**BAB 3. ANALISIS DAN DESAIN SISTEM**

**3.1 Desain Program**

Jenis program yang dibuat adalah *game* bergenre *Turn-based Strategy*. *Game* akan dijalankan dengan 2 *mode*, yaitu *mode player* melawan AI GOAP, dan AI FSMmelawan AI GOAP. 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 kubu 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.

**3.1.1 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.
* 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.
* Type : Tipe serangan yang dimiliki *unit*. Ada 2 jenis, yaitu *melee* dan *range*.

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*

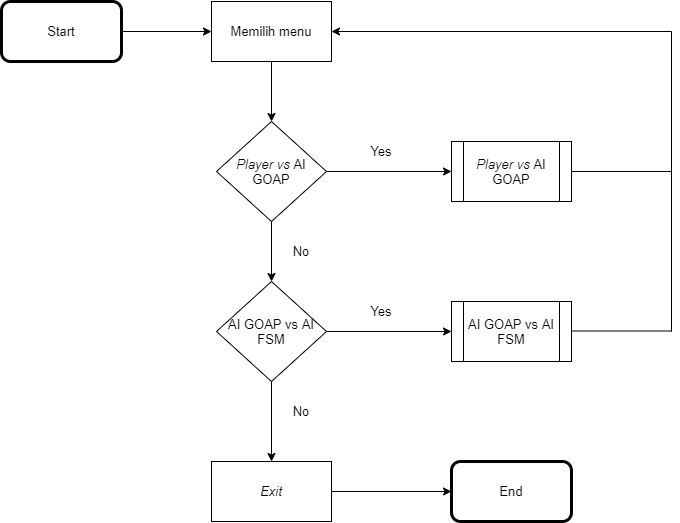
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tipe / Atribut | Health | Damage | Range | Move |
| *Ranged Unit* | 80 | 20 | 6 | 4 |
| *Melee Unit* | 110 | 30 | 1 | 6 |
| *Healer* | 65 | 0 | 4 | 3 |

**3.1.2 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.

**3.1.3 Desain *Main Menu***

Pada saat pertama kali masuk ke dalam *game*, *user* akan diberikan 3 pilihan, yaitu *player* vs AI, AI FSM vs AI GOAP, dan *exit*. Jika *user* memilih *player* vs AI, maka *user* dapat bermain melawan AI GOAP dengan fitur-fitur yang ada di dalam *game*. Jika *user* memilih AI FSM vs AI GOAP, maka *user* dapat melihat pertandingan antara AI FSM melawan AI GOAP, dan jika *user* memilih *exit*, maka *program* akan otomatis tertutup. Berikut adalah *flowchart* alur *program*.



Gambar 3.1 *Flowchart* menu

**3.1.4 Tampilan Main Menu**

Tampilan *main* menu yang ada di dalam program ini terdiri dari text dan *button* untuk memilih menu seperti *Player* vs AI GOAP, AI FSM vs AI GOAP, dan *exit*. Berikut adalah gambar tampilan *menu* secara garis besar.

**TURN-BASED STRATEGY WITH AI FSM AND AI GOAP**

**EXIT**

**AI FSM VS AI GOAP**

**PLAYER VS AI GOAP**

Gambar 3.2 Tampilan *main* menu

**3.1.5 Tampilan Gameplay**

Ketika *user* memilih *player* vs AI GOAP, *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* apabila *user* sudah selesai melakukan gerakannya.

**gameplay**

COMMAND LIST

DESCRIPTION

END TURN

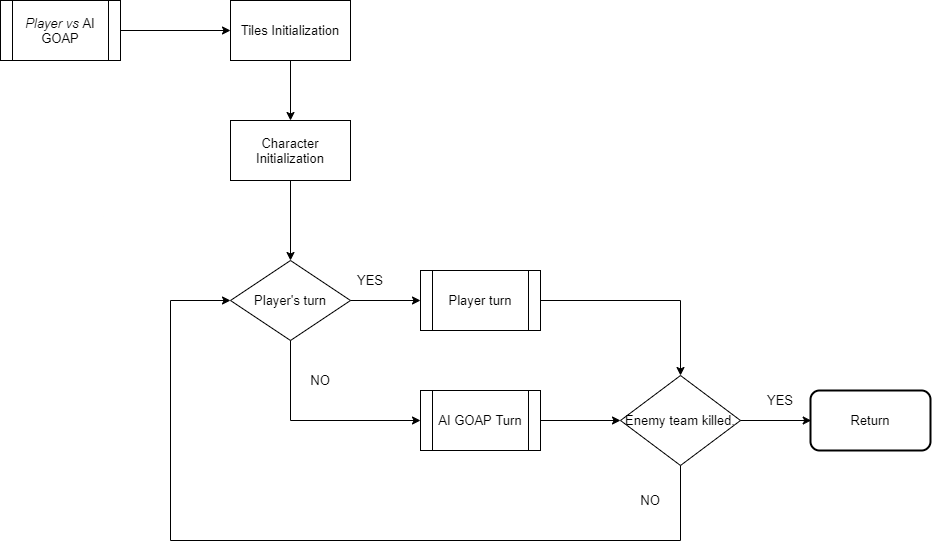
Gambar 3.3 Tampilan *Gameplay*

**3.2 Alur *Game***

**3.2.1 Alur *Player* vs AI GOAP**

Setelah *user* memilih *player vs* AI GOAP di dalam *main* menu, *program* akan melakukan inisialisasi *game*, yaitu *tiles initialization* dan *character initialization*. Setelah inisialisasi selesai, maka sistem mengecek apakah sekarang giliran *player* atau bukan. Jika iya, maka akan masuk ke proses *player turn*. Jika tidak, maka sistem akan memasuki proses AI GOAP *turn*. Ketika *player turn* atau AI GOAP *turn* selesai, maka mereka akan mengecek, apakah *unit* pasukan musuh sudah terbunuh semua atau belum. Jika belum, maka akan kembali mengecek apakah sekarang giliran *player* lagi atau tidak.

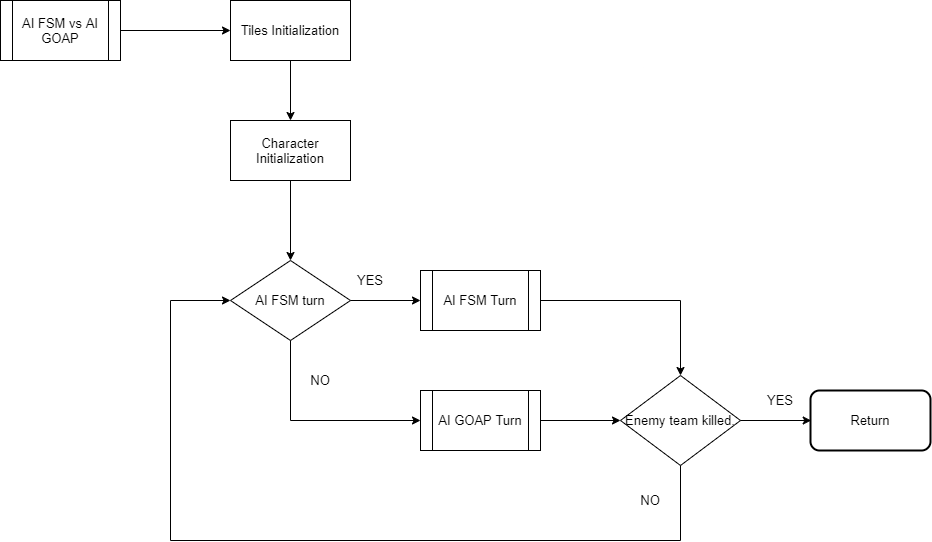
Berikut adalah *flowchart* bagaimana putaran di dalam alur *player* vs AI GOAP berjalan.



Gambar 3.4 Alur di dalam *player* vs AI GOAP

**3.2.2 Alur AI FSM vs AI GOAP**

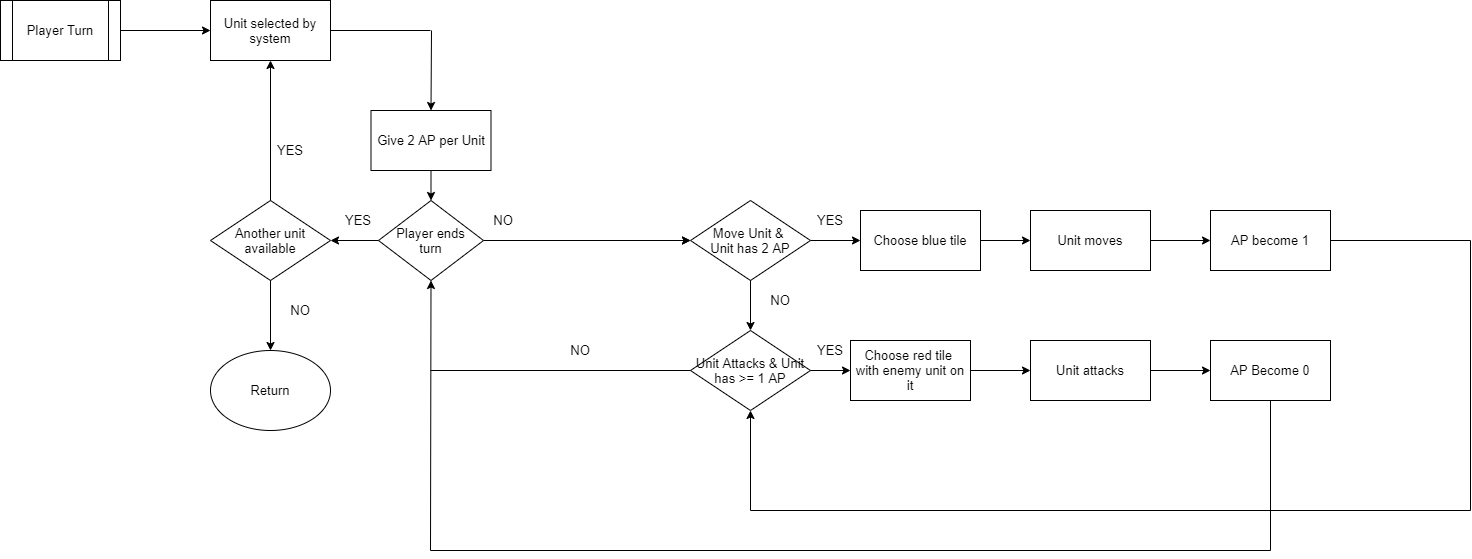
Setelah *user* memilih AI FSM *vs* AI GOAP di dalam *main* menu, *program* akan melakukan inisialisasi *game*, yaitu *tiles initialization* dan *character initialization*. Setelah inisialisasi selesai, maka sistem mengecek apakah sekarang giliran AI FSM atau bukan. Jika iya, maka akan masuk ke proses AI FSM *turn*. Jika tidak, maka sistem akan memasuki proses AI GOAP *turn*. Ketika AI FSM *turn* atau AI GOAP *turn* selesai, maka mereka akan mengecek, apakah *unit* pasukan musuh sudah terbunuh semua atau belum. Jika belum, maka akan kembali mengecek apakah sekarang giliran AI FSM lagi atau tidak. Berikut adalah *flowchart* bagaimana putaran di dalam alur AI FSMvs AI GOAP berjalan.



Gambar 3.5 Alur di dalam AI FSM vs AI GOAP

**3.2.3 *Player Turn***

Setelah memasuki *player* vs AI GOAP*,* 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 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. 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 3.6 *Flowchart* putaran *player*

**3.2.4 AI GOAP Turn**

Setelah memasuki *player* vs AI GOAP*,* 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

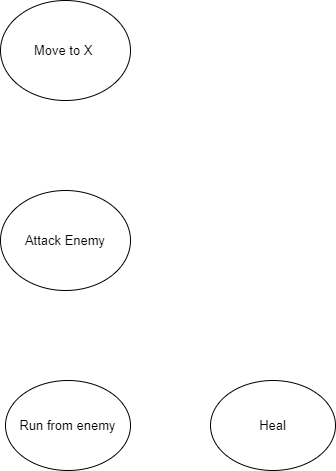
*Action* yang ada pada AI GOAP

|  |  |  |
| --- | --- | --- |
| 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 | Is not on low health |
| Run from enemy | Run from enemy X | Is on low health |
| Heal | Is not on low health | Is on low health |

Tabel 3.3

*Goals* yang ada pada AI GOAP

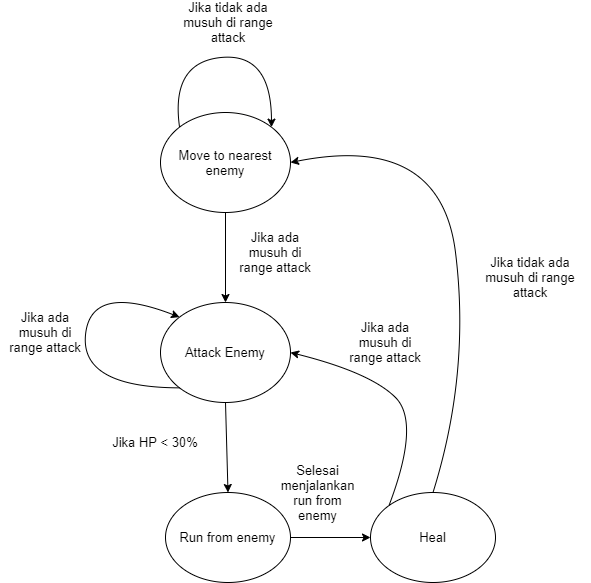
|  |  |
| --- | --- |
| Goals | Desired World State |
| Kill enemy | Attacking target X |
| Recover | Run from enemy X  Is on low health |



Gambar 3.7 State AI GOAP

**3.2.5 AI FSM Turn**

Setelah memasuki AI FSM vs AI GOAP*,* 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 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. Berikut adalah *state* AI FSM.



Gambar 3.8 *State* dari AI FSM

**3.4 Desain Pengujian**

Metode pengujian yang digunakan untuk penelitian ini yaitu dengan menggunakan AI FSM untuk melawan AI GOAP, dan membuat *player* melawan AI GOAP. Pengujian AI FSM vs AI GOAP akan dilakukan sebanyak 6 kali denganpasukan yang berbeda-beda, dengan giliran dimana AI GOAP mendapatkan giliran pertama sebanyak 3 kali, dan AI FSM mendapatkan giliran pertama sebanyak 3 kali. Pada pengujian tersebut, akan dicatat ukuran-ukuran sebagai berikut :

1. Langkah-langkah yang diambil tiap AI.
2. 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.
3. *Unit* yang dimiliki tiap-tiap AI.

Pengujian *player* vs AI GOAP akan dilakukan sebanyak 6 kali. 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.