1. PENDAHULUAN
   1. Latar Belakang Permasalahan

*Game* merupakan suatu aktivitas paling menyenangkan dan *enjoyable* apabila *game* tersebut memberikan tantangan yang cukup untuk pemainnya. Contoh tantangan tersebut misalnya, mempelajari cara bermainnya, memecahkan masalah, atau menemukan hal-hal baru ketika bermain game tersebut (Korhonen & Koivisto, 2006). Oleh karena itu, *game* membutuhkan perilaku musuh yang tepat agar *game* tersebut dapat dinikmati oleh pemain-pemainnya.

*Video games* merupakan hiburan dan tantangan. *Video games* tidak akan menyenangkan jika tidak ada tantangannya. Apabila tantangan tersebut terlalu mudah, maka *game* tersebut akan menjadi membosankan. Tetapi jika sebaliknya, *game* tersebut dapat membuat pemain frustasi. Hal ini berhubungan dengan *flow-state*, yaitu ketika kemampuan pemain dan tantangan dari game setara (Sofyan, Akbar, & Afirianto, 2019). Di dalam seri permainan *turn-based tactics game,* biasanya musuh AI memiliki fitur tambahan seperti *unit* yang lebih banyak dan *power* yang lebih kuat. Hal ini membuat *player* yang ingin mendapatkan pengalaman bermain yang optimal tanpa mengeluarkan waktu lebih banyak menjadi enggan untuk bermain permainan tersebut meskipun menyukai *genre* tersebut, karena hal-hal tersebut mengharuskan *player* untuk melakukan aktivitas bernama *grinding*. *Grinding* adalah sebuah aktivitas dimana seorang *player* melakukan *simple action* yang sama berulang kali untuk mendapatkan *resources* (Karlsen, 2011).

*Goal-Oriented Action Planning* adalah suatu metode *decision-making* yang dapat membuat suatu karakter tidak hanya melakukan apa yang akan dia lakukan, tetapi juga menentukan bagaimana cara ia melakukannya. Dengan struktur GOAP, GOAP mampu memfasilitasi suatu karakter dengan cara mempertahankan dan menggunakan ulang *behavior* tersebut disesuaikan dengan situasi dimana karakter tersebut berada. Sistem GOAP tidak akan mengganti kebutuhan akan Finite-state machine, tetapi lebih menyederhanakan FSM yang dibutuhkan (Orkin, 2003). Di dalam sisi pengimplementasian, dengan menggunakan GOAP, apabila ada *action* yang ingin ditambahkan, pembuat hanya tinggal menambahkan *action* ke dalam program, tanpa perlu mengganti *action* lainnya. Dengan menggunakan metode *Goal-Oriented Action Planning*, *agent* AI akan memiliki aksi yang cukup bervariasi dan adaptif pada *state* yang dia alami, sehingga *player* dapat bermain dengan *agent* AI yang memiliki *resources* yang sama dengan kesulitan yang cukup menantang untuk *player* sehingga *player* dapat menikmati permainan secara optimal tanpa harus melakukan *grinding*.

Penelitian tentang teori pengimplementasian GOAP ke dalam *game* pernah dilakukan oleh Jeff Orkin pada tahun 2003. Jeff Orkin menganalisa sebuah game bernama *No One Lives Forever 2: A spy in H.A.R.M.’s Way (NOLF2)*, karena *game* tersebut adalah salah satu contoh yang memiliki *goal-directed* *autonomous characters*, tanpa kemampuan untuk *planning*. Karakter di dalam *NOLF2* secara konstan melakukan evaluasi ulang tujuan mereka, dan memilih tujuan yang paling relevan untuk mengontrol perilaku mereka. Di dalam penelitiannya, Jeff Orkin mengatakan bahwa sebuah karakter akan membuat sebuah rencana secara *real-time* dengan cara memasukkan tujuan di dalamnya kepada sebuah sistem bernama *planner*. *Planner* tersebut akan mencari *action* yang dibutuhkan untuk sebuah *sequence* yang akan membawa sebuah karakter dari *starting state* hingga *goal state*. Jika *planner* tersebut sukses, *planner* akan menghasilkan sebuah rencana untuk diikuti oleh karakter tersebut untuk pengarahan perilakunya. Jika tidak, maka karakter akan meninggalkan rencana sekarang dan akan membuat yang baru.

Magnusson dan Hall pernah melakukan penelitian serupa pada sebuah *Real-time Strategy Game* pada tahun 2010. Magnusson membuat sebuah AI bernama AI Ice, yang bertujuan untuk mengalahkan musuh di dalam *game* tersebut. Untuk melakukan hal tersebut, AI Ice akan mencari informasi di dalam *gameplay* untuk menentukan, membuat, dan menjalankan sebuah tugas sesuai prioritas yang paling tinggi. Hasil dari penelitian tersebut mengatakan bahwa AI Adaptive dengan metode GOAP yang memiliki fitur yang sama dengan pemain manusia (*disabled fog of war, no extra resource, etc*) mampu mengalahkan AI statis yang memiliki fitur yang lebih dari fitur pemain manusia.

Tujuan yang diangkat dari penelitian ini adalah untuk kontribusi pengetahuan lebih lanjut tentang metode *Goal Oriented Action Planning* pada dataset yang berbeda, khususnya *Turn-based Tactics Video Game*, dengan tujuan untuk membuat *Agent* AI yang adaptif sesuai dengan state yang dialami AI pada *genre game* tersebut. Dengan aksi AI yang variatif dan adaptif, AI tidak membutuhkan *resources* tambahan, sehingga *player* tidak memerlukan *grinding* untuk mendapatkan *progress* di dalam *game* tersebut, tetapi masih mendapatkan kesulitan yang cukup menantang ketika melawan AI tersebut. Agar permainan tidak berjalan denganmonoton, maka permainan diberikan sisi *gambling* berupa *hidden personality* di tiap karakter agar gerakan dari karakter bisa berbeda dari satu pertandingan ke pertandingan lainnya. Hasil dari penelitian ini akan berupa uji coba dari pemain melawan *agent* AI GOAP dan uji coba AI GOAP melawan AI Finite-State Machine pada umumnya, untuk mengetahui apakah metode GOAP lebih baik daripada Finite-State Machine pada *game* *Turn-based tactics*.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang ada, permasalahan dapat dirumuskan sebagai berikut:

1. Apa saja yang akan dilakukan oleh AI GOAP untuk mencapai tujuannya di dalam situasi yang berbeda-beda?

2. Seberapa pengaruh performa agent AI GOAP yang tidak memiliki *resources* tambahan terhadap tingkat kepuasan pemain jika dibandingkan dengan AI FSM yang memiliki *resources* tambahan?

3. Seberapa baik performa agent AI GOAP jika dibandingkan dengan agent AI FSM yang memiliki *resources* tambahan?

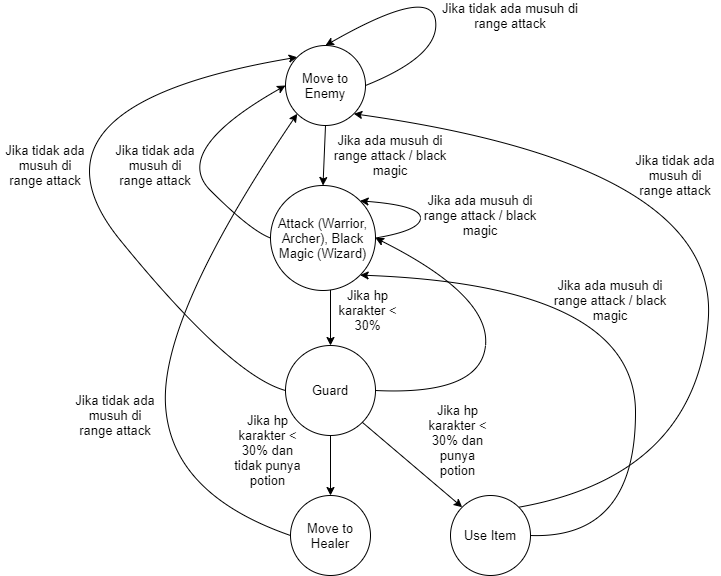
1.3 Tujuan Penelitian

Tujuan dari skripsi ini adalah pengimplementasian agent AI yang adaptif terhadap berbagai situasi menggunakan metode *Goal-Oriented Action Planning* pada *Turn-based Tactics Video Games* dengan harapan dapat memberikan kesulitan yang cukup menantang meski agent AI tidak memiliki *resources* tambahan ketika melawan *human player* sehingga *player* tidak memerlukan *grinding* untuk mendapatkan *progress* di dalam *game* tersebut.

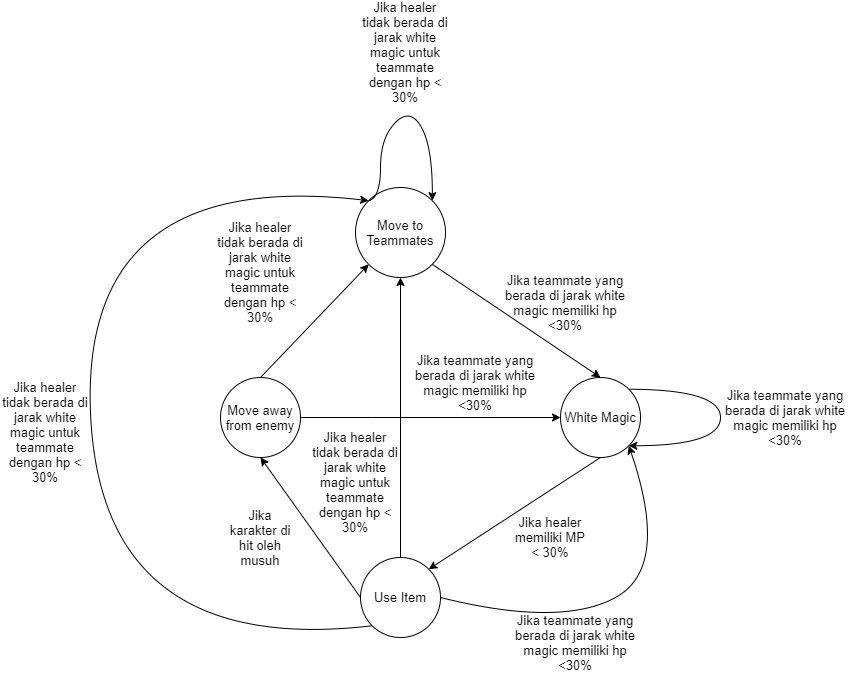
* 1. Ruang Lingkup

Ruang lingkup skripsi ini dibatasi pada :

1. *Game* dibuat menggunakan Unreal Engine 4.
2. *Game* memiliki 4 jenis karakter berbeda, yaitu Warrior, Wizard, Archer, dan Healer. Tiap karakter memiliki tingkat serangan dan pertahanan yang berbeda-beda, serta aksi yang berbeda-beda. Setiap karakter akan memiliki *personality* (*aggressive, neutral, or non-aggressive*) yang tidak diketahui oleh musuh. *Personalities* tersebut memiliki algoritma gerakan yang berbeda-beda untuk mencegah permainan bersifat monoton.
3. *Game* memiliki 2 jenis serangan dan pertahanan, yaitu *physical* dan *magical*.
4. *Game* bersifat *Tile-based system*, dimana *player* dan AI dapat menggerakan karakter ke *tile* tertentu sesuai keinginan dan batasan karakter tiap putaran.
5. Action yang ada di *game* ini untuk dilakukan karakter-karakter ada enam, yaitu :
   1. Move
   2. Attack
   3. Guard
   4. Black Magic
   5. White Magic
   6. Use Item
6. Menggunakan metode *Goal-Oriented Action Planning* dalam pembuatan agent AI untuk melawan agent AI FSM dan pemain manusia. State yang dimiliki AI FSM adalah sebagai berikut :



Gambar 1.1. *Finite-State Machines* untuk karakter *non-healer*



Gambar 1.2. *Finite-State Machines* untuk karakter *Healer*

1. Output dari penelitian ini adalah program agent AI dengan metode *Goal-Oriented Action Planning* menggunakan A\* search*.*
2. Tiap *move* yang dilakukan oleh agent AI GOAP akan dicatat sebagai perbandingan dengan move yang dilakukan oleh metode FSM.
3. Pengujian seberapa adaptif agent AI GOAP berupa sepuluh kali pertandingan melawan *player* yang memiliki pengalaman bermain yang berbeda-beda.
4. Pengujian keberhasilan performa agent AI GOAP berupa perbandingan *winrate* terhadap melawan agent AI FSM dalam 10 pertandingan dengan *resources* yang berbeda-beda.
5. Pengujian kepuasan pemain melawan agent AI GOAP dilakukan dengan cara kuisioner.

1.5 Metodologi Penelitian

Langkah-langkah dalam mengerjakan skripsi :

1. Studi Literatur

1.1 *Goal-Oriented Action Planning & Finite-State Machine*

1.2 *Unreal Engine 4*

2. Perencanaan dan Pembuatan Program

2.1 Pembuatan karakter-karakter serta aksinya menggunakan Unreal Engine 4.

2.2 Pembuatan GOAP Planner beserta *cost* dari aksi-aksinya untuk tiap jenis karakter.

2.3 Pembuatan *gameplay* dan fitur menggunakan *Unity*.

3. Pengujian dan Analisis Program

3.1 Melakukan uji coba agent AI GOAP terhadap agent AI FSM yang memiliki *resources* yang bermacam-macam dan *human player*.

3.2 Melakukan survey agent AI GOAP dan agent AI FSM terhadap kepuasan pemain.

3.3 Analisa hasil pengujian dari aplikasi.

4. Pengambilan kesimpulan

4.1 Membuat kesimpulan tentang hasil penelitian dari analisa yang sudah dilakukan.

4.2 Membuat saran untuk penelitian serupa kedepannya.

5. Pembuatan Laporan

1.6 Sistematika Penulisan

Penulisan laporan Skripsi ini dibagi menjadi beberapa bab, yaitu :

BAB I: PENDAHULUAN

Bab I berisikan judul, latar belakang, perumusan masalah, ruang lingkup, tujuan skripsi, metodologi penelitian, dan sistematika penulisan yang akan digunakan.

BAB II: LANDASAN TEORI

Bab II berisikan teori-teori serta metode-metode yang digunakan dalam pembuatan skripsi.

BAB III: ANALISIS DAN DESAIN SISTEM

Bab III berisikan analisis dan desain sistem yang dibuat

BAB IV: IMPLEMENTASI SISTEM

Bab IV berisikan tentang implementasi sistem berdasarkan desain sistem seperti pada Bab III.

BAB V: PENGUJIAN SISTEM

Bab V berisikan pengujian sistem yang telah dibuat pada Bab IV.

BAB VI: KESIMPULAN DAN SARAN

Bab VI berisikan kesimpulan yang dapat diambil terhadap hasil yang dicapai, dan saran–saran yang berguna bagi pengembangan selanjutnya.

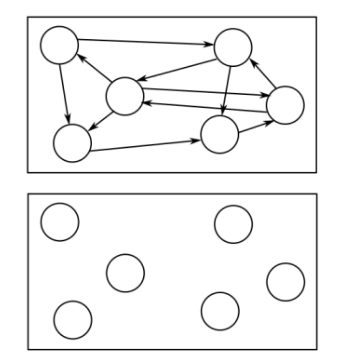
# LANDASAN TEORI

Pada bab ini akan dijelaskan mengenai teori yang dipakai untuk menyelesaikan masalah dalam pembuatan agent AI *Goal-Oriented Action Planning* di dalam *Turn-based Tactics Video Game*.

2.1 Tinjauan Pustaka

2.1.1 Goal-Oriented Action Planning

*Goal-Oriented Action Planning* adalah suatu metode *decision-making* yang dapat membuat suatu karakter tidak hanya melakukan apa yang akan dia lakukan, tetapi juga menentukan bagaimana cara ia melakukannya. Dengan struktur GOAP, GOAP mampu memfasilitasi suatu karakter dengan cara mempertahankan dan menggunakan ulang *behavior* tersebut disesuaikan dengan situasi dimana karakter tersebut berada. Sistem GOAP tidak akan mengganti kebutuhan akan Finite-state machine, tetapi lebih menyederhanakan FSM yang dibutuhkan (Orkin, 2003). GOAP merupakan teknik untuk *decision making* yang akan menghasilkan rantai aksi yang bernama *plan* untuk mencapai suatu *goal state* yang sudah ditentukan sebelumnya (Studiawan et al, 2018). GOAP biasanya memiliki 3 komponen utama dalam melakukan tugasnya, yaitu *goals* dan *actions*. *Goals* atau tujuan disini adalah sebuah kondisi yang harus dicapai oleh suatu *agent*, dengan melakukan *action*-*action* yang dapat mencapai kondisi tersebut.



Gambar 2.1. *Simplification* dari GOAP

Sumber : Studiawan, R., Hariadi, M. & Sumpeno, S. (2018). *Tactical Planning in Space Game using Goal-Oriented Action Planning.* Journal on Advanced Research in Electrical Engineering, 2(1), 5. <https://doi.org/10.12962/j25796216.v2.i1.32>

Contoh dari pemakaian GOAP adalah ketika ada seorang agen yang ingin membuat sebuah kayu bakar. Untuk membuat sebuah kayu bakar, agen tersebut memiliki tiga tindakan yang sudah ditetapkan, yaitu :

1. *getAxe.* Biaya: 2. Prekondisi: “*An axe is available*”, “*doesn’t have an axe*”. Efek: “*has an axe*”.
2. *chopLog*. Biaya: 4. Prekondisi: “*has an axe*”. Efek: “*make firewood*”.
3. *collectBranches.* Biaya: 8. Prekondisi: (tidak ada) Efek: “*make firewood*”.

*World* state yang dimiliki agen tersebut adalah “*doesn’t have an axe*“ dan “*an axe is available*“. Berikut adalah GOAP *Planner* di dalam menyusun action sesuai *world state* yang ada.

GOAL: "make firewood"

Current State: "doesn’t have an axe", "an axe is available"

Can action ChopLog run?

    NO - requires precondition "has an axe"

    Cannot use it now, try another action.

Can action GetAxe run?

    YES, preconditions "an axe is available" and "doesn’t have an axe" are true.

    PUSH action onto queue, update state with action’s effect

New State

    "has an axe"

    Remove state "an axe is available" because we just took one.

Can action ChopLog run?

    YES, precondition "has an axe" is true

    PUSH action onto queue, update state with action’s effect

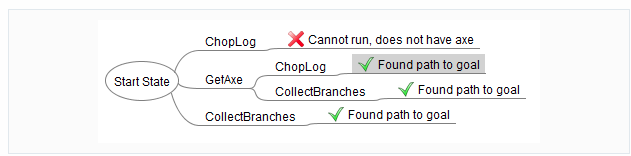
New State

    "has an axe", "makes firewood"

We have reached our GOAL of  "makes firewood"

Action sequence: GetAxe -> ChopLog

*Planner* tersebut akan berjalan melalui *action* yang lain juga, dan akan mencari *action* yang memiliki *lowest cost* yang mampu mencapai ke *goal* tersebut. *Tree* yang dibuat *planner* tersebut akan berbentuk seperti ini :



Gambar 2.2. *Tree* yang dibuat GOAP *Planner*

Sumber : Owens, B. (2014, April 23). *Goal Oriented Action Planning for a Smarter AI*. gamedevelopment.tutsplus.com/tutorials/goal-oriented-action-planning-for-a-smarter-ai--cms-20793

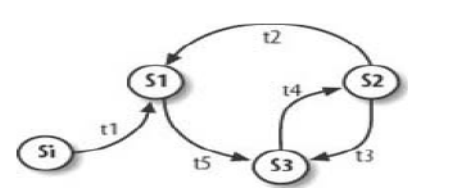
Melalui *planner* tersebut kita dapat melihat bahwa ada tiga rangkaian *action* yang dapat dijalankan, yaitu :

1. *getAxe -> chopLog* (membutuhkan biaya 6)
2. *getAxe -> collectBranches* (membutuhkan biaya 10)
3. *collectBranches* (membutuhkan biaya 8)

Dengan ini, *planner* akan mengambil *action* yang memiliki *cost* yang paling rendah, yaitu *action* pertama (*getAxe -> chopLog)* (Owens, 2014).

* + 1. **Finite State Machine**

*Finite state machine* adalah sebuah metodologi perancangan sistem control yang menggambarkan tingkah laku atau prinsip kerja sistem dengan menggunakan tiga hal berikut: *state, event,* dan *action* (Rahadian et al, 2016). Sistem dapat berpindah dari satu *state* ke *state* lain apabila suatu *event* terjadi. Biasanya transisi antar *state* disertai dengan aksi yang bisa dilakukan, baik aksi yang sederhana maupun kompleks.



Gambar 2.3. Struktur *Finite State Machine*

Sumber : Rostianingsih, S., Budhi, G., S. & Wijaya, H. K. (2013). *Game Simulasi Finite State Machine Untuk Pertanian dan Peternakan*. <http://repository.petra.ac.id/id/eprint/16408>

Pada gambar 2.3 terdapat 4 *state* {Si, S1, S2, S3} yang mungkin terjadi, setiap *state*-nya dapat berpindah *state* jika kondisi terpenuhi. Sebagai contoh *state* S1 dapat berpindah jika kondisi t5 terpenuhi (Rostianingsih, Budhi, & Wijaya, 2013).

* + 1. **Turn-Based Strategy Games (TBS)**

*Game* yang memiliki *genre turn-based strategy* (TBS) mendorong pemain untuk mengambil keputusan mengenai *action* jenis apa yang perlu dilakukan oleh pion-pion atau *unit-unit* yang dimiliki (Chandra, 2018). Turn-based Strategy Games bersifat putaran, dimana tiap pemain memiliki waktunya sendiri untuk menggerakkan *unit-unit* yang dimiliki oleh pemain tersebut. Tiap pemain memiliki objektif, yaitu mengalahkan musuh dengan *unit-unit* yang mereka miliki.



Gambar 2.4 *Advance Wars: Days of Ruin*

Sumber : Chandra, A. V. (2018). *Perbandingan Performa Turn-based Game Menggunakan Algoritma Genetika dan Logika Fuzzy*. https://dewey.petra.ac.id/catalog/digital/detail?id=42374

* + 1. **A\* Pathfinding**

*Pathfinding* adalah salah satu masalah dasar yang didapatkan ketika membuat *game*. Biasanya, *pathfinding* digunakan untuk menangani ke arah mana karakter harus berjalan untuk mencapai tujuannya. Metode yang digunakan untuk menangani *pathfinding* ada bermacam-macam, salah satunya adalah A\* (*A-star*).

A\* adalah *generic search algorithm* yang bisa digunakan untuk mencari solusi untuk berbagai masalah, *pathfinding* adalah salah satunya. Metode yang digunakan menunjukkan bahwa cara ini akan mencari suatu *path* dengan *exploring minimum number of nodes* dengan *minimum cost solution*. Karena *simplicity* yang dimilikinya, A\* hampir selalu menjadi pilihan *search method*. Ini karena A\* menjamin untuk mencari rute terpendek di graf (Mathew, 2015). Menurut Dalem, I., B., G., W., A. (2018), notasi yang dipakai oleh algoritma A\* adalah sebagai berikut :

f(n) = g(n)+h(n) (2.1)

f(n) = biaya estimasi terendah

g(n) = biaya dari *node* awal ke *node* n

h(n) = perkiraan biaya dari *node* n ke *node* akhir

* 1. **Tinjauan Studi**

**2.2.1 *Applying Goal-Oriented Action Planning to Games***(Orkin, 2003)

* Masalah yang diangkat pada penelitian tersebut adalah peneliti ingin menganalisa cara kerja agent AI di sebuah *First-person shooter game* bernama *No One Lives Forever 2: A Spy in H.A.R.M.’s Way*, yang menggunakan metode *Goal-Oriented Action Planning.*
* Hasil penelitian menunjukkan bahwa karakter di dalam NOLF2memiliki *goal* sebanyak 25. Pada suatu waktu, satu *goal* akan aktif, mengontrol tingkah laku karakter. *Goal* di dalam NOLF2 jatuh ke dalam tiga kategori, yaitu *relaxed goals*, *investigative goals*, dan *aggressive goals*. *Relaxed goals* menyangkut *Sleep, work,* dan *patrol*. *Investigative Goals* menyangkut *suspicious investigate* dan *search*. *Aggressive Goals* menyangkut *combat situations*, seperti *chase, charge*, dan *attack from cover*. Meski secara konsep hamper sama, tetapi tujuan di *NOLF2* dan GOAP pada umumnya memiliki satu perbedaan. NOLF2 memiliki *embedded plan*. Di dalam penelitiannya, *search* yang digunakan adalah A\* *algorithm*, yang membutuhkan jumlah *cost* dari *node*, dan *heuristic distance* dari sebuah *node* ke sebuah *goal*.
* Perbedaan penelitian yang bersangkutan dengan penelitian ini terletak pada cara meneliti dan dataset yang diteliti. Pada penelitian yang bersangkutan, peneliti hanya menganalisa agent AI yang ada di dalam *game* NOLF2, dan dataset yang digunakan adalah dataset *first-person shooter game*.

**2.2.2 *Adaptive Goal Oriented Action Planning for RTS Games***(Magnusson & Hall, 2010)

* Masalah yang diangkat pada penelitian tersebut adalah peneliti ingin mengimplementasi *adaptive goal-oriented* pada AI di dalam sebuah *Real-time strategy game* karena peneliti melihat bahwa AI di dalam *game* RTS itu memiliki “fitur” lebih dan bergerak secara statis melalui *state-machines* dan semacamnya.
* Metode yang digunakan pada penelitian ini adalah metode simple *goal-oriented* *system*, dengan *adaptive behavior*, yang berarti metode tersebut akan membuat sebuah task yang akan *counter* perilaku musuh.
* AI Ice adalah sebuah AI yang dibuat di dalam penelitian ini. AI Ice membuat rencana-rencana, yang secara praktek sama dengan membuat variasi tugas yang mengarahkan kemenangan kepada dirinya. Untuk melakukan hal tersebut, AI Ice harus mendapatkan informasi di dalam *game* untuk menentukan, membuat, dan menjalankan *tasks* dengan *highest priority*. AI Ice akan selalu membuat *units* berdasarkan tipe senjata dan armor *units* musuh. AI Ice dibuat menggunakan Spring Engine
* Hasil penelitian menunjukkan bahwa adaptive AI yang dibuat oleh peneliti, yang memiliki “fitur” yang sama dengan *human player*, dapat mengalahkan *static* AI yang memiliki “fitur” lebih. Tetapi, prioritas *tasks* harus lebih diperhatikan, dan bisa dibilang akan menjadi susah untuk membuat sebuah prioritas yang bekerja pada “semua” situasi.
* Perbedaan penelitian yang bersangkutan dengan penelitian ini terletak pada dataset yang diteliti. Pada penelitian yang bersangkutan, peneliti menggunakan AI buatannya untuk diuji coba dengan AI yang ada di dalam *game* RTS tersebut, tidak ada pengujian dengan *human player*. Genre *game* yang diteliti juga berbeda.

**2.2.3 *Tactical Planning in Space Game using Goal-Oriented Action Planning***(Studiawan, Hariadi, & Sumpeno, 2018)

* Masalah yang diangkat pada penelitian tersebut adalah peneliti ingin membuat sebuah AI di dalam sebuah *game* bergenre *real-time tactics*. Peneliti ingin mencari tahu apakah AI menggunakan *Goal-Oriented Action Planning* dapat digunakan pada *space tactical game*.
* Metode yang digunakan pada penelitian ini adalah metode *Goal-Oriented Action Planning* untuk mengatur perilaku AI yang akan dibuat, berdasarkan *tujuan* yang ingin dicapaidan *state* AI tersebut berada. Penelitian ini juga menggunakan *Blackboard system* untuk memanage *data communication* berdasarkan *blackboard architecture model* dimana *general knowledge base* yang bernama *blackboard* digunakan dan diperbaharui oleh komponen-komponen di dalam sistem. Dengan menggunakan *blackboard system*, *module* di dalam *game* dapat lebih fokus ke fungsi daripada fokus untuk berkomunikasi dengan *module* lainnya.
* Hasil penelitian menunjukkan bahwa di dalam pembuatannya, GOAP membantu mengurangi kompleksitas ketika men-*design* AI. Menggunakan definisi yang ada di dalam penelitian tersebut, komposisi yang dinamis dapat terlihat. Di sisi lain, AI *designer* harus mengevaluasi semua aksi yang berhubungan dengan parameter ketika ada pergantian jenis parameter di dalamnya, dan hal tersebut diakui sebagai salah satu kesulitan menggunakan GOAP.
* Perbedaan penelitian yang bersangkutan dengan penelitian ini terletak pada tujuan penelitian dan dataset yang diteliti. Pada penelitian yang bersangkutan, tujuan penelitian yang dilakukan adalah bagaimana sebuah AI melakukan tugasnya ketika dihadapkan dalam berbagai kasus dan berbagai komponen.

1. **ANALISIS DAN DESAIN SISTEM**

**3.1 Analisis Permasalahan**

Di berbagai *game* bergenre *Turn-based Strategy* seperti *Final Fantasy Tactics series*, *player* diberikan *resources* (*unit* pasukan) yang terbatas untuk melakukan *progress*. Pada *series final fantasy* tactics, apabila *player* tidak meningkatkan *level unit* yang dimiliki, mereka tidak dapat melakukan *progress* karena *resources* yang dimiliki oleh musuh jauh lebih besar, sebagai contoh, *level unit* yang dimiliki musuh bisa memiliki perbedaan sangat jauh dengan *level unit* yang dimiliki oleh *player*. Agar *player* bisa melanjutkan *story game* tersebut, *player* diharuskan melakukan aktivitas bernama *grinding*, yaitu melakukan aktivitas yang sama berkali-kali agar *resources* yang dimiliki *player* menjadi banyak, agar *player* dapat mengimbangi musuh AI. *Grinding* di sini berarti melawan musuh yang sama untuk mendapatkan *level unit* pasukan yang dimiliki *player*. Selain itu, musuh di luar *story game* tersebut juga melakukan *scaling* terhadap level *unit* yang *player* punya, sehingga *unit* musuh yang dilawan tidak akan pernah seimbang dengan *unit* yang *player* punya. Hal tersebut membuat kebanyakan *player* lebih kesulitan dalam menangani hal mengimbangi *resources* daripada mencari strategi untuk mengalahkan musuh. Setelah *player* mampu mengimbangi resources yang dimiliki musuh di dalam *story, player* tidak mengalami kesulitan ketika melawan musuh tersebut, karena AI yang dimiliki cenderung statis. Karena hal tersebut, *player* juga diharuskan untuk mengeluarkan waktu lebih banyak di dalam *game* tersebut untuk melakukan *grinding*, daripada memikirkan bagaimana strategi yang tepat untuk melawan *unit-unit* musuh yang ada.Diharapkan dengan menggunakan AI GOAP, meski *resources* yang dimiliki oleh musuh seimbang dengan *resources* yang dimiliki oleh *player*, AI GOAP dapat memberikan kesulitan yang cukup menantang sehingga *player* mendapatkan tantangan yang sesuai ketika melawan AI, dan tidak perlu menyita waktu lebih banyak untuk melakukan *grinding*.

**3.2 Desain Sistem**

Jenis program yang dibuat adalah *game* bergenre *Turn-based Strategy*. Semua pihak di dalam *game* ini memiliki satu tujuan, yaitu untuk mengalahkan semua *unit* yang musuh punya dengan *unit* yang mereka punya. *Map* yang ada di *game* ini berupa petak atau *tiles* yang berbentuk persegi dan biasanya dikenal dengan nama *grid-based map*. Untuk mencapai tujuan tersebut, *player* maupun AI dapat menggerakkan *unit*nya masing-masing. Sebelum permainan dimulai, tiap tim dapat memilih *unit* yang digunakan untuk melawan musuh nantinya.

*Game* ini dibuat dengan tujuan yaitu untuk melakukan penelitian AI mana yang lebih cocok untuk dipakai untuk *Turn-based Strategy*. AI yang akan diuji adalah AI *Finite State Machine* dan AI *Goal-Oriented Action Planning*. Pengujian performa AI akan dilakukan melalui AI FSM melawan AI GOAP, dan kepuasan pemain akan dilakukan melalui *player* melawan AI GOAP, dan hasilnya akan diambil melalui kuesioner. Untuk mendapatkan performa AI masing-masing, akan dilakukan pendataan langkah-langkah yang telah dibuat tiap AI selama pertandingan berjalan.

**3.2.1 Elemen *Game***

*Game* ini memiliki elemen-elemen atau komponen yang sebagian besar sama seperti *game* lainnya yang ber*genre* sama. Elemen-elemen yang ada diantaranya, yaitu :

1. *Map* : *Map* yang digunakan adalah *map* yang berisi petak-petak berbentuk persegi / *tiles*. Ukuran *map* yang digunakan adalah 16 baris x 20 kolom.
2. *Unit* : *Unit* yang digunakan adalah *unit* pasukan (karakter) yang digunakan untuk melakukan penyerangan terhadap *unit* pasukan musuh. *Unit* pasukan memiliki atribut. Atribut-atribut tersebut adalah :

* *Health* : Nyawa yang dimiliki oleh *unit* pasukan tersebut. Jumlah nyawa tergantung pada tipe unit tersebut.
* *Damage* : Kekuatan serangan dari *unit* tersebut untuk mengurangi *unit* pasukan musuh.
* *Move* : Jarak yang digunakan untuk menentukan berapa kotak yang bisa ditempuh *unit* pasukan untuk berpindah *tile*.
* Range : Jarak yang digunakan untuk menentukan berapa jauh kotak yang bisa ditempuh *unit* pasukan untuk menyerang *unit* musuh.
* Initiative : Semakin tinggi *initiative*, semakin cepat gerak *unit* di putaran suatu tim.
* AP : AP (*Action Point*) digunakan untuk menentukan *action* yang dapat diambil. *Default* AP tiap unit adalah 2. Ketika *command* *move* digunakan, AP akan di *set* menjadi 1. Jika AP yang dimiliki *unit* lebih dari sama dengan 1, maka *unit* dapat melaksanakan *command attack* atau *move*. Apabila *command attack* dilaksanakan, maka AP menjadi 0.
* Attack : Ada 2 jenis, yaitu Physical dan Magical. Physical menunjukkan serangan *single target* dengan *medium damage*, sedangkan *magical* menunjukkan serangan *area target* dengan *low damage*.
* Type : Tipe *unit*. Ada 2 jenis, yaitu *melee* dan *range*.
* Job : Job yang ada di dalam *game* ini ada 4, yaitu Warrior, Archer, Mage, dan Healer*.* Warrior memiliki serangan *single target* jarak dekat, Archer memiliki serangan *single target* jarak jauh, Mage memiliki serangan *area target* jarak jauh, dan Healer berguna untuk menambah *Health friendly unit*.

1. *Actions* : *Command* atau perintah yang bisa diarahkan ke *unit* pasukan untuk melakukan sesuatu sesuai dari tipe *unit*.

Berikut adalah tabel atribut per *unit* berdasarkan tipe yang dimiliki *unit*.

Tabel 3.1

Tabel atribut untuk seluruh tipe *unit*

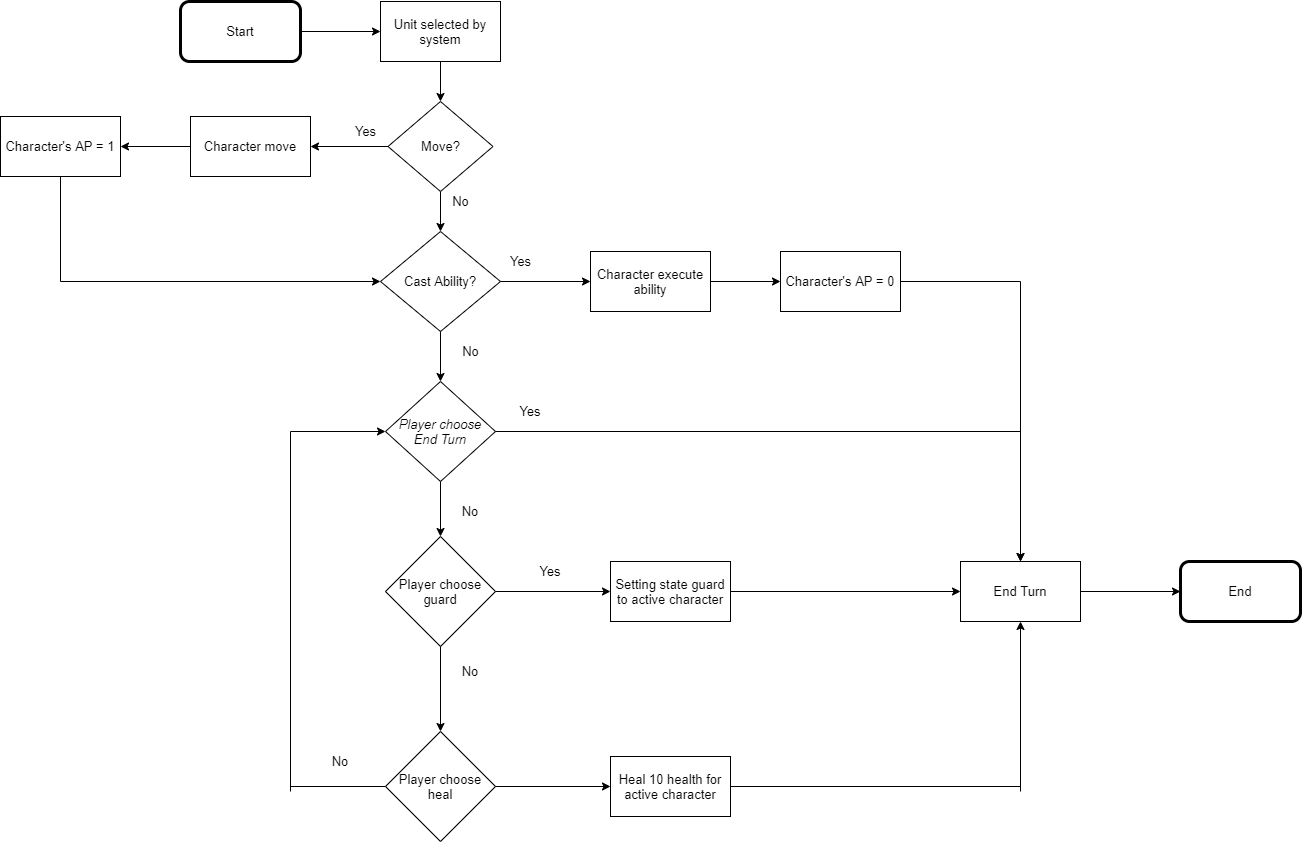
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tipe / Atribut | Health | Damage | Range | Move |
| *Archer* | 80 | 30 | 3 | 3 |
| *Warrior* | 100 | 15 | 1 | 4 |
| *Mage* | 50 | 25 | 4 | 3 |
| *Healer* | 50 | 0 | 1 | 3 |

**3.2.2 Jenis *Game***

Untuk mendapatkan data yang dibutuhkan dari penelitian ini, maka *game* akan dibagi menjadi 2 jenis, yaitu *player* melawan AI GOAP, dan AI FSM melawan AI GOAP. Di AI FSM melawan AI GOAP, tiap langkah-langkah masing-masing AI akan dicatat sebagai data agar bisa diteliti lebih lanjut.

**3.2.3 *Player Turn***

Ketika *game* berada di putaran *player*, sistem akan memilih *unit* yang memiliki speed yang paling cepat diantara semua *unit* pasukan milik *player*. Setelah sistem memilih *unit* pasukan, *unit* pasukan tersebut akan diberikan 2 AP (*Action Point*). *Action points* tersebut berguna untuk menentukan apakah *unit* tersebut dapat melaksanakan *command* yang dipilih atau tidak. Contohnya, untuk melakukan *command move*, *unit* tersebut harus memiliki minimal 2 AP. Apabila *move command* berhasil dijalankan, maka AP *unit* tersebut menjadi 1. Ketika sebuah *unit* memiliki hanya memiliki 1 AP, *unit* tersebut hanya dapat melakukan *command-command* seperti menyerang, *heal,* atau *guard.* Ketika *command* menyerang berhasil dijalankan, maka AP dari *unit* tersebut akan menjadi 0. Sebuah *unit* yang memiliki AP 0 hanya bisa melaksanakan perintah *end turn*. Ketika sebuah perintah *end turn* dilaksanakan, sistem akan mengecek, apakah ada *unit* pasukan yang masih bisa bergerak atau tidak. Jika ya, maka sistem akan memilih *unit* pasukan tersebut agar bisa diberi *command* oleh *player*. Jika tidak, maka giliran akan berpindah ke musuh. Berikut adalah *flowchart* tentang bagaimana putaran *player* berjalan.

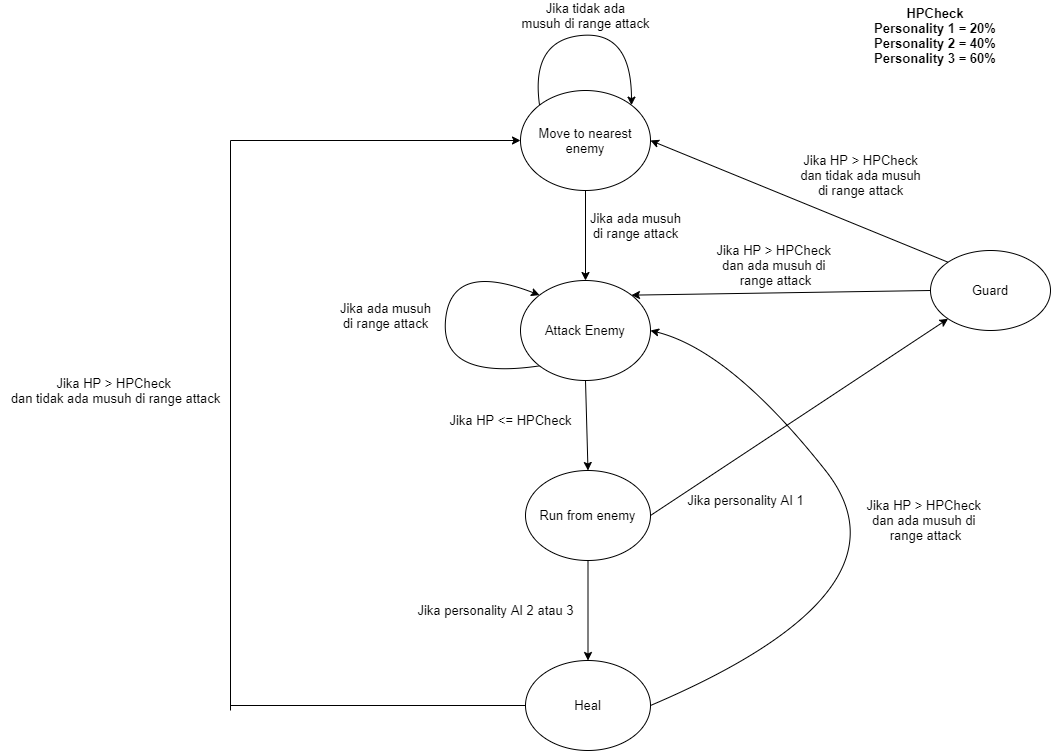


Gambar 3.1 Player Turn

Pada gambar 3.1, dapat dilihat proses saat putaran *player* berlangsung. *Player* dapat memilih apakah *player* mau menggerakkan karakter tersebut. Jika iya, maka karakter akan bergerak ke *tile* yang *player* inginkan. Jika tidak, maka *player* bisa menggunakan kemampuan karakternya. Apabila *player* tidak menggunakan keduanya, maka *player* memiliki 3 opsi untuk mengakhiri putarannya, yaitu *End Turn*, *Heal,* dan *Guard*. Apabila *player* memilih *End Turn*, maka *player* akan mengakhiri putaran karakter tersebut tanpa memiliki efek apa-apa. Apabila *player* memilih *Heal*, maka *player* dapat menyembuhkan nyawa karakternya sebanyak 10 poin. Setelah itu, putaran karakter tersebut akan berakhir. Apabila *player* memilih *guard*, maka karakter tersebut akan berada di *state guard*. *State guard* berfungsi untuk mengurangi damage yang diterima oleh karakter tersebut sebanyak 50%. *State guard* tidak akan hilang sampai putaran karakter kembali kepada karakter tersebut. Setelah karakter memasang *state guard*, putaran karakter tersebut akan berakhir.

**3.2.4 AI FSM Turn**

Ketika *game* berada di putaran AI FSM, sistem akan memilih *unit* yang memiliki speed yang paling cepat diantara semua *unit* pasukan milik AI FSM. Setelah sistem memilih *unit* pasukan, *unit* pasukan tersebut akan diberikan 2 AP (*Action Point*). *Action points* tersebut berguna untuk menentukan apakah *unit* tersebut dapat melaksanakan *command* yang dipilih atau tidak. Contohnya, untuk melakukan *command move*, *unit* tersebut harus memiliki minimal 2 AP. Apabila *move command* berhasil dijalankan, maka AP *unit* tersebut menjadi 1. Ketika sebuah *unit* memiliki hanya memiliki 1 AP, *unit* tersebut hanya dapat melakukan *command-command* seperti menyerang, *guard*, atau *heal*. Ketika *command* menyerang berhasil dijalankan, maka AP dari *unit* tersebut akan menjadi 0. Sebuah *unit* yang memiliki AP 0 hanya bisa melaksanakan perintah *end turn*. Ketika sebuah perintah *end turn* dilaksanakan, sistem akan mengecek, apakah ada *unit* pasukan yang masih bisa bergerak atau tidak. Berikut adalah *state* AI FSM.



Gambar 3.2 *State* dari AI FSM

Pada gambar 3.2, dapat dilihat proses yang dilakukan oleh AI FSM. Apabila AI tidak menemukan musuh di *range attack*, AI akan jalan ke musuh terdekat. Apabila AI masih belum menemukan musuh di *range attack*, maka AI akan terus jalan ke musuh terdekat sampai musuh tersebut mencapai *range attack*. Setelah musuh mencapai di *range attack*, maka AI akan menyerang musuh tersebut. Ketika AI mencapai HP dibawah yang sudah ditentukan sesuai personality (HPCheck), maka AI akan menjalankan fungsi “*Run from enemy*”, yaitu lari dari musuh. Setelah lari dari musuh, akan dilakukan pengecekan personality. Apabila AI mempunyai personality 1, maka AI akan melakukan *guard*. Apabila AI mempunyai personality 2 atau 3, maka AI akan melakukan *heal*.

**3.2.5 AI GOAP Turn**

Ketika *game* berada di putaran AI GOAP, sistem akan memilih *unit* yang memiliki speed yang paling cepat diantara semua *unit* pasukan milik AI GOAP. Setelah sistem memilih *unit* pasukan, *unit* pasukan tersebut akan diberikan 2 AP (*Action Point*). *Action points* tersebut berguna untuk menentukan apakah *unit* tersebut dapat melaksanakan *command* yang dipilih atau tidak. Contohnya, untuk melakukan *command move*, *unit* tersebut harus memiliki minimal 2 AP. Apabila *move command* berhasil dijalankan, maka AP *unit* tersebut menjadi 1. Ketika sebuah *unit* memiliki hanya memiliki 1 AP, *unit* tersebut hanya dapat melakukan *command-command* seperti menyerang atau mengeluarkan sebuah *skill*. Ketika *command* menyerang berhasil dijalankan, maka AP dari *unit* tersebut akan menjadi 0. Sebuah *unit* yang memiliki AP 0 hanya bisa melaksanakan perintah *end turn*. Ketika sebuah perintah *end turn* dilaksanakan, sistem akan mengecek, apakah ada *unit* pasukan yang masih bisa bergerak atau tidak. Agar AI GOAP dapat melakukan sesuatu, sistem AI GOAP harus mengecek *world state game* tersebut. Contoh dari *world state* tersebut adalah “is not on low health”, “attacking target X”, “at target X”, “no enemies remain” dan “is on low health”. Apabila *world state* terpenuhi, maka suatu *action* dapat dilakukan agar *goals* terpenuhi. Berikut adalah tabel-tabel berisi *action* yang dapat dilakukan beserta kebutuhan *world state* dan *goals*.

Tabel 3.2

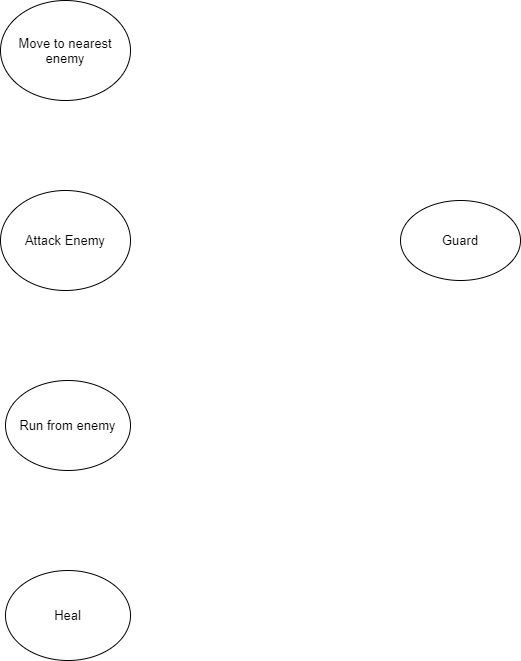
*Action* yang ada pada AI GOAP

|  |  |  |
| --- | --- | --- |
| Actions | Satisfies World State | Requires World State |
| Attack | Attacking target X | At target X  Is not on low health |
| Move to X | At target X  Going to target X | - |
| Guard | Run from enemy X | Is on low health  HP difference <= 10 |
| Heal | Is not on low health  Run from enemy X | Is on low health  HP difference >10 |

Tabel 3.3

*Goals* yang ada pada AI GOAP

|  |  |
| --- | --- |
| Goals | Desired World State |
| Kill enemy | Attacking target X |
| Recover | Run from enemy X  Is on low health |

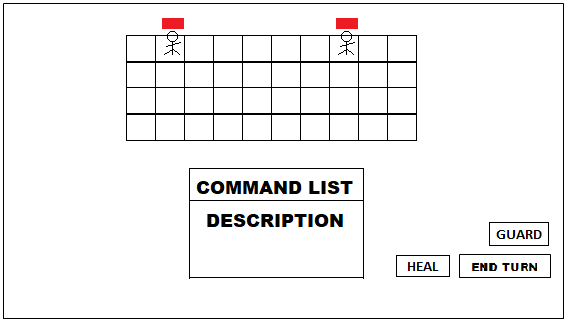


Gambar 3.3 State AI GOAP

Berdasarkan penulisan paper “Tactical Planning in Space Game using Goal-Oriented Action Planning”, state GOAP adalah state FSM yang disederhanakan.

**3.2.6 Tampilan Gameplay**

Ketika *user* masuk ke dalam *game*, *user* akan diberi tampilan *gameplay*. Di dalam tampilan *gameplay* akan ditampilkan karakter yang akan dimainkan, nyawa karakter yang tersisa, serta di *tile* mana karakter tersebut berada. Di bawah layar, terdapat *window command list* beserta *description* yang berisi *command-command* yang tersedia untuk *user* pakai kepada karakter tersebut. Di bawah kanan, terdapat tombol *End Turn*, *Guard*, dan *Heal* yang memiliki fungsi yang berbeda-beda tiap tombolnya.



Gambar 3.4 Tampilan *Gameplay*

Pada gambar 3.1, dapat dilihat bahwa pada *interface* tersebut terdapat jendela *command list* beserta deskripsi dari setiap *command*. Selain itu, terdapat tiga tombol, yaitu *End Turn, Heal*, dan *Guard*. Tiap tombolnya memiliki fungsi yang berbeda-beda. *End Turn* berfungsi untuk mengakhiri putaran karakter tersebut dan menjalankan putaran karakter selanjutnya. *Guard* berfungsi untuk mengurangi *damage* yang diterima oleh musuh sebanyak 50% sekaligus mengakhiri putaran karakter tersebut. *Heal* berfungsi untuk menambah nyawa karakter sebanyak 10 dan sekaligus mengakhiri putaran karakter tersebut.

**3.4 Desain Pengujian**

Metode pengujian yang digunakan untuk penelitian ini yaitu dengan menggunakan AI FSM untuk melawan AI GOAP, dan menggunakan AI GOAP untuk melawan *player*. Pengujian AI FSM vs AI GOAP akan dilakukan sebanyak 12 kali dengan *unit* yang berbeda-beda, dengan giliran dimana AI GOAP mendapatkan giliran pertama sebanyak 6 kali, dan AI FSM mendapatkan giliran pertama sebanyak 6 kali. Pada pengujian tersebut, akan dicatat ukuran-ukuran sebagai berikut :

1. Winrate dari masing-masing AI, kemudian dibandingkan *winrate* ketika mendapatkan giliran pertama dan *winrate* ketika tidak mendapatkan giliran pertama untuk menguji AI mana yang lebih baik untuk *game genre* ini.
2. *Unit* yang dimiliki tiap-tiap AI.

Pengujian *player* vs AI GOAP akan dilakukan kepada 10 orang. Pengujian akan dilakukan kepada orang-orang yang memiliki pengalaman berbeda bermain *game genre* ini. Tingkatan pengalaman akan berupa : 1) Tidak pernah bermain sama sekali, 2) Pernah bermain *game genre ini*, dan 3) Sering bermain *game genre* ini. Tiap orang akan bermain sebanyak 2 kali, dan bisa menggunakan *resources* yang berbeda.

**4. IMPLEMENTASI SISTEM**

Pada bab ini akan dijelaskan tentang sistem yang digunakan sesuai dengan teori-teori dan desain sistem yang ada pada bab sebelumnya. Implementasi sistem yang akan dibahas meliputi pengimplementasian AI FSM dan AI GOAP menggunakan *Unreal Engine 4* *Blueprint* beserta potongan / segmen *screenshot* dari program yang dibuat berupa *blueprint*.

**4.1 Implementasi AI FSM pada *game***

Pada tabel 4.1, terdapat daftar-daftar prosedur yang diimplementasi untuk membuat AI FSM di dalam *game* beserta keterangan dan *flowchart* dari prosedur tersebut.

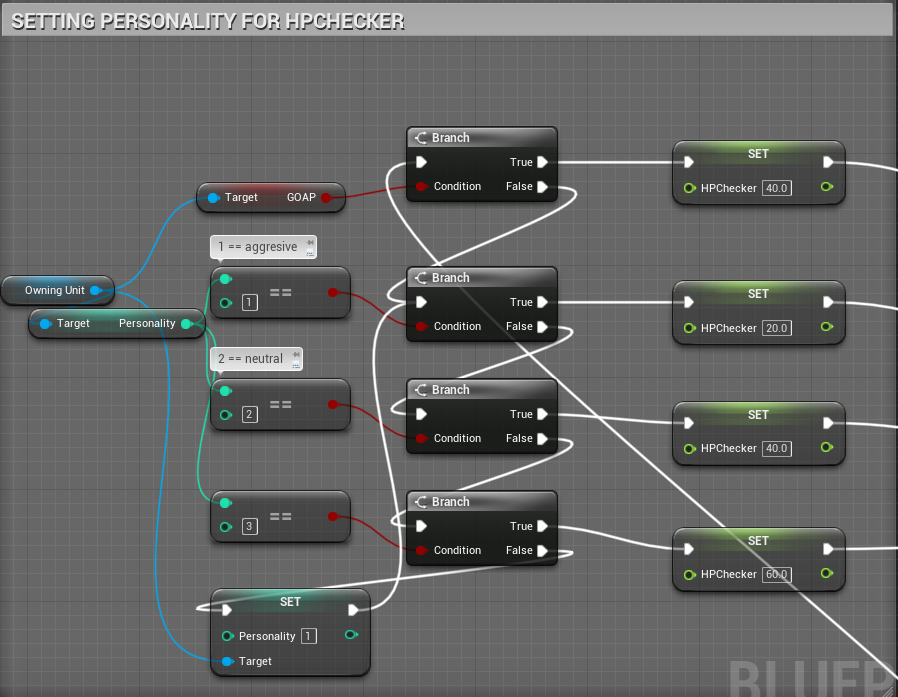
Tabel 4.1 Daftar prosedur yang digunakan untuk AI FSM

|  |  |  |  |
| --- | --- | --- | --- |
| Segmen Program | Nama Prosedur | Keterangan Prosedur | Gambar Flowchart |
| 4.1 | Pengaturan HPCheck untuk tiap personality | Mengatur HPCheck untuk tiap personality AI | - |
| 4.2 | Implementasi AI FSM | Membuat AI FSM sesuai dengan desain yang ada | Flowchart 3.2 |

**4.1.1 Implementasi Personality Karakter AI FSM**

Pada karakter AI FSM, terdapat tiga jenis personality yang digunakan agar alur permainan tidak monoton, yaitu *aggressive, neutral,* dan *non-aggressive*. Yang membedakan setiap *personality* adalah kondisi yang dibutuhkan untuk melakukan aksi “*Run away from enemy*”. Pada personality 1 (*aggressive)*, AI akan menjalankan aksi “*Run away from enemy”*, ketika HP berada di bawah atau sama dengan 20%. Pada personality 2 (*neutral*), AI akan menjalankan aksi “*Run away from enemy”*, ketika HP berada di bawah atau sama dengan 40%. Pada personality 3 (*non-aggressive*), AI akan menjalankan aksi “*Run away from enemy”*, ketika HP berada di bawah atau sama dengan 60%. Segmen program ini dapat dilihat pada Segmen 4.1.

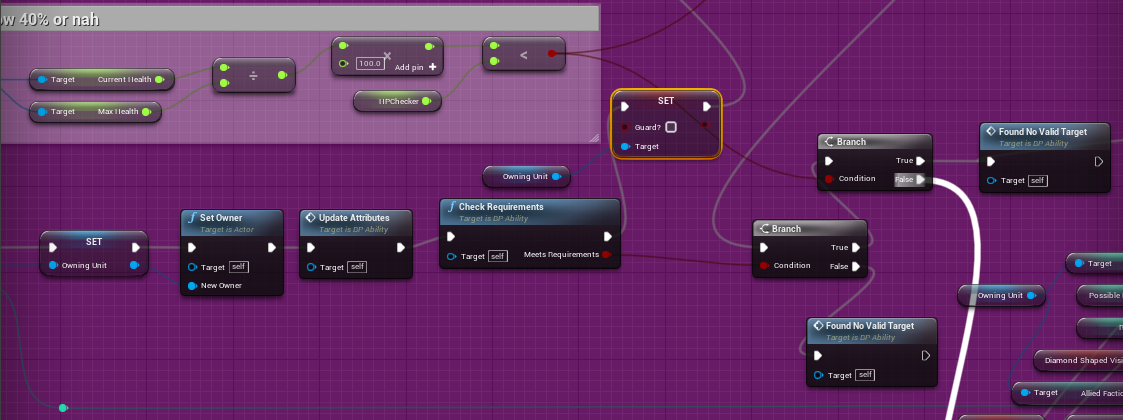
Segmen 4.1 Inisialisasi kondisi HPCheck tiap personality



**4.1.2 Implementasi AI FSM**

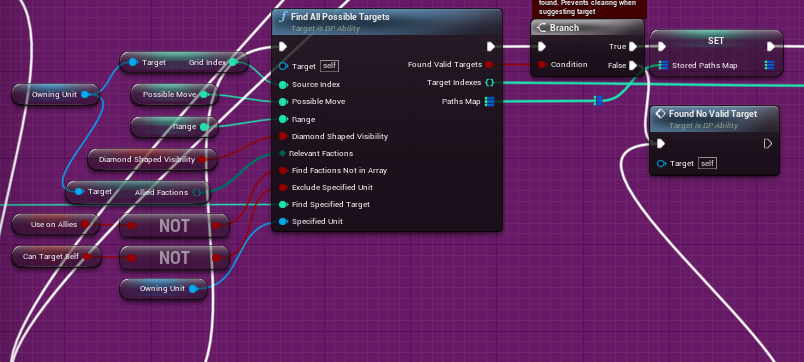
Implementasi AI FSM merupakan sebuah prosedur yang digunakan untuk membuat AI FSM yang bekerja sesuai pada gambar 3.2 yang sudah dibahas sebelumnya. AI FSM disini berfungsi sebagai bahan penelitian yang digunakan untuk melawan AI GOAP nantinya, untuk melihat apakah AI GOAP mampu melawan AI FSM. Segmen program ini dapat dilihat pada Segmen 4.2.

Segmen 4.2 Implementasi AI FSM



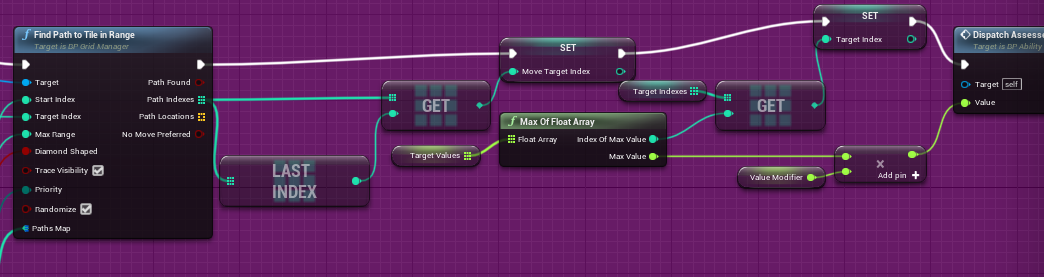
Ketika putaran suatu karakter AI dimulai, maka *state guard* yang dimiliki AI akan dinonaktifkan. Lalu akan dicek apakah HP yang dimiliki AI dibawah HPChecker atau tidak. Jika ya, maka ia akan menjalankan aksi “*Run away from enemy*”.

**Segmen 4.2 Implementasi AI FSM (Lanjutan)**

****

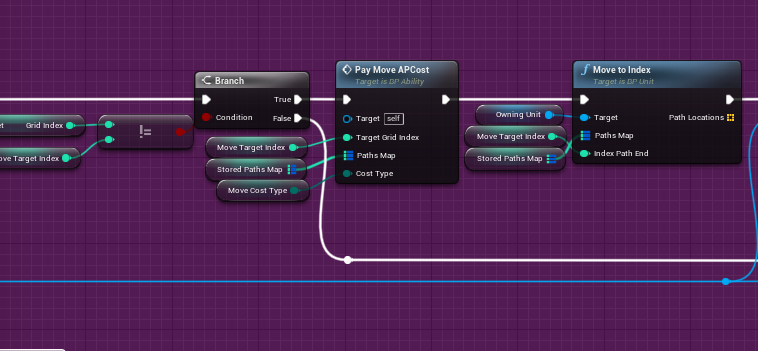
Jika tidak, maka AI akan menjalankan aksi “*Kill enemy”*. AI akan berusaha mencari musuh terdekat dari posisinya.

**Segmen 4.2 Implementasi AI FSM (Lanjutan)**



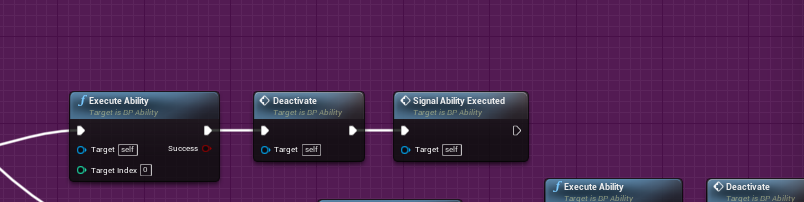
Setelah menemukan musuh terdekat, maka AI FSM akan mencari *tile* yang paling dekat dengan musuh tersebut sesuai *move range* yang dimiliki karakter. Setelah menemukan *tile* yang paling dekat dengan musuh, maka AI akan mengatur tujuan *tile* ke *tile* tersebut.

**Segmen 4.2 Implementasi AI FSM (Lanjutan)**



Setelah mengatur tujuan *tile*, AI akan berjalan ke *tile* tersebut. Apabila AP yang dimiliki AI masih 2, maka AI dapat jalan ke *tile* tersebut.

**Segmen 4.2 Implementasi AI FSM (Lanjutan)**



Apabila *range* *attack* milik AI dapat mencapai musuh, maka AI akan mengeksekusi kemampuannya sesuai dengan *job* karakter masing-masing.

**4.2 Implementasi AI GOAP pada *game***

Pada tabel 4.2, terdapat daftar-daftar prosedur yang diimplementasi untuk membuat AI FSM di dalam *game* beserta keterangan dan *flowchart* dari prosedur tersebut.

Tabel 4.2

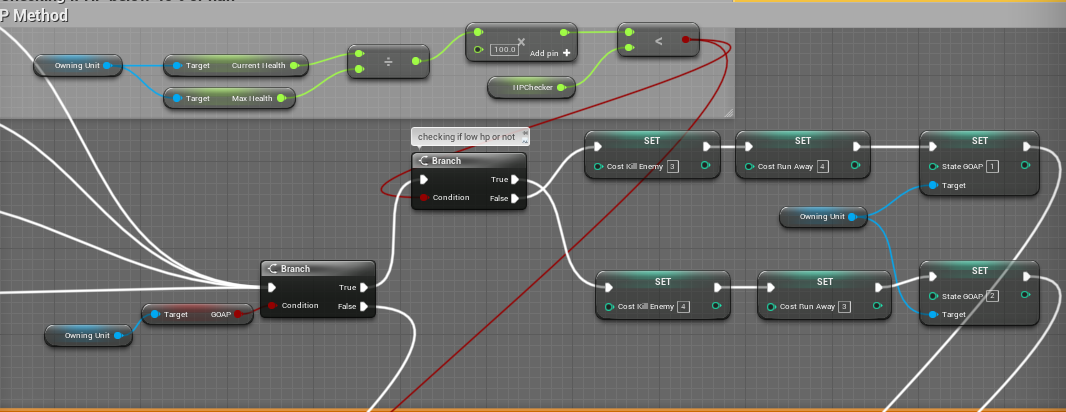
Daftar prosedur yang digunakan untuk AI GOAP

|  |  |  |  |
| --- | --- | --- | --- |
| Segmen Program | Nama Prosedur | Keterangan Prosedur | Gambar Flowchart |
| 4.3 | Mengatur *cost action* | Mengatur *cost* untuk setiap *action* yang ada di dalam AI GOAP, dan mengubah *cost-cost* tersebut di situasi tertentu (*situational costs*) | - |
| 4.4 | Implementasi AI GOAP | Membuat AI GOAP sesuai dengan desain yang ada | Flowchart 3.3 |

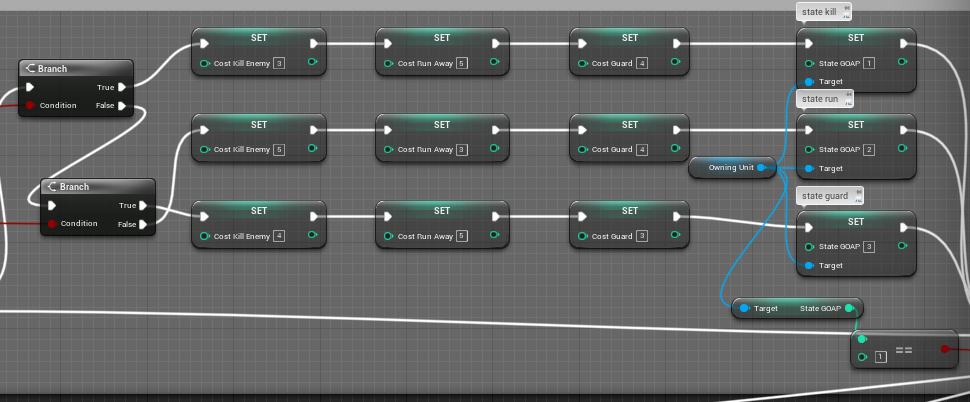
**4.2.1 Implementasi *Cost* pada AI GOAP**

Pada karakter AI GOAP, terdapat empat *action* yang dapat dieksekusi, yaitu *attack, move, guard,* dan *heal*. Ketika HP yang dimiliki AI GOAP diatas 40%, maka AI GOAP akan berusaha menyerang musuh. Dengan begitu, maka AI GOAP akan menjalankan fungsi *move to X* hingga musuh ada di dalam *range attack* AI GOAP. Setelah musuh ada di *range attack* AI GOAP, maka AI GOAP akan menjalankan fungsi *attack*. Tetapi, jika HP yang dimiliki AI GOAP dibawah 40%, maka AI GOAP akan menjalankan fungsi *move to target X* dan *heal*. Semua hal tersebut diatur melalui *cost action* masing-masing sesuai dengan kondisi yang dimiliki AI GOAP. Segmen program ini dapat dilihat pada Segmen 4.3

Segmen 4.3 Mengatur *cost action*



**Segmen 4.3 Mengatur *cost action* (Lanjutan)**

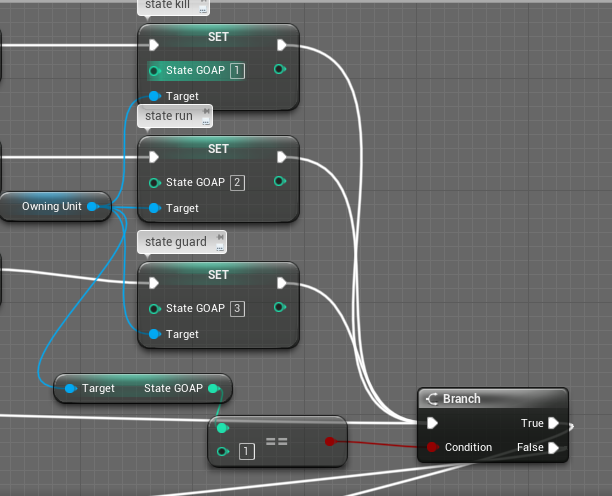


Kemudian, apabila AI GOAP sedang konflik dengan musuh, akan dilakukan pengecekan lagi untuk melakukan *situational costs*. Apabila HP yang dimiliki oleh AI GOAP dibawah 40%, dan HP yang dimiliki oleh AI GOAP lebih besar dari musuh, maka *cost attack* akan direndahkan, sehingga AI GOAP akan melakukan aksi *attack*. Apabila HP yang dimiliki oleh AI GOAP dibawah 40% dan perbedaan HP AI GOAP dan musuh diatas 10, maka AI GOAP akan melakukan *run away from enemy* dan *heal*. Tetapi, apabila perbedaan HP AI GOAP dan musuh dibawah 10, maka AI GOAP akan melakukan *run away* *from enemy* dan *guard*. Selain itu, apabila AI GOAP menemukan musuh yang memiliki nyawa yang lebih kecil di dalam *range* tersebut, maka AI GOAP akan memprioritaskan untuk menyerang musuh tersebut daripada musuh lainnya.

**4.2.2 Implementasi AI GOAP**

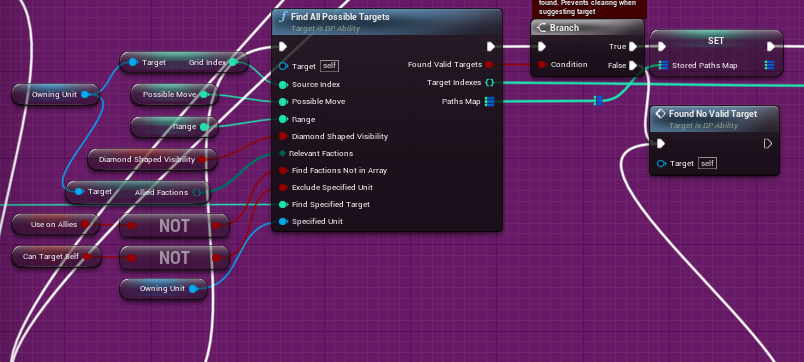
Implementasi AI GOAP merupakan prosedur pembuatan AI GOAP yang digunakan pada penelitian ini untuk menguji apakah AI GOAP mampu melawan AI FSM nantinya. Segmen program ini dapat dilihat di Segmen 4.4

Segmen 4.4 Implementasi AI GOAP



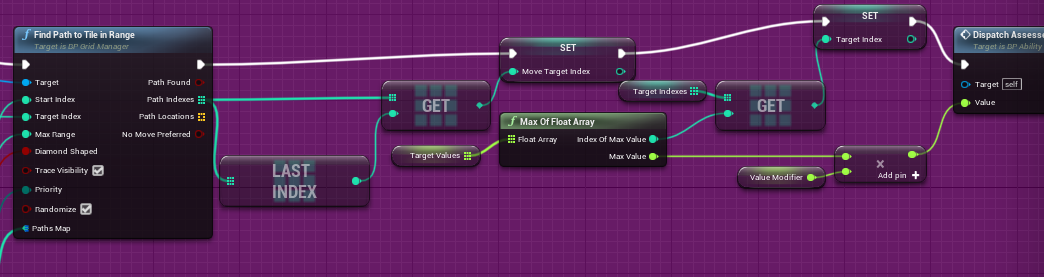
AI GOAP akan menentukan aksi mana yang akan dikerjakan sesuai dengan kondisi AI sekarang.

**Segmen 4.4 Implementasi AI GOAP (Lanjutan)**

****

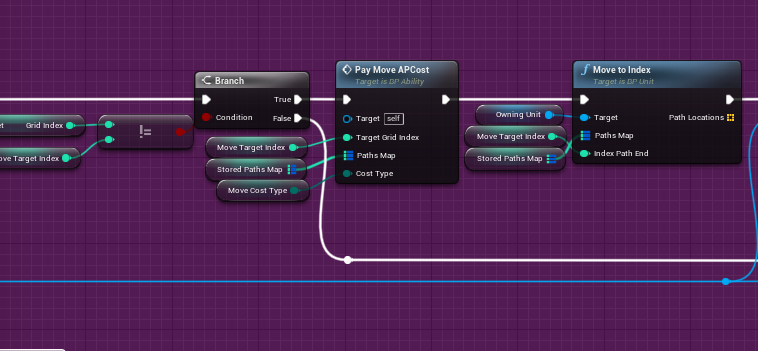
Apabila AI GOAP menentukan untuk menyerang musuh, maka AI GOAP akan mencari musuh terdekat dari posisinya.

**Segmen 4.4 Implementasi AI GOAP (Lanjutan)**



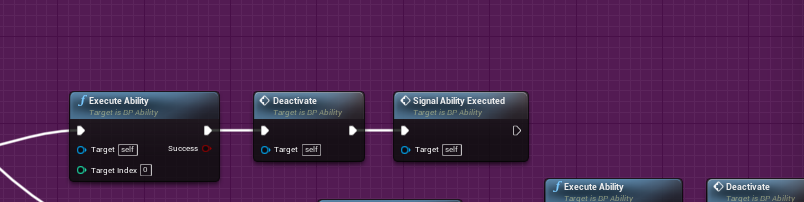
Setelah menemukan musuh terdekat, maka AI GOAP akan mencari *tile* yang paling dekat dengan musuh tersebut sesuai *move range* yang dimiliki karakter. Setelah menemukan *tile* yang paling dekat dengan musuh, maka AI akan mengatur tujuan *tile* ke *tile* tersebut.

**Segmen 4.4 Implementasi AI GOAP (Lanjutan)**



Setelah mengatur tujuan *tile*, AI akan berjalan ke *tile* tersebut. Apabila AP yang dimiliki AI masih 2, maka AI dapat jalan ke *tile* tersebut.

**Segmen 4.4 Implementasi AI GOAP (Lanjutan)**

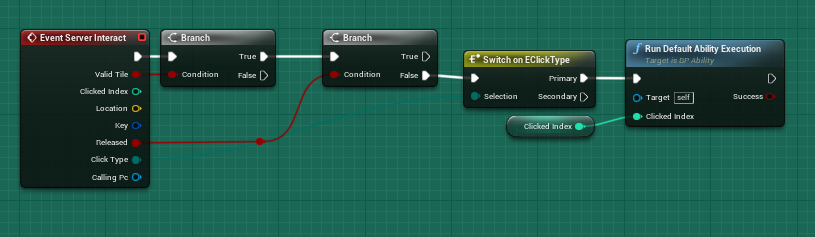


Apabila *range* *attack* milik AI dapat mencapai musuh, maka AI akan mengeksekusi kemampuannya sesuai dengan *job* karakter masing-masing.

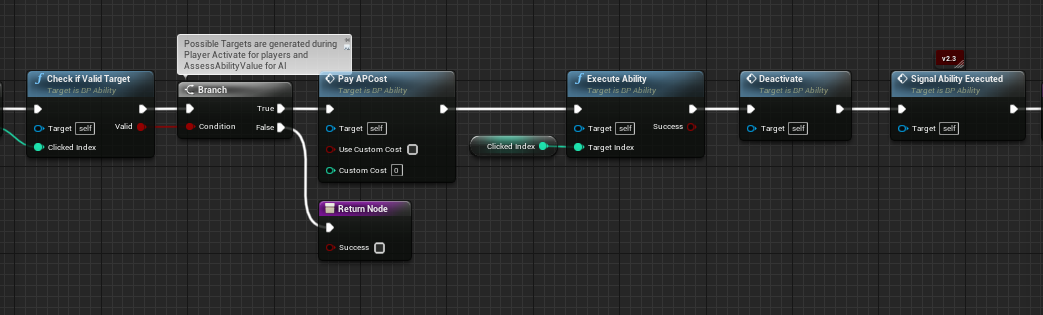
**4.3 Implementasi Player Turn**

Implementasi *Player Turn* digunakan agar *player* bisa menggerakkan karakternya ke *tile* atau musuh yang diinginkan. Segmen program ini dapat dilihat pada Segmen 4.5.

Segmen 4.5 Implementasi Player Turn



**Segmen 4.5 Implementasi Player Turn (Lanjutan)**

****

Ketika *player* memilih *tile* dan kemampuan karakter, maka sistem akan menjalankan karakter tersebut sesuai dengan yang sudah dipilih oleh *player*.

**5. IMPLEMENTASI SISTEM**

Pada bab ini akan dibahas tentang pengujian sistem aplikasi yang telah dibuat. Tahap ini meliputi pengujian terhadap *human player* dan AI *Finite State Machine* melawan AI *Goal-Oriented Action Planning*.

**5.1 Perangkat Lunak yang Digunakan**

Pengerjaan aplikasi menggunakan fitur *blueprint* yang ada pada *Unreal Engine* versi 4.26.2. Aplikasi yang dibuat mencakup AI *Finite State Machine* dan AI *Goal-Oriented Action Planning* untuk mendapatkan hasil performa dari masing-masing AI berdasarkan *winrate* dan *resources unit* yang berbeda.

**5.2 Hasil Pengujian**

Setelah pembuatan aplikasi selesai, dilakukan 2 tahap pengujian di aplikasi ini. Pengujian pertama pada skripsi ini dilakukan melalui survei yang diambil oleh 10 orang yang sudah selesai mencoba aplikasi ini. Dari tahap hasil pengujian pertama, dapat ditemukan rata-rata kepuasan pemain ketika melawan AI GOAP dan FSM, bagaimana tingkat kesulitan AI GOAP dan FSM, serta seberapa realistis gerakan yang dilakukan oleh AI GOAP dan FSM.

Tahap pengujian kedua dilakukan melalui AI *Finite State Machine* melawan AI *Goal-Oriented Action Planning*. Pengujian dilakukan dimana tiap AI menggunakan *resources* yang berbeda-beda, dimana AI *Finite State Machine* memiliki *resources* yang lebih daripada AI *Goal Oriented Action Planning*. Hasil dari pengujian kedua dapat ditemukan *winrate* AI GOAP melawan *AI FSM* dengan *resources* dan putaran yang berbeda tiap pengujian.

**5.3 Tahap Pengujian**

Pengujian pertama dilakukan untuk mendapatkan hasil kepuasan *human player* yang sudah selesai mencoba aplikasi ini dan mengambil survei setelahnya sebanyak 10 orang. Tiap orang memiliki *resources* yang berbeda untuk melawan AI *Goal-Oriented Action Planning*. Survei yang diisi berisikan skala 1-5 dalam 4 hal, yaitu seberapa *enjoy* pemain ketika melawan AI yang dilawan, seberapa sulit AI yang dilawan di dalam aplikasi ini, dan seberapa realistis gerakan yang dilakukan oleh AI yang dilawan. Dari hasil survei ini, dapat ditemukan kepuasan pemain ketika melawan AI GOAP dan AI FSM yang memiliki *resources* yang lebih.

Pengujian kedua dilakukan untuk mendapatkan performa AI *Goal-Oriented Action Planning* berdasarkan *winrate* ketika melawan AI *Finite State Machine* yang memiliki *resources* yang lebih dari AI *Goal-Oriented Action Planning*. *Resources* yang dimaksud adalah jumlah *unit*, *power* unit, dan *personality* yang dimiliki oleh AI *Finite State Machine*. Pengujian dilakukan sebanyak 12 kali. Pengujian dibagi menjadi 6 kelompok, dimana tiap kelompok terdapat 2 kali percobaan. Perbedaan pengujian yang terdapat di tiap kelompok terdapat pada siapa yang mendapatkan putaran pertama. Perbedaan satu kelompok dengan kelompok yang lain adalah *unit* yang digunakan tiap AI.

**5.3.1 Pengujian Kelompok A**

Pada kelompok ini, *unit* yang digunakan oleh AI *Finite State Machine* ada 5, yaitu, 2 *Warrior*, 2 *Archer*, dan 1 *Healer*, serta *personality aggressive*. *Unit* yang digunakan oleh AI *Goal-Oriented Action Planning* ada 4, yaitu 1 *Warrior*, 2 *Archer*, dan 1 *Healer*. Pada pengujian pertama, putaran pertama diberikan kepada AI GOAP. Pada pengujian kedua, putaran pertama diberikan kepada AI FSM. Hasil data kelompok A dapat diamati pada Tabel 5.1.

Tabel 5.1

Tabel Hasil Pengujian Kelompok A

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Pengujian ke-** | **Unit Awal GOAP** | **Unit awal FSM & Personality** | **Putaran Pertama** | **Kemenangan** | **Sisa Unit Pemenang** |
| 1 | 1 Warrior, 2 Archer, 1 Healer | 2 Warrior, 2 Archer, 1 Healer (Personality = 1) | GOAP | GOAP | 1 Archer, 1 Warrior, 1 Healer |
| 2 | 1 Warrior, 2 Archer, 1 Healer | 2 Warrior, 2 Archer, 1 Healer (Personality = 1) | FSM | FSM | 2 Warrior, 1 Archer |
| **Jumlah Kemenangan GOAP** | | | | 1 | |
| **Winrate** | | | | 50% | |

Pada 2 kali pengujian, didapatkan *winrate* sebesar 50%. Pada pengujian pertama, ketika AI GOAP mendapatkan putaran pertama, kemenangan diraih oleh AI GOAP. Di pengujian ini, AI FSM mampu mengeliminasi unit yang dimiliki oleh AI GOAP sebesar 25%. Pada pengujian kedua, ketika AI FSM mendapatkan putaran pertama, kemenangan diraih oleh AI FSM. Di pengujian ini, AI GOAP mampu mengeliminasi unit FSM sebanyak 40%.

**5.3.2 Pengujian Kelompok B**

Pada kelompok ini, *unit* yang digunakan oleh AI *Finite State Machine* ada 5, yaitu, 2 *Warrior*, 2 *Archer*, dan 1 *Healer*, personality *aggressive*, dengan catatan *power* yang dimiliki oleh beberapa *unit* dikurangi. *Power* yang direndahkan berupa HP *Warrior* dari 100 menjadi 80, HP *Archer* dari 60 menjadi 80, dan *attack* *Archer* dari 30 menjadi 20. *Unit* yang digunakan oleh AI *Goal-Oriented Action Planning* ada 4, yaitu 1 *Warrior*, 2 *Archer*, dan 1 *Healer*, tanpa ada pengurangan *power unit*. Pada pengujian ketiga, putaran pertama diberikan kepada AI GOAP. Pada pengujian keempat, putaran pertama diberikan kepada AI FSM. Hasil data kelompok B dapat diamati pada Tabel 5.2.

Tabel 5.2

Tabel Hasil Pengujian Kelompok B

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Pengujian ke-** | **Unit Awal GOAP** | **Unit awal FSM & Personality** | **Putaran Pertama** | **Kemenangan** | **Sisa Unit Pemenang** |
| 3 | 1 Warrior, 2 Archer, 1 Healer | 2 Warrior, 2 Archer, 1 Healer(Personality = 1) | GOAP | GOAP | 1 Archer, 1 Warrior, 1 Healer |
| 4 | 1 Warrior, 2 Archer, 1 Healer | 2 Warrior, 2 Archer, 1 Healer(Personality = 1) | FSM | GOAP | 1 Archer, 1 Warrior, 1 Healer |
| **Jumlah Kemenangan GOAP** | | | | 2 | |
| **Winrate** | | | | 100% | |

Pada 2 kali pengujian, didapatkan *winrate* sebesar 100%. Pada pengujian ketiga, ketika AI GOAP mendapatkan putaran pertama, kemenangan diraih oleh AI GOAP. Di pengujian ini, AI FSM mampu mengeliminasi *unit* yang dimiliki oleh AI GOAP sebesar 25%. Pada pengujian keempat, meski AI GOAP tidak mendapatkan putaran pertama, AI GOAP mampu meraih kemenangan terhadap AI FSM. Di pengujian ini, AI FSM memiliki hasil yang sama seperti pengujian sebelumnya, dimana AI FSM mampu mengeliminasi *unit* yang dimiliki oleh AI GOAP sebesar 25%.

**5.3.3 Pengujian Kelompok C**

Pada kelompok ini, *unit* yang digunakan oleh AI *Finite State Machine* ada 5, yaitu, 2 *Warrior*, 2 *Archer*, dan 1 *Mage*, serta *personality neutral*. *Unit* yang digunakan oleh AI *Goal-Oriented Action Planning* ada 4, yaitu 2 *Warrior*, 1 *Archer*, dan 1 *Mage*. Pada pengujian kelima, putaran pertama diberikan kepada AI GOAP. Pada pengujian keenam, putaran pertama diberikan kepada AI FSM. Hasil data kelompok C dapat diamati pada Tabel 5.3.

Tabel 5.3

Tabel Hasil Pengujian Kelompok C

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Pengujian ke-** | **Unit Awal GOAP** | **Unit awal FSM & Personality** | **Putaran Pertama** | **Kemenangan** | **Sisa Unit Pemenang** |
| 5 | 2 Warrior, 1 Archer, 1 Mage | 2 Warrior, 2 Archer, 1 Mage (Personality = 2) | GOAP | FSM | 1 Warrior, 1 Mage, 2 Archer |
| 6 | 2 Warrior, 1 Archer, 1 Mage | 2 Warrior, 2 Archer, 1 Mage (Personality = 2) | FSM | FSM | 1 Mage |
| **Jumlah Kemenangan GOAP** | | | | 0 | |
| **Winrate** | | | | 0% | |

Pada 2 kali pengujian, didapatkan *winrate* sebesar 0%. Pada pengujian kelima, meski AI GOAP mendapatkan putaran pertama, kemenangan diraih oleh AI FSM. Di pengujian ini, AI GOAP hanya mampu mengeliminasi *unit* AI FSM sebesar 20%. Pada pengujian keenam, kemenangan diraih oleh AI FSM. Tetapi di pengujian ini, AI GOAP mampu mengeliminasi *unit* yang dimiliki oleh AI FSM sebanyak 80%.

**5.3.4 Pengujian Kelompok D**

Pada kelompok ini, *unit* yang digunakan oleh AI *Finite State Machine* ada 5, yaitu, 2 *Warrior*, 2 *Archer*, dan 1 *Mage*, personality *neutral*, dengan catatan *power* yang dimiliki oleh beberapa *unit* dikurangi. *Power* yang dikurangi berupa HP *Warrior* dari 100 menjadi 80, HP *Archer* dari 60 menjadi 80, dan *attack* *Archer* dari 30 menjadi 20. *Unit* yang digunakan oleh AI *Goal-Oriented Action Planning* ada 4, yaitu 2 *Warrior*, 1 *Archer*, dan 1 *Mage*, tanpa ada pengurangan *power unit*. Pada pengujian ketujuh, putaran pertama diberikan kepada AI GOAP. Pada pengujian kedelapan, putaran pertama diberikan kepada AI FSM. Hasil data kelompok D dapat diamati pada Tabel 5.4.

Tabel 5.4

Tabel Hasil Pengujian Kelompok D

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Pengujian ke-** | **Unit Awal GOAP** | **Unit awal FSM & Personality** | **Putaran Pertama** | **Kemenangan** | **Sisa Unit Pemenang** |
| 7 | 2 Warrior, 1 Archer, 1 Mage | 2 Warrior, 2 Archer, 1 Mage (Personality = 2) | GOAP | GOAP | 1 Archer, 1 Warrior, 1 Mage |
| 8 | 2 Warrior, 1 Archer, 1 Mage | 2 Warrior, 2 Archer, 1 Mage (Personality = 2) | FSM | FSM | 2 Warrior, 1 Mage |
| **Jumlah Kemenangan GOAP** | | | | 1 | |
| **Winrate** | | | | 50% | |

Pada 2 kali pengujian, didapatkan *winrate* sebesar 50%. Pada pengujian ketujuh, ketika AI GOAP mendapatkan putaran pertama, kemenangan diraih oleh AI GOAP. Di pengujian ini, AI FSM mampu mengeliminasi *unit* yang dimiliki oleh AI GOAP sebesar 25%. Pada pengujian kedelapan, kemenangan diraih oleh AI FSM. Di pengujian ini, AI GOAP mampu mengeliminasi *unit* yang dimiliki oleh AI FSM sebanyak 40%.

**5.3.5 Pengujian Kelompok E**

Pada kelompok ini, *unit* yang digunakan oleh AI *Finite State Machine* ada 5, yaitu, 2 *Warrior*, 2 *Archer*, dan 1 *Mage*, serta *personality non-aggressive*. *Unit* yang digunakan oleh AI *Goal-Oriented Action Planning* ada 4, yaitu 1 *Warrior*, 2 *Archer*, dan 1 *Mage*. Pada pengujian kesembilan, putaran pertama diberikan kepada AI GOAP. Pada pengujian kesepuluh, putaran pertama diberikan kepada AI FSM. Hasil data kelompok E dapat diamati pada Tabel 5.5.

Tabel 5.5

Tabel Hasil Pengujian Kelompok E

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Pengujian ke-** | **Unit Awal GOAP** | **Unit awal FSM & Personality** | **Putaran Pertama** | **Kemenangan** | **Sisa Unit Pemenang** |
| 9 | 1 Warrior, 2 Archer, 1 Mage | 2 Warrior, 2 Archer, 1 Mage (Personality = 3) | GOAP | FSM | 1 Warrior, 1 Mage, 2 Archer |
| 10 | 1 Warrior, 2 Archer, 1 Mage | 2 Warrior, 2 Archer, 1 Mage (Personality = 3) | FSM | GOAP | 1 Warrior, 1 Mage, 2 Archer |
| **Jumlah Kemenangan GOAP** | | | | 1 | |
| **Winrate** | | | | 50% | |

Pada 2 kali pengujian, didapatkan *winrate* sebesar 50%. Pada pengujian sembilan, meski AI GOAP memiliki putaran pertama, kemenangan diraih oleh AI FSM. Di pengujian ini, AI GOAP mampu mengeliminasi *unit* AI FSM sebesar 20%. Pada pengujian kesepuluh, meski AI FSM memiliki putaran pertama, kemenangan diraih oleh AI GOAP. Di pengujian ini, AI FSM tidak mampu mengeliminasi *unit* yang dimiliki oleh AI GOAP, sehingga AI GOAP memiliki kemenangan telak.

**5.3.6 Pengujian Kelompok F**

Pada kelompok ini, *unit* yang digunakan oleh AI *Finite State Machine* ada 5, yaitu, 2 *Warrior*, 2 *Archer*, dan 1 *Mage*, personality *non-aggressive*, dengan catatan *power* yang dimiliki oleh beberapa *unit* dikurangi. *Power* yang dikurangi berupa HP *Warrior* dari 100 menjadi 80, HP *Archer* dari 60 menjadi 80, dan *attack* *Archer* dari 30 menjadi 20. *Unit* yang digunakan oleh AI *Goal-Oriented Action Planning* ada 4, yaitu 1 *Warrior*, 2 *Archer*, dan 1 *Mage*, tanpa ada pengurangan *power unit*. Pada pengujian kesebelas, putaran pertama diberikan kepada AI GOAP. Pada pengujian keduabelas, putaran pertama diberikan kepada AI FSM. Hasil data kelompok F dapat diamati pada Tabel 5.6.

Tabel 5.6

Tabel Hasil Pengujian Kelompok F

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Pengujian ke-** | **Unit Awal GOAP** | **Unit awal FSM & Personality** | **Putaran Pertama** | **Kemenangan** | **Sisa Unit Pemenang** |
| 11 | 1 Warrior, 2 Archer, 1 Mage | 2 Warrior, 2 Archer, 1 Mage (Personality = 3) | GOAP | GOAP | 2 Archer, 1 Mage |
| 12 | 1 Warrior, 2 Archer, 1 Mage | 2 Warrior, 2 Archer, 1 Mage (Personality = 3) | FSM | GOAP | 1 Warrior, 1 Mage, 2 Archer |
| **Jumlah Kemenangan GOAP** | | | | 2 | |
| **Winrate** | | | | 100% | |

Pada 2 kali pengujian, didapatkan *winrate* sebesar 100%. Pada pengujian kesebelas, kemenangan diraih oleh GOAP. Di pengujian ini, AI FSM mampu mengeliminasi *unit* AI GOAP sebesar 25%. Pada pengujian keduabelas, meski AI FSM memiliki putaran pertama, kemenangan diraih oleh AI GOAP. Di pengujian ini, AI FSM tidak mampu mengeliminasi *unit* yang dimiliki oleh AI GOAP, sehingga AI GOAP memiliki kemenangan telak.

**5.3.7 Pengujian Human Player terhadap AI GOAP**

Pengujian Human Player vs AI GOAP dilakukan dimana *human player* dapat memilih *unit* yang diinginkan maksimal 4 *unit*, melawan AI GOAP yang memiliki 4 *unit* juga. Pengujian dilakukan kepada 10 *human player* yang berbeda. Survei dilakukan setelah *human player* selesai melakukan uji coba aplikasi melawan AI GOAP. Pertanyaan yang diajukan berupa pertanyaan berbentuk skala dari 1 sampai 5 dengan tiga buah pertanyaan, yaitu tingkat kepuasan *human player* melawan AI GOAP, seberapa realistis gerakan AI GOAP, dan seberapa susah AI GOAP di dalam aplikasi tersebut. Setelah itu, *human player* juga diarahkan untuk melawan AI FSM yang memiliki *5 unit* yang memiliki personality yang berbeda. Setelah itu, akan diadakan survei yang sama untuk AI FSM. Hasil jawaban tentang AI GOAP dapat dilihat pada Tabel 5.7, dan hasil jawaban tentang AI FSM dapat dilihat pada Tabel 5.8.

Tabel 5.7

Tabel Hasil Jawaban *Human Player* tentang AI GOAP

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Email** | **Tingkat Kesulitan AI** | **Tingkat Kepuasan saat Melawan AI** | **Tingkat Kerealistisan AI** |
| 1 | luckytanjung10@gmail.com | 3 | 4 | 4 |
| 2 | [fairyxcution@gmail.com](mailto:fairyxcution@gmail.com) | 4 | 3 | 4 |
| 3 | gilang9116@gmail.com | 4 | 4 | 4 |
| 4 | jendratno@student.ciputra.ac.id | 4 | 3 | 3 |
| 5 | mamansuparman1515@gmail.com | 3 | 4 | 3 |
| 6 | christianstevan99@gmail.com | 4 | 4 | 4 |
| 7 | bumblebee.15hbr@gmail.com | 2 | 5 | 4 |
| 8 | c14170040@john.petra.ac.id | 3 | 5 | 3 |
| 9 | c14170023@john.petra.ac.id | 3 | 5 | 4 |
| 10 | albertochrissandy@gmail.com | 4 | 5 | 3 |
| Rata-rata | | 3.4 | 4.2 | 3.6 |

Dari 10 responden yang menjawab pertanyaan tentang AI GOAP, di bagian tingkat kesulitan, terdapat 1 responden yang menjawab 2 poin, 4 responden yang menjawab 3 poin, dan 5 responden yang menjawab 4 poin. Di tingkat kepuasan pemain, terdapat 2 responden yang menjawab 3 poin, 4 responden yang menjawab 4 poin, dan 4 responden yang menjawab 4 poin. Di tingkat kerealistisan AI, terdapat 4 responden yang menjawab 3 poin, dan 6 responden yang menjawab 4 poin.

Tabel 5.8

Tabel Hasil Jawaban *Human Player* tentang AI FSM

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Email** | **Tingkat Kesulitan AI** | **Tingkat Kepuasan saat Melawan AI** | **Tingkat Kerealistisan AI** |
| 1 | luckytanjung10@gmail.com | 4 | 3 | 3 |
| 2 | [fairyxcution@gmail.com](mailto:fairyxcution@gmail.com) | 5 | 3 | 3 |
| 3 | gilang9116@gmail.com | 5 | 3 | 4 |
| 4 | jendratno@student.ciputra.ac.id | 4 | 3 | 3 |
| 5 | mamansuparman1515@gmail.com | 4 | 3 | 3 |
| 6 | christianstevan99@gmail.com | 4 | 3 | 3 |
| 7 | bumblebee.15hbr@gmail.com | 3 | 4 | 4 |
| 8 | c14170040@john.petra.ac.id | 3 | 4 | 3 |
| 9 | c14170023@john.petra.ac.id | 4 | 3 | 3 |
| 10 | albertochrissandy@gmail.com | 5 | 4 | 3 |
| Rata-rata | | 4.1 | 3.3 | 3.2 |

Dari 10 responden yang menjawab pertanyaan tentang AI FSM, di bagian tingkat kesulitan, terdapat 2 responden yang menjawab 3 poin, 5 responden yang menjawab 4 poin, dan 3 responden yang menjawab 5 poin. Di tingkat kepuasan pemain, terdapat 7 responden yang menjawab 3 poin, dan 3 responden yang menjawab 4 poin Di tingkat kerealistisan AI, terdapat 8 responden yang menjawab 3 poin, dan 2 responden yang menjawab 4 poin.

**6. KESIMPULAN DAN SARAN**

**6.1 Kesimpulan**

Berdasarkan hasil pengujian AI *Goal-Oriented Action Planning* melawan AI *Finite State Machine*, *Human Player* melawan AI *Goal-Oriented Action Planning* dan *Finite State Machine*, dapat disimpulkan beberapa hal diantaranya :

* AI *Goal-Oriented Action Planning* yang terdapat di penelitian ini memiliki aksi yang disesuaikan, yaitu apabila terdapat musuh di *range* yang memiliki nyawa lebih kecil, maka AI GOAP akan memprioritaskan musuh tersebut. Apabila AI *Goal-Oriented Action Planning* dan musuh AI dalam kondisi *low health*, maka akan dilakukan pengecekan berdasarkan perbedaan HP. Apabila perbedaan HP > 10, maka AI akan memprioritaskan untuk melakukan *run away* kemudian melakukan *guard*. Tetapi, apabila perbedaan HP <= 10, maka AI *Goal-Oriented Action Planning* akan memprioritaskan untuk menyerang musuh tersebut.
* Dari perbandingan hasil survei yang dilakukan *human player* setelah melawan AI GOAP dan AI FSM, bisa disimpulkan bahwa AI FSMlebih sulit daripada AI GOAP. Selain itu, tingkat kepuasan pemain juga lebih tinggi ketika melawan AI GOAP daripada melawan AI FSM, dan kerealistisan AI GOAP lebih tinggi daripada AI FSM.
* Dari 12 kali pengujian AI GOAP melawan AI FSM, didapatkan *winrate* AI GOAP sebesar 58.33%, sedangkan AI FSM memiliki *winrate* sebesar 41.66%. Dari pengujian-pengujian tersebut, meski AI GOAP memiliki kesempatan mendapatkan putaran pertama, AI GOAP tetap bisa kalah melawan AI FSM, begitu juga sebaliknya. Selain itu, kemenangan AI GOAP juga lebih sering didapatkan ketika AI FSM memiliki jumlah *unit* yang lebih banyak, tetapi dengan *power* yang dikurangi. Karena *winrate* yang dimiliki AI GOAP tidak terlalu besar, maka AI GOAP masih belum mampu untuk menggantikan AI FSM di dalam *Turn-based Tactics*.

**6.2 Saran**

Saran yang dapat digunakan untuk penelitian serupa untuk lebih lanjut, antara lain :

* *Unit-unit* yang ada *power*nya harus diimbangkan lagi untuk mencegah ketidaksetaraan *power* antar *unit*, dan menambahkan kemampuan lebih banyak untuk tiap karakternya.
* Menggunakan skenario lebih banyak untuk AI GOAP dalam melawan AI FSM.

DAFTAR PUSTAKA

# Bibliography

Chandra, V. A. (2018). Perbandingan Performa Turn-based Game Menggunakan Algoritma Genetika dan Logika Fuzzy. Diambil kembali dari https://dewey.petra.ac.id/catalog/digital/detail?id=42374

Dalem, I. B. (2018). Penerapan Algoritma A\* (Star) Menggunakan Graph Untuk Menghitung Jarak Terpendek. *Jurnal RESISTOR (Rekayasa Sistem Komputer), vol 1, no 1*, 41-47. doi:10.31598/jurnalresistor.v1i1.253

Hall, T., & Magnusson, M. (2010). Adaptive Goal Oriented Action Planning for RTS Games. Diambil kembali dari https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A831698&dswid=-5330

Karlsen, F. (2011). Entrapment and Near Miss: A Comparative Analysis of Psycho-Structural Elements in Gambling Games and Massively Multiplayer Online Role-Playing Games. *International Journal of Mental Health and Addiction, vol 9, no 2*, 193-207. doi:10.1007/s11469-010-9275-4

Korhonen, H., & Koivisto, E. M. (2006). Playability Heuristics for Mobile Games. *ACM International Conference Proceeding Series, vol 159*, 9-16. doi:10.1145/1152215.1152218

Mathew, G. E. (2015). Direction based heuristic for pathfinding in video games. *Procedia Computer Science, vol 47*, 262-271. doi:10.1016/j.procs.2015.03.206

Orkin, J. (2003). Applying Goal-Oriented Action Planning to Games. *AI Game Programming Wisdom 2*, 217-227.

Owens, B. (2014). *Goal Oriented Action Planning for a Smarter AI*. Received from https://gamedevelopment.tutsplus.com/en/tutorials/goal-oriented-action-planning-for-a-smarter-ai--cms-20793

Rahadian, M. F., Suyatno, A., & Maharani, S. (2016). Penerapan Metode Finite State Machine Pada Game “The Relationship”. *Informatika Mulawarman : Jurnal Ilmiah Ilmu Komputer, vol 11, no 1*, 14-21. doi:10.30872/jim.v11i1.198

Rostianingsih, S., Budhi, G. S., & Wijaya, H. K. (2013). Game Simulasi Finite State Machine untuk Pertanian dan Peternakan. Received from http://repository.petra.ac.id/16408/

Sofyan, I. A., Akbar, M. A., & Afirianto, T. (2019). Implementasi Dynamic Difficulty Adjustment Pada Racing Game Menggunakan Metode Behaviour Tree. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer (J-PTIIK) Universitas Brawijaya, vol 3, no 1*, 643-650.

Studiawan, R., Hariadi, M., & Sumpeno, S. (2018). Tactical Planning in Space Game using Goal-Oriented Action Planning. *JAREE (Journal on Advanced Research in Electrical Engineering), vol 2, no 1*, 5-11. doi:10.12962/j25796216.v2.i1.32