TUGAS AKHIR – KI1502

Implementasi *Artificial Intelligence* pada *Game* Defender of Metal City dengan menggunakan *Finite State Machine*

Billy
NRP 5113100047

Dosen Pembimbing
Imam Kuswardayan, S.Kom., MT.
Wijayanti Nurul Khotimah S.Kom., M.Sc.

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

[Halaman ini sengaja dikosongkan]

TUGAS AKHIR – KI1502

IMPLEMENTASI *ARTIFICIAL INTELLIGENCE* PADA *GAME* DEFENDER OF METAL CITY DENGAN MENGGUNAKAN *FINITE STATE MACHINE*

Billy
NRP 5113100047

Dosen Pembimbing
Imam Kuswardayan, S.Kom., MT.
Wijayanti Nurul Khotimah S.Kom., M.Sc.

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

[Halaman ini sengaja dikosongkan]

UNDERGRADUATE THESES – KI1502

ARTIFICIAL INTELLIGENCE IMPLEMENTATION IN GAME DEFENDER OF METAL CITY USING FINITE STATE MACHINE

Billy
NRP 5113100047

Supervisors
Imam Kuswardayan, S.Kom., MT.
Wijayanti Nurul Khotimah S.Kom., M.Sc.

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

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

IMPLEMENTASI *ARTIFICIAL INTELLIGENCE* PADA *GAME* DEFENDER OF METAL CITY DENGAN MENGGUNAKAN *FINITE STATE MACHINE*

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
BILLY
NRP: 5113 100 047

Disetujui oleh Dosen Pembimbing Tugas Akhir:

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

Wijayanti Nurul Khotimah S.Kom., M.Sc. .....
(NIP 198603122012122004) (Pembimbing 2)

SURABAYA
JUNI, 2017

[Halaman ini sengaja dikosongkan]

IMPLEMENTASI ARTIFICIAL INTELLIGENCE PADA GAME DEFENDER OF METAL CITY DENGAN MENGGUNAKAN FINITE STATE MACHINE

Nama Mahasiswa	**:**	**Billy**
NRP	**:**	**5113 100 047**
Jurusan	**:**	**Teknik Informatika FTIf-ITS**
Dosen Pembimbing 1	**:**	**Imam Kuswardayan, S.Kom., MT.**
Dosen Pembimbing 2	**:**	**Wijayanti Nurul Khotimah S.Kom., M.Sc.**

Abstrak

Game adalah jenis perangkat lunak yang populer dewasa ini. Pada game biasa terdapat *Artificial Inteligence* (AI) yang mengendalikan pihak komputer sehingga manusia yang bermain dapat merasakan seolah mereka sedang melawan pemain manusia lain saat bermain. Salah satu jenis *game* yang ada ialah *tower defense* dimana pemain perlu mempertahankan *tower* atau markas nya agar tidak dihancurkan oleh pihak lawan. Untuk membuat AI untuk game berjenis *tower defense*, diperlukan algoritma yang baik untuk mengatur gerakan pasukan lawan.

Penelitian ini mengusulkan penerapan AI pada game berjenis tower defense menggunakan *rule based Finite State Machine*. Setiap pasukan memiliki AI dengan beberapa *state*, pada saat terjadi suatu kondisi tertentu, maka pasukan akan berpindah dari satu state ke state lainnya berdasarkan *rule* yang telah ditentukan, hanya ada satu *state* yang bisa aktif pada suatu waktu. Pada penelitian ini AI akan dibagi menjadi 3 level yaitu *easy*, *medium* dan *hard*. Pengujian dilakukan dengan mempertandingkan antar level AI pada semua

kemungkinan tingkat kesulitan, lalu dibuat tabel derajat kemenangan atau *winning* r*ate* dari AI yang ada.

Berdasarkan hasil pengujian *Finite State Machine* dapat diterapkan pada game berjenis *tower defense* dengan baik untuk membentuk AI yang kuat.

Kata kunci: Game, Artificial Intelligence, Tower Defense, Unity

ARTIFICIAL INTELLIGENCE IMPLEMENTATION IN GAME DEFENDER OF METAL CITY USING FINITE STATE MACHINE

Student Name	**:**	**Billy**
NRP	**:**	**5113 100 047**
Major	**:**	**Informatics Department FTIf – ITS**
Advisor I	**:**	**Imam Kuswardayan, S.Kom., MT.**
Advisor II	**:**	**Wijayanti Nurul Khotimah S.Kom., M.Sc.**

Abstract

Games are a popular type of software today. In ordinary games there are Artificial Intelligence (AI) that controls the computer so that humans who play can feel as if they are fighting other human players while playing. One type of game that exist is a tower defense where players need to keep its tower or headquarters so as not to be destroyed by the opponent. To create an AI for a tower defense type game, a good algorithm is needed to regulate the opponent's movement.

This research proposes the application of AI in tower defense game using rule based Finite State Machine. Each army has an AI with several states, in the event of a certain condition, then the said troop will move from one state to another state based on a predetermined rule, there is only one state that can be active at a time. In this study AI will be divided into 3 levels of easy, medium and hard. Testing is done by comparing the inter-level AI on all possible levels of difficulty, then a table of AI winning rate is created.

Based on the results of Finite State Machine testing can be applied to the type of tower defense game well to form a strong AI.

Keywords: Game, Artificial Intelligence, Tower Defense, **Unit**

KATA PENGANTAR

Puji syukur kepada Tuhan Yesus Kristus atas kasih karunia dan rahmat-Nya sehingga penulis dapat menyelesaikan tugas akhir yang berjudul

"IMPLEMENTASI ARTIFICIAL INTELLIGENCE PADA GAME DEFENDER OF METAL CITY DENGAN MENGGUNAKAN FINITE STATE MACHINE".

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. Tuhan Yesus Kristus
2. Ibu dan 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 Wijayanti Nurul Khotimah S.Kom., M.Sc. 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. Seluruh dosen dan karyawan Teknik Informatika ITS yang telah memberikan ilmu dan pengalaman kepada penulis selama menjalani masa studi di ITS.
7. 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

Billy

DAFTAR ISI

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

DAFTAR KODE SUMBER

[Halaman ini sengaja dikosongkan]

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 sangat pesat akhir-akhir ini membawa banyak perubahan dalam perkembangan perangkat lunak, salah satu jenis perangkat lunak yang sangat popular dewasa ini adalah *game*, *game* merupakan perangkat lunak yang digunakan sebagai sarana hiburan baik oleh kaum anak-anak ataupun orang dewasa.

Salah satu fitur yang umum dan dapat ditemui pada hampir semua *game* yang ada pada zaman sekarang adalah *Artificial Intelligence* atau AI, AI adalah kepintaran buatan yang dipasang pada sistem komputer sehingga memungkinkan pemain untuk bermain melawan komputer dengan keadaan seolah-olah ia melawan pemain lain, membuat AI adalah suatu tantangan tersendiri bagi developer *game.*

Defender of Metal City adalah *game* berjenis *tower defense* yang dirancang dengan AI. Tujuan dari AI adalah untuk meningkatkan antusiasme pemain dengan adanya lawan AI dengan tingkat kesulitan yang sepadan.

Tujuan dari pengerjaan Tugas Akhir ini adalah dapat menghasilkan aplikasi *game* berjenis *tower defense* yang memiliki AI dengan tingkat kesulitan yang dapat mengimbangi kemampuan pemain.

1.2. Rumusan Masalah

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

1. Bagaimana merancang strategi *game* dari AI pada *game* ‘Defender of Metal City’?
2. Bagaimana mengimplementasikan *Finite State Machine* dalam pembuatan AI pada *game* ‘Defender of Metal City’?
3. Bagaimana peraturan bermain dalam *game* ‘Defender of Metal City’?

1.3. Batasan Masalah

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

1. Pada *game* ini apabila suatu pasukan terkena serangan maka tidak dapat di sembuhkan kembali.
2. *Game* ini hanya dapat dimainkan oleh satu orang (*single player*).

1.4. Tujuan

Tugas akhir ini mempunyai beberapa tujuan, yaitu sebagai berikut:

1. Membuat *game* berjenis *tower defense* dengan *Artificial Intelligence* yang bisa bermain dengan pemain manusia.
2. Mengimplementasikan *Finite State Machine* untuk membuat *Artificial Intelligence* pada *game* ‘Defender of Metal City’.

1.5. Manfaat

Manfaat Tugas Akhir ini adalah untuk menghasilkan *game* berjenis *tower defense* yang bisa memiliki *Artificial Intelligence* yang dapat bermain dengan pemain manusia.

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

2. Studi literatur

   Studi literatur yang dilakukan pada perancangan aplikasi *game* sebagai pengerjaan tugas akhir ini adalah mengenai implementasi AI *(Artificial Inteligcence)* pada *game* berjenis *tower defense* dengan menggunakan *Finite State* Machine.

3. Analisis dan desain perangkat lunak

   Tahap ini meliputi perumusan aturan *game*, strategi untuk menang, dan penerapan AI pada *game* 'Defender of Metal City'.

4. Implementasi perangkat lunak

   Aplikasi ini diimplementasikan dengan menggunakan kakas bantu:

   1. Sistem Operasi Windows 10 Home.
   2. *Game* Engine Unity Personal versi 5.5.1f
   3. Bahasa permrograman C#.
   4. Text Editor Atom.

5. Pengujian dan evaluasi

   Pengujian dan evaluasi aplikasi perangkat lunak hasil dari tugas akhir ini diujicobakan dengan memainkan pertandingan AI melawan AI dan AI melawan manusia.

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, spesifikasi kebutuhan perangkat lunak, lingkungan perancangan, perancangan arsitektur sitem, diagram kelas, dan struktur data.

Bab IV Implementasi

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

[Halaman ini sengaja dikosongkan]

BAB II
TINJAUAN PUSTAKA

Bab ini menjelaskan teori-teori yang berkaitan dengan pembangunan aplikasi 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 Artificial Intelligence

Artificial Intelligence atau Kecerdasan Buatan adalah kecerdasan yang dimiliki oleh mesin yang dapat bertindak seperti layaknya manusia[1]. Pada *game*, AI bertindak dalam mengendalikan komputer agar dapat bermain melawan pemain manusia.

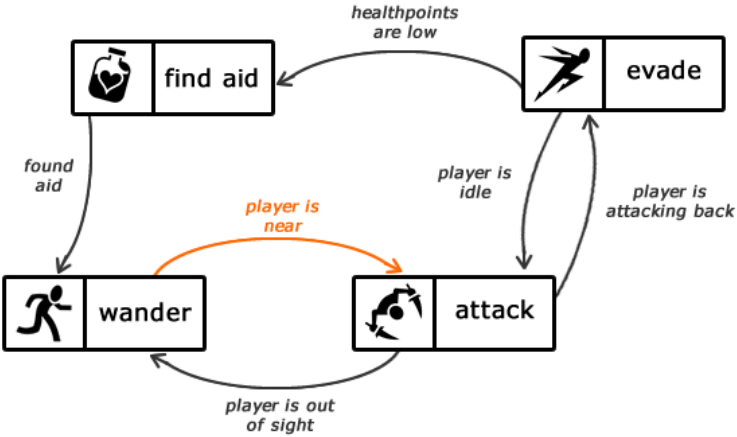
2.2 Tower Defense

Tower defense adalah *sub-genre* dari *game real-time strategy* dimana tujuan utama pemain adalah melindungi *tower* utamanya dari serangan musuh dengan menempatkan pertahanan di sepanjang jalan menuju *tower* tersebut, apabila musuh berhasil menghancurkan *tower* milik pemain maka pemain akan kalah[2]. Beberapa contoh *game* berjenis *tower defense* antara lain adalah Kingdom Rush dan Bloons TD.

2.3 Finite State Machine

Finite State Machine atau FSM adalah sebuah model yang digunakan untuk menggambarkan alur dari suatu proses berdasarkan *state* atau keadaan dari suatu proses, hanya ada satu *state* yang bisa aktif pada suatu waktu sehingga dibutuhkan adanya perpindahan ke *state* lain apabila ingin melakukan aksi yang lain[3].

FSM biasa digunakan untuk mengatur dan menentukan alur tindakan, yang berguna untuk mengimplementasikan AI dalam *game*. Contoh FSM yang digunakan sebagai AI dari musuh pada *game* dapat dilihat pada Gambar 2.1.



Gambar 2.1 Contoh Penggunaan FSM dalam AI pada *Game*

2.4 Unity

Unity adalah sebuah *Game Engine Cross-platform* yang dapat menghasilkan *game* untuk berbagai macam platform termasuk Windows, Mac, Linux, Android, iOS, dan banyak lagi. Unity pertama kali diterbitkan pada Tahun 2005 dan telah banyak berkembang sejak saat itu. Unity memiliki *Built-in* IDE yang bermana MonoDevelop.

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 yang dapat bermain layaknya pemain manusia

3.1 Spesifikasi *Game*

Pada *game* ini terdapat dua pihak yang saling menyerang satu sama lain, target nya adalah menghancurkan *tower* lawan sementara mempertahankan agar *tower* nya sendiri jangan sampai hancur, pihak yang *tower*nya hancur akan kalah.

Pada *game* ini terdapat 4 jenis pasukan yang sama pada kedua belah pihak, yaitu *minion*, *warrior*, *ranger*, dan *artillery*. Pada *game* ini kedua belah pihak berusaha saling menyerang *Tower* milik pihak lawan.

Peraturan *game* adalah sebagai berikut:

1. Terdapat 2 belah pihak yang saling menyerang satu sama lain.
2. Terdapat 4 jenis pasukan berbeda pada setiap pihak yang memiliki kemampuannya sendiri-sendiri.
3. Terdapat *tower* pada kedua belah pihak yang apabila hancur maka pihak tersebut akan kalah.
4. *Tower* dapat menyerang pasukan musuh yang mendekat
5. Apabila membunuh pasukan milik musuh maka akan mendapat sejumlah uang.
6. Kedua pihak memiliki pilihan untuk meng-*upgrade* pasukan yang dimilikinya agar lebih kuat hingga *level* 3, dengan setiap *upgrade* membutuhkan biaya sejumlah uang.

7. Kedua belah pihak memiliki pilihan untuk membuat pasukan *elite* yaitu pasukan yang jauh lebih kuat dari pasukan biasa dengan membayar sejumlah uang.
8. Terdapat 3 tingkat kesulitan pada *game* yaitu *easy*, *medium*, dan *hard*, pemain dapat memilih akan melawan AI dengan tingkat kesulitan mana dan akan menggunakan tingkat kesulitan mana untuk AI pada pihak pemain itu sendiri.

Terdapat 4 jenis pasukan pada *game* ini yaitu:

1. *Minion*
   Tipe pasukan yang lemah, memiliki *damage* yang kecil namun dapat bergerak dan menyeran dengan cepat, hanya dapat menyerang dari jarak dekat dan memiliki *health point* yang sedikit.
2. *Warrior*
   Tipe pasukan yang bergerak dengan kecepatan sedang, memiliki *health point* yang cukup sehingga dapat menyerang lawan dari jarak dekat, dan juga memiliki *damage* dan kecepatan menyerang yang lebih besar daripada tipe *minion.*
3. *Ranger*
   Tipe pasukan yang menyerang dari jarak jauh, memiliki *damage* dan *health point* yang lebih besar daripada tipe *minion* tetapi masih lebih kecil daripada tipe *warrior* sehingga tidak dapat menyerang dari jarak dekat, bergerak dan menyerang dengan kecepatan yang sama dengan tipe *warrior.*
4. *Artillery*
   Tipe pasukan terkuat yang memiliki *area damage* sehingga dapat menyerang beberapa musuh secara bersamaan, memiliki *damage* dan *health point* paling besar diantara semua pasukan namun kecepatan serangan dan kecepatan bergeraknya adalah yang paling lambat diantara semua tipe pasukan yang lain.

Pasukan akan diproduksi oleh *tower* secara periodik sehingga pasukan akan terus dihasilkan. Namun untuk pasukan *elite*, pemain harus mengendalikan kapan akan membeli pasukan *elite* yang mana karena pembelian pasukan *elite* harus menggunakan uang.

Pasukan dapat berjalan di *map* pada area yang bukan hutan. Pemain dapat mengendalikan pergerakan pasukan dengan kontrol menggunakan *mouse* untuk mengatur kemana pasukan harus pergi atau pasukan musuh mana yang harus diserang. Pada *game* ini akan ada 3 *map* yang berbeda yang dapat dipilih saat bermain.

Kondisi kemenangan *game* adalah menghancurkan *tower* milik lawan. Kondisi kekalahan *game* adalah saat *tower* milik pemain dihancurkan oleh lawan.

3.2 Strategi Memenangkan *Game*

Strategi memenangkan *game* adalah dengan mengatur agar pasukan dapat seoptimal mungkin membunuh sebanyak mungkin musuh sehingga dapat terus maju sehingga pada akhirnya menyerang *tower* musuh. Juga harus diperhatikan pertahanan terhadap *tower* sendiri karena apabila *tower* hancur maka pemain akan kalah tidak peduli berapa banyak pasukan musuh yang telah dibunuhnya.

Karena setiap pasukan memiliki kemampuan yang berbeda-beda maka penggunaan setiap pasukan juga berbeda-beda:

1. Pasukan tipe *minion* memiliki *damage* yang tidak terlalu besar namun dapat bergerak cepat sehingga dapat dimanfaatkan untuk membunuh pasukan yang sudah lemah atau *health point* nya tersisa sedikit. Apabila ingin menyerang, *minion* akan lebih efektif bila menyerang bersama-sama karena dapat dengan cepat mengurangi *health point* musuh.

2. Pasukan tipe *warrior* dapat digunakan sebagai garis depan karena memiliki pertahanan dan *damage* yang lumayan sehingga dapat melawan musuh.
3. Pasukan tipe *ranger* dapat menyerang dari jarak jauh sehingga dapat digunakan sebagai *backup* dari belakang untuk membantu pasukan yang sedang menyerang di garis depan.
4. Pasukan tipe *artillery* dapat digunakan untuk melawan musuh dalam jumlah banyak karena dapat melakukan area *damage*, dimana ia bisa menyerang lebih dari 1 musuh pada sekali serangan, pasukan ini bisa membantu tipe *warrior* di garis depan dalam menghadapi musuh.

Untuk strategi mengatur kapan melakukan *upgrade* pasukan dan men-*deploy* elite pasukan, *upgrade* harus dilakukan terlebih dahulu karena *upgrade* bersifat permanen. Apabila telah di-*upgrade*, semua pasukan akan menjadi lebih kuat, tidak seperti pasukan *elite* yang hanya akan keluar sekali dan cepat atau lambat akan mati, jadi *upgrade* harus dilakukan terlebih dahulu. Apabila semua pasukan telah di *upgrade* hingga level tertinggi, barulah uang yang tersisa digunakan untuk menghasilkan pasukan *elite.*

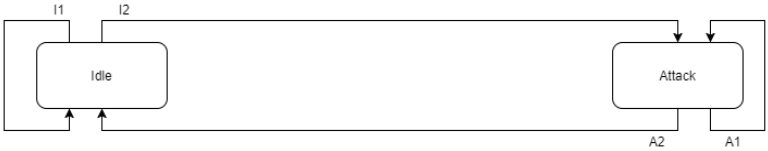
3.3 Penerapan *Artificial Intelligence*

Penerapan *Artificial Intelligence* pada *game* ini dibuat dengan menggunakan *rule-base Finite State Machine*, setiap pemain memiliki *state* tertentu dimana *state* terebut dapat berubah menjadi *state* lain apabila menemui suatu kondisi tertentu, kondisi-kondisi perubahan *state* dibuat berdasarkan startegi memenangkan *game*.

Pada bagian ini akan dibagi menjadi 5 sub-bagian, yaitu Penerapan AI pada *tower*, penerapan AI pada pasukan tipe *minion*, penerapan AI pada pasukan tipe *warrior*, penerapan AI pada pasukan tipe *ranger*, dan penerapan AI pada pasukan tipe *artillery*,

3.3.1 Penerapan AI pada *Tower*

Tower adalah bagian penting dalam *game* karena apabila *tower* hancur, pihak tersebut kalah, pada *tower*, hanya ada 2 keadaan, *idle* yaitu tidak melakukan apapaun, atau *attack* yaitu menyerang dengan menembakkan panah pada saat ada musuh mendekat dengan jarak kurang dari sama dengan 30 meter.



Gambar 3.1 FSM pada *Tower*

FSM yang digunakan oleh *tower* dapat dilihat pada Gambar 3.1. Penjelasan untuk kondisi perpindahan *state* pada FSM pada *tower* dapat dilihat sebagai berikut:

I1. Tidak ada musuh dalam jarak 30 meter.
I2. Ada musuh dalam jarak 30 meter.
A1. Ada musuh dalam jarak 30 meter.
A2. Tidak ada musuh dalam jarak 30 meter.

Semua angka jarak yang berbeda-beda didapat dari hasil percobaan. Dari beberapa kemungkinan angka yang telah dicoba telah disimpulkan bahwa angka – angka yang didapat adalah angka yang terbaik untuk menghasilkan AI *tower* yang kuat.

3.3.2 Penerapan Tingkat Kesulitan AI

Pada *game* ini terdapat 3 tingkat kesulitan AI, yaitu *easy*, *medium*, dan *hard*, terdapat beberapa hal yang membedakan ketiga tingkat kesulitan tersebut.

Pada ketiga tingkat kesulitan yang berbeda, terdapat delay saat mengambil keputusan, tingkat kesulitan yang lebih rendah akan memiliki *delay* yang lebih tinggi sehingga AI akan lebih lambat saat berpikir, pada tingkat kesulitan *easy*, pengambilan keputusan dilakukan setiap 50 *frame*, pada tingkat kesulitan *medium*, pengambilan keputusan dilakukan setiap 25 *frame*, dan pada tingkat kesulitan *hard*, pengambilan keputusan dilakukan setiap 1 *frame*.

Pada ketiga tingkat kesulitan, juga terdapat kemungkinan pengambilan keputusan secara *random*, dimana AI akan bertindak secara *random* dan tidak mengikuti *rule-base* yang telah ditetapkan, semakin rendah tingkat kesulitan AI maka akan semakin tinggi kemungkinan pengambilan keputusan secara *random* tersebut. Pada tingkat kesultian *hard*, tidak ada kemungkinan keputusan *random*. Pada tingkat kesulitan *medium*, terdapat 15% kemungkinan pengambilan keputusan *random*. Pada tingkat kesulitan *easy*, terdapat 30% kemungkinan pengambilan keputusan *random*. Pada tingkat *medium*, terdapat 4 kemungkinan keputusan *random* yang dapat diambil, yaitu bergerak secara *random*, pulang kembali ke *tower*, mengikuti teman secara *random*, dan menyerang musuh secara *random*. Pada tingkat kesulitan *easy* hanya terdapat 3 kemungkinan keputusan random yang dapat diambil, yaitu bergerak secara *random*, pulang kembali ke *tower*, dan mengikuti teman secara *random*.

Pada bagian pengambilan keputusan untuk meng-*upgrade* pasukan atau men-*deploy* pasukan *elite*, juga terdapat perbedaan. Pada tingkat kesulitan *easy*, pengambilan keputusan dilakukan secara *random* total. Pada tingkat *medium*, akan dilihat apabila ada jenis pasukan yang belum mencapai level maksimal, maka ada 70% kemungkinan pasukan tersebut akan di-*upgrade* terlebih dahulu, dan 30% sisanya adalah kemungkinan mengeluarkan pasukan *elite* secara *random*. Pada tingkat kesulitan *hard*, pasukan akan selalu di-*upgrade* terlebih dahulu hingga mencapai level maksimal, barulah berikutnya akan mengeluarkan pasukan *elite* secara *random*.

Semua angka presentase kemungkinan dan jumlah *frame* untuk semua tingkat kesulitan yang berbeda-beda didapat dari hasil percobaan. Dari beberapa kemungkinan angka yang telah dicoba telah disimpulkan bahwa angka – angka yang didapat adalah angka yang terbaik untuk menghasilkan AI yang sesuai dengan tingkat kesulitannya.

3.3.3 Penerapan AI Pasukan *Minion*

Karena *minion* adalah tipe pasukan yang lemah namun dapat bergerak cepat maka akan kurang efisien apabila menyerang sendirian apalagi saat menghadapi musuh yang lebih besar atau kuat, pada pasukan *minion* terdapat 5 *state* yaitu *idle*, *move*, *attack*, *chase*, dan *retreat*.

Rule-base yang digunakan pada pasukan tipe *minion* adalah sebagai berikut:

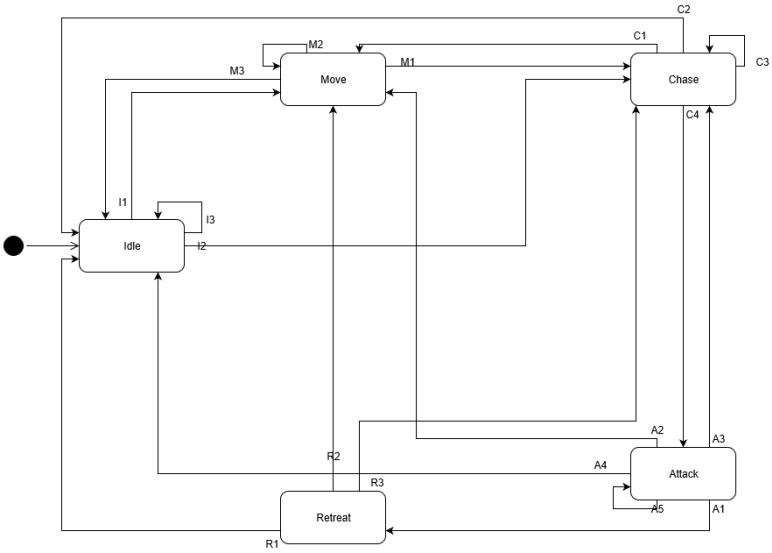
Saat tidak bertarung:

- Jika *health point* pemain lebih dari 25%.
  - Jika tak ada musuh dalam jarak 20 meter, *move*.
  - Jika ada:
    - Jika ada musuh tipe *ranger*, target musuh tipe *ranger* terdepan, *chase.*
    - Jika tidak, target musuh terlemah, *chase*
- Jika *health point* pemain kurang dari sama dengan 25%:
  - Jika dalam jarak 20 meter ada 5 atau lebih teman
    - Jika ada musuh dalam jarak 15 meter, target musuh terlemah, *chase*.
    - Jika tidak, *move*.
  - Jika tidak, *idle*.

Saat bertarung:

- Jika *health point* pemain lebih dari 25%, *attack*.
- Jika *health point* pemain kurang dari sama dengan 25%:
  - Jika musuh tipe *minion*:
    - Jika dalam jarak 20 meter terdapat 5 atau lebih teman tipe *minion*, *attack*.

    - Jika *health point* musuh lebih kecil dari *health point* pemain, *attack*.
    - Jika tidak, *retreat*.
- Jika musuh tipe *ranger*:
    - Jika *health point* musuh kurang dari sama dengan 50%, *attack*.
    - Jika tidak, *retreat.*
- Jika musuh tipe *warrior*:
    - Jika dalam jarak 20 meter terdapat 5 atau lebih teman tipe *minion*, *attack*.
    - Jika *health point* musuh kurang dari *health point* pemain, *attack*.
    - Jika tidak, *retreat*.
- Jika musuh tipe *artillery*:
    - Jika *health point* musuh kurang dari *health point* pemain, *attack*.
    - Jika tidak, *retreat*.



Gambar 3.2 FSM pada Pasukan *Minion*

FSM pada pasukan *minion* dapat dilihat pada Gambar 3.2. Penjelasan untuk kondisi perpindahan *state* pada FSM pada pasukan *minion* dapat dilihat sebagai berikut:

I1. Salah satu kondisi berikut terpenuhi:
- *Health point* lebih dari 25%, tidak ada musuh dalam jarak 20 meter.
- *Health point* kurang dari sama dengan 25%, dalam jarak 20 meter ada 5 atau lebih teman.

I2. Salah satu kondisi berikut terpenuhi:
- *Health point* lebih dari 25%, ada musuh tipe *ranger*.
- *Heallth point* lebih dari 25%, tidak ada musuh tipe *ranger*.

- *Health point* kurang dari sama dengan 25%, dalam jarak 20 meter ada 5 atau lebih teman dan ada musuh dalam jarak 15 meter.

I3. *Health point* kurang dari 25%.

M1. Salah satu kondisi berikut terpenuhi:
- *Health point* lebih dari 25%, ada musuh tipe *ranger*.
- *Heallth point* lebih dari 25%, tidak ada musuh tipe *ranger*.
- *Health point* kurang dari sama dengan 25%, dalam jarak 20 meter ada 5 atau lebih teman dan ada musuh dalam jarak 15 meter.

M2. Salah satu kondisi berikut terpenuhi:
- *Health point* lebih dari 25%, tidak ada musuh dalam jarak 20 meter.
- *Health point* kurang dari sama dengan 25%, dalam jarak 20 meter ada 5 atau lebih teman.

M3. *Health point* kurang dari 25%.

R1. *Health point* kurang dari 25%.

R2. Salah satu kondisi berikut terpenuhi:
- *Health point* lebih dari 25%, tidak ada musuh dalam jarak 20 meter.
- *Health point* kurang dari sama dengan 25%, dalam jarak 20 meter ada 5 atau lebih teman.

R3. Salah satu kondisi berikut terpenuhi:
- *Health point* lebih dari 25%, ada musuh tipe *ranger*.
- *Health point* lebih dari 25%, tidak ada musuh tipe *ranger*.
- *Health point* kurang dari sama dengan 25%, dalam jarak 20 meter ada 5 atau lebih teman dan ada musuh dalam jarak 15 meter.

C1. Salah satu kondisi berikut terpenuhi:
- *Health point* lebih dari 25%, tidak ada musuh dalam jarak 20 meter.
- *Health* point kurang dari sama dengan 25%, dalam jarak 20 meter ada 5 atau lebih teman.

C2. *Health point* kurang dari 25%.
C3. Salah satu kondisi berikut terpenuhi:
- *Health point* lebih dari 25%, ada musuh tipe ranger.
- *Heallth point* lebih dari 25%, tidak ada musuh tipe *ranger.*
- *Health point* kurang dari sama dengan 25%, dalam jarak 20 meter ada 5 atau lebih teman dan ada musuh dalam jarak 15 meter.

C4. Jarak target kurang dari sama dengan 3 meter.
A1. Salah satu kondisi berikut terpenuhi:
- *Health point* kurang dari 25%, musuh tipe *minion* atau *warrior*, tidak ada 5 atau lebih teman tipe *minion* dalam 20 meter dan *health* kurang dari *health point* musuh.
- *Health point* pemain kurang dari 25%, musuh tipe *ranger*, *health point* musuh lebih dari sama dengan 50%.
- *Health point* pemain kurang dari 25%, musuh tipe *artillery*, *health point* musuh lebih dari *health point* pemain.

A2. Salah satu kondisi berikut terpenuhi:
- Target mati atau hilang, *health point* lebih dari 25%, tidak ada musuh dalam jarak 20 meter.
- Musuh mati atau hilang, *health point* kurang dari sama dengan 25%, dalam jarak 20 meter ada 5 atau lebih teman.

A3. Salah satu kondisi berikut terpenuhi:
- Target mati atau hilang, *health point* lebih dari 25%, ada musuh tipe *ranger.*
- Target mati atau hilang, *health point* lebih dari 25%, tidak ada musuh tipe *ranger.*
- Target mati atau hilang, *health point* kurang dari sama dengan 25%, dalam jarak 20 meter ada 5 atau lebih teman dan ada musuh dalam jarak 15 meter.

A4. Target mati atau hilang, *health point* kurang dari 25%.

A5. Salah satu kondisi berikut terpenuhi:
- *Health point* lebih dari 25%.
- *Health point* kurang dari sama dengan 25%, musuh tipe *minion* atau *warrior*, dalam jarak 20 meter. terdapat 5 atau lebih teman tipe *minion.*
- *Health point* kurang dari sama dengan 25%, musuh tipe *minion* atau *warrior*, *health point* musuh lebih kecil dari *health point* pemain.
- *Health point* pemain kurang dari 25%, musuh tipe *ranger*, *health point* musuh kurang dari 50%.
- *Health point* pemain kurang dari 25%, musuh tipe *artillery*, dan *health point* musuh kurang dari *health point* pemain.

Semua angka presentase *health point* dan jarak yang berbeda-beda didapat dari hasil percobaan. Dari beberapa kemungkinan angka yang telah dicoba telah disimpulkan bahwa angka – angka yang didapat adalah angka yang terbaik untuk menghasilkan AI pasukan *minion* yang kuat.

3.3.4 Penerapan AI Pasukan *Warrior*

Warrior adalah tipe pasukan yang kuat namun dan dapat bertarung dalam jarak dekat sehingga dapat digunakan sebagai garis depan saat melawan musuh. Pada pasukan tipe *warrior* terdapat 5 *state* yaitu *idle*, *move*, *attack*, *chase*, dan *retreat*.

Rule-base yang digunakan pada pasukan tipe *warrior* adalah sebagai berikut:

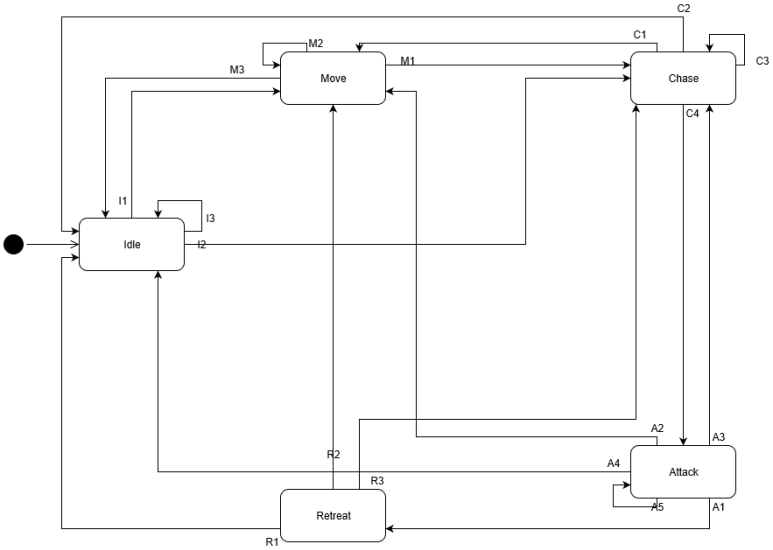
Saat tidak bertarung:

- Jika *health point* pemain lebih dari 20%:
  - Jika tak ada musuh dalam jarak 20 meter, *move*.
  - Jika ada, target musuh terdepan, *chase*.
- Jika *health point* pemain kurang dari sama dengan 20%:
  - Jika dalam jarak 20 meter ada 5 atau lebih teman:
    - Jika teman tipe *warrior* atau *ranger*:

        - Jika ada musuh dalam range, target musuh terdepan, *chase*.
        - Jika tidak, *move*.
      - Jika teman tipe *minion*:
        - Jika ada musuh dalam jarak 20 meter, target musuh terlemah, *chase*.
        - Jika tidak *move*.
    - Jika dalam jarak 20 meter ada musuh tipe *minion* atau *ranger*, target terdekat, *chase*.
    - Jika tidak, *idle*.

Saat bertarung:

- Jika *health point* pemain lebih dari 20%, *attack*.
- Jika *health point* pemain kurang dari sama dengan 20%:
    - Jika musuh tipe *minion*, *attack*.
    - Jika musuh tipe *warrior* atau *ranger*:
      - Jika dalam 15 meter ada 5 atau lebih teman tipe *warrior* atau *ranger*, *attack*.
      - Jika dalam 15 meter ada 5 atau lebih teman tipe *minion*, ganti target menjadi musuh terlemah, *chase*.
      - Jika *health point* musuh kurang dari *health point* pemain, *attack*.
      - Jika tidak *retreat*.
    - jika musuh tipe *artillery*:
      - Jika *health point* musuh kurang dari *health point* pemain, *attack*.
      - Jika tidak, *retreat*.



Gambar 3.3 FSM pada Pasukan *Warrior*

FSM yang digunakan oleh pasukan *warrior* dapat dilihat pada Gambar 3.3. Penjelasan untuk kondisi perpindahan *state* pada FSM pada pasukan *warrior* dapat dilihat sebagai berikut:

I1. Salah satu kondisi berikut terpenuhi:
- *Health point* lebih dari 20%, tidak ada musuh dalam jarak 20 meter.
- *Health point* kurang dari sama dengan 20%, ada 5 atau lebih teman dalam jarak 20 meter dan ada musuh dalam jarak 20 meter, tidak ada musuh dalam jarak 20 meter.
- *Health point* kurang dari sama dengan 20%, ada 5 atau lebih teman tipe *minion* dalam jarak 20 meter, tidak ada musuh dalam jarak 20 meter.

I2. Salah satu kondisi berikut terpenuhi:
- *Health point* lebih dari 20%, ada musuh dalam jarak 20 meter.

- *Health point* kurang dari sama dengan 20%, ada 5 atau lebih teman tipe *warrior* atau *ranger* dalam jarak 20 meter dan ada musuh dalam jarak 20 meter.
- *Health point* kurang dari sama dengan 20%, ada 5 atau lebih teman tipe *minion* dalam jarak 20 meter dan ada musuh dalam jarak 20 meter.
- *Health point* kurang dari 20%, tidak ada teman dalam jarak 20 meter, ada musuh tipe *minion* atau *ranger*.

I3. *Health point* kurang dari 20%, tidak ada 5 atau lebih teman dalam jarak 20 meter, tidak ada musuh tipe *minion* atau *ranger.*

M1. Salah satu kondisi berikut terpenuhi:

- *Health point* lebih dari 20%, ada musuh dalam jarak 20 meter.
- *Health point* kurang dari sama dengan 20%, ada 5 atau lebih teman tipe *warrior* atau *ranger* dalam jarak 20 meter dan ada musuh dalam jarak 20 meter.
- *Health point* kurang dari sama dengan 20%, ada 5 atau lebih teman tipe *minion* dalam jarak 20 meter dan ada musuh dalam jarak 20 meter.
- *Health point* kurang dari 20%, tidak ada teman dalam jarak 20 meter, ada musuh tipe *minion* atau *ranger*.

M2. Salah satu kondisi berikut terpenuhi:

- *Health point* lebih dari 20%, tidak ada musuh dalam jarak 20 meter.
- *Health point* kurang dari sama dengan 20%, ada 5 atau lebih teman dalam jarak 20 meter dan ada musuh dalam jarak 20 meter, tidak ada musuh dalam jarak 20 meter.
- *Health point* kurang dari sama dengan 20%, ada 5 atau lebih teman tipe *minion* dalam jarak 20 meter, tidak ada musuh dalam jarak 20 meter.

M3. *Heath point* kurang dari 20%, tidak ada 5 atau lebih teman dalam jarak 20 meter, tidak ada musuh tipe *minion* atau *ranger.*

R1. *Health point* kurang dari 20%, tidak ada 5 atau lebih teman dalam jarak 20 meter, tidak ada musuh tipe *minion* atau *ranger*.

R2. Salah satu kondisi berikut terpenuhi:
- *Health point* lebih dari 20%, tidak ada musuh dalam jarak 20 meter.
- *Health point* kurang dari sama dengan 20%, ada 5 atau lebih teman dalam jarak 20 meter dan ada musuh dalam jarak 20 meter, tidak ada musuh dalam jarak 20 meter.
- *Health point* kurang dari sama dengan 20%, ada 5 atau lebih teman tipe *minion* dalam jarak 20 meter, tidak ada musuh dalam jarak 20 meter.

R3. Salah satu kondisi berikut terpenuhi:
- *Health point* lebih dari 20%, ada musuh dalam jarak 20 meter.
- *Health point* kurang dari sama dengan 20%, ada 5 atau lebih teman tipe *warrior* atau *ranger* dalam jarak 20 meter dan ada musuh dalam jarak 20 meter.
- *Health point* kurang dari sama dengan 20%, ada 5 atau lebih teman tipe *minion* dalam jarak 20 meter dan ada musuh dalam jarak 20 meter.
- *Health point* kurang dari 20%, tidak ada teman dalam jarak 20 meter, ada musuh tipe *minion* atau *ranger*.

C1. Salah satu kondisi berikut terpenuhi:
- *Health point* lebih dari 20%, tidak ada musuh dalam jarak 20 meter.
- *Health point* kurang dari sama dengan 20%, ada 5 atau lebih teman dalam jarak 20 meter dan ada musuh dalam jarak 20 meter, tidak ada musuh dalam jarak 20 meter.
- *Health point* kurang dari sama dengan 20%, ada 5 atau lebih teman tipe *minion* dalam jarak 20 meter, tidak ada musuh dalam jarak 20 meter.

C2. *Health point* kurang dari 20%, tidak ada 5 atau lebih teman dalam jarak 20 meter, tidak ada musuh tipe *minion* atau *ranger*.
C3. Salah satu kondisi berikut terpenuhi:
- *Health point* lebih dari 20%, ada musuh dalam jarak 20 meter.
- *Health point* kurang dari sama dengan 20%, ada 5 atau lebih teman tipe *warrior* atau *ranger* dalam jarak 20 meter dan ada musuh dalam jarak 20 meter.
- *Health point* kurang dari sama dengan 20%, ada 5 atau lebih teman tipe *minion* dalam jarak 20 meter dan ada musuh dalam jarak 20 meter.
- *Health point* kurang dari 20%, tidak ada teman dalam jarak 20 meter, ada musuh tipe *minion* atau *ranger*.

C4. Jarak target kurang dari sama dengan 3 meter.
A1. Salah satu kondisi berikut terpenuhi:
- *Health point* pemain kurang dari 20%, musuh tipe *warrior* atau *ranger*, tidak ada teman dalam jarak 15 meter, dan *health point* musuh lebih dari *health point* pemain.
- *Health point* pemain kurang dari 20%, musuh tipe *artillery*, tidak ada teman dalam jarak 15 meter, dan *health point* musuh lebih dari *health point* pemain.

A2. Salah satu kondisi berikut terpenuhi:
- Target mati atau hilang, *health point* lebih dari 20%, tidak ada musuh dalam jarak 20 meter.
- Target mati atau hilang, *health point* kurang dari sama dengan 20%, ada 5 atau lebih teman dalam jarak 20 meter dan ada musuh dalam jarak 20 meter, tidak ada musuh dalam jarak 20 meter.
- Target mati atau hilang, *health point* kurang dari sama dengan 20%, ada 5 atau lebih teman tipe *minion* dalam jarak 20 meter, tidak ada musuh dalam jarak 20 meter.

A3. Salah satu kondisi berikut terpenuhi:

- Target mati atau hilang, *health point* lebih dari 20%, ada musuh dalam jarak 20 meter.
- Target mati atau hilang, *health point* kurang dari sama dengan 20%, ada 5 atau lebih teman tipe *warrior* atau *ranger* dalam jarak 20 meter dan ada musuh dalam jarak 20 meter.
- Target mati atau hilang, *health point* kurang dari sama dengan 20%, ada 5 atau lebih teman tipe *minion* dalam jarak 20 meter dan ada musuh dalam jarak 20 meter.
- Target mati atau hilang, *health point* kurang dari 20%, tidak ada teman dalam jarak 20 meter, ada musuh tipe *minion* atau *ranger*.

A4. Target mati atau hilang, *health point* kurang dari 20%, tidak ada 5 atau lebih teman dalam jarak 20 meter, tidak ada musuh tipe *minion* atau *ranger*.

A5. Salah satu kondisi berikut terpenuhi:

- *Health point* lebih dari 20%.
- *Health point* kurang dari 20%, musuh tipe *minion.*
- *Health point* kurang dari 20%, musuh tipe *ranger* atau *warrior*, dalam 15 meter ada 5 atau lebih teman tipe *warrior* atau *ranger*.
- *Health point* kurang dari 20%, musuh tipe *ranger* atau *warrior*, dalam 15 meter ada 5 atau lebih teman tipe *minion.*
- *Health point* pemain kurang dari 20%, musuh tipe *ranger* atau *warrior*, *health point* musuh kurang dari *health point* pemain.
- *Health point* pemain kurang dari 20%, musuh tipe *artillery*, *health point* musuh kurang dari *health point* pemain.

Semua angka presentase *health point* dan jarak yang berbeda-beda didapat dari hasil percobaan. Dari beberapa kemungkinan angka yang telah dicoba telah disimpulkan bahwa

angka – angka yang didapat adalah angka yang terbaik untuk menghasilkan AI pasukan *warrior* yang kuat.

3.3.5 Penerapan AI Pasukan *Ranger*

Ranger adalah tipe pasukan yang dapat menyerang dari jarak jauh sehingga dapat membantu pasukan jarak dekat lainnya seperti *warrior*, *minion*, atau *artillery* saat melawan musuh. Pada pasukan tipe *ranger* terdapat 6 *state* yaitu *idle*, *move*, *attack*, *chase*, *retreat*, dan *follow*.

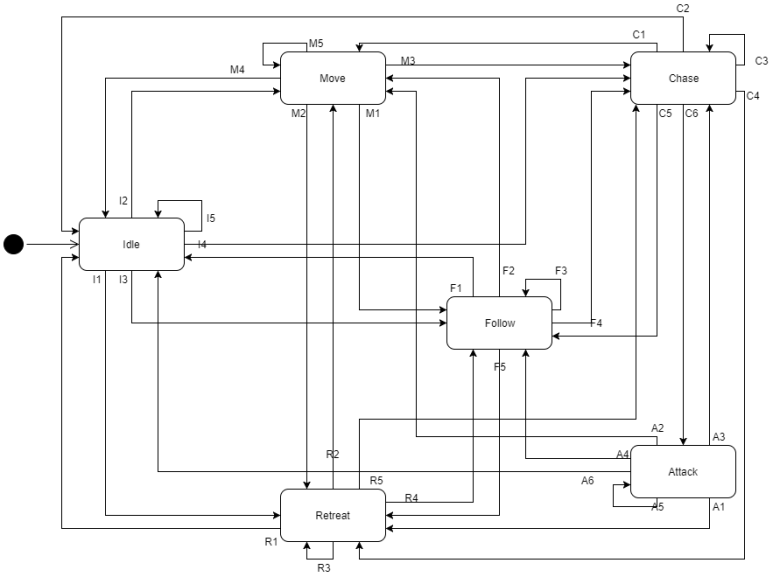
Rule-base yang digunakan pada pasukan tipe *ranger* adalah sebagai berikut:

Saat tidak bertarung:

- Jika *health point* pemain lebih dari 20%:
  - Jika tak ada musuh dalam jarak 30 meter:
    - Jika *health point* pemain kurang dari 50%:
      - Jika ada teman tipe *warrior* atau *artillery*, *follow*.
      - Jika tidak, *idle*.
    - Jika *health point* lebih dari sama dengan 50%, *move*:
      - Jika ada musuh dalam jarak 30 meter, *attack*.
- Jika *health point* pemain kurang dari sama dengan 20%:
  - Jika ada musuh dalam jarak kurang dari sama dengan 5 meter, *retreat*.
  - Jika ada musuh dalam jarak lebih dari 5 meter dan kurang dari 30 meter, pilih musuh terlemah, *attack*.
  - Jika tidak, *idle*.

Saat bertarung:

- Jika jarak musuh lebih dari sama dengan 5 meter dan kurang dari 20 meter, *attack*.
- Jika jarak musuh kurang dari 5 meter, *retreat*.



Gambar 3.4 FSM pada Pasukan *Ranger*

FSM pada pasukan *ranger* dapat dilihat pada Gambar 3.4. Penjelasan untuk kondisi perpindahan *state* pada FSM pada pasukan *ranger* dapat dilihat sebagai berikut:

I1. *Health point* kurang dari sama dengan 20% dan ada musuh dalam jarak kurang dari 5 meter.
I2. *Health point* lebih dari sama dengan 50%, tidak ada musuh dalam jarak 30 meter.
I3. *Health point* lebih dari 20% dan kurang dari 50%, dan ada teman tipe *warrior* atau *artillery*.
I4. Salah satu kondisi berikut terpenuhi:
    - *Health point* lebih dari sama dengan 50%, ada musuh dalam jale-rak 30 meter.
    - *Health point* kurang dari 20% dan ada musuh dalam jarak lebih dari 5 meter dan kurang dari 30 meter.
I5. Salah satu kondisi berikut terpenuhi:

- *Health point* lebih dari 20% dan kurang dari 50%, dan tidak ada teman tipe *warrior* atau *artillery*.
- *Health point* kurang dari 20%.

M1. *Health point* lebih dari 20% dan kurang dari 50%, dan ada teman tipe *warrior* atau *artillery*.

M2. *Health point* kurang dari sama dengan 20% dan ada musuh dalam jarak kurang dari 5 meter.

M3. Salah satu kondisi berikut terpenuhi:

- *Health point* lebih dari sama dengan 50%, ada musuh dalam jarak 30 meter.
- *Health point* kurang dari 20% dan ada musuh dalam jarak lebih dari 5 meter dan kurang dari 30 meter.

M4. Salah satu kondisi berikut terpenuhi:

- *Health point* lebih dari 20% dan kurang dari 50%, dan tidak ada teman tipe *warrior* atau *artillery*.
- *Health point* kurang dari 20%.

M5. *Health point* lebih dari sama dengan 50%, tidak ada musuh dalam jarak 30 meter.

F1. Salah satu kondisi berikut terpenuhi:

- *Health point* lebih dari 20% dan kurang dari 50%, dan tidak ada teman tipe *warrior* atau *artillery*.
- *Health point* kurang dari 20%.

F2. *Health point* lebih dari sama dengan 50%, tidak ada musuh dalam jarak 30 meter.

F3. *Health point* lebih dari 20% dan kurang dari 50%, dan ada teman tipe *warrior* atau *artillery*

F4. Salah satu kondisi berikut terpenuhi:

- *Health point* lebih dari sama dengan 50%, ada musuh dalam jarak 30 meter.
- *Health point* kurang dari 20% dan ada musuh dalam jarak lebih dari 5 meter dan kurang dari 30 meter.

F5. *Health point* kurang dari sama dengan 20% dan ada musuh dalam jarak kurang dari 5 meter.

R1. Salah satu kondisi berikut terpenuhi:

- *Health point* lebih dari 20% dan kurang dari 50%, dan tidak ada teman tipe *warrior* atau *artillery.*
- *Health point* kurang dari 20%.

R2. *Health point* lebih dari sama dengan 50%, tidak ada musuh dalam jarak 30 meter.
R3. *Health point* kurang dari sama dengan 20% dan ada musuh dalam jarak kurang dari 5 meter.
R4. *Health point* lebih dari 20% dan kurang dari 50%, dan ada teman tipe *warrior* atau *artillery*.
R5. Salah satu kondisi berikut terpenuhi:

- *Health point* lebih dari sama dengan 50%, ada musuh dalam jarak 30 meter.
- *Health point* kurang dari 20% dan ada musuh dalam jarak lebih dari 5 meter dan kurang dari 30 meter.

C1. *Health point* lebih dari sama dengan 50%, tidak ada musuh dalam jarak 30 meter.
C2. Salah satu kondisi berikut terpenuhi:

- *Health point* lebih dari 20% dan kurang dari 50%, dan tidak ada teman tipe *warrior* atau *artillery*.
- *Health point* kurang dari 20%.

C3. Salah satu kondisi berikut terpenuhi:

- *Health point* lebih dari sama dengan 50%, ada musuh dalam jarak 30 meter.
- *Health point* kurang dari 20% dan ada musuh dalam jarak lebih dari 5 meter dan kurang dari 30 meter.

C4. *Health point* kurang dari sama dengan 20% dan ada musuh dalam jarak kurang dari 5 meter.
C5. *Health point* lebih dari 20% dan kurang dari 50%, dan ada temanb tipe *warrior* atau *artillery*.
C6. Jarak target kurang dari 30 meter.
A1. Salah satu kondisi berikut terpenuhi:

- Jarak target kurang dari 5 meter.
- Target mati atau hilang, *health point* kurang dari sama dengan 20% dan ada musuh dalam jarak kurang dari 5 meter.

A2. Target mati atau hilang, *health point* lebih dari sama dengan 50%, tidak ada musuh dalam jarak 30 meter.
A3. Salah satu kondisi berikut terpenuhi:
- Target mati atau hilang, *health point* lebih dari sama dengan 50%, ada musuh dalam jarak 30 meter.
- Target mati atau hilang, *health point* kurang dari 20% dan ada musuh dalam jarak lebih dari 5 meter dan kurang dari 30 meter.

A4. Target mati atau hilang, *health point* lebih dari 20% dan kurang dari 50%, dan ada teman tipe *warrior* atau *artillery*.
A5. Jarak target lebih dari 5 meter dan kurang dari 30 meter
A6. Salah satu kondisi berikut terpenuhi:
- *Health point* lebih dari 20% dan kurang dari 50%, dan tidak ada teman tipe *warrior* atau *artillery*.
- Target mati atau hilang, *health point* kurang dari 20%.

Semua angka presentase *health point* dan jarak yang berbeda-beda didapat dari hasil percobaan. Dari beberapa kemungkinan angka yang telah dicoba telah disimpulkan bahwa angka – angka yang didapat adalah angka yang terbaik untuk menghasilkan AI pasukan *ranger* yang kuat.

3.3.6 Penerapan AI Pasukan *Artillery*

Artillery adalah tipe pasukan yang paling kuat namun paling lambat dan bisa menyerang beberapa pasukan sekaligus saat melawan musuh pada pasukan tipe *artillery* terdapat 6 *state* yaitu *idle*, *move*, *attack*, *chase*, *retreat*, dan *follow*.

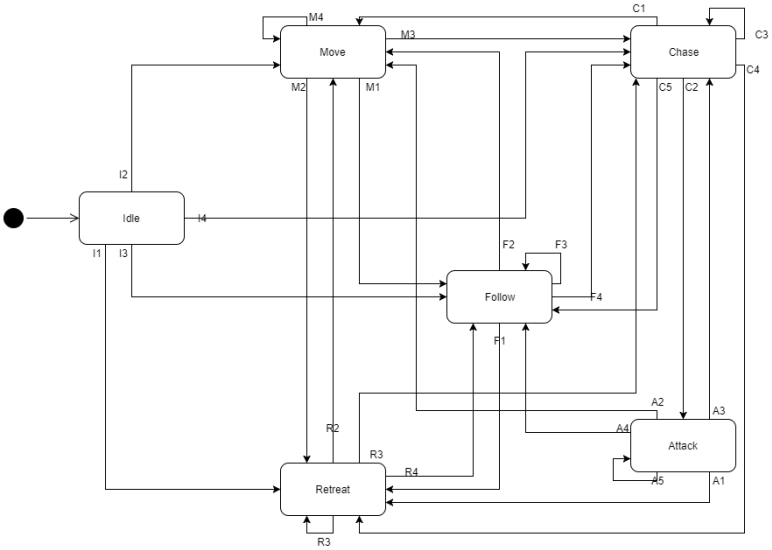
Rule-base yang digunakan pada pasukan tipe *artillery* adalah sebagai berikut:

Saat tidak bertarung:
- Jika *health point* pemain lebih dari 20%:
  - Jika tak ada musuh dalam jarak 10 meter, *move*.

    - Jika ada musuh dalam jarak 10 meter, target terdekat, *chase*.
- Jika *health point* pemain lebih dari 10% dan kurang dari sama dengan 20%:
    - Jika ada musuh dalam jarak 10 meter, target terlemah, *chase*.
    - Jika tidak, *move*.
- Jika *health point* pemain kurang dari sama dengan 10%:
    - Jika dalam jarak 10 meter ada 5 atau lebih teman, *follow*.
    - Jika tidak, *retreat*.

Saat bertarung:

- Jika *health point* pemain lebih dari 20%, *attack*.
- Jika *health point* pemain lebih dari 10% dan kurang dari sama dengan 20%:
    - jika musuh tipe *minion* atau *warrior* atau *ranger*, *attack*.
    - jika musuh tipe *artillery*:
        - Jika *health point* musuh lebih kecil dari *health point* pemain, *attack*.
        - Jika tidak, ganti target menjadi tipe *minion* atau *warrior* atau *ranger* yang terlemah, *chase*.
- Jika *health point* pemain kurang dari sama dengan 10%:
    - Jika dalam jarak 10 meter ada 5 atau lebih teman, target musuh terlemah, *chase*.
    - Jika tidak, *retreat*.



Gambar 3.5 FSM pada Pasukan *Artillery*

FSM untuk pasukan tipe *artillery* dapat dilihat pada Gambar 3.5. Penjelasan untuk kondisi perpidahan *state* pada FSM pada pasukan tipe *artillery* dapat dilihat sebagai berikut:

I1. *Health point* kurang dari 10%, tidak ada 5 atau lebih teman dalam jarak 10 meter.
I2. Salah satu kondisi berikut terpenuhi:
- *Health point* lebih dari 20%, tidak ada musuh dalam jarak 10 meter.
- *Health point* lebih dari 10% dan kurang 20%, tidak ada musuh dalam jarak 10 meter.

I3. *Health point* kurang dari 10%, dalam jarak 10 meter ada 5 atau lebih teman.
I4. Salah satu kondisi berikut terpenuhi:
- *Health point* lebih dari 20%, ada musuh dalam jarak 20%.

- *Health point* lebih dari 10% dan kurang 20%, ada musuh dalam jarak 10 meter.

M1. *Health point* kurang dari 10%, dalam jarak 10 meter ada 5 atau lebih teman.

M2. *Health point* kurang dari 10%, tidak ada 5 atau lebih teman dalam jarak 10 meter.

M3. Salah satu kondisi berikut terpenuhi:

- *Health point* lebih dari 20%, ada musuh dalam jarak 20 meter.
- *Health point* lebih dari 10% dan kurang 20%, ada musuh dalam jarak 10 meter.

M4. Salah satu kondisi berikut terpenuhi:

- *Health point* lebih dari 20%, tidak ada musuh dalam jarak 10 meter.
- *Health point* lebih dari 10% dan kurang 20%, tidak ada musuh dalam jarak 10 meter.

F1. *Health point* kurang dari 10%, tidak ada 5 atau lebih teman dalam jarak 10 meter.

F2. Salah satu kondisi berikut terpenuhi:

- *Health point* lebih dari 20%, tidak ada musuh dalam jarak 10 meter.
- *Health point* lebih dari 10% dan kurang 20%, tidak ada musuh dalam jarak 10 meter.

F3. *Health point* kurang dari 10%, dalam jarak 10 meter ada 5 atau lebih teman.

F4. Salah satu kondisi berikut terpenuhi:

- *Health point* lebih dari 20%, ada musuh dalam jarak 20 meter.
- *Health point* lebih dari 10% dan kurang 20%, ada musuh dalam jarak 10 meter.

R1. *Health point* kurang dari 10%, dalam jarak 10 meter ada 5 atau lebih teman.

R2. Salah satu kondisi berikut terpenuhi:

- *Health point* lebih dari 20%, tidak ada musuh dalam jarak 10 meter.
- *Health point* lebih dari 10% dan kurang 20%, tidak ada musuh dalam jarak 10 meter.

R3. *Health point* kurang dari 10%, tidak ada 5 atau lebih teman dalam jarak 10 meter.

R4. Salah satu kondisi berikut terpenuhi:

- *Health point* lebih dari 20%, ada musuh dalam jarak 20%.
- *Health point* lebih dari 10% dan kurang 20%, ada musuh dalam jarak 10 meter.

C1. Salah satu kondisi berikut terpenuhi:

- *Health point* lebih dari 20%, tidak ada musuh dalam jarak 10 meter.
- *Health point* lebih dari 10% dan kurang 20%, tidak ada musuh dalam jarak 10 meter.

C2. Jarak musuh kurang dari sama dengan 5 meter.

C3. Salah satu kondisi berikut terpenuhi:

- *Health point* lebih dari 20%, ada musuh dalam jarak 20%.
- *Health point* lebih dari 10% dan kurang 20%, ada musuh dalam jarak 10 meter.

C4. *Health point* kurang dari 10%, tidak ada 5 atau lebih teman dalam jarak 10 meter.

C5. *Health point* kurang dari 10%, dalam jarak 10 meter ada 5 atau lebih teman.

A1. *Health point* kurang dari 10%, tidak ada 5 atau lebih teman dalam jarak 10 meter.

A2. Salah satu kondisi berikut terpenuhi:

- Target mati atau hilang, *health point* lebih dari 20%, tidak ada musuh dalam jarak 10 meter.
- Target mati atau hilang, *health point* lebih dari 10% dan kurang 20%, tidak ada musuh dalam jarak 10 meter.

A3. Salah satu kondisi berikut terpenuhi:
- *Health point* kurang dari 20% dan lebih dari 20%, musuh tipe *artillery* dengan *health point* lebih dari pemain, ganti target ke tipe *minion* atau *ranger* atau *warrior* terlemah.
- *Health point* kurang dari 10%, ada 5 atau lebih teman dalam jarak 10 meter.
- Target mati atau hilang, *health point* lebih dari 20%, ada musuh dalam jarak 20 meter.
- Target mati atau hilang, *health point* lebih dari 10% dan kurang 20%, ada musuh dalam jarak 10 meter.

A4. Target mati atau hilang, *health point* kurang dari 10%, dalam jarak 10 meter ada 5 atau lebih teman.

A5. Salah satu kondisi berikut terpenuhi:
- *Health point* lebih dari 10%
- *Health point* kurang dari 20% dan lebih dari 10%, musuh tipe *warrior*, *ranger*, atau *minion*.
- *Health point* kurang dari 20% dan lebih dari 20%, musuh tipe *artillery* dengan *health point* kurang dari pemain.

Semua angka presentase *health point* dan jarak yang berbeda-beda didapat dari hasil percobaan. Dari beberapa kemungkinan angka yang telah dicoba telah disimpulkan bahwa angka – angka yang didapat adalah angka yang terbaik untuk menghasilkan AI pasukan *artillery* yang kuat.

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 Implementasi AI *Tower*

Implementasi AI pada *tower* untuk *game* dapat dilihat pada Kode 4.1

```
if (currentState == State.IDLE){
        if (CountEnemyInRange(40.0f) >=1){
                currentTarget = GetClosestArmyInRange(40.0f);
                currentState = State.ATTACK;
        }
}
else if (currentState == State.ATTACK){
        currentTarget = GetClosestArmyInRange(40.0f);
        if (currentTarget == null)
                currentState = State.IDLE;
        else
                Attack(currentTarget);
}
```

Kode 4.1 AI pada *Tower*

4.2 Implementasi AI Pasukan

Implementasi AI dibagi menjadi implementasi AI pada pasukan tipe *minion*, *warrior*, *ranger*, dan *artillery*, namun ada bagian-bagian yang sama dari semua tipe pasukan, yaitu untuk mengendalikan gerakan pasukan pada saat *state idle*, *chase*, *follow*, *retreat*, dan *move*, bagian ini diletakan dalam fungsi *RunBasicIntelligence().*

Setiap tipe pasukan memiliki algoritma menjalankan keputusan yang berbeda-beda yang dipanggil dalam fungsi *RunIntelligence*(), serta keadaan menyerang yang berbeda-beda juga yang dijalankan dalam fungsi *Update*() pada pasukan tersebut. Implementasi untuk AI dasar pasukan dapat dilihat pada Kode 4.2

```
protected void RunBasicIntelligence() {
	if (currentState==State.CHASE) {
		if (currentTarget!=null){
			UpdateTargetPosition(currentTarget, (curCommand=="CHASE"));
			if (IsTouching(currentTarget, 1))
				Battle(currentTarget);
		}
		else if (curCommand=="")
				RunIntelligence();
	}
	else if (currentState==State.RETREAT) {
		if (CheckDistance(baseTower.transform.position, 2)<=10)
			agent.Stop();
	}
	else if (currentState == State.FOLLOW){
		if (currentTarget!=null){
			UpdateTargetPosition(currentTarget, (curCommand == "FOLLOW"));
			if (CheckDistance(currentTarget.transform.position, 1) < 3.0f)
				agent.Stop();
			else
				agent.Resume();
		}
		else if (curCommand=="")
			RunIntelligence();
	}
	else if (currentState == State.MOVE) {
		if (IsTouching(enemyTower, 2)) {
			agent.Stop();
			Battle(enemyTower);
		}
		else if (CheckDistance(new Vector3(destX, 1.0f, destZ), 1) <3.0f) {
			if (curCommand == "MOVE")
				MakeIdle(true);
			else
				RunIntelligence();
		}
	}
}
```

Kode 4.2 Fungsi *RunBasicIntelligence*() pada Pasukan

4.2.1 Implementasi AI Pasukan *Minion*

Implementasi AI pada pasukan *minion* dibagi menjadi saat algoritma saat sedang bertarung yang diletakkan didalam fungsi *Update*() pada Unity dan algoritma pengambilan keputusan saat sedang tidak bertarung yang dijalankan dengan fungsi *RunIntelligence*().

Implementasi untuk AI untuk algoritma pasukan *minion* saat sedang bertarung dapat dilihat pada Kode 4.3 dan algoritma pengambilan keputusan dapat dilihat pada Kode 4.4.

```
if (currentState==State.ATTACK) {
	if (currentTarget==null || !IsTouching(currentTarget))
		RunIntelligence();
	else if (GetCurPercentHP() > 25)
		Attack(currentTarget, (curCommand=="ATTACK"));
	Else {
		if (currentTarget.CompareTag("Minion")) {
			if (CountArmyInRange(new int[1]{1}, 20, GetSide(), gameObject) >=5)
				Attack(currentTarget, (curCommand=="ATTACK"));
			else if (currentTargetScript.GetCurHP() < GetCurHP())
				Attack(currentTarget, (curCommand=="ATTACK"));
			else
				Retreat();
		}
		else if (currentTarget.CompareTag("Armored")) {
			if (CountArmyInRange(new int[1]{1}, 20, GetSide(), gameObject) >= 5)
				Attack(currentTarget, (curCommand=="ATTACK"));
			else if (currentTargetScript.GetCurHP() < GetCurHP())
				Attack(currentTarget, (curCommand=="ATTACK"));
			else
				Retreat();
		}
		else if (currentTarget.CompareTag("Ranger")) {
			if (currentTargetScript.GetCurPercentHP() < 50)
				Attack(currentTarget, (curCommand=="ATTACK"));
			else
				Retreat();
		}
		else if (currentTarget.CompareTag("Artillery")) {
			if (currentTargetScript.GetCurHP() < GetCurHP())
				Attack(currentTarget, (curCommand=="ATTACK"));
			else
				Retreat();
		}
	}
}
```

Kode 4.3 Algoritma Saat Bertarung untuk Pasukan *Minion*

```
public override void RunIntelligence() {
        float ranProb = ms.GetRandomPercentage(difficulty);
        int ranNum = UnityEngine.Random.Range(1, 101);
        if (ranNum <= ranProb)
                DoSomethingRandom(20);
        else if (GetCurPercentHP() > 25) {
                if (CountArmyInRange(new int[4]{1,2,3,4}, 20, (1-GetSide()), gameObject) == 0)
                        MoveToTower(enemyTower, false);
                Else {
                        ChangeTarget(GetWeakestEnemyInRange(new int[4]{1,2,3,4}, 20));
                        Chase(currentTarget, false);
                }
        }
        else if (GetCurPercentHP() <= 25) {
                if (CountArmyInRange(new int[4]{1,2,3,4}, 20, GetSide(), gameObject) >= 5) {
                        if (CountArmyInRange(new int[4]{1,2,3,4}, 15, (1-GetSide()), gameObject) >= 1) {
                                ChangeTarget(GetWeakestEnemyInRange(new int[4]{1,2,3,4}, 15));
                                Chase(currentTarget, false);
                        }
                        else
                                MakeIdle(false);
                }
        }
}
```

Kode 4.4 Fungsi *RunIntelligence*() pada Pasukan *Minion*

4.2.2 Implementasi AI Pasukan *Warrior*

Implementasi AI pada pasukan *warrior* dibagi menjadi saat algoritma saat sedang bertarung yang diletakkan didalam fungsi *Update*() pada Unity dan algoritma pengambilan keputusan saat sedang tidak bertarung yang dijalankan dengan fungsi *RunIntelligence*().

Implementasi untuk AI untuk algoritma pasukan *minion* saat sedang bertarung dapat dilihat pada Kode 4.5 dan algoritma pengambilan keputusan dapat dilihat pada Kode 4.6.

```
if (currentState ==State.ATTACK){
	if (currentTarget==null || !IsTouching(currentTarget))
		RunIntelligence();
	else if (GetCurPercentHP() > 20)
		Attack(currentTarget, (curCommand=="ATTACK"));
	Else {
		if (currentTarget.CompareTag("Minion"))
			Attack(currentTarget, (curCommand=="ATTACK"));
		else if(currentTarget.CompareTag("Armored")|| currentTarget.CompareTag("Ranger")){
			if (CountArmyInRange(new int[2]{2,3}, 15, GetSide(), gameObject) >= 5)
				Attack(currentTarget, (curCommand=="ATTACK"));
			else if (CountArmyInRange(new int[1]{1}, 15, GetSide(), gameObject) >= 5){
				ChangeTarget(GetWeakestEnemyInRange(new int[4]{1,2,3,4}, 15));
				Chase(currentTarget, false);
			}
			else if (currentTargetScript.GetCurHP() < GetCurHP())
				Attack(currentTarget, (curCommand=="ATTACK"));
			else
				Retreat();
		}
		else if (currentTarget.CompareTag("Artillery")){
			if (currentTargetScript.GetCurHP() < GetCurHP())
				Attack(currentTarget, (curCommand=="ATTACK"));
			else
				Retreat();
		}
	}
}
```

Kode 4.5 Algoritma Saat Bertarung untuk Pasukan *Warrior*

```
public override void RunIntelligence() {
	float ranProb = ms.GetRandomPercentage(difficulty);
	int ranNum = UnityEngine.Random.Range(1, 101);
	if (ranNum <= ranProb)
		DoSomethingRandom(20);
	else if (GetCurPercentHP() > 20) {
		if (CountArmyInRange(new int[4]{1,2,3,4}, 20, (1-GetSide()), gameObject) == 0)
			MoveToTower(enemyTower, false);
		else{
			ChangeTarget(GetClosestArmyInRange(new int[4]{1,2,3,4}, 20, (1-GetSide()),
gameObject));
			Chase(currentTarget, false);
		}
	}
	Else {
		if (CountArmyInRange(new int[2]{2,3}, 20, GetSide(), gameObject) >= 5) {
			if (CountArmyInRange(new int[4]{1,2,3,4}, 20, (1-GetSide()), gameObject) >= 1) {
				ChangeTarget(GetClosestArmyInRange(new int[4]{1,2,3,4}, 20, (1-
GetSide()), gameObject));
				Chase(currentTarget, false);
			}
			else
				MoveToTower(enemyTower, false);
		}
		else if (CountArmyInRange(new int[1]{1}, 20, GetSide(), gameObject) >= 5) {
			if (CountArmyInRange(new int[4]{1,2,3,4}, 20, (1-GetSide()), gameObject) >= 1) {
				ChangeTarget(GetWeakestEnemyInRange(new int[4]{1,2,3,4}, 20));
				Chase(currentTarget, false);
			}
			else
				MoveToTower(enemyTower, false);
		}
		else if (CountArmyInRange(new int[2]{1,3}, 20, (1-GetSide()), gameObject) >= 1) {
			ChangeTarget(GetWeakestEnemyInRange(new int[2]{1,3}, 20));
			Chase(currentTarget, false);
		}
		else
			MakeIdle(false);
	}
}
```

Kode 4.6 Fungsi *RunIntelligence*() pada Pasukan *Warrior*

4.2.3 Implementasi AI Pasukan *Ranger*

Implementasi AI pada pasukan *ranger* dibagi menjadi saat algoritma saat sedang bertarung yang diletakkan didalam fungsi *Update*() pada Unity dan algoritma pengambilan keputusan saat sedang tidak bertarung yang dijalankan dengan fungsi *RunIntelligence*().

Implementasi untuk AI untuk algoritma pasukan *ranger* saat sedang bertarung dapat dilihat pada Kode 4.7 dan algoritma pengambilan keputusan dapat dilihat pada Kode 4.8.

```
if (currentState == State.ATTACK) {
        if (currentTarget == null || !IsTouching(currentTarget, 1))
                RunIntelligence();
        else if (CheckDistance(currentTarget.transform.position, 1) < 5.0f)
                Retreat();
        else if (CheckDistance(currentTarget.transform.position, 1) >= 5.0f &&
CheckDistance(currentTarget.transform.position, 1) <= 30.0f)
                Attack(currentTarget, (curCommand=="ATTACK"));
        else if (CheckDistance(currentTarget.transform.position, 1) > 30.0f)
                Chase(currentTarget, false);
}
```

Kode 4.7 Algoritma Saat Bertarung untuk Pasukan *Ranger*

```
public override void RunIntelligence() {
        float ranProb = ms.GetRandomPercentage(difficulty);
        int ranNum = UnityEngine.Random.Range(1, 101);
        if (ranNum <= ranProb)
                DoSomethingRandom(30);
        else if (GetCurPercentHP() > 20) {
                if (CountArmyInRange(new int[4]{1,2,3,4}, 30, (1-GetSide()), gameObject) == 0) {
                        if (GetCurPercentHP() < 50) {
                                if (CountArmyInRange(new int[2]{2,4}, 30, GetSide(),gameObject) >= 1) {
                                        ChangeTarget(GetClosestArmyInRange(new int[2]{2,4}, 30,
GetSide(), gameObject));
                                        Follow(currentTarget, false);
                                }
                                else
                                        MakeIdle(false);
                        }
                        else
                                MoveToTower(enemyTower, false);
                }
                Else {
                        ChangeTarget(GetWeakestEnemyInRange(new int[4]{1,2,3,4}, 30));
                        Chase(currentTarget, false);
                }
        }
        else if (GetCurPercentHP() <= 20) {
                if (CountArmyInRange(new int[4]{1,2,3,4}, 5, (1-GetSide()), gameObject) >= 1)
                        Retreat();
                else if (CountArmyInRange(new int[4]{1,2,3,4}, 30, (1-GetSide()), gameObject) >= 1) {
                        ChangeTarget(GetWeakestEnemyInRange(new int[4]{1,2,3,4},30));
                        Chase(currentTarget, false);
                }
                else
                        MakeIdle(false);
        }
}
```

Kode 4.8 Fungsi *RunIntelligence*() pada Pasukan *Ranger*

4.2.4 Implementasi AI Pasukan *Artillery*

Implementasi AI pada pasukan *artillery* dibagi menjadi saat algoritma saat sedang bertarung yang diletakkan didalam fungsi Update() pada Unity dan algoritma pengambilan keputusan saat sedang tidak bertarung yang dijalankan dengan fungsi *RunIntelligence*().

Implementasi untuk AI untuk algoritma pasukan *artillery* saat sedang bertarung dapat dilihat pada Kode 4.9 dan algoritma pengambilan keputusan dapat dilihat pada Kode 4.10.Kode 4.8

```
if (currentState == State.ATTACK) {
	if (currentTarget ==null || !IsTouching(currentTarget, 1))
		RunIntelligence();
	else if (GetCurPercentHP() >20)
		Attack(currentTarget, (curCommand=="ATTACK"));
	else if (GetCurPercentHP() >10 && GetCurPercentHP()<=20) {
		if (currentTarget.CompareTag("Artillery"))  {
			if (currentTargetScript.GetCurHP() < GetCurHP())
				Attack(currentTarget, (curCommand=="ATTACK"));
			else {
				if (CountArmyInRange(new int[]{1,2,3}, attackDistance, (1-GetSide()),
gameObject) >=1) {
					ChangeTarget(GetWeakestEnemyInRange(new int[3]{1, 2,3},
attackDistance));
					Chase(currentTarget, false);
				}
				else
					Retreat();
			}
		}
	}
	else if (GetCurPercentHP() < 10) {
		if (CountArmyInRange(new int[4]{1,2,3,4}, 10, (1-GetSide()), gameObject) >=5) {
			ChangeTarget(GetWeakestEnemyInRange(new int[3]{1,2,3}, attackDistance));
			Attack(currentTarget, (curCommand=="ATTACK"));
		}
		else
			Retreat();
	}
}
```

Kode 4.9 Algoritma Saat Bertarung untuk Pasukan *Artillery*

```
public override void RunIntelligence() {
	float ranProb = ms.GetRandomPercentage(difficulty);
	int ranNum = UnityEngine.Random.Range(1, 101);
	if (ranNum <= ranProb)
		DoSomethingRandom(10);
	else if (GetCurPercentHP() >20) {
		if (CountArmyInRange(new int[4]{1,2,3,4}, 10, (1-GetSide()), gameObject) >= 1) {
			ChangeTarget (GetClosestArmyInRange(new int[4]{1,2,3,4}, 10, (1-GetSide()),
gameObject));
			Chase(currentTarget, false);
		}
		else
			MoveToTower(enemyTower, false);
	}
	else if (GetCurPercentHP() >10 && GetCurPercentHP() <=20) {
		if (CountArmyInRange(new int[4]{1,2,3,4}, 10, (1-GetSide()), gameObject) >= 1) {
			ChangeTarget (GetWeakestEnemyInRange(new int[4]{1,2,3,4}, 10));
			Chase(currentTarget, false);
		}
		else
			MoveToTower(enemyTower, false);
	}
	else if (GetCurPercentHP() <= 10) {
		if (CountArmyInRange(new int[4]{1,2,3,4}, 10, (1-GetSide()), gameObject) >= 5)
			MoveToTower(enemyTower, false);
		else
			Retreat();
	}
}
```

Kode 4.10 Fungsi *RunIntelligence*() pada Pasukan *Artillery*

4.3 Pembuatan Keputusan *Random*

Implementasi untuk membuat keputusan secara *random* pada AI level *easy* dan *medium* dilakukan dengan memanggil fungsi *DoSomethingRandom*(*int distance*, *int diff*). Terdapat 2 parameter pada fungsi tersebut yaitu *distance* dan *diff*, *distance* adalah jarak aman dari pasukan yang bersangkutan tersebut, sedangkan untuk *diff* adalah tingkat kesulitan dari pasukan tersebut, untuk implemenatsi dari fungsi *DoSomethingRandom*() dapat dilihat pada Kode 4.11 dan Kode 4.12

```
int ranNum = UnityEngine.Random.Range(1, 101);
if (ranNum >0 && ranNum<=20 ) {
        if (diff == 2 && CountArmyInRange(new int[4]{1,2,3,4}, distance, 1-GetSide(),
gameObject)>=1) {
                GameObject randomEnemy = GetRandomUnit(distance, 1-GetSide(),
gameObject);
                ChangeTarget(randomEnemy);
                Chase(randomEnemy, false);
        }
        Else {
                int ranNum2 = UnityEngine.Random.Range(1, 91);
                if (ranNum2 >0 && ranNum2<=30 )
                        Retreat();
                else if (ranNum2 >30 && ranNum2<=60 )
                        MakeIdle(false);
                else if (ranNum2 >60 && ranNum2<=90 ) {
                        Vector3 randomPlace = GetRandomPlace(distance);
                        Move(randomPlace.x, randomPlace.z, false);
                }
        }
}
```

Kode 4.11 Fungsi Pembuat Keputusan *Random* Bagian Pertama

```
else if (ranNum >20 && ranNum<=40) {
        if (diff == 2 &&  CountArmyInRange(new int[4]{1,2,3,4}, distance, GetSide(), gameObject)>=1) {
                GameObject randomFriend = GetRandomUnit(distance, GetSide(), gameObject);
                ChangeTarget(randomFriend);
                Follow(randomFriend, false);
        }
        Else {
                int ranNum2 = UnityEngine.Random.Range(1, 91);
                if (ranNum2 >0 && ranNum2<=30 )
                        Retreat();
                else if (ranNum2 >30 && ranNum2<=60 )
                        MakeIdle(false);
                else if (ranNum2 >60 && ranNum2<=90 ) {
                        Vector3 randomPlace = GetRandomPlace(distance);
                        Move(randomPlace.x, randomPlace.z, false);
                }
        }
}
else if (ranNum >40 && ranNum<=60 )
        Retreat();
else if (ranNum >60 && ranNum<=80 )
        MakeIdle(false);
else if (ranNum >80 && ranNum<=100 ) {
        Vector3 randomPlace = GetRandomPlace(distance);
        Move(randomPlace.x, randomPlace.z, false);
}
```

Kode 4.12 Fungsi Pembuat Keputusan *Random* Bagian Kedua

4.4 *Upgrade* dan Pasukan *Elite*

Implementasi AI untuk menentukan kapan akan melakukan *upgrade* dan men-*deploy* pasukan *elite* dapat dilihat pada Kode 4.13 untuk AI *easy*, Kode 4.14 untuk AI *medium*, dan Kode 4.15 untuk AI *hard*

```
if(this.difficulty == 1) {
        int randomProb = UnityEngine.Random.Range(0, 101);
        if (randomProb<=30)
                return 0;
        else {
                int angka1 = UnityEngine.Random.Range(0, 2);
                int angka2 = UnityEngine.Random.Range(1, 5);
                return ((angka1*10) + angka2);
        }
}
```

Kode 4.13 Fungsi Pembuat Keputusan *Upgrade* dan Pasukan *Elite* untuk AI *Easy*

```
else if (this.difficulty == 2) {
        int randomProb = UnityEngine.Random.Range(0, 101);
        if (randomProb<70) {
                int tipe;
                for (tipe=1;tipe<=4;tipe++) {
                        if (armyLevel[tipe-1] < 3) {
                                int tot = armyLevel[tipe-1];
                                foreach (int isi in taskQueue.ToList()) {
                                   if (isi % 10 == tipe && (isi/10) == armyLevel[tipe-1])
                                        tot++;
                                }
                                if (tot<3) return (10 + tipe);
                        }
                }
                int randomNum = UnityEngine.Random.Range(1, 5);
                return randomNum;
        }
        Else {
                int angka1 = UnityEngine.Random.Range(0, 2);
                int angka2 = UnityEngine.Random.Range(1, 5);
                return ((angka1*10) + angka2);
        }
}
```

Kode 4.14 Fungsi Pembuat Keputusan *Upgrade* dan Pasukan *Elite* untuk AI *Medium*

```
else if (this.difficulty == 3) {
        int tipe;
        for (tipe=1;tipe<=4;tipe++) {
                if (armyLevel[tipe-1] < 3) {
                        int tot = armyLevel[tipe-1];
                        foreach (int isi in taskQueue.ToList()) {
                           if (isi % 10 == tipe && (isi/10) == armyLevel[tipe-1])
                                tot++;
                        }
                        if (tot<3) return (10 + tipe);
                }
        }
        int randomNum = UnityEngine.Random.Range(1, 5);
        return randomNum;
}
```

Kode 4.15 Fungsi Pembuat Keputusan *Upgrade* dan Pasukan *Elite* untuk AI *Hard*

[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 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 Spesifikasi Lingkungan Pengujian

Spesifikasi	**Deskripsi**
Jenis Perangkat	*Laptop*
Merek Perangkat	Lenovo E531
Sistem Operasi	Windows 10
Procesor	Core i7
RAM	8 GB

5.2 Skenario Uji Coba Peraturan *Game*

Pengujian ini adalah pengujian terhadap peraturan permainan yang telah diterapkan pada *game*, dengan menguji apakah peraturan yang ada telah berhasil diterapkan dengan baik pada *game*.

Pengujian pada peraturan permainan dapat dilihat sebagai berikut:

1. Terdapat 2 belah pihak yang saling menyerang satu sama lain.

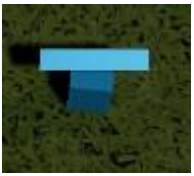
   Pada Gambar 5.1 terlihat kedua belah pihak yang sedang saling menyerang, pihak pemain digambarkan dengan pasukan berwarna biru dan pihak musuh digambarkan dengan pasukan berwarna merah.

Gambar 5.1 Kedua Belah Pihak Saling Menyerang

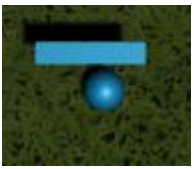
2. Terdapat 4 jenis pasukan berbeda pada setiap pihak yang memiliki kemampuannya sendiri-sendiri.

   Gambar dari keempat jenis pasukan yang ada pada *game* dapat dilihat pada Gambar 5.2 untuk pasukan *minion*, Gambar 5.3 untuk pasukan *warrior*, Gambar 5.4 untuk pasukan *ranger*, dan Gambar 5.5 untuk pasukan *artillery*.

Gambar 5.2 Pasukan *Minion*



Gambar 5.3 Pasukan *Warrior*



Gambar 5.4 Pasukan *Ranger*

Gambar 5.5 Pasukan *Artillery*

3. Terdapat *tower* pada kedua belah pihak yang apabila hancur maka pihak tersebut akan kalah.

   Gambar dari tower dapat dilihat pada Gambar 5.6 untuk tower dari pihak pemain yang ditandai oleh bendera dan *health bar* berwarna biru di puncak *tower*. Sedangkan untuk gambar dari *tower* pihak musuh dapat dilihat

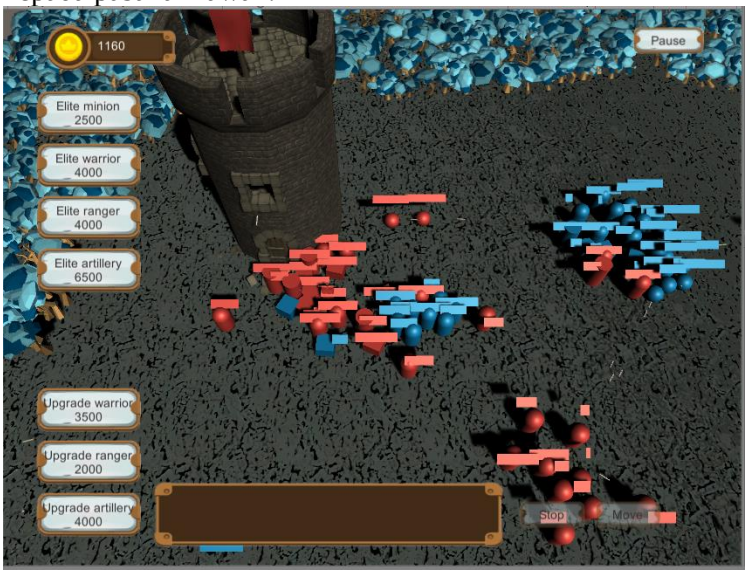
pada Gambar 5.7 yang ditandai oleh bendera dan *health bar* berwarna merah di puncak *tower*.

Gambar 5.6 *Tower* Pemain

Gambar 5.7 *Tower* Musuh

4. *Tower* dapat menyerang pasukan musuh yang mendekat.

   Pada Gambar 5.8 dapat terlihat *tower* sedang menyerang pasukan lawan dengan menembakkan panah kepada pasukan lawan.



Gambar 5.8 *Tower* Menyerang Pasukan Musuh

5. Apabila membunuh pasukan milik musuh maka akan mendapat sejumlah uang.

   Uang yang dimiliki oleh pemain dapat dilihat pada pojok kiri atas layar permainan seperti terlihat pada Gambar 5.9. Untuk uang yang dimiliki oleh musuh tidak dapat terlihat di layar permainan karena dikendalikan oleh AI musuh.

Gambar 5.9 Tampilan Uang

6. Kedua pihak memiliki pilihan untuk meng-*upgrade* pasukan yang dimilikinya agar lebih kuat hingga *level* 3, dengan setiap *upgrade* membutuhkan biaya sejumlah uang.

   Tombol untuk kontrol pemain meng-*upgrade* pasukannya dapat dilihat pada Gambar 5.10, dimana pada pada tampilan *game* terdapat 4 tombol yang berguna untuk meng-*upgrade* pasukannya. Pada tombol tertulis tipe pasukan yang akan di-*upgrade* dan biaya yang dibutuhkan untuk melakukan *upgrade* tersebut. Apabila tipe pasukan tersebut telah di-*upgrade* hingga level maksimal, tombol untuk *upgrade* tersebut akan hilang dari layar. Setelah melakukan *upgrade*, notifikasi akan muncul pada *panel* yang ada di bagian bawah layar permainan seperti ditunjukkan pada Gambar 5.11.
   Untuk *upgrade* pada pihak musuh tidak dapat dilihat pada tampilan layar pemain karena dikendalikan oleh pihak AI musuh.

Gambar 5.10 Tombol untuk *Upgrade* Pasukan

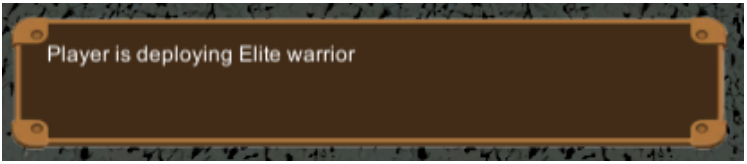
Gambar 5.11 Notifikasi *Upgrade*

7. Kedua belah pihak memiliki pilihan untuk membuat pasukan *elite* yaitu pasukan yang jauh lebih kuat dari pasukan biasa dengan membayar sejumlah uang.

   Pada layar dapat terlihat 4 tombol yang dapat digunakan untuk men-*deploy* pasukan *elite* beserta biaya yang dibutuhkan untuk men-*deploy* pasukan *elite* tersebut. Gambar tombol tersebut dapat dilihat pada Gambar 5.12 dan setelah melakukan *deploy* pasukan *elite* akan muncul notifikasi pada panel yang ada di bagian bawah layar permainan seperti ditunjukkan pada Gambar 5.13.

Untuk *deploy* pasukan *elite* pada pihak musuh tidak dapat dilihat pada tampilan layar pemain karena dikendalikan oleh pihak AI musuh.

Gambar 5.12 Tombol untuk *Deploy* Pasukan *Elite*



Gambar 5.13 Notifikasi Pasukan *Elite*

8. Terdapat 3 tingkat kesulitan pada *game* yaitu *easy*, *medium*, dan *hard*, pemain dapat memilih akan melawan AI dengan tingkat kesulitan mana dan akan menggunakan tingkat kesulitan mana untuk AI pada pihak pemain itu sendiri.

Pada menu untuk memilih *map* dan tingkat kesulitan permainan, dapat dilihat terdapat pilihan untuk 3 tingkat kesulitan yaitu *easy*, *medium*, dan *hard* pada kedua belah pihak yang ada pada *game* seperti terlihat pada Gambar 5.14.

Gambar 5.14 Tombol untuk Memilih Tingkat Kesulitan

5.3 Skenario Uji Coba FSM

Pengujian ini adalah pengujian terhadap *rule-based* FSM yang telah dibuat dengan menguji kesesuaian *rule-based* FSM dengan jalan pasukan di permainan.

Untuk menguji kesesuaian dengan FSM, penulis mengkatifkan sebuah tulisan di atas pasukan yang akan menampilkan informasi yang diperlukan untuk menguji kondisi yang ada, dan juga untuk pengujian akan digunakan AI level *hard* agar tidak terjadi tindakan *random* yang dapat mengganggu hasil percobaan.

5.3.1 Skenario Uji Coba FSM pada Pasukan *Minion*

Pengujian ini dilakukan dengan cara menguji jalannya pasukan tipe *minion* pada permainan dan menguji kesesuaian dengan *rule-base* FSM yang telah ditetapkan, pada pengujian ini tidak semua kondisi akan diuji melainkan hanya beberapa kondisi saja.

Berikut kondisi yang diuji pada pasukan tipe *minion*:

- Saat tidak bertarung, jika *health point* pemain lebih dari 25% dan tidak ada musuh dalam jarak 20 meter, *move*.

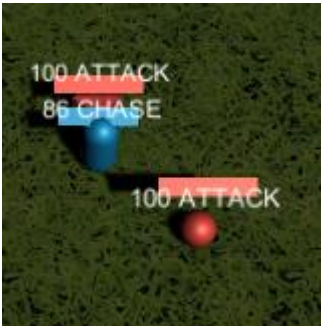
  Pada Gambar 5.15 dapat dilihat seorang pasukan *minion* yang setelah di-*deploy* langsung memasuki *state move* karena *health point* nya lebih dari 25%, yaitu 100%, *health point* pasukan dapat dilihat tertulis di atas pasukan dan di sampingnya tertulis state dari pasukan tersebut yaitu *move*.

Gambar 5.15 Pasukan *Minion* Memasuki *State Move*

- Saat tidak bertarung, jika *health point* lebih dari 25% dan ada musuh tipe *ranger* dalam jarak 20 meter target musuh tipe *ranger* terdepan, *chase.*

  Pada Gambar 5.16 terlihat pasukan *minion* yang akan berhadapan dengan pasukan musuh dengan tipe *warrior* dan *ranger*, pada Gambar 5.17 terlihat state dari pasukan *minion* tersebut berubah menjadi *chase* karena akan mengejar pasukan musuh bertipe *ranger*.

Gambar 5.16 Pasukan *Minion* dalam *State Move* yang akan Bertemu 2 Musuh yaitu *Warrior* dan *Ranger*

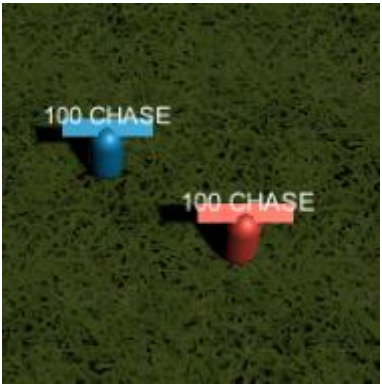


Gambar 5.17 Pasukan *Minion* dalam *State Chase* Sedang Mengejar Musuh Tipe *Ranger*

- Saat tidak bertarung, jika *health point* lebih dari 25% dan tidak ada musuh tipe *ranger* dalam jarak 20 meter, target musuh terlemah, *chase.*

Pada Gambar 5.18 terlihat 2 pasukan *minion* yang semula berada pada *state move*, pada gambar berikutnya yaitu Gambar 5.19 *state* nya berubah menjadi *chase* Karena berada dalam jarak 20 meter

Gambar 5.18 Pasukan *Minion* dalam *State Move* yang akan Bertemu Musuh



Gambar 5.19 Pasukan *Minion* dalam *State Chase*

- Saat bertarung, Jika *health* point kurang dari sama dengan 25%, tidak dan teman dalam jarak 20 meter dan *health point* musuh lebih besar dari *health point* pemain, *retreat*.

  Pada Gambar 5.20 terlihat pasukan *minion* berwarna merah berpindah menjadi *state retreat* setelah melawan pasukan *minion* biru karena *health point* nya yang telah jatuh menjadi 25%.

Gambar 5.20 Pasukan *Minion* dalam S*tate Retreat*

5.3.2 Skenario Uji Coba FSM pada Pasukan *Warrior*

Pengujian ini dilakukan dengan cara menguji jalannya pasukan tipe *warrior* pada permainan dan menguji kesesuaian dengan *rule-base* FSM yang telah ditetapkan, pada pengujian ini tidak semua kondisi akan diuji melainkan hanya beberapa kondisi saja.

Berikut kondisi yang diuji dari pasukan tipe *warrior*:

- Saat tidak bertarung, *health point* lebih dari 20%, tidak ada musuh dalam jarak 20 meter, *move.*

  Pada Gambar 5.21 terlihat pasukan *warrior* yang memiliki *health point* 100% berpindah ke *state move* setelah di-*deploy*

Gambar 5.21 Pasukan Warrior dalam *State Move*

- Saat tidak bertarung, Jika *health point* lebih dari 20%, ada musuh dalam jarak 20 meter, *chase.*

  Pada Gambar 5.22 terlihat pasukan *warrior* yang sedang dalam *state move* dengan *health point* 100%, lalu pada Gambar 5.23 terlihat pasukan tersebut berubah *state* menjadi *chase* karena telah mendekati musuh dalam jarak 20 meter.

Gambar 5.22 Pasukan *Warrior* dalam *State Move* yang akan Bertemu Musuh

Gambar 5.23 Pasukan *Warrior* dalam *State Chase*

- Saat bertarung, Jika *health point* pemain lebih dari 20%, *attack.*

Pada Gambar 5.24 terlihat kedua pasukan *warrior* yang sedang bertarung, *state* mereka tetap pada *attack* karena *health point* kedua pasukan masih diatas 20%.

Gambar 5.24 Pasukan *Warrior* dalam *State Attack*

5.3.3 Skenario Uji Coba FSM pada Pasukan *Ranger*

Pengujian ini dilakukan dengan cara menguji jalannya pasukan tipe *ranger* pada permainan dan menguji kesesuaian dengan *rule-base* FSM yang telah ditetapkan, pada pengujian ini tidak semua kondisi akan diuji melainkan hanya beberapa kondisi saja.

Berikut beberapa kondisi yang diuji pada pasukan tipe *ranger*:

- Saat tidak bertarung, Jika *health point* lebih dari sama dengan 50%, tidak ada musuh dalam jarak 30 meter, *move.*

  Pada Gambar 5.25 terlihat pasukan *ranger* memasuki *state move* setelah di-*deploy* karena *health point* nya lebih dari 50% yaitu 100%.

Gambar 5.25 Pasukan *Ranger* dalam *State Move*

- Saat tidak bertarung, jika *health point* lebih dari 50% dan ada musuh dalam jarak 30 meter, *attack.*

  Pada Gambar 5.26 terlihat pasukan *ranger* yang semula berada dalam *state move* dengan *health point* 100% akan bertemu musuh, lalu Gambar 5.27 terlihat *state* nya telah berubah menjadi *attack* karena musuh telah mendekat dalam jarak kurang dari 30 meter.

Gambar 5.26 Pasukan *Ranger* dalam *State Move* yang akan Bertemu Musuh

Gambar 5.27 Pasukan *Ranger* dalam *State Attack* setelah Bertemu Musuh dalam Jarak 30 Meter

- Saat tidak bertarung, jika *health point* kurang dari sama dengan 20% dan ada musuh dalam jarak kurang dari 5 meter, *retreat.*

  Pada Gambar 5.28 terlihat pasukan *ranger* yang telah berpindah *state* menjadi *retreat* karena *health point* nya telah menjadi 20% dan ada musuh yaitu pasukan *warrior* dalam jarak 5 meter.

Gambar 5.28 Pasukan *Ranger* dalam *State Retreat*

5.3.4 Skenario Uji Coba FSM pada Pasukan *Artillery*

Pengujian ini dilakukan dengan cara menguji jalannya pasukan tipe *artillery* pada permainan dan menguji kesesuaian dengan *rule-base* FSM yang telah ditetapkan, pada pengujian ini tidak semua kondisi akan diuji melainkan hanya beberapa kondisi saja.

Berikut beberapa kondisi yang diuji pada pasukan tipe *artillery*:

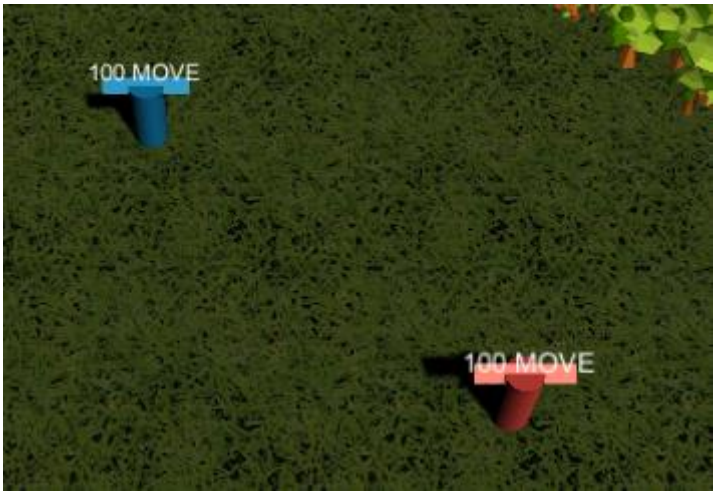
- Saat tidak bertarung, Jika *health point* lebih dari 20%, tidak ada musuh dalam jarak 10 meter, *move.*

  Pada Gambar 5.29 terlihat pasukan *artillery* yang memasuki *state move* karena *health point* nya masih lebih dari 20% yaitu 100%.

Gambar 5.29 Pasukan *Artillery* dalam *State Move*

- Saat tidak bertarung, Jika *health point* lebih dari 20%, ada musuh dalam jarak 10 meter, *chase.*

  Pada Gambar 5.30 terlihat kedua pasukan *artillery* pada *state move* akan berhadapan. Lalu pada Gambar 5.31 terlihat keduanya memasuki *state chase* karena telah bertemu musuh dalam jarak 10 meter dengan *health point* yang masih lebih dari 20%.

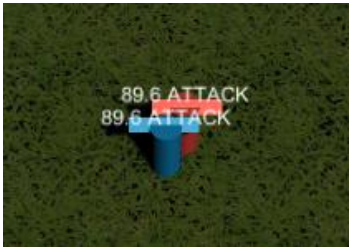


Gambar 5.30 Pasukan *Artillery* pada *State Move* akan Berhadapan dengan Musuh

Gambar 5.31 Pasukan *Artillery* Berpindah ke *State Chase*

- Saat bertarung, Jika *health point* lebih dari 10%, *attack.*

  Pada Gambar 5.32 terlihat kedua pasukan *artillery* yang sedang bertarung tetap pada *state attack* karena *health point* nya masih diatas 10%.



Gambar 5.32 Kedua Pasukan *Artillery* dalam *State Attack*

5.4 Skenario Uji Coba AI Melawan AI

Pengujian ini adalah pengujian dengan mempertandingkan AI melawan AI pada semua kemungkinan *map* dan tingkat kesulitan dengan mencatat hasil kemenangan dan waktu yang dibutuhkan untuk permainan selesai.

5.4.1 Uji Coba *Map* 1

Uji coba dilakukan pada *map* 1 dengan mempertandingkan setiap level AI yaitu *easy* vs *easy*, *easy* vs *medium*, *easy* vs *hard*, *medium* vs *medium*, *hard* vs *medium*, dan *hard* vs *hard*.

Hasil pengujian pada *ma*p 1 pada setiap level AI 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 Uji Coba *Easy* vs *Easy* pada *Map* 1

Percobaan ke	Pemenang	Waktu
1	*Easy 2*	49 menit 58 detik
2	*Easy 2*	58 menit 28 detik
3	*Easy 2*	44 menit 5 detik
4	*Easy 1*	23 menit 15 detik
5	*Easy 1*	23 menit 24 detik

Tabel 5.3 Uji Coba *Easy* vs *Medium* pada *Map* 1

Percobaan ke	Pemenang	Waktu
1	*Medium*	30 menit 14 detik
2	*Medium*	78 menit 4 detik
3	*Easy*	82 menit 44 detik
4	*Medium*	46 menit 5 detik
5	*Medium*	50 menit 8 detik

Tabel 5.4 Uji Coba *Easy* vs *Hard* pada *Map* 1

Percobaan ke	Pemenang	Waktu
1	*Hard*	10 menit 39 detik
2	*Hard*	8 menit 54 detik
3	*Hard*	10 menit 31 detik
4	*Hard*	9 menit 2 detik
5	*Hard*	10 menit 2 detik

Tabel 5.5 Uji Coba *Medium* vs *Medium* pada *Map* 1

Percobaan ke	Pemenang	Waktu
1	*Medium* 1	53 menit 32 detik
2	*Medium* 1	41 menit 4 detik
3	*Medium* 2	40 menit 54 detik
4	*Medium* 1	76 menit 53 detik
5	Medium 2	61 menit 35 detik

Tabel 5.6 Uji Coba *Medium* vs *Hard* pada *Map* 1

Percobaan ke	Pemenang	Waktu
1	*Hard*	14 menit 27 detik
2	*Hard*	11 menit 6 detik
3	*Hard*	11 menit 1 detik

Percobaan ke	Pemenang	Waktu
4	*Hard*	9 menit 9 detik
5	*Hard*	12 menit 18 detik

Tabel 5.7 Uji Coba *Hard* vs *Hard* pada *map* 1

Percobaan ke	Pemenang	Waktu
1	*Hard 1*	26 menit 28 detik
2	*Hard 2*	37 menit 0 detik
3	*Hard 1*	31 menit 49 detik
4	*Hard 1*	39 menit 50 detik
5	*Hard 2*	23 menit 8 detik

5.4.2 Uji Coba *Map* 2

Uji coba dilakukan pada *map* 2 dengan mempertandingkan setiap level AI yaitu *easy* vs *easy*, *easy* vs *medium*, *easy* vs *hard*, *medium* vs *medium*, *medium* vs *hard*, dan *hard* vs *hard*.

Hasil pengujian pada *map* 2 pada setiap level AI 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 Uji Coba *Easy* vs *Easy* pada *Map* 2

Percobaan ke	Pemenang	Waktu
1	*Easy 1*	35 menit 35 detik
2	*Easy 1*	54 menit 29 detik
3	*Easy 2*	49 menit 25 detik
4	*Easy 2*	39 menit
5	*Easy 2*	46 menit 26 detik

Tabel 5.9 Uji Coba *Easy* vs *Medium* pada *Map* 2

Percobaan ke	Pemenang	Waktu
1	*Medium*	25 menit 48 detik

Percobaan ke	Pemenang	Waktu
2	*Easy*	29 menit 2 detik
3	*Medium*	22 menit 9 detik
4	*Medium*	30 menit 21 detik
5	*Medium*	25 menit 23 detik

Tabel 5.10 Uji Coba *Easy* vs *Hard* pada *Map* 2

Percobaan ke	Pemenang	Waktu
1	*Hard*	8 menit 27 detik
2	*Hard*	7 menit 3 detik
3	*Hard*	7 menit 16 detik
4	*Hard*	7 menit 35 detik
5	*Hard*	6 menit 56 detik

Tabel 5.11 Uji Coba *Medium* vs *Medium* pada *Map* 2

Percobaan ke	Pemenang	Waktu
1	*Medium* 2	35 menit 35 detik
2	*Medium* 2	54 menit 29 detik
3	*Medium* 2	49 menit 25 detik
4	*Medium* 1	39 menit
5	*Medium* 1	46 menit 26 detik

Tabel 5.12 Uji Coba *Medium* vs *Hard* pada *Map* 2

Percobaan ke	Pemenang	Waktu
1	*Hard*	6 menit 54 detik
2	*Hard*	7 menit 16 detik
3	*Hard*	8 menit 4 detik
4	*Hard*	6 menit 42 detik
5	*Hard*	7 menit 32 detik

Tabel 5.13 Uji Coba *Hard* vs *Hard* pada *Map* 2

Percobaan ke	Pemenang	Waktu
1	*Hard 2*	14 menit 43 detik
2	*Hard 2*	19 menit 20 detik
3	*Hard 2*	14 menit 27 detik
4	*Hard 1*	17 menit 47 detik
5	*Hard 1*	16 menit 43 detik

5.4.3 Uji Coba *Map* 3

Uji coba dilakukan pada *map* 3 dengan mempertandingkan setiap level AI yaitu *easy* vs *easy*, *easy* vs *medium*, *easy* vs *hard*, *medium* vs *medium*, *medium* vs *hard*, dan *hard* vs *hard*.

Hasil pengujian pada *map* 3 pada setiap level AI dapat dilihat pada Tabel 5.14, Tabel 5.15, Tabel 5.16, Tabel 5.17 , Tabel 5.18, dan Tabel 5.19.

Tabel 5.14 Uji Coba *Easy* vs *Easy* pada *Map* 3

Percobaan ke	Pemenang	Waktu
1	*Easy 1*	35 menit 39 detik
2	*Easy 2*	39 menit 36 detik
3	*Easy 1*	40 menit 9 detik
4	*Easy 2*	36 menit 38 detik
5	*Easy 2*	34 menit 49 detik

Tabel 5.15 Uji Coba *Easy* vs *Medium* pada *Map* 3

Percobaan ke	Pemenang	Waktu
1	*Medium*	29 menit 24 detik
2	*Medium*	33 menit 6 detik
3	*Medium*	26 menit 16 detik
4	*Medium*	29 menit 25 detik

Percobaan ke	Pemenang	Waktu
5	*Medium*	30 menit 2 detik

Tabel 5.16 Uji Coba *Easy* vs *Hard* pada *Map* 3

Percobaan ke	Pemenang	Waktu
1	*Hard*	12 menit 20 detik
2	*Hard*	11 menit 17 detik
3	*Hard*	11 menit 43 detik
4	*Hard*	10 menit 20 detik
5	*Hard*	12 menit 38 detik

Tabel 5.17 Uji Coba *Medium* vs *Medium* pada *Map* 3

Percobaan ke	Pemenang	Waktu
1	*Medium 2*	35 menit 39 detik
2	*Medium 1*	39 menit 36 detik
3	*Medium 1*	40 menit 9 detik
4	*Medium 2*	36 menit 38 detik
5	*Medium 2*	34 menit 49 detik

Tabel 5.18 Uji Coba *Medium* vs *Hard* pada *Map* 3

Percobaan ke	Pemenang	Waktu
1	*Hard*	12 menit 1 detik
2	*Hard*	11 menit 25 detik
3	*Hard*	11 menit 22 detik
4	*Hard*	11 menit 8 detik
5	*Hard*	12 menit 34 detik

Tabel 5.19 Uji Coba *Hard* vs *Hard* pada *Map* 3

Percobaan ke	Pemenang	Waktu
1	*Hard 1*	61 menit 7 detik
2	*Hard 2*	137 menit 36 detik

Percobaan ke	Pemenang	Waktu
3	*Hard 1*	60 menit 10 detik
4	*Hard 2*	87 menit 46 detik
5	*Hard 2*	171 menit 36 detik

5.3 Skenario Uji Coba AI Melawan Pemain Manusia

Pengujian ini adalah pengujian dengan mempertandingkan AI melawan pemain manusia pada beberapa kemungkinan *map* dan tingkat kesulitan dengan mencatat hasil kemenangan dan waktu selesainya permainan.

Tabel 5.20 Uji Coba Pemain Manusia ke-1

Pemain	Musuh	Pemenang	Waktu
Medium	*Easy*	Pemain	10 menit 28 detik
Medium	*Medium*	Musuh	20 menit 3 detik
Medium	*Hard*	Musuh	6 menit 4 detik

Tabel 5.21 Uji Coba Pemain Manusia ke-2

Pemain	Musuh	Pemenang	Waktu
Medium	*Easy*	Pemain	11 menit 11 detik
Medium	*Medium*	Pemain	15 menit 43 detik
Medium	*Hard*	Musuh	9 menit 4 detik

Tabel 5.22 Uji Coba Pemain Manusia ke-3

Pemain	Musuh	Pemenang	Waktu
Medium	*Easy*	Pemain	23 menit 11 detik
Medium	*Medium*	Musuh	21 menit 23 detik
Medium	*Hard*	Musuh	12 menit 4 detik

5.4 Evaluasi Pengujian

Berdasarkan pengujian yang telah dilakukan pada bab 5.2, maka *winning rate* AI dapat dirangkum pada Tabel 5.23.

Tabel 5.23 *Winning Rate* AI

	Easy	*Medium*	*Hard*
Easy	40%	13.333%	0%
Medium	-	53.333%	0%
Hard	-	-	46.667%

Dari hasil analisa dapat dilihat bahwa pada setiap kemungkinan AI, apabila ia melawan AI yang tingkat kesulitannya lebih tinggi kemungkinan menangnya cenderung kecil. Apabila melawan AI yang sama maka kemungkinan menangnya rata-rata mendekati hampir sama. Dan apabila ia melawan AI yang lebih mudah maka kemungkinan menangnya cenderung lebih besar.

Dapat dilihat juga bahwa dengan semakin jauhnya perbedaaan tingkat kesulitan, kecepatan permainan akan semakin cepat. AI *hard* melawan *easy* akan lebih cepat daripada AI *hard* melawan AI *medium*. Hal itu disebabkan Karena AI dengan tingkat kesultian tinggi akan dengan cepat membunuh AI dengan tingkat kesulitan yang rendah.

Dengan itu dapat disimpulkan bahwa AI telah terbentuk sesuai dengan tingkat kesulitannya yang sesuai. Semakin tinggi tingkat AI, maka kemungkinan kemenangannya akan semakin besar.

[Halaman ini sengaja dikosongkan]

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. Penggunaan *Finite State Machine* dalam *game* berjenis *tower defense* yang dikombinasikan dengan strategi untuk menang dan aturan *game* dapat membentuk AI yang kuat.
2. AI yang memiliki tingkat kemungkinan tindakan *random* yang rendah akan lebih kuat daripada AI yang memiliki kemungkinan tindakan *random* yang lebih tinggi.
3. AI yang terbentuk telah sesuai dengan tingkat kesulitannya, dimana AI yang lebih susah akan memiliki kemungkinan kemenangan yang lebih besar.

6.2 Saran

1. Penambahan variasi pasukan agar AI dapat bekerja lebih baik.
2. Penambahan variasi strategi untuk memenangkan *game* dengan *Finite State Machine*.

[Halaman ini sengaja dikosongkan]

DAFTAR PUSTAKA

[1] "What is Artificial Intelligence (AI)? - Definition from Techopedia," *Techopedia.com*. [Online]. Available: https://www.techopedia.com/definition/190/artificial-intelligence-ai. [Accessed: 12-Feb-2017].

[2] "*Game* Genres Defined: *Tower* Defense," *Vita Player - the one-stop resource for PS Vita owners*. [Online]. Available: http://www.vitaplayer.co.uk/*game*-genres-defined-*tower*-defense/. [Accessed: 12-Feb-2017].

[3] "Finite-*State* Machines: Theory and Implementation," *Game Development Envato Tuts+*. [Online]. Available: https://*game*development.tutsplus.com/tutorials/finite-*state*-machines-theory-and-implementation--*game*dev-11867. [Accessed: 12-Feb-2017].

[Halaman ini sengaja dikosongkan]

BIODATA PENULIS

Billy Lahir pada 25 Juli 1995 di kota Surabaya. Penulis menempuh Pendidikan SD di Vita School Surabaya (2001-2007), SMP di Vita School Surabaya (2007-2010), SMA di SMAK Katolik St Louis 1 Surabaya (2010-2013), dan S1 Teknik Informatika Fakultas Teknologi Informasi di Institut Teknologi Sepuluh Nopember Surabaya (2013-2017).

Selama kuliah di Teknik Informatika ITS, penulis mengambil bidang minat Interaksi Grafika dan Seni (IGS). Penulis dapat dihubungi melalui alamat surel **billy.adam1@gmail.com**