TUGAS AKHIR – KI1502

PERANCANGAN DAN IMPLEMENTASI *ARTIFICIAL INTELLIGENCE* PADA *GAME STRATEGY TURN-BASED ROLE PLAYING GAME* MENGGUNAKAN ALGORITMA A*

GIDEON ADIPRANA TIGOR SIBURIAN
NRP 5113100052

Dosen Pembimbing
Imam Kuswardayan, S.Kom., MT.
Dr. Eng. Nanik Suciati, S.Kom, M.Kom.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

TUGAS AKHIR – KI1502

PERANCANGAN DAN IMPLEMENTASI *ARTIFICIAL INTELLIGENCE* PADA *GAME STRATEGY TURN-BASED ROLE PLAYING GAME* MENGGUNAKAN ALGORITMA A*

GIDEON ADIPRANA TIGOR SIBURIAN
NRP 5113100052

Dosen Pembimbing
Imam Kuswardayan, S.Kom., MT.
Dr. Eng. Nanik Suciati, S.Kom, M.Kom.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

[Halaman ini sengaja dikosongkan]

UNDERGRADUATE THESES –KI1502

DESIGN AND IMPLEMENTATION *ARTIFICIAL INTELLIGENCE* IN STRATEGY TURN-BASED ROLE PLAYING GAME USING A* ALGORITHM

GIDEON ADIPRANA TIGOR SIBURIAN
NRP 5113100052

Supervisors
Imam Kuswardayan, S.Kom., MT.
Dr. Eng. Nanik Suciati, S.Kom, M.Kom.

DEPARTMENT OF INFORMATICS
Faculty of Information Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2017

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

PERANCANGAN DAN IMPLEMENTASI *ARTIFICIAL INTELLIGENCE* PADA *GAME STRATEGY TURN-BASED ROLE PLAYING GAME* MENGGUNAKAN ALGORITMA A*

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Interaksi Grafika dan Seni
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh
GIDEON ADIPRANA TIGOR SIBURIAN
NRP: 5113 100 052

Disetujui oleh Dosen Pembimbing Tugas Akhir:

Imam Kuswardayan, S.Kom., MT. .....
(NIP 197612152003121001) (Pembimbing 1)

Dr. Eng. Nanik Suciati S.Kom., M.Kom. .....
(NIP 197104281994122001) (Pembimbing 2)

SURABAYA
JULI, 2017

v

[Halaman ini sengaja dikosongkan]

PERANCAGAN DAN IMPLEMENTASI *ARTIFICIAL INTELLIGENCE* PADA *GAME STRATEGY TURN-BASED ROLE PLAYING GAME* MENGGUNAKAN ALGORITMA A*

Nama Mahasiswa	**:**	**Gideon Adiprana Tigor Siburian**
NRP	**:**	**5113 100 052**
Jurusan	**:**	**Teknik Informatika FTIf-ITS**
Dosen Pembimbing 1	**:**	**Imam Kuswardayan, S.Kom., MT.**
Dosen Pembimbing 2	**:**	**Dr. Eng. Nanik Suciati, S.Kom, M.Kom.**

Abstrak

Game yang menarik adalah game yang dapat memberikan tantangan bagi pemainnya. Pada game yang berjenis Role Playing Game (RPG) terdapat Artificial Inteligence (AI) yang mengendalikan skenario pertarungan. Salah satu jenis pertarungan yang biasa digunakan adalah dengan cara strategy turn-based. Pertarungan dilakukan secara bergantian dimana pemain akan dihadapkan oleh karakter yang dikendalikan oleh AI. Untuk menghasilkan AI yang baik, diperlukan algoritma yang baik juga dalam membantu pergerakan karakter.

Penelitian ini mengusulkan penerapan AI pada game berjenis strategy turn-based role playing game menggunakan algoritma A. Algoritma A* adalah algoritma pencarian graph yang mencari jalan dari satu titik awal ke titik akhir yang telah ditentukan. Algoritma ini menggunakan pendekatan heuristik yang memberikan peringkat ke tiap – tiap neighbor dengan cara memperkirakan rute terbaik yang dapat dilalui dari titik tersebut. Setelah itu tiap-tiap titik tersebut akan dicek satu-persatu berdasarkan urutan yang dibuat dengan pendekatan heuristik tersebut. Pada penelitian ini AI akan dibagi menjadi*

4 level yaitu noob, easy, medium dan hard. Pengujian dilakukan dengan mempertandingkan antar level AI dan AI melawan pemain. Pada pengujian AI melawan pemain, diambil lima responden dan diberikan kuisioner terkait pertarungan dalam game. Berdasarkan hasil pengujian algoritma A dapat diterapkan pada game strategy RPG dengan baik dan dapat membentuk AI yang kuat.*

Kata Kunci* : *algoritma A*, graph, heuristik, strategy turn-based role playing game.

DESIGN AND IMPLEMENTATION *ARTIFICIAL INTELLIGENCE* IN STRATEGY TURN-BASED ROLE PLAYING GAME USING A* ALGORITHM

Student Name	**:**	**Gideon Adiprana Tigor Siburian**
NRP	**:**	**5113 100 052**
Major	**:**	**Informatics Department FTIf – ITS**
Advisor I	**:**	**Imam Kuswardayan, S.Kom., MT.**
Advisor II	**:**	**Dr. Eng. Nanik Suciati, S.Kom, M.Kom.**

Abstract

An exciting game is a game that can challenge players. In the Role Playing Game (RPG) game there is Artificial Intelligence (AI) that controls battle scenarios. One type of battle that is commonly used is by way of turn-based strategy. The battle is done alternately where players will be confronted by characters controlled by AI. To produce a good AI, good algorithms are needed also in assisting the movement of characters.

This research proposes the application of AI in game type turn-based role playing game strategy using A * algorithm. The A * algorithm is a graph search algorithm that searches the path from one starting point to the specified end point. The algorithm uses a heuristic approach that assigns rankings to each neighbor by estimating the best route that can pass from that point. After that each point will be checked one by one based on the order made with the heuristic approach. In this study AI will be divided into 4 levels of noob, easy, medium and hard. Testing is done by matching between AI and AI levels against players. In AI testing against players, five respondents were taken and given questionnaires related to in-game battles. Based on the results of A * algorithm testing can be applied to the game strategy RPG well and can form a strong AI.

Keywords: ***A* algorithm, graph, heuristic, strategy turn-based role playing game.***

KATA PENGANTAR

Puji dan syukur bagi Tuhan yang Maha Kuasa sehingga penulis dapat menyelesaikan tugas akhir yang berjudul **"PERANCANGAN DAN IMPLEMENTASI *ARTIFICIAL INTELLIGENCE* PADA *GAME STRATEGY TURN-BASED ROLE PLAYING GAME* MENGGUNAKAN ALGORITMA A*".**

Pengerjaan tugas akhir ini merupakan suatu kesempatan yang sangat baik bagi penulis. Dengan pengerjaan tugas akhir ini, penulis bisa belajar lebih banyak untuk memperdalam dan meningkatkan apa yang telah didapatkan penulis selama menempuh perkuliahan di Teknik Informatika ITS.

Selesainya tugas akhir ini tidak lepas dari bantuan dan dukungan beberapa pihak, sehingga pada kesempatan ini penulis mengucapkan syukur dan terima kasih kepada:

1. Allah yang Maha Kuasa
2. Ayah, yang selalu mendoakan penulis dan mendukung setiap pilihan yang penulis ambil.
3. Bapak Imam Kuswardayan, S.Kom., MT. selaku pembimbing I yang selalu memberikan motivasi dan membimbing penulis selama pengerjaan tugas akhir.
4. Ibu Dr. Eng. Nanik Suciati , S.Kom., M.Kom. selaku pembimbing II yang selama ini telah membantu dan membimbing penulis selama pengerjaan tugas akhir.
5. Bapak Dr.Eng Darlis Herumurti, S.Kom.,M.Kom. selaku Kepala Jurusan Teknik Informatika ITS, Bapak Dr. Radityo Anggoro, S.Kom.,M.Sc. selaku koordinator TA, dan segenap dosen Teknik Informatika yang telah banyak memberikan ilmu kepada penulis.

6. Teman-teman ABISS (Alumni Budi Mulia Siantar-Surabaya) angkatan 2013 yang telah membantu, berbagi ilmu, menjaga kebersamaan, dan memberi motivasi kepada penulis
7. Serta semua pihak yang yang telah turut membantu penulis dalam menyelesaikan tugas akhir ini.

Penulis menyadari bahwa tugas akhir ini masih memiliki banyak kekurangan, sehingga dengan kerendahan hati, penulis mengharapkan kritik dan saran dari pembaca untuk perbaikan ke depannya.

Surabaya, Mei 2017

Gideon Adiprana Tigor Siburian

DAFTAR ISI

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

DAFTAR KODE SUMBER

BAB I
PENDAHULUAN

Bab ini menjelaskan garis besar tugas akhir yang meliputi latar belakang, tujuan, rumusan, batasan permasalahan, metodologi pembuatan tugas akhir, dan sistematika penulisan.

1.1. Latar Belakang

Perkembangan teknologi yang pesat telah membawa banyak perubahan pada pengembangan *game*. Salah satu jenis *game* yang paling banyak digemari adalah yang berjenis RPG (*Role Playing Game*). Dalam *game* tersebut pasti ada AI (*Artificial Intelligence*) yang dibuat untuk mengendalikan skenario pertarungan [1]. Salah satu jenis pertarungan yang biasa digunakan adalah dengan cara *Strategy Turn-Based*. Pertarungan dilakukan secara bergantian dimana pemain akan dihadapkan oleh karakter yang dikendalikan oleh AI. Hal ini menjadi pertimbangan bagi para developer *game* untuk menghasilkan *game* yang dapat memenuhi kemampuan pemain dari berbagai kalangan. Pembuatan AI berdasarkan kemampuan pemain merupakan solusi dari permasalahan tersebut.

Tujuan dari pengerjaan Tugas Akhir ini adalah merancang dan mengimplementasikan aplikasi *game strategy turn-based* RPG menggunakan algoritma A*.

.

1.2. Rumusan Masalah

Rumusan masalah yang diangkat dalam tugas akhir ini dapat dipaparkan sebagai berikut:

1. Bagaimana merancang aturan *game* dan perancangan level pada *game Strategy Turn Based* RPG ?
2. Bagaimana strategi untuk menang dan menerapkan AI dalam *game Strategy Turn Based* RPG ?

1.3. Batasan Masalah

Permasalahan yang dibahas dalam tugas akhir ini memiliki beberapa batasan, yaitu sebagai berikut:

1. Perangkat lunak dibangun dengan menggunakan bahasa pemograman C# dan aplikasi Unity Free License.
2. Aplikasi yang dibuat merupakan aplikasi *game* yang berjalan di PC.

1.4. Tujuan

Tugas akhir ini bertujuan untuk membuat aplikasi *game* bergenre *strategy turn-based* RPG berbasis PC dan pemanfaatan algoritma A* dalam mengimplementasikan AI pada game tersebut.

1.5. Manfaat

Manfaat penelitian dari tugas akhir ini adalah terciptanya aplikasi *game* yang dapat dimainkan dengan nyaman oleh pemain dari berbagai kalangan karena diterapkannya tingkat kesulitan menggunakan AI sesuai dengan kemampuan pemain. Penelitian AI pada *game* strategy turn-based RPG ini diharapkan dapat memberikan solusi untuk penerapan AI pada *game* genre lain.

1.6. Metodologi Pembuatan Tugas Akhir

Tahapan-tahapan yang dilakukan dalam pengerjaan tugas akhir ini adalah sebagai berikut:

1. Penyusunan proposal tugas akhir.

   Proposal tugas akhir ini terdiri dari deskripsi pendahuluan yang menjabarkan latar belakang dan rumusan masalah yang mendasari dibangunnya aplikasi ini, batasan masalah dalam pembangunan aplikasi ini, serta tujuan dan manfaat yang diharapkan dapat dicapai dengan dibangunnya aplikasi ini. Selain itu, pada proposal tugas akhir ini juga terdapat tinjauan pustaka yang menjelaskan teori-teori yang menjadi dasar pembuatan tugas akhir ini, yaitu Pembuatan *Game*,konsep

Strategy Turn-Based RPG, Penggunaan Algoritma A* dalam pencarian jarak terdekat.

2. Studi literatur

   Studi literatur yang dilakukan pada perancangan aplikasi *game* sebagai pengerjaan tugas akhir ini adalah mengenai implementasi AI *(Artificial Inteligcence)* pada genre *game Strategy Turn-Based RPG* dengan menggunakan algoritma A * . Selain itu, dilakukan juga studi literatur mengenai strategi untuk menang pada *game* lain dengan genre yang sama.

3. Analisis dan desain perangkat lunak

   Tahap ini meliputi perumusan aturan *game*, strategi untuk menang, dan penerapan AI pada *game* 'Zero *State*'

4. Implementasi perangkat lunak

   Aplikasi ini diimplementasikan dengan menggunakan kakas bantu :

   1. Sistem operasi Windows dengan spesifikasi minimal Windows 7.
   2. Bahasa pemrograman C#.
   3. Unity Free License.

5. Pengujian dan evaluasi

   Pengujian dan evaluasi aplikasi perangkat lunak hasil dari tugas akhir ini diujicobakan pada mahasiswa Institut Teknologi Sepuluh Nopember (ITS)

6. Penyusunan buku tugas akhir

   Pada tahap ini dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam tugas akhir ini serta hasil dari implementasi aplikasi perangkat lunak yang telah dibuat. Sistematika penulisan buku tugas akhir secara garis besar antara lain:

   1. Pendahuluan

a. Latar Belakang
b. Rumusan Masalah
c. Batasan Masalah
d. Tujuan
e. Manfaat
f. Metodologi Pembuatan Tugas Akhir
g. Sistematika Penulisan Laporan Tugas Akhir

2. Tinjauan Pustaka
3. Analisis dan Perancangan Sistem
4. Pengujian dan Evaluasi
5. Kesimpulan dan Saran
6. Daftar Pustaka

1.7. Sistematika Penulisan Laporan Tugas Akhir

Buku tugas akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan tugas akhir ini. Selain itu, diharapkan dapat berguna untuk pembaca yang tertarik untuk melakukan pengembangan lebih lanjut. Secara garis besar, buku tugas akhir terdiri atas beberapa bagian seperti berikut ini.

Bab I Pendahuluan

Bab yang berisi mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, dan manfaat dari pembuatan tugas akhir. Selain itu metodologi yang digunakan dan sistematika penulisan laporan akhir juga merupakan bagian dari bab ini.

Bab II Tinjauan Pustaka

Bab ini berisi penjelasan secara detail mengenai dasar-dasar penunjang dan teori-teori yang digunakan untuk mendukung pembuatan tugas akhir ini.

Bab III Analisis dan Perancangan Sistem

Bab ini berisi tentang analisis permasalahan, deskripsi umum sistem, aturan *game*, strategi untuk menang, dan penerapan AI.

Bab IV Implementasi

Bab ini membahas implementasi dari perancangan yang telah dibuat pada bab sebelumnya. Penjelasan berupa kode sumber yang digunakan untuk proses implementasi.

Bab V Pengujian dan Evaluasi

Bab ini menjelaskan kemampuan perangkat lunak dengan melakukan pengujian kebenaran dan pengujian kinerja dari sistem yang telah dibuat.

Bab VI Kesimpulan dan Saran

Bab ini berisi kesimpulan dari hasil pengujian dan evaluasi yang dibuat pada bab sebelumnya serta saran untuk pengembangan berikutnya

[Halaman ini sengaja dikosongkan]

BAB II
TINJAUAN PUSTAKA

Bab ini menjelaskan teori-teori yang berkaitan dengan pembangunan AI pada *game strategy turn-based RPG* yang diajukan untuk tugas akhir ini. Penjelasan ini bertujuan untuk memberikan gambaran secara umum terhadap perangkat lunak yang dibuat dan berguna sebagai penunjang dalam pengembangan perangkat lunak.

2.1 Game Engine (Unity)

Unity merupakan suatu aplikasi yang digunakan untuk mengembangkan *game* multi platform yang didesain untuk mudah digunakan. Unity itu bagus dan penuh perpaduan dengan aplikasi yang profesional. Editor pada Unity dibuat dengan *user interface* yang sederhana. Unity cocok dengan versi 64-bit dan dapat beroperasi pada Mac OS x dan windows dan dapat menghasilkan *game* untuk Mac, Windows, Wii, iPhone, iPad dan Android.

2.2 Bahasa Pemograman C#

C# (dibaca: C sharp) merupakan sebuah bahasa pemrograman yang berorientasi objek yang dikembangkan oleh Microsoft sebagai bagian dari inisiatif kerangka. .NET Framework. Bahasa pemrograman ini dibuat berbasiskan bahasa C++ yang telah dipengaruhi oleh aspek-aspek ataupun fitur bahasa yang terdapat pada bahasa-bahasa pemrograman lainnya seperti Java, Delphi, Visual Basic, dan lain-lain) dengan beberapa penyederhanaan [2].

2.3 Fungsi Heuristik

Heuristik adalah sebuah teknik yang mengembangkan efisiensi dalam proses pencarian (pencarian yang lebih sederhana), namun kemungkinan juga dapat mengorbankan kelengkapan (*completeness*). Salah satu fungsi heuristik yang dipakai adalah Best First Search. Fungsi heuristik yang digunakan merupakan

prakiraan(estimasi) *cost* dari *initial state* ke *goal state* yang dinyatakan dengan

$$f(n) = g(n) + h(n) \quad [3]$$

Keterangan :
f(n) = fungsi evaluasi
g(n) = *cost* dari *initial state* ke *current state*
h(n) = prakiraan *cost* dari *current state* ke *goal state*.

2.4 Algoritma A*

Algoritma A* merupakan format pencarian heuristik untuk menghitung efisiensi solusi optimal. Algoritma A* adalah algoritma best-first search dimana *cost* yang terkait dengan *node* dapat dijelaskan sesuai dengan persamaan (3), dimana *G* adalah *cost of the path* dari keadaan awal ke *node* n dan *H* adalah perkiraan heuristik atau *cost* atau *path* dari *node* n ke tujuan. Jadi *F* adalah perkiraan total *cost* terendah dari setiap *path* yang akan dilalui *node* n ke *node* tujuan [4].

Algortima A* secara ringkas langkah demi langkahnya adalah sebagai berikut.

1) Tambahkan *starting point* ke dalam *open set*.
2) Ulangi langkah berikut:
    a) Carilah biaya *F* terendah pada setiap simpul dalam *open set*. *Node* dengan biaya *F* terendah kemudian disebut *current node*.
    b) Masukkan ke dalam *closed set*.
    c) Untuk setiap 8 simpul (*neighbor node*) yang berdekatan dengan *current node* :
        i) Jika tidak walkable atau jika termasuk *closed set*, maka abaikan.
        ii) Jika tidak ada pada *open set*, tambahkan ke *open set*.

iii) Jika sudah ada pada *open set*, periksa apakah ini jalan dari simpul ini ke *current node* yang lebih baik dengan menggunakan biaya *G* sebagai ukurannya. Simpul dengan biaya *G* yang lebih rendah berarti bahwa ini adalah jalan yang lebih baik. Jika demikian, buatlah simpul ini (*neighbor node*) sebagai came from dari *current node*, dan menghitung ulang nilai *G* dan *F* dari simpul ini.

d) Stop ketika:

i) Menambahkan *target point* ke dalam *closed set*, dalam hal ini jalan telah ditemukan, atau

ii) Gagal untuk menemukan *target point*, dan *open set* kosong. Dalam kasus ini, tidak ada jalan.

3) Simpan jalan. Bekerja mundur dari *target point*, pergi dari masing-masing simpul ke simpul *came from* sampai mencapai *starting point*.

2.5 Role-Playing *Game*

Role-Playing Game disingkat RPG adalah sebuah *game* yang para pemainnya memainkan peran tokoh-tokoh khayalan dan berkolaborasi untuk merajut sebuah cerita bersama [5]. Para pemain memilih aksi tokok-tokoh mereka berdasarkan karakteristik tokoh tersebut, dan keberhasilan aksi mereka tergantung dari sistem peraturan *game* yang telah ditentukan. Asal tetap mengikuti peraturan yang ditetapkan, para pemain bisa berimprovisasi membentuk arah dan hasil akhir *game* ini.

2.6 Tactical Role-Playing *Game*

Tactical role-playing game merupakan subgenre dari *game role-playing game* yang mengacu pada *game* yang menggabungkan unsur-unsur strategi alternatif [6]. Perbedaan yang jelas antara RPG taktis dan RPG tradisional adalah kurangya eksplorasi. Misalnya pada *game* Final Fantasy Tactics tidak jauh

dengan eksplorasi orang ketiga khas kota dan ruang bawah tanah yang khas dalam *game*. Pertempuran memiliki kondisi kemenangan yang spesifik, seperti mengalahkan musuh atau bertahan dari sejumlah putaran tertentu, yang harus dicapai pemain sebelum peta berikutnya akan tersedia. Di sela-sela pertarungan pemain bisa mengakses karakter mereka untuk dibekali, mengubah kelas , maupun melatih unit karakternya.

BAB III
ANALISIS DAN PERANCANGAN SISTEM

Bab ini membahas mengenai perancangan dan pembuatan perangkat lunak. Perangkat lunak yang dibuat pada tugas akhir ini adalah *game* dengan AI menggunakan algoritma A*.

3.1 Spesifikasi Game

Pada *game* ini terdapat dua pihak yang saling menyerang satu sama lain, masing-masing pihak harus menghabisi seluruh pasukan lawan untuk mendapatkan kemenangan.

Pada *game* ini terdapat empat jenis pasukan yang komposisinya sama pada tiap pihak, yaitu *Spearman*, *Archer* dan *Paladin* serta satu *Hero* yang kekuatannya lebih besar dibanding pasukan lainnya.

Adapun aturan pada *game* ini adalah sebagai berikut :

1. Terdapat 2 belah pihak yang saling menyerang satu sama lain.
2. Terdapat 4 jenis pasukan yaitu *Spearman*, *Archer* dan *Paladin* yang memiliki kemampuan masing-masing serta 1 *Hero* yang kemampuannya lebih kuat dari pasukan lainnya.
3. Tiap pihak menyerang berdasarkan giliran yang ditentukan secara bergantian. Setiap giliran masing-masing pihak dapat menggerakan seluruh unitnya.
4. Masing – masing pasukan memiliki kelemahan yang apabila menyerang tipe pasukan tesebut menghasilkan serangan dua kali lipat kecuali *Hero*.
5. Terdapat 4 tingkat kesulitan pada *game* yaitu *Noob*, *Easy*, *Medium*, dan *Hard*. Pemain dapat memilih untuk melawan AI dengan tingkat kesulitan yang sesuai kemampuannya.

Terdapat 4 jenis pasukan pada *game* ini yaitu :

1. *Spearman*
   Tipe pasukan jarak dekat yang memiliki *damage* dan *defense* yang stabil, memiliki *movements points* yang kecil dan *health point* yang cukup besar. Kelamahannya adalah *archer.*

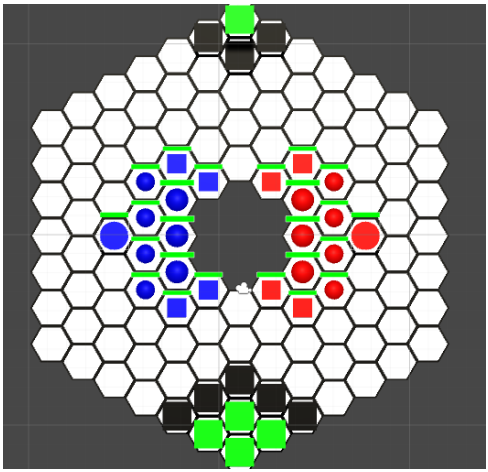
2. *Archer*
   Tipe pasukan jarak jauh dengan attack range dua kotak, memiliki *damage* yang besar namun *defense* yang kecil. *movements* yang kecil dan *health point* yang kecil. kelemahannya adalah *paladin.*

3. *Paladin*
   Tipe pasukan jarak dekat berkuda yang memiliki *damage* dan *defense* yang stabil, memiliki *movements points* yang besar dan *health point* yang besar. Kelemahannya adalah *spearman.*

4. *Hero*
   Tipe pasukan jarak dekat yang memiliki *damage* dan *defense* yang seimbang, *movements points* yang kecil dan *health point* yang besar. Tipe pasukan ini tidak memliki kelemahan dan memiliki kemampuan khusus.

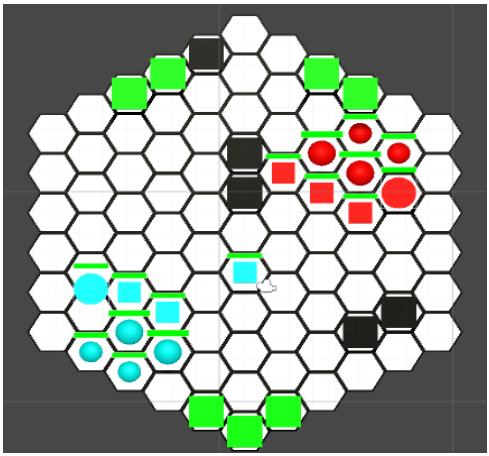
Pada *game* ini terdapat 3 *map* yang berbeda yang dapat dipilih saat bermain. Masing – masing *map* memiliki spesifikasi tersendiri.

1. *Field* 1
   Map ini dirancang untuk pertarungan 12 vs 12 unit , di *map* ini tidak ada *random obstacles.* Gambar *map field* 1 seperti gambar dibawah ini



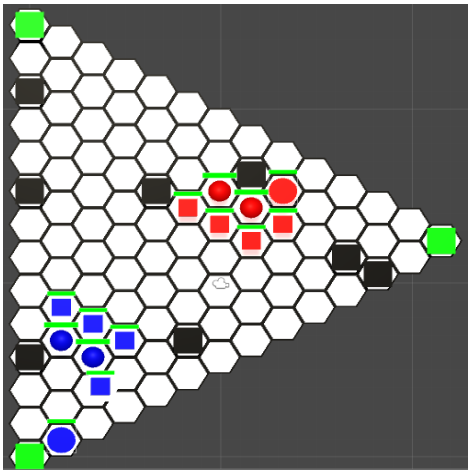
Gambar 3. 1 Gambar *Map Field* 1

Map ini dirancang untuk pertarungan 8 VS 8 unit, memiliki 5 *random obstacles.* Gambar dari *map field* 2 seperti gambar dibawah ini



Gambar 3. 2 Gambar *Map Field* 2

2. *Field* 3
   Map ini dirancang untuk pertarungan 7 VS 7 unit, memiliki 8 *random obstacles*. Gambar dari *map field* 3 seperti gambar dibawah ini



Gambar 3. 3 Gambar *Map Field* 3

Kondisi kemenangan *game* adalah mengalahkan seluruh pasukan lawan. Kondisi kekalahan *game* adalah seluruh pasukan *player* mati atau tidak ada yang tersisa.

3.2. Strategi Memenangkan *Game*

Strategi untuk memenangkan *game* adalah mengatur agar pasukan dapat seoptimal mungkin untuk membunuh sebanyak mungkin musuh pada tiap giliran strategi ini disebut juga sebagai *naive strategy*.

Setiap pasukan memiliki kemampuan yang berbeda-beda maka harus dilakukan pengaturan agar dapat sebanyak mungkin mengalahkan lawan dan sedikit kehilangan pasukan.

1. Unit tipe jarak dekat yaitu *spearman* dan *paladin* memiliki *damage* dan *defense* yang stabil, serta *health point* yang cukup besar, sehingga pasukan ini harus dijalankan terlebih dahulu sebagai pertahanan untuk unit tipe jarak jauh menyerang. Jika ada celah untuk masuk menyerang musuh tipe jarak jauh maka ini didahulukan.
2. Unit tipe jarak jauh yaitu *archer* memiliki *damage* yang besar di posisikan dibelakang unit tipe jarak dekat sebagai barisan backup untuk menyerang garis depan.
3. Unit tipe *hero* digunakan sebagai kontrol, untuk membantu penyerangan sekaligus pertahanan. Unit tipe *hero* bisa digunakan untuk membantu penyerangan di baris depan karena memilki *defense* serta *health point* yang besar juga.

Untuk strategi mengatur unit akan jalan ke posisi yang mana dilihat dari masing-masing kelemahan unit, unit yang memiliki *health point* yang paling sedikit, serta kemungkinan jalan agar unit tersebut aman untuk tidak diserang pada saat giliran musuh. Utamakan untuk bergerak dan menyerang ke posisi musuh yang *health point* sedikit sampai memusnahkan unitnya, jika tidak ada maka cari pasukan musuh yang dapat diserang dan menghasilkan serangan dua kali lipat. Posisikan unit jarak dekat seperti *spearman* dan *paladin* untuk menghalangi jalan pasukan mush menyerang pasukan jarak jauh sehingga musuh harus menghabisi pasukan jarak dekat terlebih dahulu untuk dapat masuk ke barisan unit jarak jauh.

3.3. Penerapan *Artificial Intelligence*

Penerapan *Artificial Intelligence* pada *game* ini dibuat dengan bantuan Algoritma A* sebagai algoritma *path*finding. Algoritma tersebut dikombinasikan dengan aturan *game* pada *game* ini. Kondisi – kondisi yang dibuat berdasarkan pada strategi untuk memenangkan *game*. Pada bagian ini akan dibagi menjadi 5 sub-bagian, yaitu penerapan *Artificial Intelligence* menggunakan

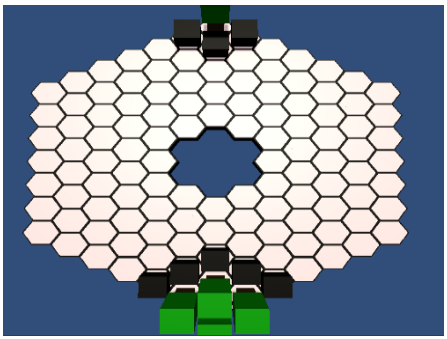
A* *pathfinding*, penerapan *Artificial Intelligence* pada level *noob*, penerapan *Artificial Intelligence* pada level *easy*, penerapan *Artificial Intelligence* pada level *medium*, dan penerapan *Artificial Intelligence* pada level *hard*.

3.3.1. Penerapan *Artificial Intelligence* menggunakan Algoritma A* *Pathfinding*

Penerapan AI dibuat dengan menggabungkan algoritma A* *pathfinding* dengan aturan *game*.

1. *Cell Grid*

   Grid map merupakan representasi *map* yang paling umum digunakan dalam *game Turn-Based Strategy* . *Grid map* ini merupakan kumpulan kotak yang saling berhubungan satu sama lain. Setiap kota dapat diberikan informasi *cost* yang diperlukan dan juga dapat diterapkan aturan tambahan mengenai pergerakan dari unit yang ada di *map*.



Gambar 3. 4 *Cell Grid* pada *Game*

2. Penerapan Algoritma A* *pathfinding*

Penerapan Algoritma A* digunakan dalam menemukan jalan yang terpendek menuju tujuan. Setiap unit akan mencari jalan tersingkat untuk menghasilkan kerugian pasukan lawan terbanyak dengan cara menyerang.

- *Initial state*
  Posisi awal dari setiap unit pemain.

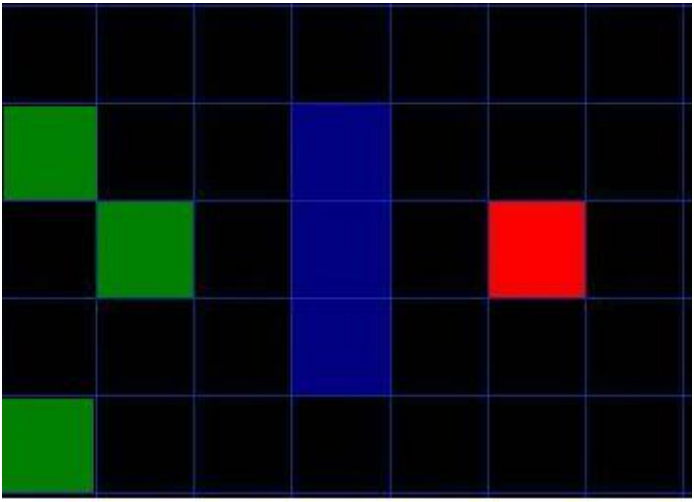
- *Cost*
  Movement *cost* yang dibutuhkan untuk bergerak ke cell yang dituju. Masing – masing cell memiliki *movement cost*. Pada setiap *map* ketika unit bergerak secara horizontal atau vertikal dikenakan *cost* sebesar 10, ketika bergerak secara diagonal dikenakan cost sebesar 14.

- Fungsi Evaluasi
  Fungsi evaluasi dilakukan sesuai dengan persamaan (3).

- Goal State
  Posisi dimana unit dapat membuat kerugian pasukan lawan terbesar dengan cara menyerang.

Penerapan algoritma A* pada AI dapat ditunjukkan melalui penjelasan berikut

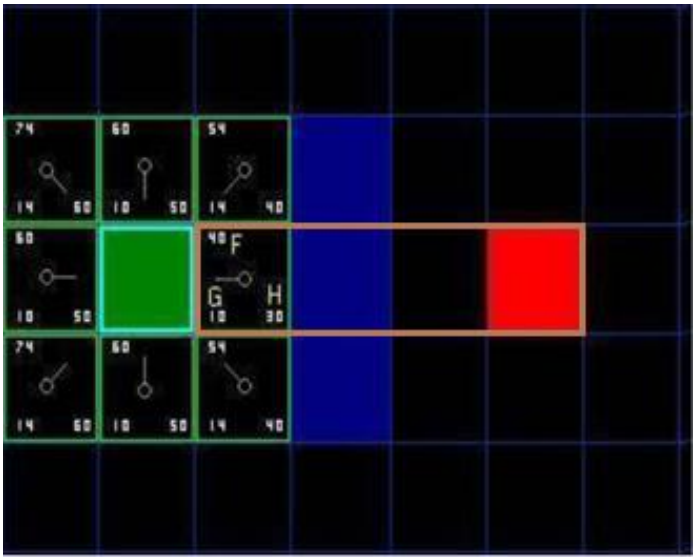
a. Kondisi awal



Gambar 3. 5 Kondisi Awal *Game*

Pada Gambar 3. 5 terdapat kotak hitam , kotak biru, kotak merah dan kotak hijau. Kotak merah merupakan unit pasukan musuh, kotak hijau merupakan unit pasukan *player*, kotak biru merupakan *obstacles* atau penghalang dan kotak hitam merupakan *cell*. Gambar diatas merupakan kondisi awal dari *game*.

b. Langkah awal



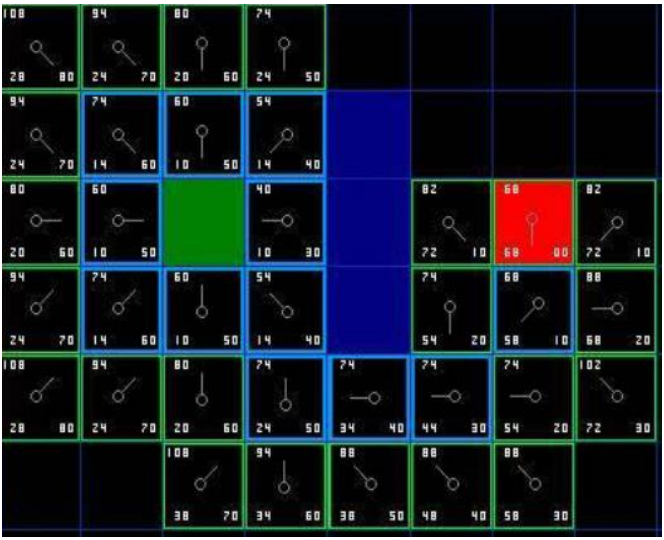
Gambar 3.6 Langkah Awal

Pada Gambar 3.6 AI akan mencoba seluruh simpul yang berdekatan dengan posisi awal. Untuk setiap pergerakan horizontal dan vertikal diberi *cost* 10, dan diagonal diberi *cost* 14. Pada gambar diatas untuk setiap kotak yang berdekatan dengan unit terdapat tiga buah nilai yaitu nilai F di pojok kiri atas, nilai G di pojok kiri bawah dan nilai H di pojok kanan bawah. G merupakan *cost* yang dibutuhkan dari *initial state* ke *next state*, H merupakan *cost* dari *goal state* ke *next state* dengan hanya bergerak secara horizontal dan vertikal dengan mengabaikan *obstacles*, sedangkan F merupakan skor untuk masing – masing kotak dengan cara menjumlah G dan H.

Pada gambar tersebut dapat dilihat contoh perhitungan nilai F, G dan H pada kotak disamping kanan kotak hijau atau unit pasukan sendiri. Nilai G didapat dari unit bergerak secara

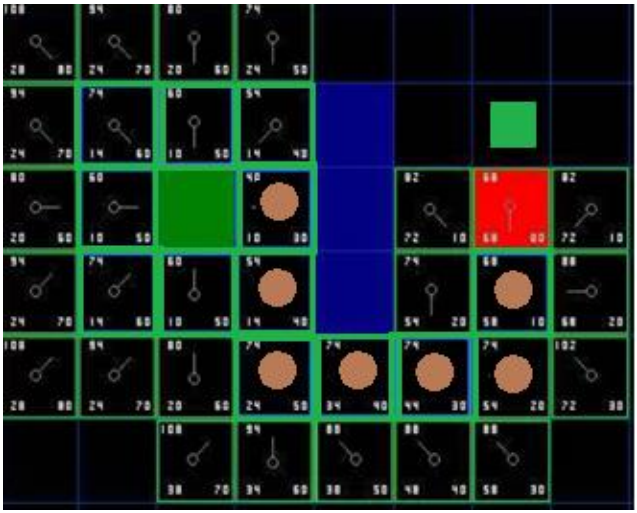
horizontal ke kanan dari posisi awal sehingga diberikan *cost* 10, sedangkan nilai H didapat dari estimasi *cost* terkecil yang ditandai dengan kotak berwarna coklat yang menghubungkan antara kotak merah dengan kemungkinan *cell* yang dituju. H diberikan *cost* 30 dihitung dari kotak yang sedang dituju, unit bergerak secara horizontal sebanyak tiga langkah ke kanan untuk sampai ke kotak merah. F diberikan *cost* 40 dihitung dengan menjumlahkan nilai dari G dan H.

c. Trace dari initial state sampai goal state

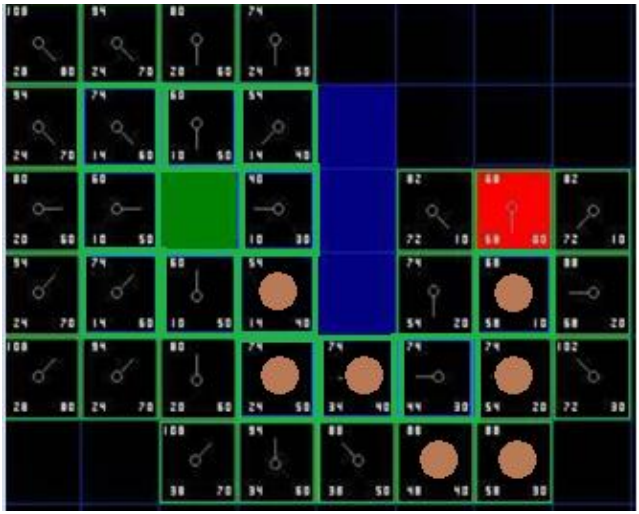


Gambar 3.7 *Trace Initial State* ke *Goal State*

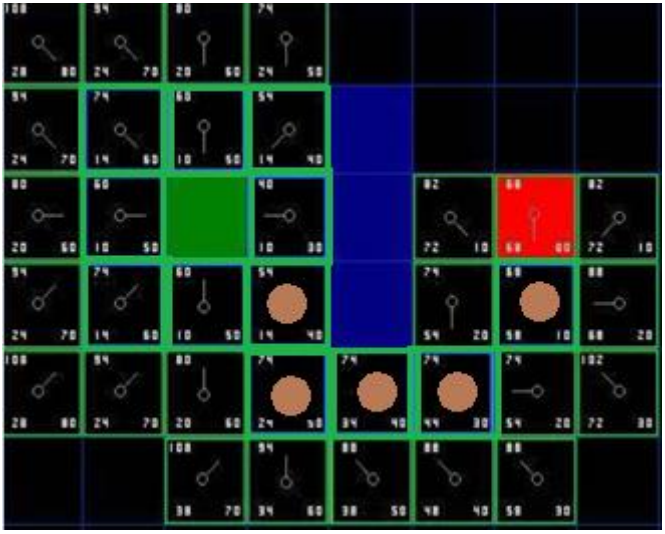
Pada gambar 3. 7 setiap kotak yang dilalui sebagai kemungkinan jalan akan diberikan skor berdasarkan fungsi heuristik dengan cara melakukan *looping* untuk mencoba semua kemungkinan. Dari hasil *looping* maka akan ditemukan beberapa *path* solusi untuk sampai ke tujuan.



Gambar 3. 8 *Path* Solusi 1



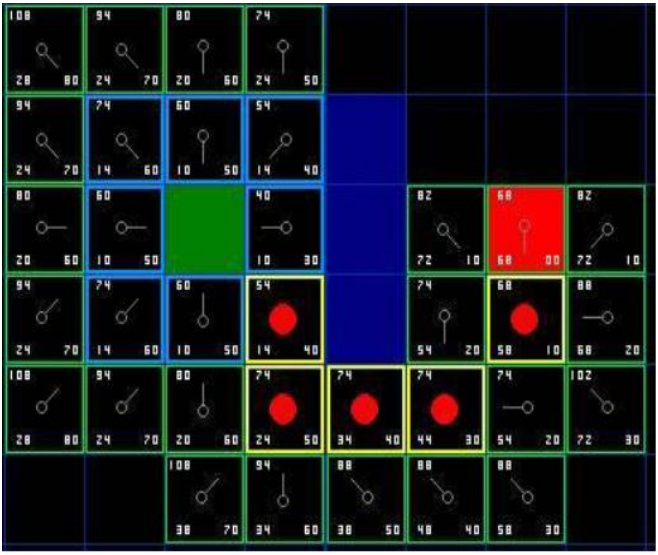
Gambar 3. 9 *Path* Solusi 2



Gambar 3. 10 *Path* Solusi 3

Pada Gambar 3. 8, Gambar 3. 9, dan Gambar 3. 10 dapat dilihat beberapa macam kemungkinan *path* yang dijadikan solusi untuk sampai ke tujuan, *path* solusi ditandai dengan titik-titik berwarna coklat . Dari beberapa *path* solusi tersebut kemudian AI akan memilih solusi yang paling optimal yaitu *path* dengan nilai F paling kecil.

d. Memilih *path* solusi



Gambar 3. 11 *Path* Solusi yang Dipilih

Pada Gambar 3. 11 setelah melakukan seluruh *looping* untuk mencari seluruh kemungkinan untuk mencapai ke *goal state* AI menemukan beberapa *path* solusi setelah itu AI akan mempertimbangkan bergerak kemana dengan cara mencari jarak terpendek untuk dapat sampai ke *goal state* dengan melihat nilai F terkecil dari setiap *cell* yang dituju sampai akhirnya AI sampai ke tujuan, *path* solusi ditandai dengan titik-titik merah.

3.3.2. Penerapan *Artificial Intelligence* pada level *Noob*

AI pada level ini adalah level yang paling rendah dimana pada level ini AI akan bergerak dan menyerang secara *random* sepenuhnya.

3.3.3. Penerapan *Artificial Intelligence* pada level *Easy*

AI pada level ini adalah AI yang bergerak dengan penggabungan strategi untuk menang dan algoritma A*. Pada level ini AI akan menghitung tiap kemungkinan jalan yang menghasilkan kerugian pasukan lawan terbesar. Pada level ini juga AI mempunyai probabilitas untuk melakukan gerakan dan menyerang *random* sebesar 40%. Gerakan *random* yang dimaksud adalah ketika setiap unit melakukan pencarian jalan menggunakan A*, setelah menemukan *path* yang dituju unit tersebut akan memiiki presentase *random* untuk bergerak tidak sesuai dengan solusi yang optimal, menyerang unit yang di dalam *attack range* secara *random* dan keputusan untuk menyerang atau tidak. AI pada level ini dapat dijelaskan dengan tahapan sebagai berikut :

1. Untuk setiap unit, apabila terdapat unit pasukan musuh dalam *attack range* maka lakukan serangan.
2. Apabila tidak ada musuh dalam *attack range,* AI akan meyimpan seluruh lokasi *cell* dimana setiap unit dapat bergerak ke tempat tersebut dengan cara melihat apakah *cell* tersebut sudah berisi penghalang atau unit lain.
3. AI akan membandingkan jarak lokasi *cell* pasukan musuh dengan *movements points* dan *attack range* yang dimiliki setiap unit pasukan sendiri dengan menggunakan algoritma a* *pathfinding*, lalu menyimpannya ke dalam *potential destinations.* Apabila tidak ada unit pasukan musuh dalam *potential destinations* maka AI akan bergerak ke *cell* yang paling mendekati unit pasukan lawan dengan

probabilitas 60% atau melakukan gerakan *random* dengan probabilitas 40%.

4. Jika isi dari *potential destinations* dari suatu unit pasukan sendiri hanya satu, maka AI akan bergerak untuk menyerang unit pasukan lawan tersebut. Jika lebih dari satu maka AI akan melihat unit lawan yang dapat diserang dan mati , jika tidak ada maka akan mencari unit dengan *health point* yang paling sedikit, jika *health point* unit lawan sama maka AI akan menyerang unit pasukan lawan yang paling dekat.

3.3.4. Penerapan *Artificial Intelligence* pada level *Medium*

AI pada level ini adalah AI yang bergerak dengan penggabungan strategi untuk menang dan algoritma A*. Pada level medium, AI akan menghitung kemungkinan jalan yang menghasilkan kerugian pasukan lawan terbesar, memiliki probabilitas gerakan *random* sebesar 10%. Gerakan *random* yang dimaksud adalah ketika setiap unit melakukan pencarian jalan menggunakan A*, setelah menemukan *path* yang dituju unit tersebut akan memiiki presentase *random* untuk bergerak tidak sesuai dengan solusi yang optimal, menyerang unit yang di dalam *attack range* secara *random* dan keputusan untuk menyerang atau tidak. AI pada level ini dapat dijelaskan dengan tahapan sebagai berikut :

1. Untuk setiap unit, apabila terdapat unit pasukan musuh dalam *attack range* maka lakukan serangan.
2. Apabila tidak ada musuh dalam *attack range*, AI akan meyimpan seluruh lokasi *cell* dimana setiap unit dapat bergerak ke tempat tersebut dengan cara melihat apakah *cell* tersebut sudah berisi penghalang atau unit lain.
3. AI akan membandingkan jarak lokasi *cell* pasukan musuh dengan *movements points* dan *attack range*

yang dimiliki setiap unit pasukan sendiri dengan menggunakan algoritma a* *pathfinding*, lalu menyimpannya ke dalam *potential destinations.* Apabila tidak ada unit pasukan musuh dalam *potential destinations* maka AI akan bergerak ke *cell* yang paling mendekati unit pasukan lawan dengan probabilitas 90% atau melakukan gerakan *random* dengan probabilitas 10%.

4. Jika isi dari *potential destinations* dari suatu unit pasukan sendiri hanya satu, maka AI akan bergerak untuk menyerang unit pasukan lawan tersebut. Jika lebih dari satu maka AI akan melihat unit lawan yang dapat diserang dan mati , jika tidak ada maka akan mencari unit dengan *health point* yang paling sedikit, jika *health point* unit lawan sama maka AI akan menyerang unit pasukan lawan yang paling dekat.

3.3.5. Penerapan *Artificial Intelligence* pada level *Hard*

AI pada level ini adalah yang paling tinggi levelnya, yang bergerak dengan penggabungan strategi untuk menang dan algoritma A*. Pada level hard, AI akan menghitung kemungkinan jalan yang menghasilkan kerugian pasukan lawan terbesar, tidak memiliki gerakan *random* dan mencari kemungkinan untuk mencari pasukan yang merupakan unit untuk menghasilkan serangan dua kali lipat. AI pada level ini dapat dijelaskan dengan tahapan sebagai berikut :

1. Untuk setiap unit, apabila terdapat unit pasukan musuh dalam *attack range* maka lakukan serangan.
2. Apabila tidak ada musuh dalam *attack range,* AI akan meyimpan seluruh lokasi *cell* dimana setiap unit dapat bergerak ke tempat tersebut dengan cara melihat apakah *cell* tersebut sudah berisi penghalang atau unit lain.

3. AI akan membandingkan jarak lokasi *cell* pasukan musuh dengan *movements points* dan *attack range* yang dimiliki setiap unit pasukan sendiri dengan menggunakan algoritma a* *pathfinding*, lalu menyimpannya ke dalam *potential destinations*. Apabila tidak ada unit pasukan musuh dalam *potential destinations* maka AI akan bergerak ke *cell* yang paling mendekati unit pasukan lawan.
4. Jika isi dari *potential destinations* dari suatu unit pasukan sendiri hanya satu, maka AI akan bergerak untuk menyerang unit pasukan lawan tersebut. Jika lebih dari satu maka AI akan melihat unit lawan yang dapat diserang dan mati , jika tidak ada maka akan mencari unit lawan yang dapat menghasilkan serangan dua kali lipat, jika tidak ada maka akan mencari unit dengan *health point* yang paling sedikit, jika *health point* unit lawan sama maka AI akan menyerang unit pasukan lawan yang paling dekat.

[Halaman ini sengaja dikosongkan]

BAB IV
IMPLEMENTASI

Bab ini membahas implementasi yang dilakukan berdasarkan rancangan yang telah dijabarkan pada bab sebelumnya. Sebelum penjelasan implementasi akan ditunjukkan terlebih dahulu lingkungan untuk melakukan implementasi.

Pada bagian implementasi ini juga akan dijelaskan mengenai fungsi-fungsi yang digunakan dalam program tugas akhir ini dan disertai dengan kode sumber masing-masing fungsi utama.

4.1 Lingkungan Implementasi Perangkat Lunak

Spesifikasi perangkat keras serta perangkat lunak yang digunakan dalam tahap implementasi perangkat lunak tugas akhir ini seperti dijelaskan pada Tabel 4.1.

Tabel 4.1 Lingkungan Impementasi Perangkat Lunak

Perangkat Keras	Komputer	ASUS A46CB
	Prosesor	Intel® Core™ i5-33170 CPU @ 1.70GHz
	Memori Primer	4 GB
	Memori Sekunder	500 GB
Perangkat Lunak	Sistem Operasi	*Windows* 10 64-bit
	Perangkat Lunak	Unity Free License, Mo*node*velop-Unity, Microsoft Word 2013

4.2 Implementasi Algoritma A* (Star) *Pathfinding*

Implementasi algoritma A*(Star) sebagai pencarian jalan pada *game* ini dapat dilihat pada kode berikut

```
public override List<T> FindPath<T>(Dictionary<T, Dictionary<T, int>> edges, T originNode, T destinationNode)
{
    IPriorityQueue<T> frontier = new HeapPriorityQueue<T>();
    frontier.Enqueue(originNode, 0);

    Dictionary<T, T> cameFrom = new Dictionary<T, T>();
    cameFrom.Add(originNode, default(T));
    Dictionary<T, int> costSoFar = new Dictionary<T, int>();
    costSoFar.Add(originNode, 0);

    while (frontier.Count != 0)
    {
        var current = frontier.Dequeue();
        if (current.Equals(destinationNode)) break;

        var neighbours = GetNeigbours(edges, current);
        foreach (var neighbour in neighbours)
        {
            var newCost = costSoFar[current] + edges[current][neighbour];
            if (!costSoFar.ContainsKey(neighbour) || newCost < costSoFar[neighbour])
            {
                costSoFar[neighbour] = newCost;
                cameFrom[neighbour] = current;
                var priority = newCost + Heuristic(destinationNode, neighbour);
                frontier.Enqueue(neighbour, priority);
            }
        }
    }
```

Kode Sumber 4. 1 Potonga Kode A* *Pathfinding* I

Pada bagian ini algoritma a*(star) diimplementasikan dengan menggunakan *priority queue* yaitu pasangan dari setiap neighbour dan *cost* yang harus dikeluarkan untuk mencapai kesana. Pada bagian ini akan menghitung seluruh kemungkinan jalan terdekat untuk mencapai tujuan.

```
    List<T> path = new List<T>();
    if (!cameFrom.ContainsKey(destinationNode))
        return path;

    path.Add(destinationNode);
    var temp = destinationNode;

    while (!cameFrom[temp].Equals(originNode))
    {
        var currentPathElement = cameFrom[temp];
        path.Add(currentPathElement);

        temp = currentPathElement;
    }

    return path;
}
private int Heuristic<T>(T a, T b) where T : IGraphNode
{
    return a.GetDistance(b);
}
```

Kode Sumber 4. 2 Potongan Kode A* *Pathfinding* II

Pada bagian ini *path* akan terisi kosong jika tidak ada jalan mencapai *node* destinasi, jika tidak maka *path* akan bergerak mundur dari destinasi dan pergi ke masing-masing simpul sampai ke *starting point*. Fungsi heuristic digunakan untuk melakukan estimasi *cost* yang dibutuhkan untuk bergerak ketempat tersebut.

4.3 Implementasi *Artificial Intelligence* Level *Noob*

Implementasi *Artificial Intelligence* pada level *noob* untuk *game* dapat dilihat pada kode berikut

```
var myUnits = _cellGrid.Units.FindAll(u => u.PlayerNumber.Equals(Pla
yerNumber)).ToList();
        foreach (var unit in
            myUnits.OrderByDescending(u => u.Cell.GetNeighbours(_cel
lGrid.Cells).FindAll(u.IsCellTraversable).Count))
        {
            var enemyUnits = _cellGrid.Units.Except(myUnits).ToList(
);
            var unitsInRange = new List<Unit>();
            foreach (var enemyUnit in enemyUnits)
            {
                if (unit.IsUnitAttackable(enemyUnit,unit.Cell))
                {
                    unitsInRange.Add(enemyUnit);
                }
            }
            if (unitsInRange.Count != 0)
            {
                var index = _rnd.Next(0, unitsInRange.Count);
                unit.DealDamage(unitsInRange[index]);
                yield return new WaitForSeconds(0.5f);
                continue;
            }

            List<Cell> potentialDestinations = new List<Cell>();
```

Kode Sumber 4. 3 Potongan Kode *Noob* AI I

Pada bagian ini AI akan melihat selurruh unit pasukan sendiri dan unit pasukan lawan lalu melihat seluruh kemungkinan pasukan lawan yang masuk ke dalam jangkauan serangan jika ada maka lakukan serangan.

```
foreach (var enemyUnit in enemyUnits)
{
    potentialDestinations.AddRange(_cellGrid.Cells.FindAll(c=> unit.IsCellMovableTo(c)));
}

var notInRange = potentialDestinations.FindAll(c => c.GetDistance(unit.Cell) > unit.MovementPoints);
potentialDestinations = potentialDestinations.Except(notInRange).ToList();

if (potentialDestinations.Count == 0 && notInRange.Count !=0)
{
    potentialDestinations.Add(notInRange.ElementAt(_rnd.Next(0,notInRange.Count-1)));
}

potentialDestinations = potentialDestinations.OrderBy(h => _rnd.Next()).ToList();
List<Cell> shortestPath = null;

foreach (var potentialDestination in potentialDestinations)
{
    var path = unit.FindPath(_cellGrid.Cells, potentialDestination);

        unit.Move(potentialDestination, path);
        while (unit.isMoving)
            yield return 0;
        shortestPath = null;
        break;

    yield return 0;
}
```

Kode Sumber 4. 4 Potongan Kode *Noob* AI II

Pada bagian ini AI akan mencari kemungkinan jalan yang sesuai dengan *movements points* yang dimiliki dan akan bergerak secara random.

```
int randomone;

foreach (var enemyUnit in enemyUnits)
{
    randomone = Random.Range (0, 3);

    if (randomone == 2)
    {
        var enemyCell = enemyUnit.Cell;
        if (unit.IsUnitAttackable (enemyUnit, unit.Cell)) {
            unit.DealDamage (enemyUnit);
            yield return new WaitForSeconds (0.5f);
            break;
        }
    }
}
```

Kode Sumber 4. 5 Potongan Kode *Noob* AI III

Pada bagian ini AI akan melihat seluruh unit lawan yang masuk ke dalam area serangan, lalu secara random akan menyerang.

4.4 Implementasi *Artificial Intelligence* Level *Easy*

Implementasi *Artificial Intelligence* pada level *easy* untuk *game* dapat dilihat pada kode berikut

```
var myUnits = _cellGrid.Units.FindAll(u => u.PlayerNumber.Equals(Pla
yerNumber)).ToList();
 foreach (var unit in myUnits.OrderByDescending(u => u.Cell.GetNeigh
bours(_cellGrid.Cells).FindAll(u.IsCellTraversable).Count))
        {
            var enemyUnits = _cellGrid.Units.Except(myUnits).ToList(
);
            var unitsInRange = new List<Unit>();
            foreach (var enemyUnit in enemyUnits)
            {
                if (unit.IsUnitAttackable(enemyUnit,unit.Cell))
                {
                    unitsInRange.Add(enemyUnit);
                }
            }

            if (unitsInRange.Count != 0)
            {
                var index = _rnd.Next(0, unitsInRange.Count);
                unit.DealDamage(unitsInRange[index]);
                yield return new WaitForSeconds(0.5f);
                continue;
            }
```

Kode Sumber 4. 6 Potongan Kode *Easy* AI I

Pada bagian ini AI akan melihat seluruh unit pasukan sendiri dan unit pasukan lawan, lalu melihat seluruh kemungkinan pasukan lawan yang masuk ke dalam jangkauan serangan jika ada maka lakukan serangan.

```
List<Cell> potentialDestinations = new List<Cell>();

foreach (var enemyUnit in enemyUnits)
{
    potentialDestinations.AddRange(_cellGrid.Cells.FindAll(c=> unit.IsCellMovableTo(c)));
}

var notInRange = potentialDestinations.FindAll(c => c.GetDistance(unit.Cell) > unit.MovementPoints);
potentialDestinations = potentialDestinations.Except(notInRange).ToList();

if (potentialDestinations.Count == 0 && notInRange.Count !=0)
{
    potentialDestinations.Add(notInRange.ElementAt(_rnd.Next(0,notInRange.Count-1)));
}

potentialDestinations = potentialDestinations.OrderBy(h => _rnd.Next()).ToList();
List<Cell> shortestPath = null;
```

Kode Sumber 4. 7 Potongan Kode *Easy* AI II

Pada bagian ini AI akan mendaftarkan seluruh kemungkinan bergerak lalu membandingkannya dengan *movements points* yang dimiliki.

```
foreach (var potentialDestination in potentialDestinations)
{
    var path = unit.FindPath(_cellGrid.Cells, potentialDestination);
    if ((shortestPath == null && path.Sum(h => h.MovementCost) > 0))
        shortestPath = path;

    var pathCost = path.Sum(h => h.MovementCost);
    if (pathCost > 0 && pathCost <= unit.MovementPoints)
    {
        unit.Move(potentialDestination, path);
        while (unit.isMoving)
            yield return 0;
        shortestPath = null;
        break;
    }
    yield return 0;
}
```

Kode Sumber 4. 8 Potongan Kode *Easy* AI III

Pada bagian ini AI akan melihat lokasi *cell* dimana AI tersebut dapat bergerak dan menyerang, kemungkinan yang diambil adalah kemungkinan yang pertama kali di dapat

```
    if (shortestPath != null)
    {
        foreach (var potentialDestination in
            shortestPath.Intersect(unit.GetAvailableDestinations(_cellGrid.Cells)))
        {
            var path = unit.FindPath(_cellGrid.Cells, potentialDestination);
            var pathCost = path.Sum(h => h.MovementCost);
            if (pathCost > 0 && pathCost <= unit.MovementPoints)
            {
                unit.Move(potentialDestination, path);
                while (unit.isMoving)
                    yield return 0;
                break;
            }
            yield return 0;
        }
    }

    foreach (var enemyUnit in enemyUnits)
    {
        var enemyCell = enemyUnit.Cell;
        if (unit.IsUnitAttackable(enemyUnit,unit.Cell))
        {
            unit.DealDamage(enemyUnit);
            yield return new WaitForSeconds(0.5f);
            break;
        }
    }
}
_cellGrid.EndTurn();
```

Kode Sumber 4. 9 Potongan Kode *Easy* AI IV

pada bagian ini AI akan melihat jika kemungkinan bergerak ke posisi yang dapat menyerang melebihi *movements points* maka AI akan bergerak secara random ke tempat lain yang terjauh. Jika ada unit musuh dalam jarak serang ke unit pasukan sendiri maka serang unit tersebut .

4.5 Implementasi *Artificial Intelligence* Level *Medium*

Implementasi AI pada level medium untuk *game* dapat dilihat pada kode berikut

```
var myUnits = _cellGrid.Units.FindAll(u => u.PlayerNumber.Equals(Pla
yerNumber)).ToList();
        foreach (var unit in myUnits.OrderByDescending(u => u.Cell.G
etNeighbours(_cellGrid.Cells).FindAll(u.IsCellTraversable).Count))
        {
            var enemyUnits = _cellGrid.Units.Except(myUnits).ToList(
);
            var unitsInRange = new List<Unit>();
            foreach (var enemyUnit in enemyUnits)
            {
                if (unit.IsUnitAttackable(enemyUnit,unit.Cell))
                {
                    unitsInRange.Add(enemyUnit);
                }
            }
            if (unitsInRange.Count != 0)
            {
                var index = _rnd.Next(0, unitsInRange.Count);
                unit.DealDamage(unitsInRange[index]);
                yield return new WaitForSeconds(0.5f);
                continue;
            }
```

Kode Sumber 4. 10 Potongan Kode *Medium* AI I

Pada bagian ini ini AI akan melihat seluruh pasukan sendiri dan seluruh pasukan lawan dan melihat seluruh kemungkinan pasukan lawan yang masuk ke dalam jangkauan serangan. Jika ada maka lakukan serangan

```
List<Cell> potentialDestinations = new List<Cell>();

foreach (var enemyUnit in enemyUnits)
{
    potentialDestinations.AddRange(_cellGrid.Cells.FindAll(c=> unit.IsCellMovableTo(c) &&
    unit.IsUnitAttackable(enemyUnit, c)));
}

var notInRange = potentialDestinations.FindAll(c => c.GetDistance(unit.Cell) > unit.MovementPoints);
potentialDestinations = potentialDestinations.Except(notInRange).ToList();

if (potentialDestinations.Count == 0 && notInRange.Count !=0)
{
    potentialDestinations.Add(notInRange.ElementAt(_rnd.Next(0,notInRange.Count-1)));
}

potentialDestinations = potentialDestinations.OrderBy(h => _rnd.Next()).ToList();
List<Cell> shortestPath = null;
```

Kode Sumber 4. 11 Potongan Kode *Medium* AI II

Pada bagian ini AI akan melihat seluruh kemungkinan jalan dan menghabiskan pasukan lawan terbanyak dengan membandingkan ke *movements points* yang dimiliki.

```
foreach (var potentialDestination in potentialDestinations)
{
    var path = unit.FindPath(_cellGrid.Cells, potentialDestination);
    if ((shortestPath == null && path.Sum(h => h.MovementCost) > 0))
        shortestPath = path;

    var pathCost = path.Sum(h => h.MovementCost);
    if (pathCost > 0 && pathCost <= unit.MovementPoints)
    {
        unit.Move(potentialDestination, path);
        while (unit.isMoving)
            yield return 0;
        shortestPath = null;
        break;
    }
    yield return 0;
}
```

Kode Sumber 4. 12 Potongan Kode *Medium* AI III

Pada bagian ini AI akan melihat seluruh kemungkinan lokasi cell dimana unit pasukan dapat melakukan serangan ke unit lawan.

```
if (shortestPath != null)
{
    foreach (var potentialDestination in
    shortestPath.Intersect(unit.GetAvailableDestinations(_cellGrid.Cells)))
    {
        var path = unit.FindPath(_cellGrid.Cells, potentialDestination);
        var pathCost = path.Sum(h => h.MovementCost);
        if (pathCost > 0 && pathCost <= unit.MovementPoints)
        {
            unit.Move(potentialDestination, path);
            while (unit.isMoving)
                yield return 0;
            break;
        }
        yield return 0;
    }
}

foreach (var enemyUnit in enemyUnits)
{
    var enemyCell = enemyUnit.Cell;
    if (unit.IsUnitAttackable(enemyUnit,unit.Cell))
    {
        unit.DealDamage(enemyUnit);
        yield return new WaitForSeconds(0.5f);
        break;
    }
}
```

Kode Sumber 4.13 Potongan Kode *Medium* AI IV

Pada bagian ini AI akan melihat apabila tidak ada kemungkinan jalan dan menyerang, maka unit akan bergerak ke destinasi yang paling jauh yang paling memungkinkan untuk menyerang lawan. Jika ada unit musuh dalam jangkauan serangan maka lakukan serangan.

4.6 Implementasi *Artificial Intelligence* Level *Hard*

Implementasi AI pada level hard untuk *game* dapat dilihat pada kode berikut

```
var myUnits = _cellGrid.Units.FindAll(u => u.PlayerNumber.Equals(Pla
yerNumber)).ToList();
        foreach (var unit in myUnits.OrderByDescending(u => u.Cell.G
etNeighbours(_cellGrid.Cells).FindAll(u.IsCellTraversable).Count))
        {
            var enemyUnits = _cellGrid.Units.Except(myUnits).ToList(
);
            var unitsInRange = new List<Unit>();
            foreach (var enemyUnit in enemyUnits)
            {
                if (unit.IsUnitAttackable(enemyUnit,unit.Cell))
                {
                    unitsInRange.Add(enemyUnit);
                }
            }
            if (unitsInRange.Count != 0)
            {
               int i;
                for (i = 0; i < unitsInRange.Count; i++) {
                    if (unit is Archer && unitsInRange [i] is Spearm
an) {break; }
                    if (unit is Paladin && unitsInRange[i] is Archer
)  {break; }
                    if (unit is Spearman && unitsInRange [i] is Pala
din) {break; }
                }
                if (i == unitsInRange.Count)
                {var index = _rnd.Next(0, unitsInRange.Count);
                    unit.DealDamage(unitsInRange[index]);
                    yield return new WaitForSeconds(0.5f);
                    continue;}
                else
                {unit.DealDamage(unitsInRange[i]);
                    yield return new WaitForSeconds(0.5f);
                    continue; }
        }
```

Kode Sumber 4. 14 Potongan Kode *Hard* AI I

Pada bagian ini AI akan melihat seluruh pasukan unit sendiri dan pasukan lawan. Jika ada unit pasukan lawan dalam jangkauan serangan AI akan memprioritaskan untuk menyerang unit yang menghasilkan serangan dua kali lipat jika tidak ada maka lakukan serangan pada unit lain.

```
List<Cell> potentialDestinations = new List<Cell>();

foreach (var enemyUnit in enemyUnits)
{
    potentialDestinations.AddRange(_cellGrid.Cells.FindAll(c=> unit.IsCellMovableTo(c)
    && unit.IsUnitAttackable(enemyUnit, c)));
}

var notInRange = potentialDestinations.FindAll(c => c.GetDistance(unit.Cell) > unit.MovementPoints);
potentialDestinations = potentialDestinations.Except(notInRange).ToList();

if (potentialDestinations.Count == 0 && notInRange.Count !=0)
{
    potentialDestinations.Add(notInRange.ElementAt(_rnd.Next(0,notInRange.Count-1)));
}

potentialDestinations = potentialDestinations.OrderBy(h => _rnd.Next()).ToList();
List<Cell> shortestPath = null;
```

Kode Sumber 4. 15 Potongan Kode *Hard* AI III

Pada bagian ini AI akan melihat destinasi tempat bergerak yang akan megnhasilkan kerugian pasukan musuh terbanyak atau dapat melakukan serangan dengan membandingkan pada *movements points* yang dimiliki.

```
foreach (var potentialDestination in potentialDestinations)
{
    var path = unit.FindPath(_cellGrid.Cells, potentialDestination);
    if ((shortestPath == null && path.Sum(h => h.MovementCost) > 0)
        || shortestPath != null && (path.Sum(h => h.MovementCost) < shortestPath.Sum(h => h.MovementCost)
        && path.Sum(h => h.MovementCost) > 0))
        shortestPath = path;

    var pathCost = path.Sum(h => h.MovementCost);
    if (pathCost > 0 && pathCost <= unit.MovementPoints)
    {
        unit.Move(potentialDestination, path);
        while (unit.isMoving)
            yield return 0;
        shortestPath = null;
        break;
    }
    yield return 0;
}
```

Kode Sumber 4. 16 Potongan Kode *Hard* AI III

Pada bagian ini AI akan melihat destinasi tempat bergerak pada list destinasi yang telah dibuat yang dapat menyerang pasukan lawan yang terdekat.

```
if (shortestPath != null)
{
    foreach (var potentialDestination in
        shortestPath.Intersect(unit.GetAvailableDestinations(_cellGrid.Cells)).OrderByDescending(
            h => h.GetDistance(unit.Cell)))
    {
        var path = unit.FindPath(_cellGrid.Cells, potentialDestination);
        var pathCost = path.Sum(h => h.MovementCost);
        if (pathCost > 0 && pathCost <= unit.MovementPoints)
        {
            unit.Move(potentialDestination, path);
            while (unit.isMoving)
                yield return 0;
            break;
        }
        yield return 0;
    }
}

foreach (var enemyUnit in enemyUnits)
{
    var enemyCell = enemyUnit.Cell;
    if (unit.IsUnitAttackable(enemyUnit,unit.Cell))
    {
        unit.DealDamage(enemyUnit);
        yield return new WaitForSeconds(0.5f);
        break;
    }
}
```

Kode Sumber 4. 17 Potongan Kode *Hard* AI IV

Pada bagian ini AI akan melihat kemungkinan jalan jika tidak dapat menyerang, maka akan bergerak ke jarak terjauh yang dapat memungkinkan kerugian pasukan lawan terbanyak. Jika ada unit musuh dalam jarak serang, maka lakukan serangan ke unit tersebut.

4.7 Implementasi GUI *Controller*

Implementasi GUI controller untuk *game* dapat dilihat pada kode berikut

```
private void OnGameStarted(object sender, EventArgs e)
{
    foreach (Transform unit in UnitsParent.transform)
    {
        unit.GetComponent<Unit>().UnitHighlighted += OnUnitHighlighted;
        unit.GetComponent<Unit>().UnitDehighlighted += OnUnitDehighlighted;
        unit.GetComponent<Unit>().UnitAttacked += OnUnitAttacked;
    }

    foreach (Transform cell in CellGrid.transform)
    {
        cell.GetComponent<Cell>().CellHighlighted += OnCellHighlighted;
        cell.GetComponent<Cell>().CellDehighlighted += OnCellDehighlighted;
    }

    OnTurnEnded(sender,e);
}
```

Kode Sumber 4. 18 Potongan Kode GUI *Controller* I

Pada bagian ini akan mengatur pada saat *game* akan dimulai yang tampilannya akn tampak pada gambar 4.7. dibawah

```
private void OnGameEnded(object sender, EventArgs e)
{
    InfoText.text = "Player " + ((sender as CellGrid).CurrentPlayerNumber + 1) + " wins!";
}
private void OnTurnEnded(object sender, EventArgs e)
{
    NextTurnButton.interactable = ((sender as CellGrid).CurrentPlayer is HumanPlayer);

    InfoText.text = "Player " + ((sender as CellGrid).CurrentPlayerNumber +1);
}
private void OnCellDehighlighted(object sender, EventArgs e)
{
    UnitImage.color = Color.gray;
    StatsText.text = "";
}
private void OnCellHighlighted(object sender, EventArgs e)
{
    UnitImage.color = Color.gray;
    StatsText.text = "Movement Cost: " + (sender as Cell).MovementCost;
}
```

Kode Sumber 4. 19 Potongan Kode GUI *Controller* II

Pada bagian ini mengatur pemberitahuan saat salah satu pemain menang, pemberitahuan saat berganti giliran pemain mana yang jalan saat ini dan mengatur perubahan saat sebuah cell di klik dan tidak diklik.

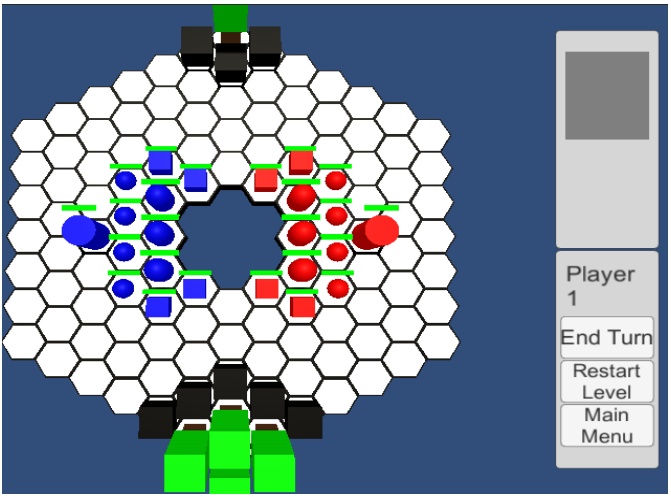
```
private void OnUnitDehighlighted(object sender, EventArgs e)
{
    StatsText.text = "";
    UnitImage.color = Color.gray;
}
private void OnUnitHighlighted(object sender, EventArgs e)
{
    var unit = sender as MyUnit;
    StatsText.text = unit.UnitName
        + "\nHit Points: " + unit.HitPoints +"/"+unit.TotalHitPoints
        + "\nAttack: " + unit.AttackFactor
        + "\nDefence: " + unit.DefenceFactor
        + "\nRange: " + unit.AttackRange;
    UnitImage.color = unit.PlayerColor;

}

public void RestartLevel()
{
    Application.LoadLevel(Application.loadedLevel);
}
```

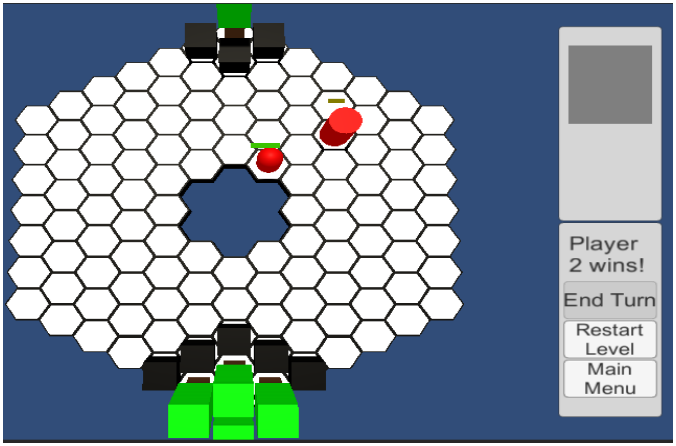
Kode Sumber 4. 20 Potongan Kode GUI *Controller* III

Pada bagian ini mengatur saat sebuah unit diklik akan menampilkan status unit, dan saat tidak dklik akan mengosongkan statusnya, serta tombol restart level yang dklik akan mengulang pertandingan dari awal.



Gambar 4. 1 UI pada saat *game* dimulai

Pada bagian ini gambar diatas menggambarkan kondisi *game* saat pertama dimulai. Terdapat field yang berisi pasukan , bar controller berisi tombol end turn, restart level, dan main menu.



Gambar 4. 2 UI pada saat *game* berakhir

Pada bagian ini gambar diatas menggambarkan situasi saat *game* berakhir dan dimenangkan oleh *player* 2.

4.8 Implementasi UI *Main Menu*

Implementasi UI main menu pada *game* dijelaskan pada kode berikut

```
public void clickPlay()
{
    int field_check = PlayerPrefs.GetInt ("field");
    int player_one = PlayerPrefs.GetInt ("playerone");
    if (player_one == 5) {
        if (field_check == 1) {SceneManager.LoadScene ("Field1_2");}
        else if (field_check == 2) {SceneManager.LoadScene ("Field2_2");}
        else if (field_check == 3) {SceneManager.LoadScene ("Field3_2");}
    }
    else {
        if (field_check == 1) {SceneManager.LoadScene ("Field1");}
        else if (field_check == 2) {SceneManager.LoadScene ("Field2");}
        else if (field_check == 3) {SceneManager.LoadScene ("Field3");}
    }
}
public void clickMainMenu()
{
    SceneManager.LoadScene("MainMenu");
}
public void ClickExit()
{
    Application.Quit();
}
```

Kode Sumber 4. 21 *Main Menu*

Pada bagian ini melakukan pengaturan saat klik pada tombol start *game* sesuai field yang diinginkan, tombol kembali pada main menu dan tombol exit *game*.

```
public void SelectToggle (int id)
{
    var toggles = toggleGroupInstance.GetComponentsInChildren<Toggle> ();
    toggles [id].isOn = true;
}

void Update()
{
    if (currentSelection.name == "Noob") {
        PlayerPrefs.SetInt ("playerone", 1);
    } else if (currentSelection.name == "Easy") {
        PlayerPrefs.SetInt ("playerone", 2);
    }
    else if (currentSelection.name == "Medium") {
        PlayerPrefs.SetInt ("playerone", 3);
    }
    else if (currentSelection.name == "Hard") {
        PlayerPrefs.SetInt ("playerone", 4);
    }
    else if (currentSelection.name == "Player") {
        PlayerPrefs.SetInt ("playerone", 5);
    }
}
```

Kode Sumber 4. 22 Potongan Kode *Toggle Group* I

Pada bagian ini melakukan pengaturan toggle group 1 untuk menyimpan level kekuatan *player* pertama.

```
public void SelectToggle (int id)
{
    var toggles = toggleGroupInstance.GetComponentsInChildren<Toggle> ();
    toggles [id].isOn = true;
}
void Update()
{

    Debug.Log (currentSelection.name);
    if (currentSelection.name == "Noob") {
        PlayerPrefs.SetInt ("playertwo", 1);
    } else if (currentSelection.name == "Easy") {
        PlayerPrefs.SetInt ("playertwo", 2);
    }
    else if (currentSelection.name == "Medium") {
        PlayerPrefs.SetInt ("playertwo", 3);
    }
    else if (currentSelection.name == "Hard") {
        PlayerPrefs.SetInt ("playertwo", 4);
    }
}
```

Kode Sumber 4. 23 Potongan Kode *Toggle Group* II

Pada bagian ini melakukan pengaturan toggle group 2 untuk menyimpan level kekuatan *player* kedua.

```
public void SelectToggle (int id)
{
    var toggles = toggleGroupInstance.GetComponentsInChildren<Toggle> ();
    toggles [id].isOn = true;
}

void Update()
{

    if (currentSelection.name == "field1") {
        PlayerPrefs.SetInt ("field", 1);
    } else if (currentSelection.name == "field2") {
        PlayerPrefs.SetInt ("field", 2);
    }
    else if (currentSelection.name == "field3") {
        PlayerPrefs.SetInt ("field", 3);
    }
}
```

Kode Sumber 4. 24 Potongan Kode *Toggle Group* III

Pada bagian ini melakukan pengaturan toggle group 3 untuk menyimpan field yang dijadikan tempat pertarungan.

Gambar 4. 3 UI *Main Menu*

Gambar diatas adalah gambar antarmuka halaman depan dari *game* "Zero *State*" *strategy turn-based RPG Game.*

4.9 Implementasi Unit

Implemntasi unit dalam *game* dapat dilihat pada kode berikut

```
public Cell Cell { get; set; }
public int HitPoints;
public int AttackRange;
public int AttackFactor;
public int DefenceFactor;
public int MovementPoints;
public float MovementSpeed;
public int ActionPoints;
public int PlayerNumber;

public bool isMoving { get; set; }

private static IPathfinding _pathfinder = new AStarPathfinding();
```

Kode Sumber 4. 25 Potongan Kode Program Unit I

Pada bagian ini terdapat nilai – nilai yang perlu diisi untuk dimasukan ke dalam tiap unit yaitu,

- *Hit Points*
  Hit points sering juga disebut *health points* yaitu total maksimum serangan yang dapat ditanggung oleh suatu unit.

- *Attack Range*
  Attack range merupakan jarak jangakauan serangan dari suatu unit, ada unit yang bertipe jarak jauh dan ada yang bertipe jarak dekat. *Attack range* menentukan apa suatu unit dapat menyerang unit dengan jangkauan yang unit itu miliki.

- *Attack Factor*
  Attack factor merupakan kekuatan serangan yang dihasilkan ketika unit menyerang unit lawan.

- *Defence factor*
  Defence factor merupakan kekuatan pertahanan yang dimiliki unit ketika unit lawan menyerang unit tersebut.

- *Movement points*
  Movement points adalah jangkauan pergerakan yang dimiliki oleh setiap pasukan. *Movements point* menentukan seberapa jauh sebuah unit dapat bergerak dibandingkan dengan movement *cost* dari tiap cell dalam field yang harus dilalui.

- *Movement speed*
  Movement speed adalah animasi bergerak yang dilakukan. Movement speed yang dipakai menentukan seberapa cepat pergerakan didalam *game*.

- *Action points*
  Action points adalah poin yang dibutuhkan untuk melakukan gerakan atau menyerang. Action *point* masing-masing unit adalah 1

- *Player number*
  Player number merupakan nilai yang menentukan unit tersebut berpihak kepada siapa.

- *Is Moving*
  Merupakan variabel yang menentukan apakah unit tersebut sedang bergerak atau tidak.

- *Pathfinder*
  Pathfinder merupakan fungsi yang dipakai untuk mencari jalan, algoritma dipakai adalah A* *pathfinding*.

```
public class MyUnit : Unit
{
    public Color PlayerColor;
    public string UnitName;
    private Transform Highlighter;
    public override void Initialize()
    {
        base.Initialize();
        SetColor(PlayerColor);

        Highlighter = transform.Find("Highlighter");
        if (Highlighter != null)
        {
            Highlighter.position = transform.position + new Vector3(0, 0, 1.5f);
            foreach (Transform cubeTransform in Highlighter)
                Destroy(cubeTransform.GetComponent<BoxCollider>());
        }
        gameObject.transform.position = Cell.transform.position + new Vector3(0, 0, -1.5f);
    }
    protected override void Defend(Unit other, int damage)
    {
        base.Defend(other, damage);
        UpdateHpBar();
    }
    public override void Move(Cell destinationCell, List<Cell> path)
    {
        base.Move(destinationCell, path);
    }
```

Kode Sumber 4. 26 Potongan Kode Program Unit II

Pada bagian ini mengatur inisialisasi unit dengan cara melakukan highlight, pada bagian ini juga terdapat fungsi untuk bertahan dengan cara memasukan *damage* dan target unit serta terdapat fungsi move untuk bergerak menuju cell yang dituju dan list *path* yang dilalui.

```
public override void MarkAsFriendly()
{
    SetHighlighterColor(new Color(0.8f,1,0.8f));
}
public override void MarkAsReachableEnemy()
{
    SetHighlighterColor(Color.red);
}
public override void MarkAsSelected()
{
    SetHighlighterColor(new Color(0,1,0));
}
public override void MarkAsFinished()
{
    SetColor(PlayerColor - Color.gray);
    SetHighlighterColor(new Color(0.8f, 1, 0.8f));
}
public override void UnMark()
{
    SetColor(PlayerColor);
    SetHighlighterColor(Color.white);
    if (Highlighter == null) return;
        Highlighter.position = transform.position + new Vector3(0, 0, 1.52f);
}
```

Kode Sumber 4. 27 Potongan Kode Program Unit III

Pada bagian ini terdapat beberapa fungsi untuk menandakan sebuah unit sebagai unit teman, sebagai musuh yang dapat dituju , sebagai unit yang dipilih, sebagai unit yang telah selesai dijalankan, dan unit yang tidak dipilih. Masing-masing diberikan perbedaan dengan memberikan warna highlight yang berbeda.

```
private void UpdateHpBar()
{
    if (GetComponentInChildren<Image>() != null)
    {
        GetComponentInChildren<Image>().transform.localScale =
            new Vector3((float)((float)HitPoints / (float)TotalHitPoints), 1, 1);
        GetComponentInChildren<Image>().color = Color.Lerp(Color.red, Color.green,
            (float)((float)HitPoints / (float)TotalHitPoints));
    }
}
private void SetColor(Color color)
{
    GetComponent<Renderer>().material.color = color;
}
private void SetHighlighterColor(Color color)
{
    if (Highlighter == null) return;
    Highlighter.position = transform.position + new Vector3(0, 0, 1.48f);
    for (int i = 0; i < Highlighter.childCount; i++)
    {
        var rendererComponent = Highlighter.transform.GetChild(i).GetComponent<Renderer>();
        if (rendererComponent != null)
            rendererComponent.material.color = color;
    }
}
```

Kode Sumber 4. 28 Potongan Kode Program Unit IV

Pada bagian ini terdapat fungsi untuk memperbarui hp bar yang dimiliki tiap unit, mengatur warna pada tiap unit dan warna highlight ketika suatu unit melakukan suatu.

4.10 Implementasi Variasi unit

Implementasi variasi unit pada *game* dapat dilihat pada kode berikut

```
public class Archer : MyUnit
{
    protected override void Defend(Unit other, int damage)
    {
        var realDamage = damage;
        if (other is Paladin)
            realDamage *= 2;

        base.Defend(other, realDamage);
    }
}
```

Kode Sumber 4. 29 Kode Program Unit *Archer*

Pada bagian ini unit *archer* memiliki seluruh fungsi yang ada pada class MyUnit dan fungsi override yaitu ketika melawan unit *paladin*, *paladin* menghasilkan serangan dua kali lipat.

```
public class Paladin : MyUnit
{
    protected override void Defend(Unit other, int damage)
    {
        var realDamage = damage;
        if (other is Spearman)
            realDamage *= 2;

        base.Defend(other, realDamage);
    }
}
```

Kode Sumber 4. 30 Kode Program Unit *Paladin*

Pada bagian ini unit *paladin* memiliki seluruh fungsi yang ada pada class MyUnit dan fungsi override yaitu ketika melawan unit *spearman*, *spearman* menghasilkan serangan dua kali lipat.

```
public class Spearman : MyUnit
{
    protected override void Defend(Unit other, int damage)
    {
        var realDamage = damage;
        if (other is Archer)
            realDamage *= 2;

        base.Defend(other, realDamage);
    }
}
```

Kode Sumber 4. 31 Kode Program Unit *Spearman*

Pada bagian ini unit *spearman* memiliki seluruh fungsi yang ada pada class MyUnit dan memiliki fungsi override yaitu ketika melawan *archer* , *archer* menghasilkan serangan dua kali lipat.

```
public class Hero : MyUnit
{
    private BuffSpawner _buffSpawner;
    private Button _specialAbilityButton;
    private bool _abilityUsed;

    public override void Initialize()
    {
        base.Initialize();
        _buffSpawner = new BuffSpawner();
        _specialAbilityButton = GetComponentInChildren<Button>();
        _specialAbilityButton.gameObject.SetActive(false);
        _specialAbilityButton.onClick.AddListener(TriggerSpecialAbility);
    }

    public override void OnTurnEnd()
    {
        base.OnTurnEnd();
    }
    public override void OnUnitSelected()
    {
        if (!_abilityUsed)
        {
            Invoke("EnableSpecialAbilityButton",0.1f);
        }
    }
    public override void OnUnitDeselected()
    {
        _specialAbilityButton.gameObject.SetActive(false);
    }
```

Kode Sumber 4. 32 Potongan Kode Program Unit *Hero* I

Pada bagian ini *hero* memiliki seluruh fungsi yang ada pada class MyUnit, memiliki fungsi inisialisasi, dan memiliki kemampuan buff khusus. Buff khusus tersebut akan muncul didekat unit *hero* jika belum pernah digunakan. Dalam satu *game* hanya dapat menggunakan satu kali.

```
private void EnableSpecialAbilityButton()
{
    _specialAbilityButton.gameObject.SetActive(true);
    _specialAbilityButton.interactable = true;
}
private void TriggerSpecialAbility()
{

    if (!_abilityUsed)
    {
        _abilityUsed = true;
        var buff = new AttackBuff(3, 2);
        buff.Apply(this);
        Buffs.Add(buff);

        _specialAbilityButton.gameObject.SetActive(false);
    }
}
```

Kode Sumber 4. 33 Potongan Kode Program Unit *Hero* II

Pada bagian ini terdapat fungsi untuk memunculkan buff kemampuan khusus dan trigger untuk mengaktifkan untuk menaikan kekuatan serangan dari unit *hero* yang berlaku 1 kali giliran.

[Halaman ini sengaja dikosongkan]

BAB V
PENGUJIAN DAN EVALUASI

Bab ini membahas tentang pengujian dan evaluasi pada perangkat lunak yang dibangun untuk tugas akhir ini. Pengujian dilakukan pada kasus penggunaan dari sistem perangkat lunak.

5.1 Lingkungan Pengujian

Pada proses pengujian perangkat lunak, dibutuhkan suatu lingkungan pengujian yang sesuai dengan standar kebutuhan. Lingkungan pengujian dalam tugas akhir ini dilakukan pada setiap level kemampuan AI. Spesifikasi lingkungan pengujian dijabarkan pada Tabel 5.1

Tabel 5. 1 Lingkungan Pengujian Fungsionalitas Perangkat Lunak

Spesifikasi	**Deskripsi**
Jenis Perangkat	*PC*
Merek Perangkat	ASUS A46CB
Sistem Operasi	Windows 10
Procesor	Core i-5
RAM	4 GB

5.2 Skenario Uji Coba

Sebelum melakukan uji coba, perlu ditentukan skenario yang akan digunakan dalam uji coba. Melalui skenario ini kecerdasan tiap level AI akan diuji dan bagaimana performa dari masing – masing AI pada tiap skenario. Terdapat 4 macam skenario uji coba , yaitu :

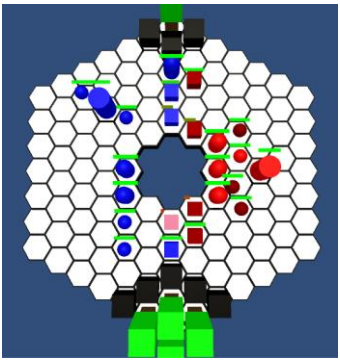
1. Pengujian aturan *game* pada ‘Zero State’.
2. Penghitungan kemenangan dan jumlah sisa pasukan pada masing – masing AI dengan cara mempertandingkan antar level AI pada *map* Field 1 tanpa menggunakan *random obstacles* dengan kombinasi pasukan 12 vs 12. Masing – masing AI akan dipertandingkan sebanyak 10 kali untuk tiap level.
3. Penghitungan kemenangan dan jumlah sisa pasukan pada masing – masing AI dengan cara mempertandingkan antar level AI pada *map* Field 2 dengan menggunakan *random obstacles* sebanyak 5 buah dengan kombinasi pasukan 8 vs 8. Masing-masing AI akan dipertandingkan sebanyak 10 kali untuk tiap level.
4. Penghitungan kemenangan dan jumlah sisa pasukan pada masing – masing kubu dengan cara mempertandingkan AI tiap level ke *player* pada *map* Field 1 tanpa menggunkan *random obstacles* dengan kombinasi pasukan 12 vs 12. Masing – masing *player* akan bertanding sebanyak 1 kali untuk tiap level AI. Level AI yang diujikan yaitu level *noob, easy, medium* dan *hard.*

5.2.1. Skenario Uji Coba 1

Pada skenario uji coba 1 ini pengujian dilakukan dengan cara melihat kesesuaian aturan *game* pada ‘Zero State’. Pengujian dapat ditunjukkan pada penjelasan berikut

1. Terdapat 2 belah pihak yang saling menyerang satu sama lain.
   Pada Gambar 5. 1 dapat dilihat terdapat 2 belah pihak yang saling menyerang satu sama lain.

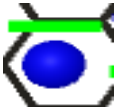
Player 1 adalah unit yang berwarna biru dan *Player* 2 adalah unit yang berwarna merah. Kedua belah pihak saling menyerang satu sama lain



Gambar 5. 1 Dua pihak saling menyerang

2. Terdapat 4 jenis pasukan yaitu *Spearman*, *Archer* dan *Paladin* yang memiliki kemampuan masing-masing serta 1 *Hero* yang kemampuannya lebih kuat dari pasukan lainnya.
   Pada Gambar 5. 2, Gambar 5. 3, Gambar 5. 4, dan Gambar 5. 5 dapat dilihat 4 jenis pasukan yaitu *spearman,archer*, *paladin*, dan *hero.*

Gambar 5. 2 *Spearman*

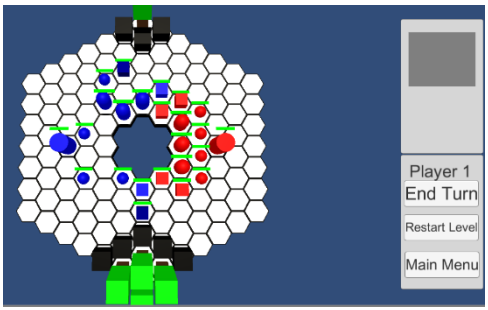


Gambar 5. 3 *Archer*

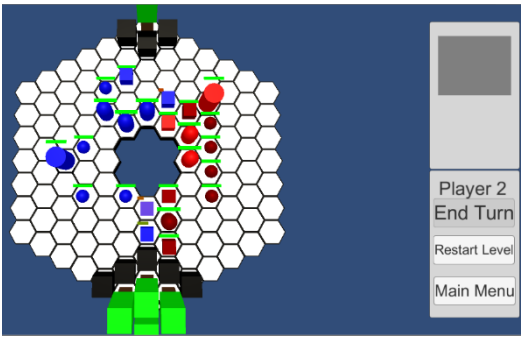
Gambar 5. 4 *Paladin*

Gambar 5. 5 *Hero*

3. Tiap pihak menyerang berdasarkan giliran yang ditentukan secara bergantian. Setiap giliran masing-masing pihak dapat menggerakan seluruh unitnya.
   Pada Gambar 5.6 dan Gambar 5.7 dapat dilihat giliran *player* 1 dan *player* 2. Pada masing-masing giliran *player* dapat menggerkaan seluruh unit yang dia miliki.

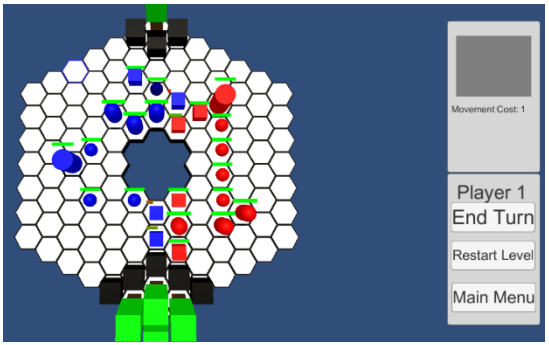


Gambar 5. 6 Giliran *Player* 1



Gambar 5. 7 Giliran *Player* 2

4. Masing – masing pasukan memiliki kelemahan yang apabila menyerang tipe pasukan tesebut menghasilkan serangan dua kali lipat kecuali *Hero.* Pada Gambar 5. 8 dapat dilihat pada bagian atas *player* 1 menggunakan unit *Archer* untuk menyerang unit *Spearman* lawan dan menghasilkan serangan dua kali lipat.



Gambar 5. 8 *Archer* menyerang *spearman*

5. Terdapat 4 tingkat kesulitan pada *game* yaitu *Noob*, *Easy*, *Medium*, dan *Hard*. Pemain dapat memilih untuk melawan AI dengan tingkat kesulitan yang sesuai kemampuannya
   Pada Gambar 5. 9 dapat dilihat pada *player* 1 dan *player* 2 terdapat pilihan tingkat kesulitan. Pada *player* 1 ada tambahan untuk tingkat kesulitan *player* apabila digunakan untuk dikendalikan oleh pemain.

Gambar 5. 9 Tingkat kesulitan pada *game*

5.2.2. Skenario Uji Coba 2

Pada skenario uji coba 2 ini pengujian dilakukan dengan cara mempertandingkan antar level AI dan melihat kemenangan serta jumlah pasukan yang tersisa. Pada pengujian ini menggunakan *map Field* 1 dengan jumlah pasukan masing-masing pihak adalah 12 unit, dan tidak menggunakan *random obstacles*. Hasil pengujian dapat dilihat pada Tabel 5. 2, Tabel 5. 3, Tabel 5. 4, Tabel 5. 5, Tabel 5. 6, dan Tabel 5. 7

Tabel 5. 2 Pengujian AI *hard* vs *medium*

No	AI Hard	AI Medium	Jumlah sisa pasukan
1	Menang	Kalah	8
2	Menang	Kalah	7
3	Menang	Kalah	3
4	Menang	Kalah	8
5	Menang	Kalah	6
6	Menang	Kalah	6
7	Menang	Kalah	6
8	Menang	Kalah	5
9	Menang	Kalah	8
10	Menang	Kalah	5

Tabel 5. 3 Pengujian AI *hard* vs *easy*

No	AI Hard	AI Easy	Jumlah sisa pasukan
1	Menang	Kalah	9
2	Menang	Kalah	7
3	Menang	Kalah	9
4	Menang	Kalah	8
5	Menang	Kalah	7
6	Menang	Kalah	9
7	Menang	Kalah	8
8	Menang	Kalah	8
9	Menang	Kalah	7
10	Menang	Kalah	7

Tabel 5. 4 Pengujian AI *hard* vs *noob*

No	AI Hard	AI Noob	Jumlah sisa pasukan
1	Menang	Kalah	10

2	Menang	Kalah	8
3	Menang	Kalah	6
4	Menang	Kalah	8
5	Menang	Kalah	4
6	Menang	Kalah	8
7	Menang	Kalah	6
8	Menang	Kalah	9
9	Menang	Kalah	4
10	Menang	Kalah	8

Tabel 5. 5 Pengujian AI *medium* vs *easy*

No	AI Medium	AI Easy	Jumlah sisa pasukan
1	Menang	Kalah	8
2	Menang	Kalah	7
3	Menang	Kalah	8
4	Menang	Kalah	9
5	Menang	Kalah	8
6	Menang	Kalah	9
7	Menang	Kalah	5
8	Menang	Kalah	8
9	Menang	Kalah	8
10	Menang	Kalah	8

Tabel 5. 6 Pengujian AI *medium* vs *noob*

No	AI Medium	AI Noob	Jumlah sisa pasukan
1	Menang	Kalah	7
2	Menang	Kalah	4
3	Menang	Kalah	7

4	Menang	Kalah	7
5	Menang	Kalah	5
6	Menang	Kalah	4
7	Menang	Kalah	5
8	Menang	Kalah	7
9	Menang	Kalah	5
10	Menang	Kalah	5

Tabel 5. 7 Pengujian AI *easy* vs *noob*

No	AI Easy	AI Noob	Jumlah sisa pasukan
1	Menang	Kalah	1
2	Menang	Kalah	4
3	Menang	Kalah	3
4	Menang	Kalah	4
5	Kalah	Menang	2
6	Menang	Kalah	3
7	Menang	Kalah	1
8	Menang	Kalah	4
9	Kalah	Menang	1
10	Menang	Kalah	6

5.2.3. Skenario Uji Coba 3

pada skenario uji coba 3 ini pengujian dilakukan dengan cara mempertandingkan antar level AI dan melihat kemenangan serta jumlah pasukan yang tersisa. Pada pengujian ini menggunakan *map Field* 2 dengan jumlah pasukan masing-masing pihak adalah 8 unit, dan menggunakan *random obstacles* sebanyak 5 buah. Hasil pengujian dapat dilihat pada Tabel 5. 8, Tabel 5. 9, Tabel 5. 10, Tabel 5. 11, Tabel 5. 12, dan Tabel 5. 13

Tabel 5. 8 Pengujian AI *hard* vs *medium*

No	AI Hard	AI Medium	Jumlah sisa pasukan
1	Kalah	Menang	2
2	Kalah	Menang	4
3	Kalah	Menang	2
4	Menang	Kalah	4
5	Menang	Kalah	3
6	Menang	Kalah	5
7	Menang	Kalah	4
8	Menang	Kalah	1
9	Menang	Kalah	2
10	Kalah	Menang	2

Tabel 5. 9 Pengujian AI *hard* vs *easy*

No	AI Hard	AI Easy	Jumlah sisa pasukan
1	Menang	Kalah	5
2	Menang	Kalah	7
3	Menang	Kalah	4
4	Menang	Kalah	4
5	Menang	Kalah	3
6	Menang	Kalah	5
7	Menang	Kalah	5
8	Menang	Kalah	6
9	Menang	Kalah	5
10	Menang	Kalah	7

Tabel 5. 10 Pengujian AI *hard* vs *noob*

No	AI Hard	AI Noob	Jumlah sisa pasukan
1	Menang	Kalah	7

2	Menang	Kalah	7
3	Menang	Kalah	5
4	Menang	Kalah	5
5	Menang	Kalah	3
6	Menang	Kalah	5
7	Menang	Kalah	6
8	Menang	Kalah	3
9	Menang	Kalah	7
10	Menang	Kalah	4

Tabel 5. 11 Pengujian AI *medium* vs *easy*

No	AI Medium	AI Easy	Jumlah sisa pasukan
1	Menang	Kalah	4
2	Menang	Kalah	5
3	Menang	Kalah	3
4	Menang	Kalah	3
5	Menang	Kalah	4
6	Menang	Kalah	5
7	Menang	Kalah	5
8	Menang	Kalah	5
9	Menang	Kalah	4
10	Menang	Kalah	3

Tabel 5. 12 Pengujian AI *medium* vs *noob*

No	AI Medium	AI Noob	Jumlah sisa pasukan
1	Menang	Kalah	3
2	Menang	Kalah	5
3	Menang	Kalah	6
4	Menang	Kalah	6
5	Menang	Kalah	7

6	Menang	Kalah	2
7	Menang	Kalah	3
8	Menang	Kalah	5
9	Menang	Kalah	5
10	Menang	Kalah	5

Tabel 5. 13 Pengujian AI *easy* vs *noob*

No	AI Easy	AI Noob	Jumlah sisa pasukan
1	Menang	Kalah	3
2	Menang	Kalah	3
3	Menang	Kalah	4
4	Kalah	Menang	3
5	Kalah	Menang	2
6	Menang	Kalah	2
7	Menang	Kalah	1
8	Menang	Kalah	2
9	Kalah	Menang	2
10	Menang	Kalah	3

5.2.4. Skenario Uji Coba 4

Pengujian kali ini menggunakan sampel sebanyak 5 orang mahasiswa ITS. Berikut data mengenai sampel :

1. Tester 1
   Nama : Richard Alvin Sianturi
   NRP : 5113100078

2. Tester 2
   Nama : Billy Adam
   NRP : 5113100047

3. Tester 3
   Nama : Rigold Nainggolan
   NRP : 5113100139

4. Tester 4
   Nama : Belli Martha J. Silaban
   NRP : 2313100046

5. Tester 5
   Nama : Peniel Simaremare
   NRP : 2713100095

Kelima tester tersebut melakukan pengujian terhadap tiap level AI dengan bertanding sebanyak 1 kali. Berikut data hasil pengujian dapat dilihat pada Tabel 5. 14, Tabel 5. 15 , Tabel 5. 16, dan Tabel 5. 17

Tabel 5. 14 Pengujian *player* vs AI *noob*

No	*Player*	AI *noob*	Jumlah sisa pasukan
1	Tester 1	Menang	9
2	Tester 2	Menang	11
3	Tester 3	Menang	7
4	Tester 4	Menang	7
5	Tester 5	Menang	9

Tabel 5. 15 Pengujian *player* vs AI *easy*

No	*Player*	AI *easy*	Jumlah sisa pasukan
1	Tester 1	Menang	4
2	Tester 2	Menang	9
3	Tester 3	Kalah	1

4	Tester 4	Menang	2
5	Tester 5	Menang	3

Tabel 5. 16 Pengujian *player* vs AI *medium*

No	*Player*	AI *medium*	Jumlah sisa pasukan
1	Tester 1	Kalah	3
2	Tester 2	Menang	7
3	Tester 3	Kalah	5
4	Tester 4	Menang	2
5	Tester 5	Menang	5

Tabel 5. 17 Pengujian *player* vs AI *hard*

No	*Player*	AI *hard*	Jumlah sisa pasukan
1	Tester 1	Menang	1
2	Tester 2	Menang	2
3	Tester 3	Kalah	4
4	Tester 4	Kalah	5
5	Tester 5	Kalah	2

5.3. Evaluasi

Berdasarkan Pengujian pada 5.2.2 dan 5.2.3 maka *winning rate* dari suatu AI berdasarkan kemenangan dapat dirangkum pada tabel berikut

Tabel 5. 18 Winning Rate AI vs AI

P1/P2	AI *Noob*	AI *Easy*	AI *Medium*	AI *Hard*
AI *Noob*	-	25%	0%	0%

AI *Easy*	75%	-	0%	0%
AI *Medium*	100%	100%	-	20%
AI *Hard*	100%	100%	80%	-

Evaluasi terhadap pengujian yang dilakukan dapat dijelaskan sebagai berikut :

1. Berdasarkan pengujian pada skenario uji coba 1 dapat dilihat bahwa aturan *game* pada 'Zero State' telah diterapkan dengan baik.
2. Berdasarkan pengujian pada skenario uji coba 2 dan skenario uji coba 3, dapat dilihat bahwa terdapat jenjang kekuatan antar level AI, terbutki pada probabilitas kemenangan AI dengan level yang lebih kuat. Hal ini dikarenakan semakin tinggi level AI probabilitas gerakan random semakin berkurang.
3. Berdasarkan pengujian pada skenario uji coba 2 dan uji coba 3, terdapat perbedaan field dan random obstacles namun hasil probabilitas kemenangan AI dengan level yang lebih tinggi cenderung tidak berubah, hal ini menunjukan bahwa strategi yang digunakan untuk memenangkan *game* pada AI tergolong baik.
4. Berdasarkan pengujian pada skenario uji coba 2 dan uji coba 3, AI dengan level Noob yang menggunakan algoritma random memiliki probabilitas yang sangat kecil untuk dapat menang dari AI yang menggunakan strategi untuk menang yaitu level Easy,Medium dan Hard. Hal ini menunjukan bahwa AI yang menggunakan algoritma strategi menang dapat diimplementasikan dengan baik dibanding tanpa algoritma.

5. Berdasarkan pengujian pada skenario uji coba 4, tiap tester memiliki hasil yang berbeda-beda saat bertanding melawan AI dengan level yang berbeda.Dari pengujian tersebut terlihat bahwa semakin tinggi level dari AI , tester cenderung kesulitan untuk memenangkan *game*, tidak hanya dari faktor itu, tapi dapat dilihat dari sisa pasukan yang dimiliki tester saat menang tergolong sedikit ketika level AI yang dilawan semakin tinggi. Hal ini membuktikan bahwa perancangan level AI dibuat dengan baik.
6. Dari tabel 5.18 dapat dilihat presentase kemenangan AI, ketika AI melawan level yang lebih tinggi presentase kemenangannya semakin kecil. Hal ini mennjukan bahwa perancangan level antar AI dibuat dengan baik.
7. Berdasarkan pengujian pada tabel 5.13 dapat dilihat bahwa terdapat sebanyak tiga kali kekalahan AI *easy* melawan AI *noob* yang tidak menggunakan algoritma, setelah dilakukan analisa hal ini disebabkan karena faktor gerakan *random* yang dimiliki AI *easy* sering muncul menyebabkan kerugian pasukan sendiri semakin banyak, yang seharusnya melakukan serangan ternyata tidak melakukan serangan sehingga hal tersebut menyebabkan kekalahan di pihak AI *easy*.

BAB VI
KESIMPULAN DAN SARAN

Bab ini membahas mengenai kesimpulan yang dapat diambil dari hasil pengujian yang telah dilakukan sebagai jawaban dari rumusan masalah yang dikemukakan. Selain kesimpulan, juga terdapat saran yang ditujukan untuk pengembangan perangkat lunak di masa mendatang.

6.1 Kesimpulan

1. Penggabungan penggunaan algoritma A* *pathfinding* dan aturan *game* dapat diterapkan dalam *game strategy turn-based* RPG dengan baik.
2. Berdasarkan pengujian bahwa AI yang menggunakan strategi memiliki probabilitas kemenangan jauh dibanding AI yang bergerak secara random.
3. Algoritma A* dapat membantu untuk membentuk AI yang kuat pada *game strategy turn-based* RPG.

6.2 Saran

1. Penambahan variasi unit pasukan agar AI dapat bekerja lebih baik.
2. Penambahan variasi strategi untuk memenangkan *game* yang dikombinasikan ke algoritma A*.

[Halaman ini sengaja dikosongkan]

DAFTAR PUSTAKA

[1] E. Adams, Fundamental Of *Game* Design , 2nd Edition, Berkeley: Pearson Education, Inc, 2010.

[2] Wikipedia, "C Sharp," 15 February 2015. [Online]. Available: http://id.wikipedia.org/wiki/C_sharp.. [Diakses 13 Desember 2016].

[3] R. M, Variasi Penggunaan Fungsi Heuristik Dalam Pengaplikasian Algoritma A*, Teknik Informatika, Institut Teknologi Bandung, 2007.

[4] Robin, "A Star Algorithm," November 2011. [Online]. Available: http://Intelligence.worldofcomputing.net/ai-search/a-star-algorithm/html. [Diakses 12 Desember 2016].

[5] Wikipedia, "Role-Playing *Game*," 30 April 2010. [Online]. Available: https://en.wikipedia.org/wiki/Role-playing_*game*. [Diakses 15 Desember 2016].

[6] Wikipedia, "Tactical Role Playing *GAME*," January 2012. [Online]. Available: https://en.wikipedia.org/wiki/Tactical_role-playing_*game*. [Diakses 15 Desember 2016].

[Halaman ini sengaja dikosongkan]

BIODATA PENULIS

Gideon Adiprana Tigor Siburian lahir di Jakarta 26 Agustus 1995. Penulis menempuh pendidikan formal dimulai dari TK Mutiara Indonesia (200-2001), SD Mutiara Indonesia (2001-2006), SMP Budi Mulia Pematangsiantar (2007-2010), SMA Budi Mulia Pematangsiantar (2010-2013) dan S1 Teknik Informatika ITS (2013-2017). Bidang studi yang diambil oleh penulis pada saat berkuliah di Teknik Informatika ITS adalah Interaksi Grafika dan Seni (IGS). Penulis dapat dihubungi melalui surel pribadi pada dionadiprana@gmail.com atau melalui surel formal pada gideon13@mhs.if.its.ac.id