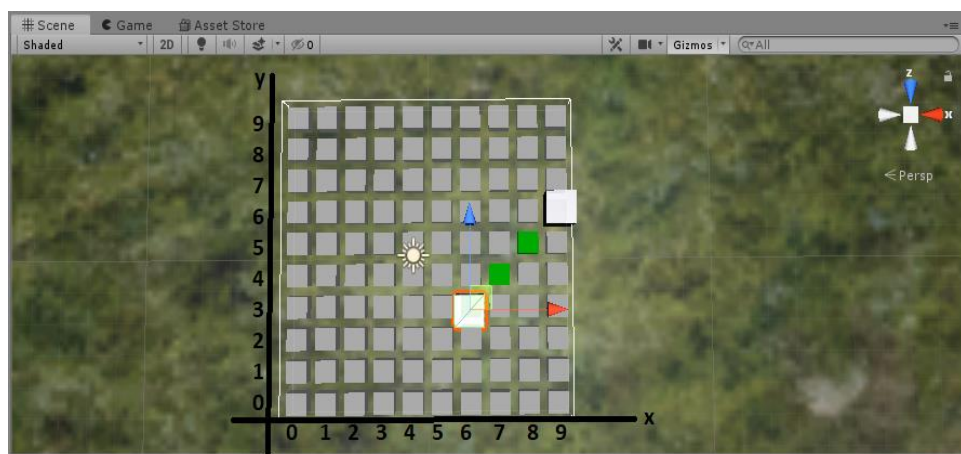
5.2 Pengujian Sistem

Pengujian merupakan tahap yang utama dalam pembuatan suatu aplikasi. Hasil dari pengujian yang didapat akan dijadikan sebagai tolak ukur dalam proses pengembangan selanjutnya.

5.2.1 Pengujian Algoritma A*

Pada tahap ini dilakukan pengujian Algoritma A* yang diterapkan pada *player* untuk mencari rute terpendek menuju fakultas, untuk *enemy* akan berlaku sama saja.

Pada pengujian ini di ambil sampel dengan *grid* 10 x 10, dengan posisi *start node* berada pada koordinat (6,3) dan *target node* berada pada koordinat (9,6), untuk *grid* nya dimulai dari index 0 jadi panjang *grid* x dan y adalah dari 0 sampai 9.



Gambar 5.22 Contoh *grid* 10 x 10 pada game

1. Aturan algoritma A*

A. Menentukan nilai *G cost* dengan menggunakan fungsi jarak manhatan, menggunakan rumus :

a. $\text{jarakX} = (X1 - X2)$

b. $\text{jarakY} = (Y1 - Y2)$

c. $\text{jarak} = \text{jarakX} + \text{jarakY}$

dengan titik yang dihitung adalah titik *current node* dan *node neighbour*.

Nilai tersebut di absolutkan sehingga mendapatkan angka positif. Dan nilai awal *G cost current node* adalah 0.

- B. menentukan Nilai *H cost* dengan menggunakan fungsi jarak manhatan, dengan titik yang dihitung adalah titik *node neighbour* dan titik *node target*.
- C. Setelah mendapatkan nilai *G cost* dan *H cost*, maka langsung jumlahkan dengan menggunakan rumus A^* : $F cost = G cost + H cost$.
- D. Apabila didapatkan nilai *F cost* yang sama, maka *node* yang menjadi rute terpendek adalah *node* dengan *H cost* terkecil.

Untuk menghitung *node* yang ada didalam *grid* maka digunakan aturan diatas, *current node* yang sedang dihitung akan di masukkan ke dalam *openSet* sebagai penyimpanan *node* yang sedang dihitung, ketika *node* tersebut sudah dihitung maka akan di masukkan kedalam *closedSet* agar tidak ada perulangan perhitungan *node* yang terulang.

2. Perhitungan manual A^*

- A. Menentukan posisi *start node* dan *target node*, contoh *start node* (6,3) dan *target node* (9,6) pada koordinat x dan y.
- B. *start node* akan menjadi *current node* di awal.
- C. menentukan *node neighbour* dari *current node* dan mencari *node neighbour* nya.
- D. mencari *G cost* dan *H cost* dari masing-masing *node neighbour*. Untuk $G cost = current node G cost + jarak$.
- E. Mencari *F cost* dengan menjumlah kan *G cost* dan *H cost*.
- F. Setelah mendapatkan nilai *F cost* dari masing *node neighbour*, maka nilai dari *F cost* terkecil adalah merupakan rute terpendek dan akan menjadi *current node*.

Mencari rute pertama, *current node* (6,3):

menentukan *node neighbour* dari *current node* (6,3), sehingga didapatkan *node neighbour* nya adalah (5,2), (5,3), (5,4), (6,2), (6,4), (7,2), (7,3) dan (7,4), dengan nilai *current node G cost* = 0.

Mencari *node neighbour* (5,2)

a. Mencari *G cost node neighbour*(5,2) dengan *current node* (6,3)

$$\text{jarakX} = 5 - 6 = 1$$

$$\text{jarakY} = 2 - 3 = 1$$

$$\text{jarak} = 3 + 1 = 2$$

$$G \text{ cost} = \text{current node } G \text{ cost} + \text{jarak} = 0 + 2 = 2$$

b. Mencari *H cost node neighbour*(5,2) dengan *target node* (9,6)

$$\text{jarakX} = 5 - 9 = 4$$

$$\text{jarakY} = 2 - 6 = 4$$

$$H \text{ cost} = 4 + 4 = 8$$

c. Mencari *F cost*

$$F \text{ cost} = G \text{ cost} + H \text{ cost} = 2 + 8 = 10$$

Mencari *node neighbour* (5,3)

a. Mencari *G cost node neighbour*(5,3) dengan *current node* (6,3)

$$\text{jarakX} = 5 - 6 = 1$$

$$\text{jarakY} = 3 - 3 = 0$$

$$\text{jarak} = 1 + 0 = 1$$

$$G \text{ cost} = \text{current node } G \text{ cost} + \text{jarak} = 0 + 1 = 1$$

b. Mencari *H cost node neighbour*(5,3) dengan *target node* (9,6)

$$\text{jarakX} = 5 - 9 = 4$$

$$\text{jarakY} = 3 - 6 = 3$$

$$H \text{ cost} = 4 + 3 = 7$$

c. Mencari *F cost*

$$F \text{ cost} = G \text{ cost} + H \text{ cost} = 1 + 7 = 8$$

Mencari *node neighbour* (5,4)

a. Mencari *G cost node neighbour*(5,4) dengan *current node* (6,3)

$$\text{jarakX} = 5 - 6 = 1$$

$$\text{jarakY} = 4 - 3 = 1$$

$$\text{jarak} = 1 + 1 = 2$$

$$G \text{ cost} = \text{current node } G \text{ cost} + \text{jarak} = 0 + 2 = 2$$

- b. Mencari *H cost node neighbour*(5,4) dengan *target node* (9,6)

$$\text{jarakX} = 5 - 9 = 4$$

$$\text{jarakY} = 4 - 6 = 2$$

$$H \text{ cost} = 4 + 2 = 6$$

- c. Mencari *F cost*

$$F \text{ cost} = G \text{ cost} + H \text{ cost} = 2 + 6 = 8$$

Mencari *node neighbour* (6,2)

- a. Mencari *G cost node neighbour*(6,2) dengan *current node* (6,3)

$$\text{jarakX} = 6 - 6 = 0$$

$$\text{jarakY} = 2 - 3 = 1$$

$$\text{jarak} = 0 + 1 = 1$$

$$G \text{ cost} = \text{current node } G \text{ cost} + \text{jarak} = 0 + 1 = 1$$

- b. Mencari *H cost node neighbour*(6,2) dengan *target node* (9,6)

$$\text{jarakX} = 6 - 9 = 3$$

$$\text{jarakY} = 2 - 6 = 4$$

$$H \text{ cost} = 3 + 4 = 7$$

- c. Mencari *F cost*

$$F \text{ cost} = G \text{ cost} + H \text{ cost} = 1 + 7 = 8$$

Mencari *node neighbour* (6,4)

- a. Mencari *G cost node neighbour*(6,4) dengan *current node* (6,3)

$$\text{jarakX} = 6 - 6 = 0$$

$$\text{jarakY} = 4 - 3 = 1$$

$$\text{jarak} = 0 + 1 = 1$$

$$G \text{ cost} = \text{current node } G \text{ cost} + \text{jarak} = 0 + 1 = 1$$

- b. Mencari *H cost node neighbour*(6,4) dengan *target node* (9,6)

$$\text{jarakX} = 6 - 9 = 3$$

$$\text{jarakY} = 4 - 6 = 2$$

$$H \text{ cost} = 3 + 2 = 5$$

- c. Mencari *F cost*

$$F \text{ cost} = G \text{ cost} + H \text{ cost} = 1 + 5 = 6$$

Mencari *node neighbour* (7,2)

a. Mencari *G cost node neighbour*(7,2) dengan *current node* (6,3)

$$\text{jarakX} = 7 - 6 = 1$$

$$\text{jarakY} = 2 - 3 = 1$$

$$\text{jarak} = 1 + 1 = 2$$

$$G \text{ cost} = \text{current node } G \text{ cost} + \text{jarak} = 0 + 2 = 2$$

b. Mencari *H cost node neighbour*(7,2) dengan *target node* (9,6)

$$\text{jarakX} = 7 - 9 = 2$$

$$\text{jarakY} = 2 - 6 = 4$$

$$H \text{ cost} = 2 + 4 = 6$$

c. Mencari *F cost*

$$F \text{ cost} = G \text{ cost} + H \text{ cost} = 2 + 6 = 8$$

Mencari *node neighbour* (7,3)

a. Mencari *G cost node neighbour*(7,3) dengan *current node* (6,3)

$$\text{jarakX} = 7 - 6 = 1$$

$$\text{jarakY} = 3 - 3 = 0$$

$$\text{jarak} = 1 + 0 = 1$$

$$G \text{ cost} = \text{current node } G \text{ cost} + \text{jarak} = 0 + 1 = 1$$

b. Mencari *H cost node neighbour*(7,3) dengan *target node* (9,6)

$$\text{jarakX} = 7 - 9 = 2$$

$$\text{jarakY} = 3 - 6 = 3$$

$$H \text{ cost} = 2 + 3 = 5$$

c. Mencari *F cost*

$$F \text{ cost} = G \text{ cost} + H \text{ cost} = 1 + 5 = 6$$

Mencari *node neighbour* (7,4)

a. Mencari *G cost node neighbour*(7,4) dengan *current node* (6,3)

$$\text{jarakX} = 7 - 6 = 1$$

$$\text{jarakY} = 4 - 3 = 1$$

$$\text{jarak} = 1 + 1 = 2$$

$$G \text{ cost} = \text{current node } G \text{ cost} + \text{jarak} = 0 + 2 = 2$$

b. Mencari *H cost node neighbour*(7,4) dengan *target node* (9,6)

$$\text{jarakX} = 7 - 9 = 2$$

$$\text{jarakY} = 4 - 6 = 2$$

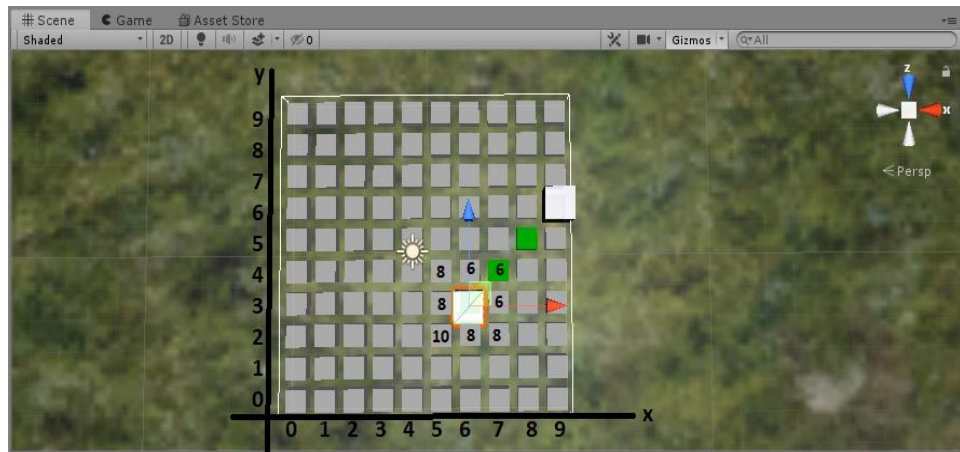
$$H \text{ cost} = 2 + 2 = 4$$

c. Mencari *F cost*

$$F \text{ cost} = G \text{ cost} + H \text{ cost} = 2 + 4 = 6$$

Tabel 5.1 Hasil perhitungan manual rute pertama

No.	Node(x,y)	G cost	H cost	F cost = G cost + H cost
1	5,2	2	8	10
2	5,3	1	7	8
3	5,4	2	6	8
4	6,2	1	7	8
5	6,4	1	5	6
6	7,2	2	6	8
7	7,3	1	5	6
8	7,4	2	4	6



Gambar 5.23 Hasil perhitungan rute pertama

Node dengan *F cost* terkecil adalah *node* (7,4), *node* ini akan menjadi *current node* dengan nilai *G cost* = 2. Nilai ini akan ditambahkan pada nilai *G cost* dari *node neighbour* selanjutnya.

Mencari rute kedua, *current node* (7,4):

menentukan *node neighbour* dari *current node* (7,4), sehingga didapatkan *node neighbour* nya adalah (8,3), (8,4), (8,5), (7,5) dan (6,5).

Mencari *node neighbour* (8,3), dengan nilai *current node G cost* = 2.

- a. Mencari *G cost node neighbour* (8,3) dengan *current node* (7,4)

$$\text{jarakX} = 8 - 7 = 1$$

$$\text{jarakY} = 3 - 4 = 1$$

$$\text{jarak} = 1 + 1 = 2$$

$$G \text{ cost} = \text{current node } G \text{ cost} + \text{jarak} = 2 + 2 = 4$$

- b. Mencari *H cost node neighbour*(7,4) dengan *target node* (9,6)

$$\text{jarakX} = 7 - 9 = 2$$

$$\text{jarakY} = 4 - 6 = 2$$

$$H \text{ cost} = 2 + 2 = 4$$

- c. Mencari *F cost*

$$F \text{ cost} = G \text{ cost} + H \text{ cost} = 4 + 4 = 8$$

Mencari *node neighbour* (8,4), dengan nilai *current node G cost* = 2.

- a. Mencari *G cost node neighbour* (8,4) dengan *current node* (7,4)

$$\text{jarakX} = 8 - 7 = 1$$

$$\text{jarakY} = 4 - 4 = 0$$

$$\text{jarak} = 1 + 0 = 1$$

$$G \text{ cost} = \text{current node } G \text{ cost} + \text{jarak} = 2 + 1 = 3$$

- b. Mencari *H cost node neighbour*(8,4) dengan *target node* (9,6)

$$\text{jarakX} = 8 - 9 = 1$$

$$\text{jarakY} = 4 - 6 = 2$$

$$H \text{ cost} = 1 + 2 = 3$$

- c. Mencari *F cost*

$$F \text{ cost} = G \text{ cost} + H \text{ cost} = 3 + 3 = 6$$

Mencari *node neighbour* (8,5), dengan nilai *current node G cost* = 2.

- a. Mencari *G cost node neighbour* (8,5) dengan *current node* (7,4)

$$\text{jarakX} = 8 - 7 = 1$$

$$\text{jarakY} = 5 - 4 = 1$$

$$\text{jarak} = 1 + 1 = 2$$

$$G \text{ cost} = \text{current node } G \text{ cost} + \text{jarak} = 2 + 2 = 4$$

b. Mencari *H cost node neighbour*(8,5) dengan *target node* (9,6)

$$\text{jarakX} = 8 - 9 = 1$$

$$\text{jarakY} = 5 - 6 = 1$$

$$H \text{ cost} = 1 + 1 = 2$$

c. Mencari *F cost*

$$F \text{ cost} = G \text{ cost} + H \text{ cost} = 4 + 2 = 6$$

Mencari *node neighbour* (7,5), dengan nilai *current node G cost* = 2.

a. Mencari *G cost node neighbour* (7,5) dengan *current node* (7,4)

$$\text{jarakX} = 7 - 7 = 0$$

$$\text{jarakY} = 5 - 4 = 1$$

$$\text{jarak} = 0 + 1 = 1$$

$$G \text{ cost} = \text{current node } G \text{ cost} + \text{jarak} = 2 + 1 = 3$$

b. Mencari *H cost node neighbour*(7,5) dengan *target node* (9,6)

$$\text{jarakX} = 7 - 9 = 2$$

$$\text{jarakY} = 5 - 6 = 1$$

$$H \text{ cost} = 2 + 1 = 3$$

c. Mencari *F cost*

$$F \text{ cost} = G \text{ cost} + H \text{ cost} = 3 + 4 = 6$$

Mencari *node neighbour* (6,5), dengan nilai *current node G cost* = 2.

a. Mencari *G cost node neighbour* (6,5) dengan *current node* (7,4)

$$\text{jarakX} = 6 - 7 = 1$$

$$\text{jarakY} = 5 - 4 = 1$$

$$\text{jarak} = 1 + 1 = 2$$

$$G \text{ cost} = \text{current node } G \text{ cost} + \text{jarak} = 2 + 2 = 4$$

b. Mencari *H cost node neighbour*(6,5) dengan *target node* (9,6)

$$\text{jarakX} = 6 - 9 = 3$$

$$\text{jarakY} = 5 - 6 = 1$$

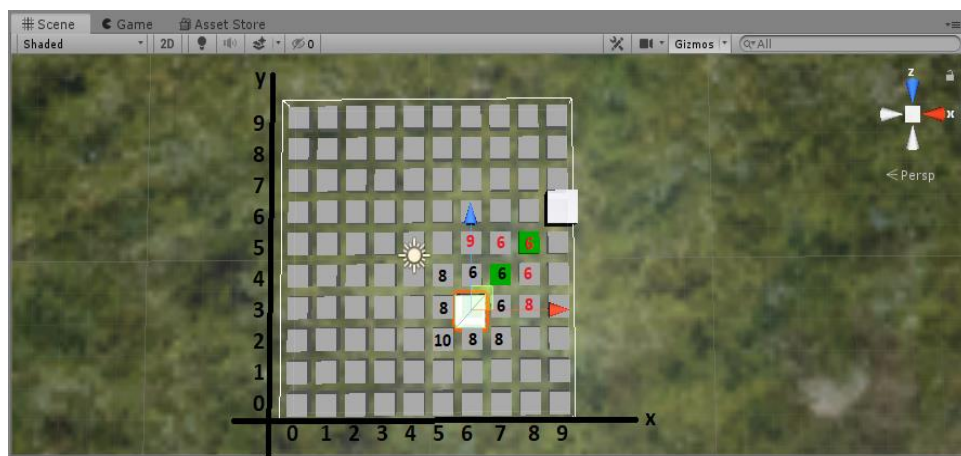
$$H \text{ cost} = 4 + 1 = 5$$

c. Mencari *F cost*

$$F \text{ cost} = G \text{ cost} + H \text{ cost} = 4 + 5 = 9$$

Tabel 5.2 Hasil perhitungan manual rute kedua

No.	Node(x,y)	G cost	H cost	F cost = G cost + H cost
1	8,3	4	4	8
2	8,4	3	3	6
3	8,5	4	2	6
4	7,5	3	3	6
5	6,5	5	4	9



Gambar 5.24 Hasil perhitungan rute kedua

Node dengan *F cost* terkecil adalah *node* (8,5), *node* ini akan menjadi *current node* dengan nilai *G cost* = 4. Nilai ini akan ditambahkan pada nilai *G cost* dari *node neighbour* selanjutnya.

Mencari rute ketiga, *current node* (8,5):

menentukan *node neighbour* dari *current node* (8,5), sehingga didapatkan *node neighbour* nya adalah (7,6), (8,6), (9,4), (9,5) dan (9,6).

Mencari *node neighbour* (7,6), dengan nilai *current node G cost* = 4.

- a. Mencari *G cost node neighbour* (7,6) dengan *current node* (8,5)

$$\text{jarakX} = 7 - 8 = 1$$

$$\text{jarakY} = 6 - 5 = 1$$

$$\text{jarak} = 1 + 1 = 2$$

$$G \text{ cost} = \text{current node } G \text{ cost} + \text{jarak} = 4 + 2 = 6$$

- b. Mencari *H cost node neighbour*(7,6) dengan *target node* (9,6)

$$\text{jarakX} = 7 - 9 = 2$$

$$\text{jarakY} = 6 - 6 = 0$$

$$H \text{ cost} = 2 + 0 = 2$$

- c. Mencari *F cost*

$$F \text{ cost} = G \text{ cost} + H \text{ cost} = 6 + 2 = 8$$

Mencari *node neighbour* (8,6), dengan nilai *current node G cost* = 4.

- a. Mencari *G cost node neighbour* (8,6) dengan *current node* (8,5)

$$\text{jarakX} = 8 - 8 = 0$$

$$\text{jarakY} = 6 - 5 = 1$$

$$\text{jarak} = 0 + 1 = 1$$

$$G \text{ cost} = \text{current node } G \text{ cost} + \text{jarak} = 4 + 1 = 5$$

- b. Mencari *H cost node neighbour*(8,6) dengan *target node* (9,6)

$$\text{jarakX} = 8 - 9 = 1$$

$$\text{jarakY} = 6 - 6 = 0$$

$$H \text{ cost} = 1 + 0 = 1$$

- c. Mencari *F cost*

$$F \text{ cost} = G \text{ cost} + H \text{ cost} = 5 + 1 = 6$$

Mencari *node neighbour* (9,4), dengan nilai *current node G cost* = 4.

- a. Mencari *G cost node neighbour* (9,4) dengan *current node* (8,5)

$$\text{jarakX} = 9 - 8 = 1$$

$$\text{jarakY} = 4 - 5 = 1$$

$$\text{jarak} = 1 + 1 = 2$$

$$G \text{ cost} = \text{current node } G \text{ cost} + \text{jarak} = 4 + 2 = 6$$

b. Mencari *H cost node neighbour*(9,4) dengan *target node* (9,6)

$$\text{jarakX} = 9 - 9 = 0$$

$$\text{jarakY} = 4 - 6 = 2$$

$$H \text{ cost} = 0 + 2 = 2$$

c. Mencari *F cost*

$$F \text{ cost} = G \text{ cost} + H \text{ cost} = 6 + 2 = 8$$

Mencari *node neighbour* (9,5), dengan nilai *current node G cost* = 4.

a. Mencari *G cost node neighbour* (9,5) dengan *current node* (8,5)

$$\text{jarakX} = 9 - 8 = 1$$

$$\text{jarakY} = 5 - 5 = 0$$

$$\text{jarak} = 1 + 0 = 1$$

$$G \text{ cost} = \text{current node } G \text{ cost} + \text{jarak} = 4 + 1 = 5$$

b. Mencari *H cost node neighbour*(9,5) dengan *target node* (9,6)

$$\text{jarakX} = 9 - 9 = 0$$

$$\text{jarakY} = 5 - 6 = 1$$

$$H \text{ cost} = 0 + 1 = 1$$

c. Mencari *F cost*

$$F \text{ cost} = G \text{ cost} + H \text{ cost} = 5 + 1 = 6$$

Mencari *node neighbour* (9,6), dengan nilai *current node G cost* = 4.

a. Mencari *G cost node neighbour* (9,6) dengan *current node* (8,5)

$$\text{jarakX} = 9 - 8 = 1$$

$$\text{jarakY} = 6 - 5 = 1$$

$$\text{jarak} = 1 + 1 = 2$$

$$G \text{ cost} = \text{current node } G \text{ cost} + \text{jarak} = 4 + 2 = 6$$

b. Mencari *H cost node neighbour*(9,6) dengan *target node* (9,6)

$$\text{jarakX} = 9 - 9 = 0$$

$$\text{jarakY} = 6 - 6 = 0$$

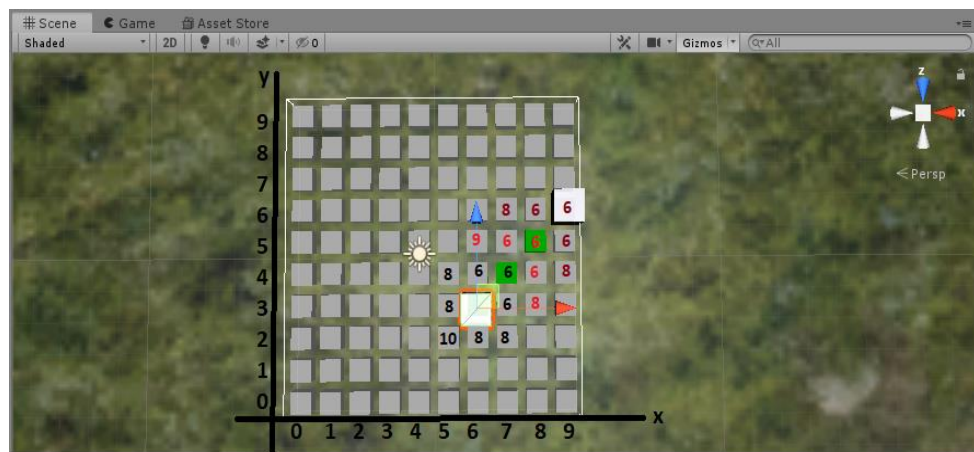
$$H \text{ cost} = 0 + 0 = 0$$

c. Mencari $F cost$

$$F cost = G cost + H cost = 6 + 0 = 6$$

Tabel 5.3 Hasil perhitungan manual rute ketiga

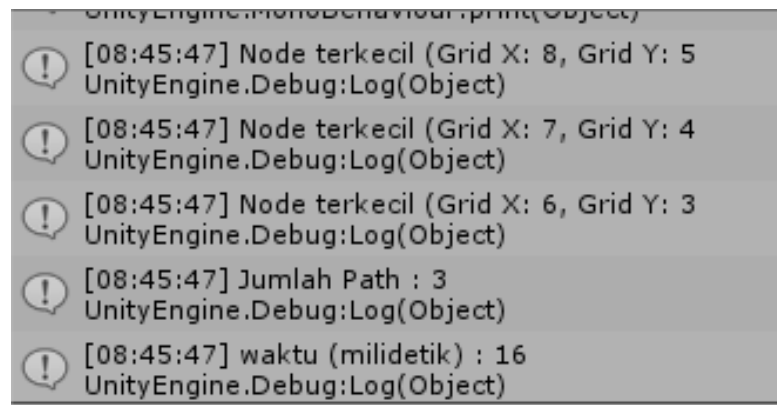
No.	Node(x,y)	$G cost$	$H cost$	$F cost = G cost + H cost$
1	7,6	6	2	8
2	8,6	5	1	6
3	9,4	6	2	8
4	9,5	5	1	6
5	9,6	6	0	6



Gambar 5.25 Hasil perhitungan rute ketiga

Node dengan $F cost$ terkecil adalah node (9,6), kemudian di cek apakah *current node* sama dengan *target node*, apabila sama maka pencarian berhenti.

Node dengan F cost terkecil di dapatkan adalah node (6,3), (7,4) dan (8,5), dengan jumlah path adalah 3 dan waktu pencarian 16 milidetik.



Gambar 5.26 Node yang didapat sebagai rute terpendek

5.4 Implementasi *Artificial Intelligence* pada *enemy*

Berisi source code yang akan membentuk *Artificial Intelligence* musuh pada *game* ini.

```
public class AturanNPC : MonoBehaviour
{
    // Use this for initialization
    Transform dicari;
    public string tagPlayer = "Player";
    public float putaran = 1f;
    Animation anim;
    private float jarakPlayer = 0f;
    public float jarakJalan = 70f;
    public float jarakSerang = 1f;
    public float jarakSuara = 20f;
    public static int nyawa = 3;

    AudioSource audioData;
    NavMeshAgent agent;
    void Start()
    {
        anim = GetComponent<Animation>();
        audioData = GetComponent<AudioSource>();
        audioData.Play(0);
        Debug.Log("started");
        agent = GetComponent<NavMeshAgent>();
    }
    void Update()
    {
        if (GameObject.FindGameObjectWithTag(tagPlayer))
        {
            dicari =
GameObject.FindGameObjectWithTag(tagPlayer).GetComponent<Transform>();
            jarakPlayer = Vector3.Distance(dicari.position,
transform.position);
        }
        else
        {
            Debug.Log("player tidak ditemukan");
            jarakPlayer = 0f;
        }
    }
}
```

```

if (GameObject.FindGameObjectWithTag(tagPlayer))
{
    if (jarakPlayer < jarakJalan)
    {
        arahPlayer();
        jalan();
        audioData.Pause();
        agent.SetDestination(dicari.position);
    }

    if (jarakPlayer <= jarakSerang)
    {
        serang();
    }
    if (jarakPlayer <= jarakSuara)
    {
        audioData.UnPause();
    }
    else
    {
        audioData.Pause();
    }

}
else
{
    idle();
    audioData.Pause();
    Debug.Log("Player tidak ketemu");
}
if (nyawa <= 0)
{
    ObjectToCollect.objects = 0;
    Cursor.visible = true;
    SceneManager.LoadScene("GameOver");
}
}
void arahPlayer()
{
    Quaternion rotasi = Quaternion.LookRotation(dicari.position -
transform.position);
    transform.rotation = Quaternion.Slerp(transform.rotation,
rotasi, Time.deltaTime * putaran);
}

```

```
void jalan()
{
    anim.GetComponent<Animation>().wrapMode = WrapMode.Loop;
    anim.GetComponent<Animation>().CrossFade("");
}

void idle()
{
    anim.GetComponent<Animation>().wrapMode = WrapMode.Loop;
    anim.GetComponent<Animation>().CrossFade("");
}

void serang()
{
    anim.GetComponent<Animation>().wrapMode = WrapMode.Loop;
    anim.GetComponent<Animation>().CrossFade("");
}
private void OnTriggerEnter(Collider Player)
{
    if (Player.gameObject.tag == "Player")
    {
        nyawa = nyawa-1;
    }
}
}
```

BAB VI

PENUTUP

6.1 Kesimpulan

Kesimpulan yang dapat diambil dari penulisan tugas akhir ini adalah sebagai berikut :

1. Untuk menentukan jalur terpendek pada pencarian ruangan dengan menggunakan algoritma *All star*, dibutuhkan data yaitu, posisi *node* awal dan *node* target. Dengan data tersebut, maka pencarian jalur terpendek pada dapat diimplementasikan.
2. Aplikasi *Game Ghost Adventure* dibangun menggunakan aplikasi *Unity* dengan bahasa pemrograman C# agar dapat diimplementasikan pada sebuah game *3D simulator* dengan sistem operasi *Windows*.
3. Kelebihan algoritma *A** pada sistem ini yaitu dapat mencari rute terpendek menuju ke fakultas yang ingin dituju, ketika ada penghalang di jalan maka sistem otomatis akan menghitung otomatis *rute* selanjutnya.
4. Kelemahan algoritma *A** pada sistem yaitu apabila terlalu banyak jumlah grid yang dibuat maka komputer atau laptop akan lemot dikarenakan *A** pada game ini akan terus *terupdate* dalam mencari rute.
5. Rute yang terbuat tidak dapat mengikuti alur jika jalan melengkung.

6.2 Saran

Saran yang perlu diperhatikan untuk penelitian lebih lanjut adalah :

1. Algoritma yang digunakan pada penelitian dapat dikembangkan dengan algoritma lain. Sehingga dapat dibandingkan kinerjanya dalam pencarian jalur terpendek.
2. Aplikasi pencarian jalur terpendek dapat dikembangkan dengan memanfaatkan fitur *Virtual Reality* dengan menggunakan *smartphone*

android atau menggunakan *Oculus Rift* sebagai medianya. Sehingga dapat memberikan kesan yang lebih nyata pada aplikasi *Game*.

LAMPIRAN

Script Algoritma All star :

Node.cs

```
public class Node
{
    public bool walkable;
    public Vector3 worldPosition;
    public int gridX;
    public int gridY;

    public int gCost;
    public int hCost;
    public Node parent;

    public Node(bool _walkable, Vector3 _worldPos, int _gridX, int
_gridY)
    {
        walkable = _walkable;
        worldPosition = _worldPos;
        gridX = _gridX;
        gridY = _gridY;
    }

    public int fCost
    {
        get
        {
            return gCost + hCost;
        }
    }
}
```

Grid.cs

```
public class Grid : MonoBehaviour
{
    public LayerMask halangan;
    public Vector2 gridWorldSize;
    public float nodeRadius;
    Node[,] grid;

    float nodeDiameter;
    int gridSizeX, gridSizeY;

    void Start()
    {
        nodeDiameter = nodeRadius * 2;
        gridSizeX = Mathf.RoundToInt(gridWorldSize.x / nodeDiameter);
        gridSizeY = Mathf.RoundToInt(gridWorldSize.y / nodeDiameter);
    }

    void Update()
    {
        grid = new Node[gridSizeX, gridSizeY];
        Vector3 worldBottomLeft = transform.position - Vector3.right *
            gridWorldSize.x / 2
            - Vector3.forward * gridWorldSize.y / 2;

        for (int x = 0; x < gridSizeX; x++)
        {
            for (int y = 0; y < gridSizeY; y++)
            {
                Vector3 worldPoint = worldBottomLeft + Vector3.right *
                    (x * nodeDiameter + nodeRadius)
                    + Vector3.forward * (y * nodeDiameter +
                    nodeRadius);
                bool walkable = !(Physics.CheckSphere(worldPoint,
                    nodeRadius, halangan));
                grid[x, y] = new Node(walkable, worldPoint, x, y);
            }
        }
    }

    public List<Node> GetNeighbours(Node node)
    {
        List<Node> neighbours = new List<Node>();

        for (int x = -1; x <= 1; x++)
        {
            for (int y = -1; y <= 1; y++)
            {
                if (x == 0 && y == 0)
                    continue;
            }
        }
    }
}
```

Lanjutan

```
int checkX = node.gridX + x;
    int checkY = node.gridY + y;

    if (checkX >= 0 && checkX < gridSizeX && checkY > 0 &&
checkY <= gridSizeY)
    {
        neighbours.Add(grid[checkX, checkY]);
    }
}

return neighbours;
}

public Node NodeFromWorldPoint(Vector3 worldPosition)
{
    float percentX = (worldPosition.x + gridWorldSize.x / 2) /
gridWorldSize.x;
    float percentY = (worldPosition.z + gridWorldSize.y / 2) /
gridWorldSize.y;
    percentX = Mathf.Clamp01(percentX);
    percentY = Mathf.Clamp01(percentY);

    int x = Mathf.RoundToInt((gridSizeX - 1) * percentX);
    int y = Mathf.RoundToInt((gridSizeY - 1) * percentY);
    return grid[x, y];
}

public List<Node> path;
void OnDrawGizmos()
{
    Gizmos.DrawWireCube(transform.position, new
Vector3(gridWorldSize.x, 1, gridWorldSize.y));

    if (grid != null)
    {
        foreach (Node n in grid)
        {
            if (n.walkable)//If the current node is a wall node
            {
                Gizmos.color = Color.clear;//Set the color of the
node

            }

            else
            {
                Gizmos.color = Color.clear;//Set the color of the
node

            }
        }
    }
}
```

Lanjutan

```
if (path != null) //If the final path is not empty
{
    if (path.Contains(n)) //If the current node is in
the final path
    {
        Gizmos.color = Color.red; //Set the color of
that node
    }
}

Gizmos.DrawCube(n.worldPosition, Vector3.one *
(nodeDiameter - nodeRadius)); //Draw the node at the position of the
node.
}
}
```

Astar.cs

```
public class Astar1 : MonoBehaviour
{
    Transform dicari;
    public Transform PosisiPlayer;
    public GameObject sistemGrid;
    public string Target = "";
    public int maxJarak = 10;
    public GameObject inputField;
    public GameObject textDisplay;
    public GameObject canvas;
    public Stopwatch timer;

    Grid grid;

    public List<Node> path;
    Node[,] gridjalur;
    void Awake()
    {
        timer = new Stopwatch();
        grid = sistemGrid.GetComponent<Grid>();
    }
    void Start()
    {
    }
    public void cariFakultas() {
        Target = inputField.GetComponent<Text>().text;
        if (GameObject.FindGameObjectWithTag(Target))
        {
            dicari =
GameObject.FindGameObjectWithTag(Target).GetComponent<Transform>();
        }

        if (this.gameObject && dicari)
        {
            FindPath(dicari.position, PosisiPlayer.transform.position);
        }

        if (Input.GetButtonDown("Fire2")) {
            canvas.SetActive(false);
            Time.timeScale = 1;
        }
    }

    void Update()
    {
        cariFakultas();
    }
}
```

Lanjutan

```
void FindPath(Vector3 startPos, Vector3 targetPos)
{
    timer.Reset();
    timer.Start();
    Node startNode = grid.NodeFromWorldPoint(startPos);
    Node targetNode = grid.NodeFromWorldPoint(targetPos);

    UnityEngine.Debug.Log("Posisi Awal Player: " +
PosisiPlayer.transform.position);
    UnityEngine.Debug.Log("Posisi Gedung / Fakultas: " +
dicari.transform.position);

    List<Node> openSet = new List<Node>();
    HashSet<Node> closedSet = new HashSet<Node>();
    openSet.Add(startNode);

    while (openSet.Count > 0)
    {
        Node currentNode = openSet[0];
        for (int i = 1; i < openSet.Count; i++)
        {
            if (openSet[i].fCost < currentNode.fCost ||
openSet[i].fCost == currentNode.fCost
            && openSet[i].hCost < currentNode.hCost)
            {
                currentNode = openSet[i];
            }
        }

        openSet.Remove(currentNode);
        closedSet.Add(currentNode);

        if (currentNode == targetNode)
        {
            RetracePath(startNode, targetNode);
            timer.Stop();
            UnityEngine.Debug.Log("waktu (milidetik) : " +
timer.ElapsedMilliseconds);
            UnityEngine.Debug.Log("- - - - -");
            return;
        }

        foreach (Node neighbour in grid.GetNeighbours(currentNode))
        {
            if (!neighbour.walkable ||
closedSet.Contains(neighbour))
            {
                continue;
            }
        }
    }
}
```

Lanjutan

```
int newMovementCostToNeighbour = currentNode.gCost +
GetDistance(currentNode, neighbour);
    if (newMovementCostToNeighbour < neighbour.gCost ||
!openSet.Contains(neighbour))
    {
        neighbour.gCost = newMovementCostToNeighbour;
        neighbour.hCost = GetDistance(neighbour,
targetNode);
        neighbour.parent = currentNode;

        if (!openSet.Contains(neighbour))
            openSet.Add(neighbour);
    }
}

void RetracePath(Node startNode, Node endNode)
{
    List<Node> path = new List<Node>();
    Node currentNode = endNode;
    Vector3 posisiNode = new Vector3(1.0f, 1.0f, 1.0f);

    while (currentNode != startNode)
    {
        path.Add(currentNode);
        currentNode = currentNode.parent;
        UnityEngine.Debug.Log("Node (Grid X: " + currentNode.gridX
+ ", Grid Y: " + currentNode.gridY);
        posisiNode = currentNode.worldPosition - new Vector3(-0.5f,
0.1f, -0.5f);
        UnityEngine.Debug.Log("Posisi Node:" + posisiNode);
    }
    path.Reverse();
    UnityEngine.Debug.Log("Jumlah Path : " + path.Count);
    grid.path = path;
}

int GetDistance(Node nodeA, Node nodeB)
{
    int dstX = Mathf.Abs(nodeA.gridX - nodeB.gridX);
    int dstY = Mathf.Abs(nodeA.gridY - nodeB.gridY);

    if (dstX > dstY)
        return 20 * dstY + 10 * (dstX - dstY);

    return 20 * dstX + 10 * (dstY - dstX);
}

}
```

DAFTAR PUSTAKA

- Ayu, S. F., Sutardi, & Tajidun, L. M. (2017). *Rancang Bangun Game Edukasi Puzzle Kebudayaan Sulawesi Tenggara Dengan Algoritma Fisher-Yates Shuffle*. pp. 29–38. Retrieved from <http://ojs.uho.ac.id/index.php/semantik/article/view/2588>
- Fajar, M., Ningrum, I. P., & Ramadhan, R. (2015). *Membangun aplikasi game “dua satu (21)” berbasis android menggunakan metode depth first search*. pp. 63–68. Retrieved from <http://ojs.uho.ac.id/index.php/semantik/article/view/494>
- Hayati, E. N., & Yohanes, A. (2014). *PENCARIAN RUTE TERPENDEK MENGGUNAKAN ALGORITMA GREEDY* Enty Nur Hayati 1 , Antoni Yohanes 2 1,2. pp. 391–397. Retrieved from <https://publikasiilmiah.ums.ac.id/handle/11617/4535>
- Hermawan, L., & Bendi, R. K. J. (2013). Penerapan Algoritma A * pada Aplikasi Puzzle. *Seminar Nasional Teknologi Informasi Dan Komunikasi*, (Snastikom), 23–28.
- Marwan, A., Sutardi, & Ramadhan, R. (2015). Penerapan Metode Linear Congruent Method (Lcm) Dalam Perancangan Dan Pembuatan Game Monopoli Edukasi Untuk Tokoh Pahlawan Nasional. *Bianglala Informatika*, pp. 229–236. Retrieved from <http://ojs.uho.ac.id/index.php/semantik/article/view/2585>
- Pamungkas, A., Widiyanto, E. P., & Angreni, R. (2011). Penerapan Algoritma A* (A Star) Pada Game Edukasi The Maze Island Berbasis Android. *Penerapan Algoritma*, (x), 1–11. Retrieved from <http://journals.ums.ac.id/index.php/khif/article/view/5221>
- Pramono, A. (2015). Algoritma Pathfinding A* Pada Game RPG Tanaman Higienis. *Jurnal Edukasi Dan Penelitian Informatika (JEPIN)*. <https://doi.org/10.26418/jp.v1i2.12517>
- Pugas, D. O., Somantri, M., & Satoto, K. I. (2011). Pencarian Rute Terpendek Menggunakan Algoritma Dijkstra dan Astar (A*) pada SIG Berbasis Web untuk Pemetaan Pariwisata Kota Sawahlunto. *Transmisi*, (Vol 13, No 1 (2011): TRANSMISI), 27–32. Retrieved from <http://ejournal.undip.ac.id/index.php/transmisi/article/view/3632/pdf>
- Siregar, A. S., & Nasution, S. D. (2017). Penerapan Algoritma A* untuk Pencarian Solusi Terbaik pada Game Congklak. *Informasi Dan Teknologi Ilmiah (INTI)*, pp. 150–154. Retrieved from <https://ejurnal.stmik-budidarma.ac.id/index.php/inti/article/view/372>
- Sunarto, D., & Krisdiawan, R. A. (2015). Rancang Bangun Game Kumbang Kum-Oid Berbasis Android Menggunakan Algoritma A* (A Star). *Jurnal Ilmiah Teknologi Informasi Asia*, pp. 40–48. Retrieved from <https://jurnal.stmikasia.ac.id/index.php/jitika/article/view/124>

- Susilawati. (2014). Perancangan Game Space Ship Dengan Metode Quad Tree. *Pelita Informatika Budi Dharma*, pp. 80–84. Retrieved from <https://www.ilmuskripsi.com/2016/06/jurnal-perancangan-game-space-ship.html>
- Syuhada, A. (2014). *Perancangan Aplikasi Game Rancang Bangun Angka*. pp. 1–6. Retrieved from <https://www.ilmuskripsi.com/2016/06/jurnal-perancangan-aplikasi-game.html>
- Syukriah, Y., Falahah, F., & Solihin, H. (2016). Penerapan Algoritma a* (star) untuk Mencari Rute Tercepat dengan Hambatan. *Seminar Nasional Telekomunikasi Dan Informatika (SELISIK)*, (1), 219–224. <https://doi.org/ISSN : 2503-2844>