BAB 3

METODOLOGI PENELITIAN

3.1 Metodologi Penelitian

Metodelogi yang digunakan pada penelitian ini mempunyai beberapa tahapan yang harus dilakukan, yaitu:

a. Observasi

   Pada tahap observasi, dilakukan analisis terhadap permasalahan yang sedang diteliti, yaitu penggunaan PCG untuk *game open world* dengan menggunakan algoritma *Markov Chain*.

b. Telaah Literatur

   Pada telaah literatur, dilakukan pengumpulan informasi mengenai *game* secara general, lalu dilanjutkan dengan mengumpulkan informasi mengenai *video game*. Kemudian dilanjutkan dengan mencari informasi mengenai PCG dan algoritma *Markov Chain* yang berfungsi untuk melakukan generasi konten yang ada pada *video game*. Lalu dilanjutkan dengan mencari informasi mengenai GUESS untuk mengukur tingkat kepuasa pemain pada *video game*.

c. Analisi dan Perancangan *Game*

   Pada tahap ini, dilakukan analisis serta perancangan mengenai *game* yang akan melakukan generasi terhadap konten cuaca dengan menggunakan PCG dan mempelajari serta melihat bagaimana cara kerja pada saat melakukan generasi konten pada *game* yang dibangun. Dan juga mempelajari cara kerja algoritma *Markov Chain* dalam menentukan cuaca yang akan terjadi pada konten *game*.

d. Implementasi

Pada tahap ini, rancangan dari *game* yang sudah menerapkan PCG dan algoritma *Markov Chain* akan dibuat dengan menggunakan bahasa *C#* dengan *game engine Unity*.

e. Pengujian

Setelah rancang bangun *game* selesai, maka selanjutnya adalah tahap pengujian di mana akan dilakukan pengujian dengan memainkan *game* ini untuk menguji apakah sudah berfungsi terhadap *game* yang dibangun. Serta memastikan cuaca yang adaptif sudah bekerja setelah PCG menggunakan algoritma *Markov Chain* sudah berfungsi dengan baik selama permainan berlangsung.

f. Evaluasi

Pada tahap ini, dilakukan evaluasi menggunakan GUESS terhadap *game* dengan menyebarkan kuisioner terhadap pemain yang telah memainkan untuk mengukur tingkat kepuasan pemain. Serta mengevaluasi hasil, baik itu jika hasil sesuai dengan keinginan maupun tidak, menulis baik atau buruknya dari algoritma yang digunakan serta jika terdapat kendala akan dijelaskan pada tahap ini.

g. Dokumentasi

Pada tahap ini akan dilakukan dokumentasi selama progress pengerjaan baik itu jika ada catatan - catatan penting, apa yang terjadi selama proses rancang bangun *game open world*, serta beberapa gambar dari apa yang dilakukan selama proses pengerjaan maupun saat proses mendapatkan data.

3.2 Perancangan Game

Perancangan *game* dilakukan dengan menggunakan struktur *game* dan *flowchart* serta menampilkan *asset* yang dalam pembuatan *game*, beserta dengan *mockup* sebagai gambaran dasar *game*. Struktur game merupakan gambar awal dari game yang dimainkan oleh pemain. Pada *flowchart* terdapat dibagi menjadi beberapa bagian yang dipecah dalam beberapa modul. Lalu untuk asset game akan digunakan tabel - tabel yang dapat diurutkan.

3.2.1 Struktur Game

Judul *Game* : Gammon

Game ini merupakan *game open world*, dimana *game* akan bersifat bebas sesuai dengan keinginan pemain.

Formal elements yang terdapat dalam *game* ini dijelaskan sebagai berikut, yaitu:

1. Player

   Merupakan *game single player*, dimana *game* dimainkan oleh 1 pemain.

2. Objectives

   Pemain dapat bergerak secara bebas, berpindah dari satu tempat ke tempat lain dengan tujuan untuk mengalahkan musuh yang ada.

3. Procedures

   Pada *game* ini terdapat langkah - langkah yang dapat dilakukan oleh pemain, yaitu:

   a. Pemain menjalankan *game*, kemudian masuk ke *main menu*. *Game* akan dimulai ketika pemain memilih pilihan *start game*.

b. Setelah itu pemain masuk ke dalam *game*, *game* akan men-*generate* cuaca menggunakan algoritma *Markov Chain*, merupakan cuaca hasil *random* dengan menggunakan algoritma *Markov Chain* yang telah selesai dibangun. Cuaca pada *game* ini akan mempengaruhi cara mengalahkan musuh. Setiap berganti hari maka cuaca yang dihasilkan juga ikut berganti.

c. Pemain harus menyelesaikan *game* sebelum waktu yang diberikan telah selesai. Pemain yang telah masuk ke dalam game dapat bergerak secara bebas dari 1 tempat ke tempat lain. Jika pemain bertemu musuh, pemain memilih opsi untuk mengalahkan musuh tersebut atau tidak. Jika pemain memilih untuk mengalahkan musuh, maka pemain naik *level* dan *gold* bertambah.

d. Pemain yang telah mempunyai gold dapat membeli senjata dan item yang dapat membantu dalam memainkan *game* ini.

e. Pada permainan ini terdapat 1 jenis *boss*. Jika pemain berhasil mengalahkan *boss* maka *game* akan berakhir dan kembali pada main menu.

f. Jika darah pemain menyentuh 0, maka permainan berakhir dan muncul halaman *game over*.

4. Rules

a. Untuk mengontrol *game*, pemain harus menggunakan *keyboard* dan *mouse*.

b. Pemain harus mengalahkan boss untuk mengakhiri *game*.

5. Resources

a. *Gold*: menunjukkan jumlah *gold* yang dipunyai oleh pemain. Pemain mendapatkan *gold* dari mengalahkan musuh dan dapat dibelanjakan untuk senjata dan *item*.

b. *Player Health*: menunjukan darah yang dipunyai oleh pemain. Darah akan bertambah seiring dengan level pemain yang semakin membesar. Dan darah akan berkurang jika pemain mengenai musuh.

6. Conflict

   Pemain harus dapat bertahan hidup dari musuh dan menyelesaikan *game* dengan waktu tertentu.

7. Boundaries

   Pemain bebas berkeliling namun terbatas pada daerah yang sudah disediakan.

8. Outcome

   Pemain akan menang setelah mengalahkan *boss* pada *game* ini.

Selain *formal elements*, berikut ini merupakan *dramatic elements* yang terdapat pada *game*, yaitu:

1. Challenge

   Tantangan di dalam *game* ini adalah pemain bisa mengalahkan musuh serta *boss* tanpa mengalami kematian. Setiap *health* dari musuh dipengaruhi oleh cuaca yang berbeda, bisa memperkuat dan bisa memperlemah musuh sehingga mempengaruhi jalannya permainan.

   Berikut ini merupakan daftar cuaca yang mempengaruhi setiap musuh :

   - *Rainy* memperkuat *slime* dan memperlemah *snake* serta *dragon*.
   - *Snowy* semua musuh menjadi mode *default* atau tidak terjadi apa – apa.

- *Cloudy* dan *windy* memperlemah musuh *beast*.
- *Sunny* memperkuat musuh *bamboo*.

2. Play

   Pemain dapat dengan bebas untuk menentukan aktivitas yang diinginkan baik bergerak ke tempat lain serta bebas untuk berhadapan dengan musuh atau tidak.

3. Premis

   Premis di dalam game ini, *player* merupakan seorang *ninja* yang dibekali kemampuan bertarung untuk menolong orang lain.

4. Character

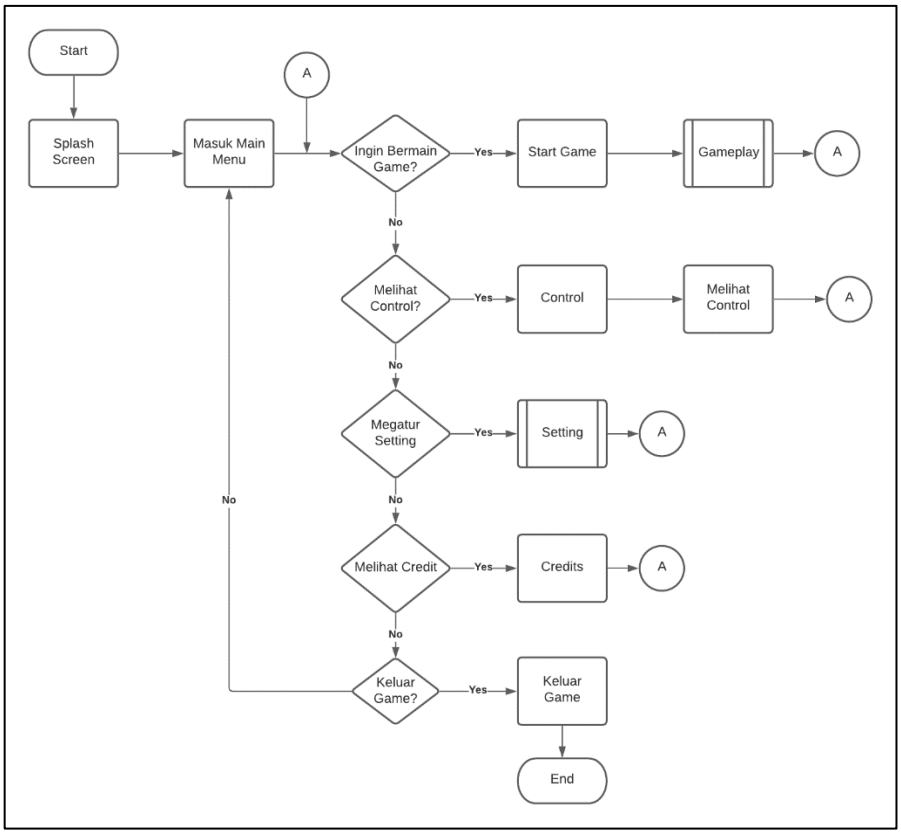
   Merupakan seorang ninja yang bernama Green Ninja, dibekali oleh *kunai* yang dapat dilemparkan kepada musuh.

5. Story

   Green Ninja merupakan seorang *ninja* yang sedang mencari jati diri. Dia bertualang ke berbagai tempat untuk mencari makna tentang kekuatan. Berbekal pesan dari gurunya untuk menolong orang yang sedang kesusahan dan kemampuan untuk bertarung Green Ninja memulai petualangannya.

3.2.2 Flowchart

Berikut ini adalah beberapa *flowchart* yang digunakan dalam pembuatan *game open world* menggunakan algoritma Markov Chain. Pada gambar 3.1 merupakan *flowchart main menu*.

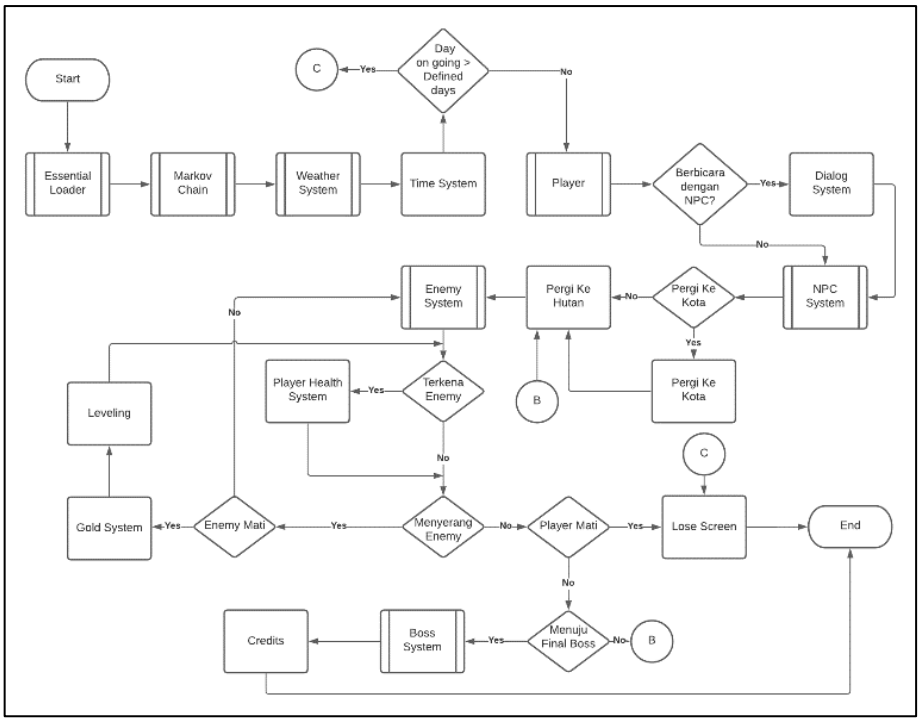


Gambar 3.1 Flowchart Main Menu

Pada *flowchart* diatas pemain akan membuka *game*, kemudian akan tampil sebuah *splash screen* dan akhirnya masuk ke dalam *main menu*. Di dalam main menu pemain diberikan pilihan untuk melakukan aksi. Jika pemain ingin bermain *game* akan memilih *start game* dan masuk ke modul *gameplay*, dan jika sudah selesai bermain *game* proses akan kembali ke dalam *main menu*. Jika pemain tidak ingin bermain *game* pemain dapat memilih tombol *control*, setelah pemain memilih tombol *control* akan diarahkan ke halaman *control* untuk memperlihatkan *control* apa saja yang dapat dilakukan didalam *game*, dan setelah selesai melihat *control* pemain dapat menekan tombol *back* dan kembali ke *main menu*.

Jika pemain tidak ingin melihat *control* maka, pemain dapat menekan tombol *setting* dan kemudian masuk ke modul *setting*, dan akan kembali ke dalam

main menu jika sudah selesai di dalam modul *setting*. Lalu, jika pemain tidak ingin masuk ke dalam *setting* maka pemain dapat melihat *credit* dan kemudian kembali ke dalam *main menu*. Dan pemain tidak memilih semua pilihan yang tersedia, maka pemain dapat menekan tombol *exit* sehingga pemain dapat keluar dari *game*.

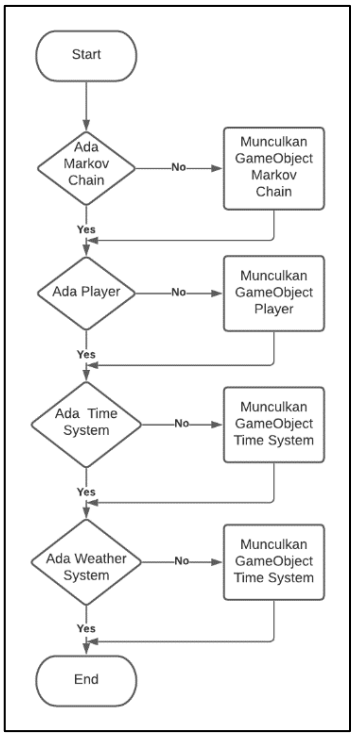


Gambar 3.2 *Flowchart* Modul *Gameplay*

Berikut ini merupakan gambar 3.2 merupakan *flowchart* modul *gameplay* yang akan menjelaskan keseluruhan alur *gameplay*. Pada flowchart modul *gameplay*, akan menjalankan beberapa modul, yaitu modul *essential loader*, *markov chain* dan *weather system*. *Time system* akan menjalankan sistem waktu yang ada didalam *game* yang dimulai dari hari ke 1 selama 24 jam, dan jika hari yang telah ditentukan lebih besar dari jumlah hari maka, *player* akan meninggal dan menuju *lose screen*. Lalu modul *player system* juga akan berjalan bersamaan dengan *gameplay* dimulai.

Player diberikan kebebasan untuk berbicara dengan NPC, jika *player* memilih berbicara dengan NPC maka *dialog system* akan bekerja. *Dialog system* akan muncul dilayar dengan menampilkan perkataan dari NPC maupun *Player* dan kemudian modul NPC *system* akan berjalan baik setelah berbicara dengan NPC maupun tidak. Lalu setelah itu *player* juga diberikan kebebasan dengan apakah *player* ingin pergi ke kota atau tidak, jika *player* memilih untuk pergi ke kota, maka *player* pergi ke kota. Jika *player* sudah pergi ke kota atau tidak pergi ke kota, *player* dapat pergi ke hutan. Di dalam hutan player akan bertemu musuh dan *enemy system* akan berjalan.

Lalu pemain diberikan kebebasan untuk menghindari serangan *enemy*, jika terkena serangan maka, *player health system* akan bekerja. *Player* yang terkena serangan ataupun tidak, mempunyai opsi untuk menyerang *enemy*. Jika player memilih untuk menyerang *enemy*, lalu jika *enemy* mati akan menghasilkan gold, kemudian *leveling system* akan bekerja dengan menambah *level player* lalu akan kembali pada serangan *enemy*. Jika tidak mati, maka akan kembali pada *enemy system*. Jika *player* memilih untuk tidak menyerang *enemy*, apakah *player* telah mati, jika iya, akan muncul *lose screen* dan kembali pada *main menu*. Jika *player* tidak mati, maka *player* dapat melanjutkan menuju tempat *final boss* berada atau player dapat kembali ke kota. *Player* yang melanjutkan menuju *final boss* akan berhadapan memangil modul *boss system* dan kemudian memanggil proses *credit* lalu kembali pada *main menu*.

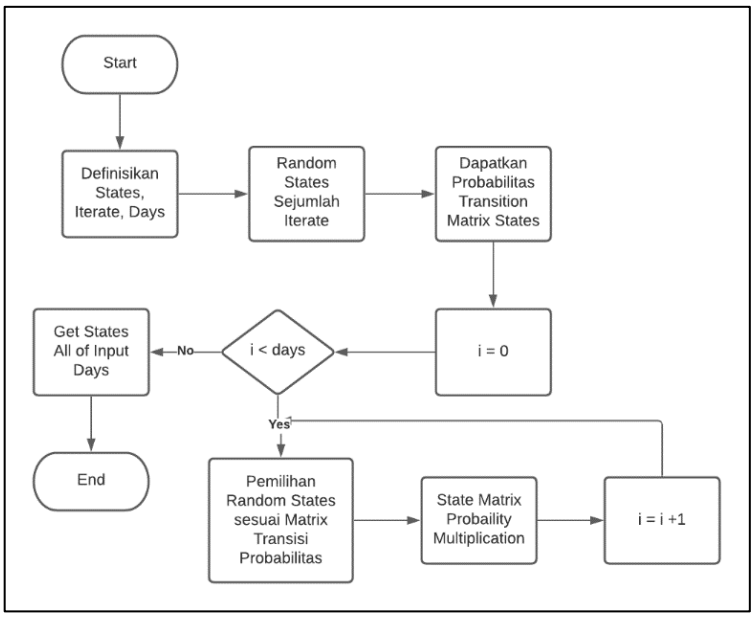


Gambar 3.3 *Flowchart* Modul *Essential Loader*

Pada gambar 3.3 merupakan *flowchart* tentang modul *essential loader*. Pada modul *essential loader* proses dimulai dengan melakukan pengecekan pada *object markov chain*, jika sudah ada akan berlanjut ke proses berikutnya, jika belum maka lalu masuk ke dalam proses munculkan *gameobject markov chain* dan menuju proses berikutnya. Proses berikutnya merupakan melakukan pengecekan terhadap *object player*, jika belum ada akan masuk ke dalam proses *munculkan gameobject player* dan menuju proses berikutnya.

Proses berikutnya merupakan pengecekan terhadap object time system, jika sudah ada akan menuju proses berikutnya dan jika belum akan menuju proses munculkan *gameobject time system* lalu menuju proses berikutnya. Lalu proses berikutnya adalah melakukan pengecekan terhadap *object weather system*, jika

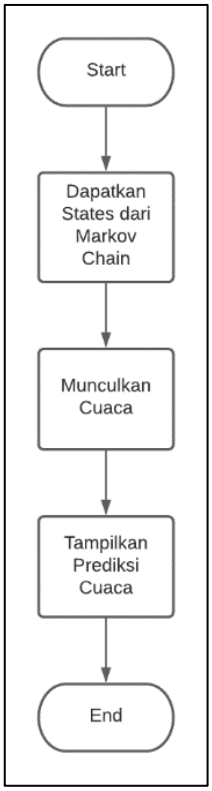
object system sudah ada, maka akan kembali pada *gameplay*, dan jika belum akan menuju proses munculkan *gameobject weather system* dan kembali pada *gameplay*.



Gambar 3.4 *Flowchart* Modul *Markov Chain*

Gambar 3.4 merupakan gambar *flowchart* mengenai modul *markov chain* yang digunakan pada *game*. Pada proses awal markov chain, diawali dengan mendefinisikan 3 variable, yaitu: *states* yang merupakan cuaca, *iterate* yang merupakan variabel untuk melakukan *looping random* sesuai dengan jumlah variabel *iterate* dan *days* yang merupakan variabel untuk menentukan jumlah hari yang dilalui *player*. Kemudian proses berlanjut pada *random states*, dimana *states* akan diacak sesuai dengan jumlah variabel *iterate*. Fungsi dari *random states* ini untuk mendapatkan probabilitas secara acak. Selanjutnya, setelah semua *states* diacak, dilakukan perhitungan dengan menghitung jumlah states yang muncul berdasarkan proses *random states* yang sudah dilakukan sehingga didapatkan matriks transisi probabilitas. Kemudian proses berlanjut kepada pendefinisian variabel i yang diisi dengan nilai 0. Selanjutnya proses bergerak kepada *looping*

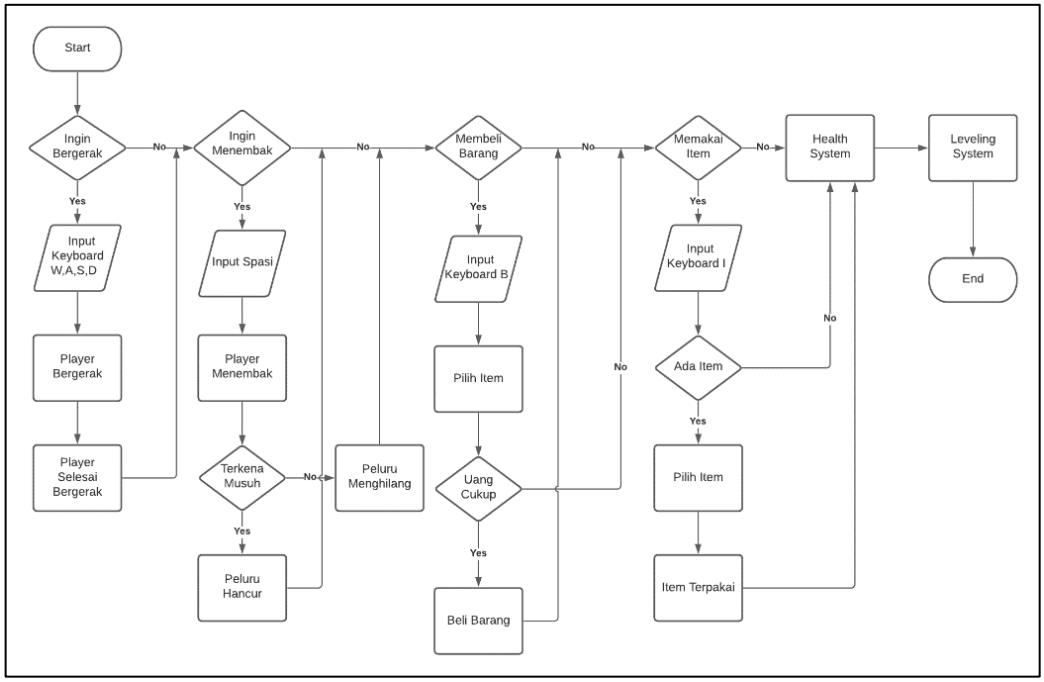
dimana, jika jumlah i lebih kecil dari jumlah *days*, maka proses menuju pemilihan states yang telah disesuai dengan bobot probabilitas matrik transisi. Sehingga saat terjadi proses *random* akan dipilih sesuai dengan bobot probabilitas yang dipunya pada setiap *state* sesuai dengan *state* yang sedang dipilih. Kemudian probabilitas transisi matriks dilakukan *multiplication* dengan dirinya sendiri, dan i bertambah dengan 1. Jika i sudah lebih besar sama dengan jumlah *days* maka, proses kembali menuju modul *gameplay*.



Gambar 3.5 *Flowchart* Modul *Weather System*

Pada gambar 3.5 merupakan *flowchart* modul *weather system*. Proses dimulai setelah modul *markov chain* selesai menjalankan prosesnya. Pertama mendapatkan *states* yang akan ditampilkan per hari yang berasal dari *markov chain*. Lalu setelah mendapatkan *states*, maka cuaca akan ditampilkan sesuai dengan hari

yang ada pada *states*. Selain itu, untuk *states* yang didapat akan ditampilkan dalam bentuk prediksi kepada *player*. Kemudian proses akan kembali pada *gameplay*.



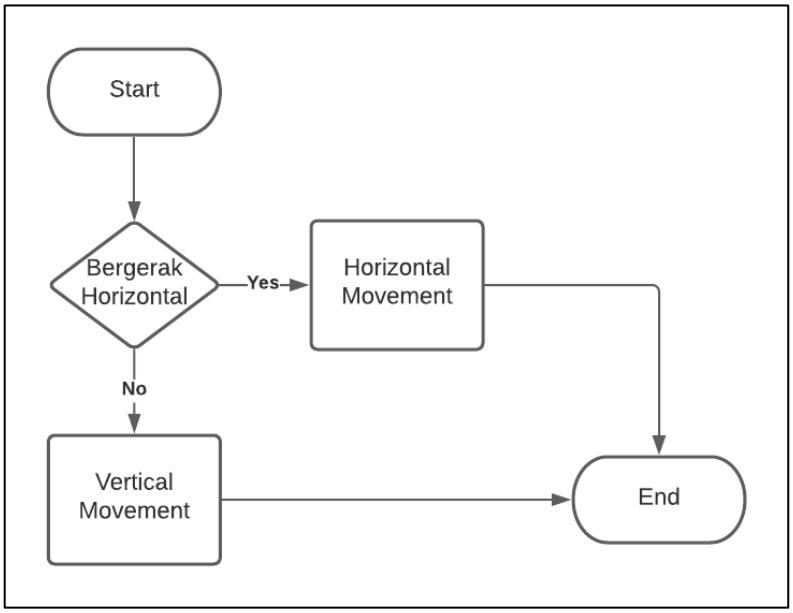
Gambar 3.6 *Flowchart* Modul *Player*

Pada gambar 3.6 merupakan *flowchart* modul *player*. Proses dimulai dengan *player* diberikan pilihan untuk bergerak. Jika *player* ingin bergerak, maka *player* dapat memasukkan *input* dari *keyboard* yaitu: w, a, s, d. Lalu proses akan berlanjut dimana *character* akan bergerak dan berlanjut pada proses *character* selesai bergerak. *Player* yang memilih untuk bergerak maupun tidak akan masuk pada pilihan ingin menembak.

Jika *player* memilih untuk menembak, *player* dapat memasukkan *input* spasi pada *keyboard*, lalu proses berlanjut pada *player* menembak dan jika mengenai *enemy* maka peluru hancur, jika tidak peluru akan hilang. Setelah itu, jika *player* memilih menembak atau tidak, proses berlanjut kepada pilihan untuk membeli barang. Jika *player* memilih untuk membeli barang maka, *player* dapat

memasukan input *keyboard* b, lalu akan masuk kepada proses untuk memilih *item* dan jika uang *player* cukup maka *item* akan terbeli dan berlanjut pada proses berikutnya.

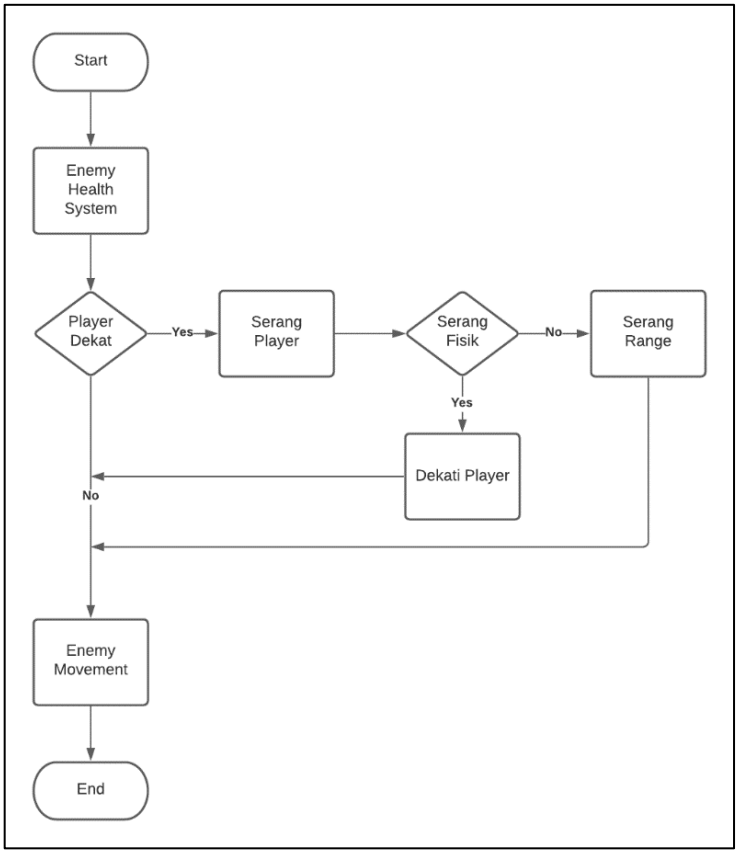
Jika *player* sudah membeli barang atau *player* yang tidak mempunyai uang atau tidak membeli barang, selanjutnya berada pada proses untuk pilihan memakai item. Jika *player* memilih untuk memakai *item*, maka *player* memasukkan input keyboard i. Jika *player* tidak mempunyai item, akan berlanjut pada proses berikutnya. Jika iya, maka pilih *item* dan *item* akan terpakai. Setelah itu baik *player* yang memilih *item* atau tidak, *player health system* akan bekerja, serta *leveling system* akan bekerja. Lalu proses akan kembali pada *gameplay*.



Gambar 3.7 *Flowchart* Modul NPC *System*

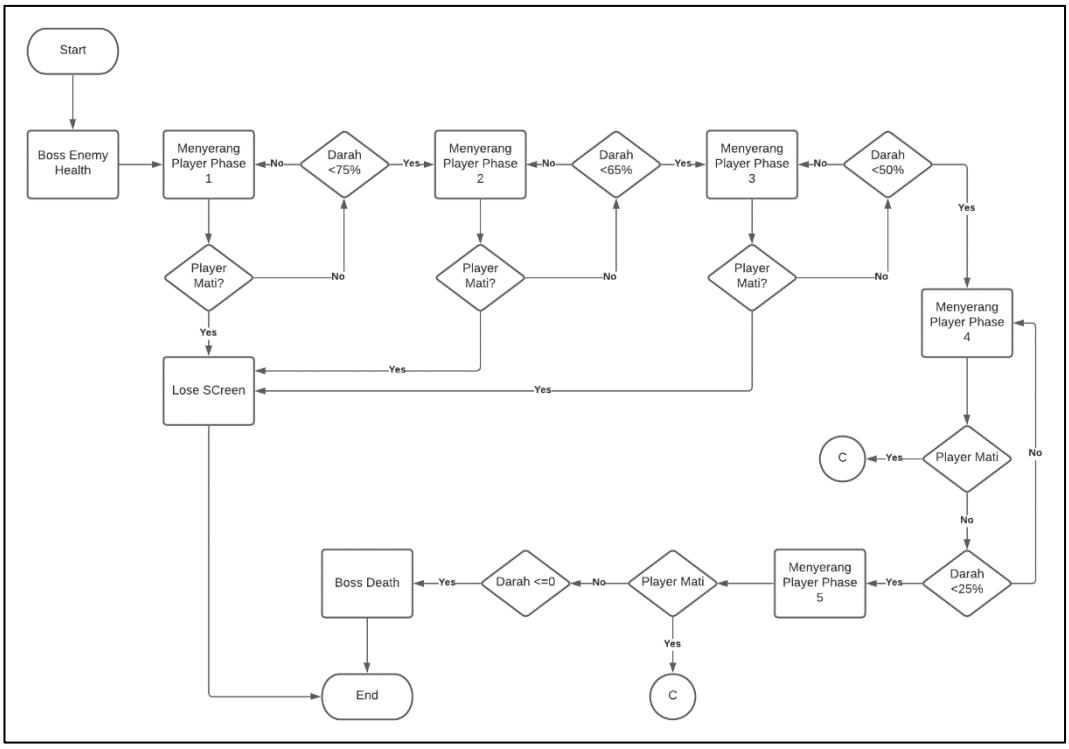
Pada gambar 3.7 merupakan modul *flowchart* NPC *system*, proses dimulai dengan pilihan apakah NPC bergerak secara *horizontal*, jika iya NPC bergerak secara *horizontal*, lalu kembali pada modul *gameplay*. Jika NPC tidak bergerak

secara *horizontal*, maka NPC akan bergerak secara *vertical* lalu kembali pada *modul gameplay*.



Gambar 3.8 *Flowchart* Modul *Enemy System*

Pada gambar 3.8 merupakan *flowchart* modul *enemy system*, dimana proses pertama dimulai dengan *enemy health system* yang berjalan, kemudian proses berpindah pada pilihan apakah *player* ada didekat enemy. Jika tidak, akan menuju proses selanjutnya. Jika iya, maka *enemy* akan menyerang *player*, dan proses selanjutnya melakukan pengecekan apakah serang fisik, jika iya serang fisik, jika tidak serangan *range*. Lalu proses akan berlanjut pada *enemy movement* dimana *enemy* akan bergerak sesuai dengan *movement* yang ada. Kemudian proses kembali pada modul *gameplay*.



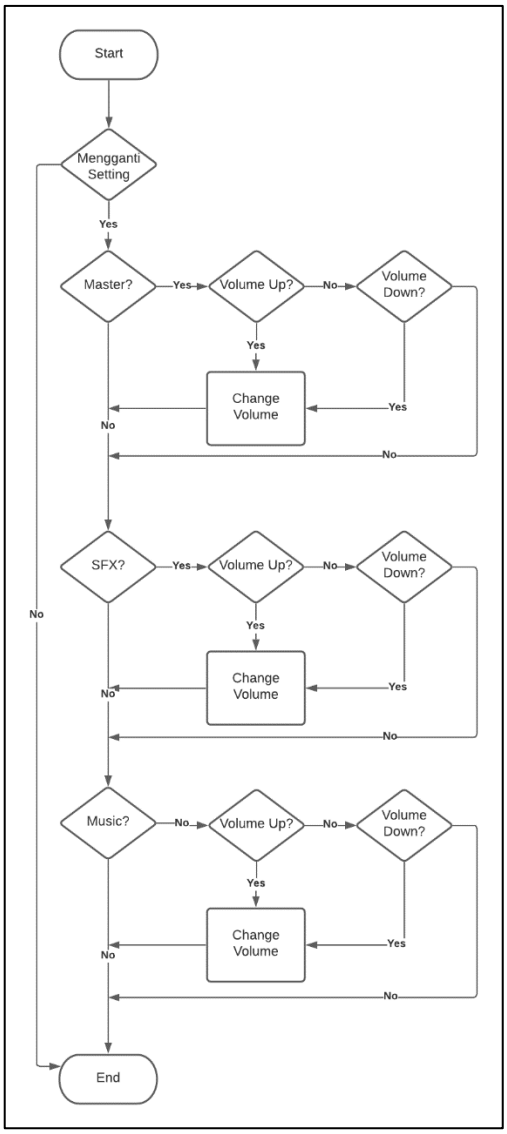
Gambar 3.9 *Flowchart* Modul *Boss System*

Pada gambar 3.9 merupakan *flowchart* modul *boss system*. Proses yang pertama kali dilakukan adalah *boss enemy health*, kemudian proses berlanjut pada menyerang *player phase* 1 dimana boss akan mendekati *player* lalu menyerang *player*. Jika *player* mati berlanjut pada proses *lose screen*. Jika tidak, dilakukan pengecekan terhadap darah *boss* berada dibawah 75%, jika tidak akan kembali pada *phase* 1.

Kemudian jika darah *boss* berada dibawah 75%, maka *boss* bergerak kepada *proses* menyerang *player phase* 2, dimana *boss* akan berada 1 tempat lalu menembak bola api kepada *player*. Jika *player* mati berlanjut pada proses *lose screen.* Jika tidak, dilakukan pengecekan terhadap darah *boss* berada dibawah 65%, jika tidak akan kembali pada proses *phase* 2. Dan jika iya maka proses akan berlanjut kepada menyerang *player phase* 3.

Pada *phase* 3, *boss* menyerang dengan mengeluarkan bola tanah yang lebih berbahaya terhadap *player*. Jika *player* mati berlanjut pada proses *lose screen.* Jika tidak, dilakukan pengecekan terhadap darah *boss* berada dibawah 50%, jika tidak akan kembali pada proses *phase* 3. Lalu jika darah *boss* berada dibawah 50% maka proses akan berlanjut kepada menyerang *player phase* 4. Pada *phase* 4, *boss* berganti kembali ke serangan api, namun jumlah serangan bertambah menjadi 4 serta mengikuti *player* pada saat ditembakkan.

Kemudian, jika *player* mati berlanjut pada proses *lose screen*, jika *player* tidak mati maka, dilakukan pengecekan terhadap darah *boss*. Jika darah *boss* tidak berada dibawah 25%, maka *boss* menyerang player kembali diphase 4. Jika iya, maka proses berlanjut kepada *phase* 5. Pada *phase* 5, boss akan berjalan ke beberapa arah sambil menyerang dengan 4 bola api kepada *player*. Jika *player* mati, proses selanjutnya adalah proses *lose screen*. Namun jika *player* tidak mati, dilakukan pengecekan terhadap darah *boss*. Jika darah *boss* tidak lebih kecil sama dengan 0, maka boss kembali berada di*phase* 5. Jika darah boss lebih kecil sama dengan 0 maka akan berlanjut pada proses *boss lose*. Setelah proses *boss lose*, maka akan kembali pada modul *gameplay*.



Gambar 3.10 *Flowchart* Modul *Setting*

Pada gambar 3.10 merupakan *flowchart* modul *setting*. Proses dimulai dengan apakah *player* ingin mengganti *master volume*. Jika iya, apakah *player* ingin mengganti *volume up*, lalu jika ingin mengganti *volume up* maka, *change volume* dan berlanjut pada proses berikutnya. Jika tidak, apakah player ingin mengganti *volume down*, lalu jika *player* ingin mengganti *volume down* maka, *change volume* dan berlanjut pada proses berikutnya. Jika *player* tidak ingin mengganti *volume*

down atau *player* tidak ingin mengganti *master volume* sehingga proses akan berlanjut pada proses berikutnya yaitu mengganti SFX.

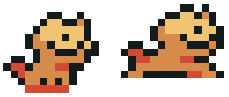
Jika *player* ingin mengganti SFX, *player* akan masuk ke dalam proses pengecekan *volume up*, jika iya, maka proses *change volume* akan bekerja dan menuju proses berikutnya. Selanjutnya jika *player* tidak memilih untuk *volume up*, maka dilakukan pengecekan kembali terhadap *volume down*, jika *player* memilih tidak, maka proses bergerak menuju proses selanjutnya. Namun jika *player* memilih *volume down*, maka proses *change volume* bekerja dan menuju proses berikutnya. Setelah proses SFX, selanjutnya merupakan proses pengecekan terhadap *volume music*. *Player* yang tidak mengubah *volume music* akan kembali pada *main menu*. Namun, *player* yang mengubah *volume music* akan masuk ke dalam proses pengecekan *volume up*, jika iya maka proses akan berlanjut pada proses *change volume* dan kembali pada *main menu*. Dan jika *player* tidak melakukan *volume up*, maka menuju proses pengecekan *volume down*, jika pemain tidak memilih *volume down* maka kembali pada *main menu*. Jika pemain memilih *volume down*, maka proses berlanjut pada *change volume* dan kembali pada *main menu*.

3.2.3 Aset

Berikut ini merupakan aset - asset dalam bentuk tabel - tabel yang akan digunakan pada *game open world* dengan menggunakan algoritma *markov chain*.

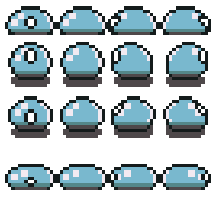
Tabel 3.1 Daftar Aset Karakter

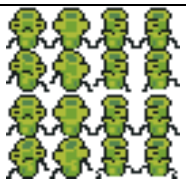
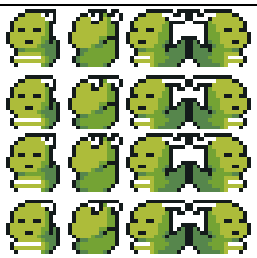
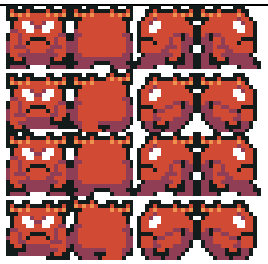
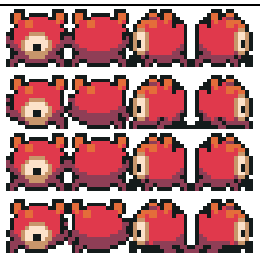
No	Nama	Gambar	Deskripsi	Sumber
1	Green Ninja		Merupakan tokoh utama yang digunakan oleh pemain.	pixel-boy

2	Old Man		Merupakan NPC pengghuni di desa awal.	pixel-boy
3	Child		Merupakan NPC pengghuni di desa awal.	pixel-boy
4	Cat		Merupakan NPC binatang di desa awal.	pixel-boy
5	Villager		Merupakan NPC di perbatasan kota dan hutan.	pixel-boy
6	Dog		Merupakan NPC binatang di kota.	pixel-boy
7	Old Man 2		Merupakan NPC di kota.	pixel-boy
8	Old Woman		Merupakan NPC di kota.	pixel-boy
9	Princess		Merupakan NPC penting di kota.	pixel-boy
10	Master		Merupakan NPC di dekat gua.	pixel-boy

Pada tabel 3.1 dijabarkan mengenai aset - aset karakter yang digunakan didalam pembangunan *game*.

Tabel 3.2 Daftar Aset Musuh

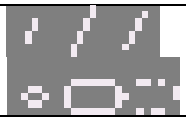
No	Nama	Gambar	Deskripsi	Sumber
1	Slime		Merupakan musuh yang hadir di awal permainan. Menyerang pemain dengan cara menabrak pemain.	pixel-boy
2	Slime 2		Merupakan musuh yang hadir di tengah permainan. Berdarah lebih tebal, menyerang pemain dengan cara menabraknya.	pixel-boy

3	Bamboo		Merupakan musuh yang hadir di tengah permainan. Menyerang pemain dengan menembakan jarum.	pixel-boy
4	Snake 1		Merupakan musuh yang hadir di akhir permainan. Menembak pemain dengan bola api serta jarak tembak yang luas.	pixel-boy
5	Dragon		Merupakan musuh yang hadir di akhir permainan. Menembak pemain dengan bola api serta jarak tembak yang lebih pendek dibanding *snake* 1.	pixel-boy
6	Beast		Merupakan musuh yang hadir di akhir permainan. Menyerang pemain dengan cara nabrak pemain dan mempunyai darah paling tebal diantara musuh lain	pixel-boy
7	Cyclopes		Merupakan bos akhir dari permainan. Mempunyai 5 fase pola serangan yang berbeda sesuai dengan jumlah darah yang dipunya	pixel-boy

Pada tabel 3.2 dijabarkan mengenai aset - aset musuh yang dihadapi oleh pemain didalam *game*.

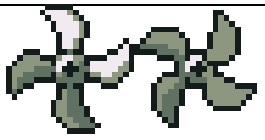
Tabel 3.3 Daftar Aset Cuaca

No	Nama	Gambar	Deskripsi	Sumber

1	Rainy		Merupakan partikel cuaca pada saat hujan.	pixel-boy
2	Windy		Merupakan partikel cuaca pada saat berangin.	pixel-boy
3	Cloudy		Merupakan partikel cuaca pada saat berawan.	pixel-boy
4	Snowy		Merupakan partikel cuaca pada saat bersalju.	pixel-boy

Pada tabel 3.3 dijabarkan mengenai aset - aset cuaca yang digunakan dalam pembuatan *game*, dimana aset sunny merupakan aset yang berbentuk cahaya yang berasal dari *unity*.

Tabel 3.4 Daftar Aset Senjata

No	Nama	Gambar	Deskripsi	Sumber
1	Kunai		Merupakan senjata awal yang dimiliki oleh pemain. Memiliki *damage* serang sebesar 5.	pixel-boy
2	Shuriken		Merupakan senjata yang dapat dibeli seharga 100 *gold*. Memiliki *damage* serang sebesar 10.	pixel-boy
3	Big Kunai		Merupakan senjata yang dapat dibeli seharga 275 *gold*. Memiliki *damage* serang sebesar 20.	pixel-boy
4	Big Shuriken		Merupakan senjata yang dapat dibeli seharga 500 *gold*. Memiliki *damage* serang sebesar 30.	pixel-boy

Pada tabel 3.4 merupakan jenis senjata yang dapat digunakan oleh pemain sepanjang *game*.

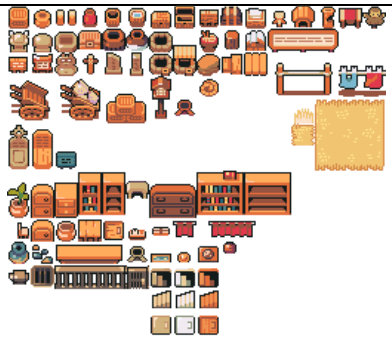
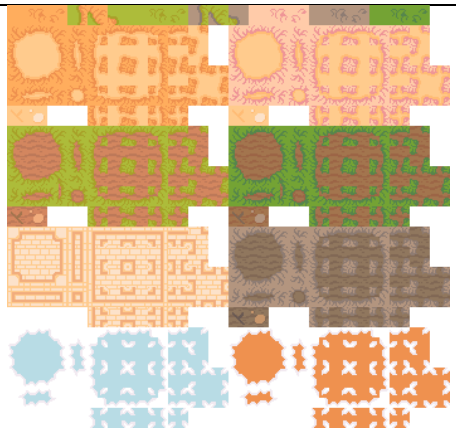
Tabel 3.5 Daftar Aset *Item*

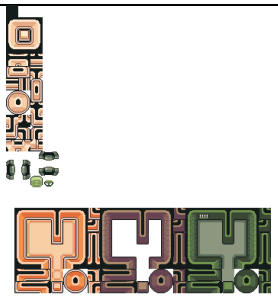
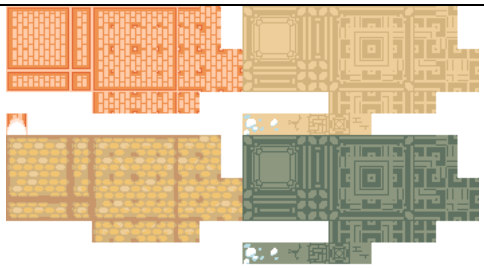
No	Nama	Gambar	Deskripsi	Sumber
1	Water Pot		Merupakan *item* untuk mempercepat gerakan pemain. Seharga 200 *gold*.	pixel-boy
2	Milk Pot		Merupakan *item* untuk menambah darah sebesar 50 pada pemain. Seharga 100 *gold*.	pixel-boy
3	Health Pot		Merupakan *item* untuk menambah darah sebesar 25-50 pada pemain. Seharga 50 *gold*.	pixel-boy
4	Medipack		Merupakan *item* untuk menambah darah sebesar sampai penuh pada pemain. Seharga 250 *gold*.	pixel-boy

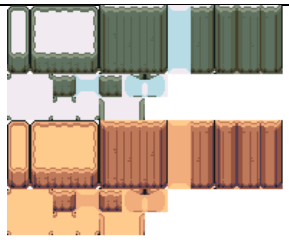
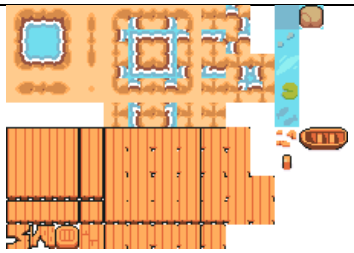
Pada tabel 3.5 merupakan jenis *item* yang dapat digunakan oleh pemain sepanjang *game*.

Tabel 3.6 Daftar Aset *Tileset Background*

No	Nama	Gambar	Sumber
1	Dungeon		pixel-boy

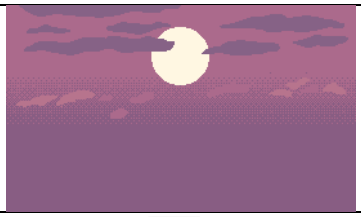
2	Element		pixel-boy
3	Floor		pixel-boy
4	Floor B		pixel-boy
5	Floor Detail		pixel-boy
6	Hole		pixel-boy

7	House		pixel-boy
8	Interior		pixel-boy
9	Interior Floor		pixel-boy
10	Logic		pixel-boy
11	Nature		pixel-boy

12	Relief		pixel-boy
13	Water		pixel-boy

Pada tabel 3.6 merupakan kumpulan *tileset background* yang digunakan dalam pembangunan *game*.

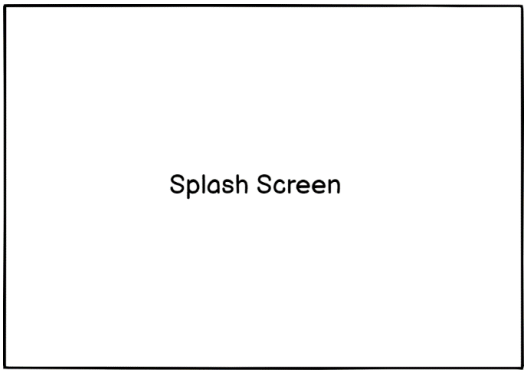
Tabel 3.7 Daftar Aset *UI* dan *Music*/SFX

No	Nama	Gambar	Sumber
1	Background Parallax		Luis Zuno
2	Normal Font	 NormalFont.ttf	pixel-boy
3	Music and SFX		pixel-boy

Pada tabel 3.7 merupakan kumpulan aset yang digunakan untuk UI serta *music* maupun SFX di dalam *game*.

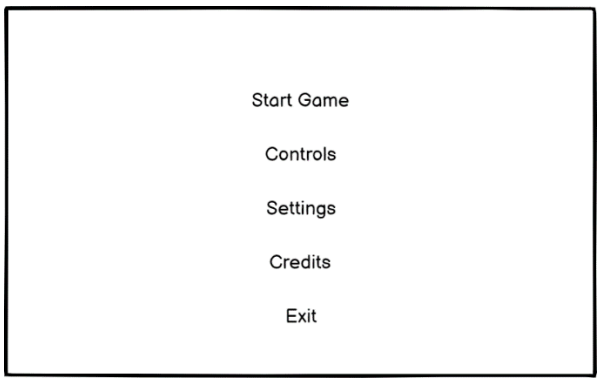
3.2.4 Mockup Design

Berikut ini merupakan wujud dari mockup design untuk game. Pada gambar 3.11 merupakan halaman awal yang berupa *splash screen*. Splash screen yang ditampilkan merupakan logo dari Unity dan logo dari Universitas Multimedia Nusantara.



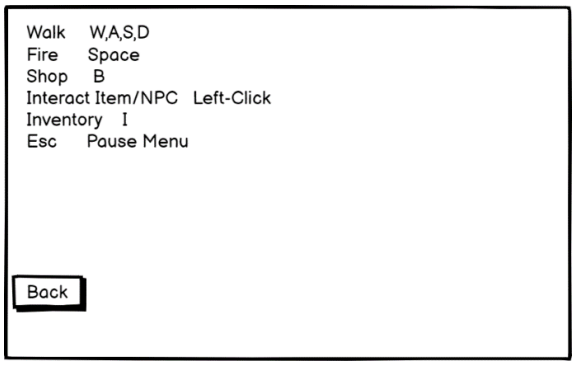
Gambar 3.11 *Mockup Splash Screen*

Kemudian setelah melalui halaman splash screen akan berlanjut menuju halaman main menu. Pada gambar 3.12 merupakan tampilan mockup pada main menu. Tampilan tersebut berisi tentang *start game* yang berfungsi untuk memulai *game, controls* penjelasan kontrol yang dapat digunakan oleh pemain, *settings* untuk mengatur pengaturan didalam *game, credits* menampilkan nama - nama kontributor dan *exit* untuk keluar dari permainan.



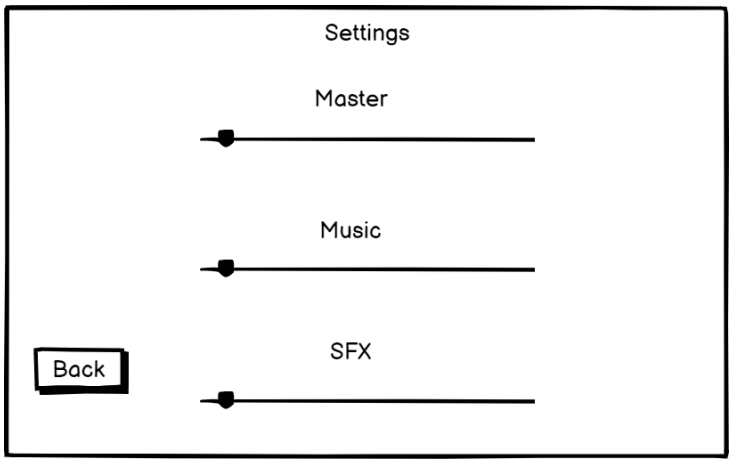
Gambar 3.12 *Mockup Main Menu*

Halaman yang bisa diakses pemain pada *main menu* salah satunya adalah *controls*. Pada gambar 3.13 merupakan *mockup* tentang *controls*, dimana pada *controls* berisi tentang cara untuk memainkan *game* dan juga *button back* untuk kembali ke *main menu*.



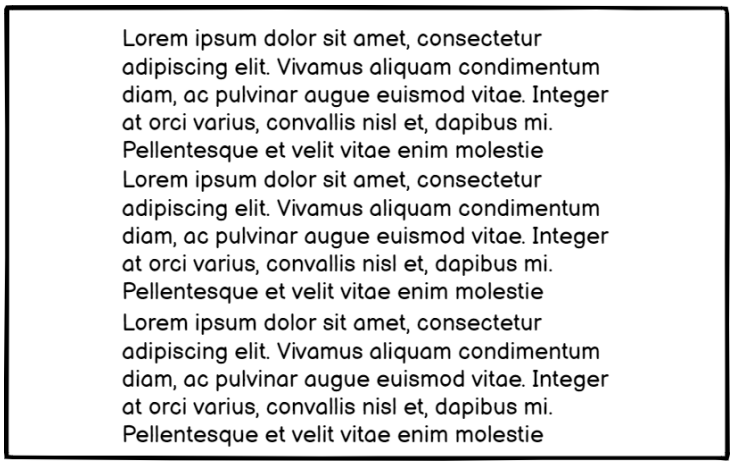
Gambar 3.13 *Mockup Controls*

Selain *controls* pemain juga bisa mengakses halaman *settings*. Pada gambar 3.14 merupakan *mockup* halaman *settings*. *Settings* terdiri dari 3 buah *slider* untuk mengatur *audio*, yaitu: *master*, *music*, dan SFX serta *button back* untuk kembali ke *main menu*.



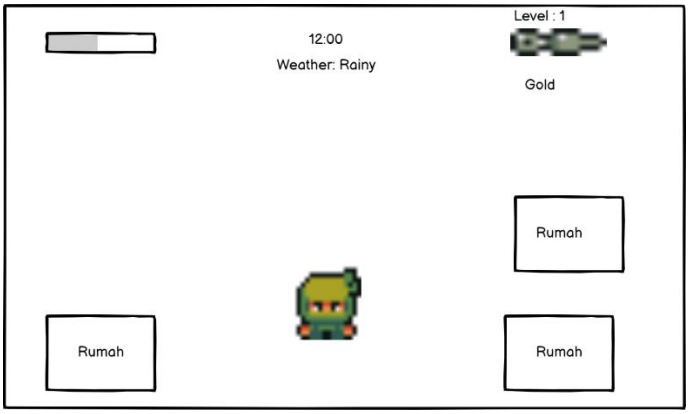
Gambar 3.14 *Mockup Settings*

Selain *settings* pemain juga bisa mengakses halaman *credits*. Pada gambar 3.15 merupakan *mockup* untuk halaman *credits.* Pada halaman *credits* menampilkan daftar - daftar kontributor.



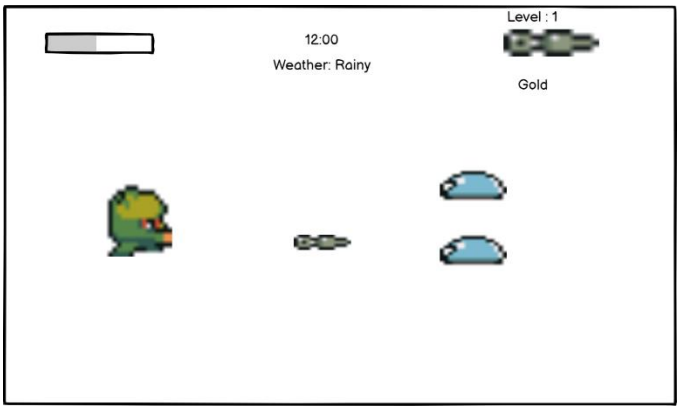
Gambar 3.15 *Mockup Credits*

Jika pemain menekan tombol *start* akan diarahkan ke halaman *gameplay* kota. Pada gambar 3.16 merupakan *mockup* untuk *gameplay* di kota. Pada *mockup* ini ditampilkan tampilan karakter yang digunakan oleh pemain serta rumah - rumah sebagai gambaran. Lalu terdapat tampilan *bar* darah pemain, jam, cuaca yang ditampilkan, *level*, senjata yang digunakan dan uang yang dipunya oleh pemain.



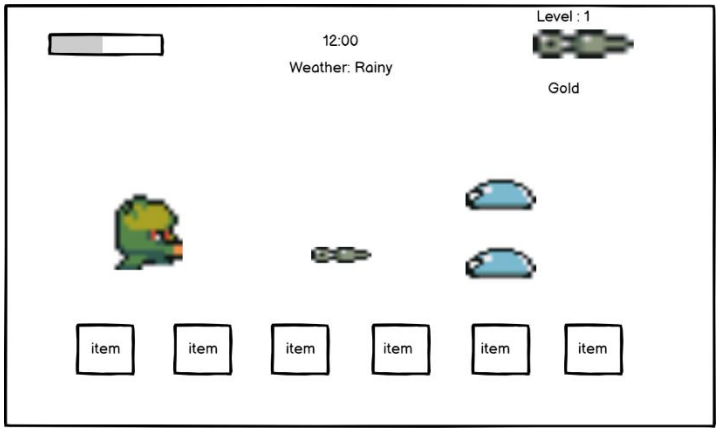
Gambar 3.16 *Mockup Gameplay* di Kota

Selain gameplay di kota terdapat juga gameplay, yaitu pada saat bertarung dengan musuh. Pada gambar 3.17 merupakan gambar mockup tentang gameplay battle saat bertarung melawan musuh. Tetap memilik informasi yang sama dengan gameplay di kota.



Gambar 3.17 *Mockup Gameplay Battle*

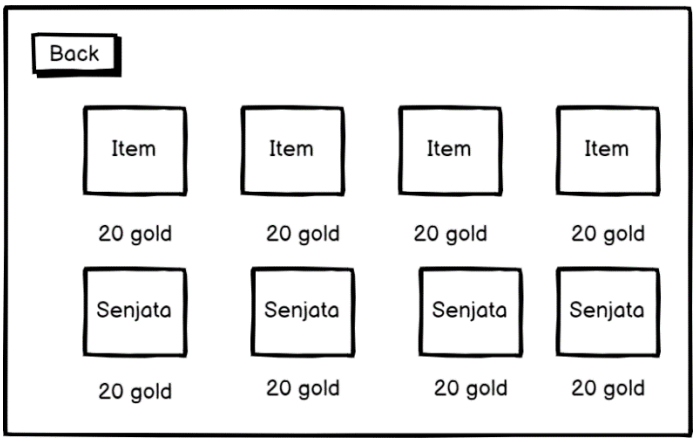
Jika pemain menekan I pada *keyboard*, maka pemain bisa mengakses *inventory* untuk melihat *item* yang dipunyai oleh pemain. Pada gambar 3.18 merupakan tampilan *inventory* jika pemain membukanya.



Gambar 3.18 *Mockup Inventory*

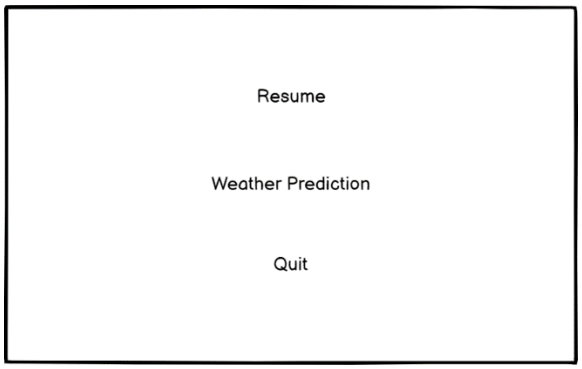
Jika pemain menekan b pada *keyboard*, maka pemain bisa mengakses halaman *shop* untuk membeli barang dan senjata. Pada gambar 3.19 merupakan tampilan *mockup*

untuk halaman *shop* yang terdiri dari *list item* dan senjata serta harga, dan juga mempunyai *button back* untuk kembali ke gameplay.



Gambar 3.19 *Mockup Shop*

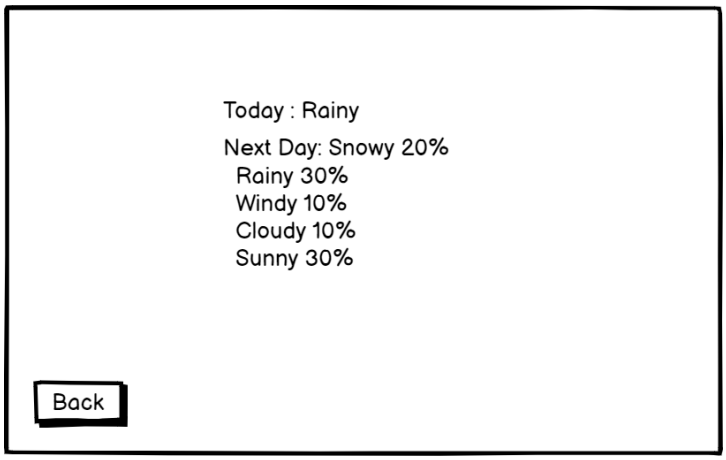
Jika pemain menekan esc pada keyboard, maka pemain akan membantu pause menu. Pada gambar 3.20 merupakan tampilan *pause menu*, yang terdiri dari *resume* yang berfungsi melanjutkan permainan, *weather prediction*, dan *quit* untuk kembali ke *main menu*.



Gambar 3.20 *Mockup Pause Menu*

Jika pemain dari main menu memilih *weather predic*tion, maka pemain akan dibawa kehalaman tersebut. Pada gambar 3.21 merupakan tampilan *mockup* untuk *weather prediction* yang terdiri dari cuaca hari ini serta prediksi cuaca untuk besok

sesuai dengan presentase yang ditampilkan serta *button back* untuk kembali ke *pause menu*.



Gambar 3.21 *Mockup Weather Prediction*