3. ANALISA DAN DESAIN SISTEM

Pada bab ini akan dijelaskan mengenai desain dari sistem yang akan dibuat dari aplikasi *game* ini, beserta garis besar cara kerja sistem secara keseluruhan aplikasi *game* yang akan dibuat. Selain itu, akan dibahas juga mengenai cara kerja serta desain metode *Artificial Intelligence* (AI) yang dipakai dalam sistem yaitu *behaviour tree*.

3.1 Analisa Desain

Pada sub bab ini akan dijelaskan penyesuaian yang perlu dilakukan dalam pembuatan desain aplikasi *game* ini. Ada beberapa perubahan yang telah dilakukan dari ruang lingkup sebagai penyesuaian:

- Adanya perubahan kecil pada mekanisme yang ada pada *game*, yang di mana, tidak akan adanya mekanisme pulau terlantar yang nantinya akan tenggelam dalam jangka waktu tertentu. Perubahan ini dilakukan karena adanya kesulitan pada saat mau menerapkan ide tersebut di karenakan kurangnya realistis mekanisme tenggelamnya sebuah pulau pada *game survival*, di mana ada juga sebuah NPC yang mengejar sang *player*.
- Ada penambahan pada jenis tipe *item* yang dimiliki oleh *game* yaitu *usable Items*. Sebelumnya, *game* ini hanya memiliki 2 macam jenis tipe *item* yaitu *materials* dan *tools.* Penambahan pada jenis tipe *item* diperlukan agar dapat menambah fitur yang dimiliki *game*. Dengan adanya penambahan fitur ini, *player* dapat memakai *item* yang telah diperoleh dari sekeliling pulau atau dibuat melalui *crafting tools.*
- Adanya perubahan pada kondisi untuk menyelesaikan *game,* di mana *player* dengan cara memecahkan misteri yang ada di sekeliling *desert island* agar dapat kabur. Misteri yang di maksud adalah berupa teka-teki yang akan muncul disekitar pulau yang nantinya akan menuntun *player* untuk kabur dari *desert island*. Sebelumnya, game ini hanya direncanakan untuk *player* mencari jalan kabur dengan cara harus mencari *materials* yang dibutuhkan, yang nantinya akan dipakai untuk membentuk *item* baru berdasarkan *crafting*. Perubahan ini terjadi di karenakan agar dapat membuat *player* lebih tertantang dengan adanya teka-teki pada saat memainkan *game* ini.

- Ada perubahan yang besar pada mekanisme yang dimiliki pada *difficulties* (Tingkat kesulitan). Hal-hal yang hanya dapat mempengaruhi *gameplay* dari AI NPC pada saat *player* memilih tingkat kesulitan adalah kecepatan NPC pada saat mengejar *player*, radius mendeteksi *player* berdasarkan pengelihatan yang dimiliki oleh NPC, dan juga jangka waktu *player* dalam menyelesaikan *puzzle* (teka-teki). Sebelumnya, hal yang direncanakan saat merancang tingkat kesulitan yang dimiliki oleh *game* ini adalah *player* akan mempunyai *hit point* berdasarkan jumlah nyawa dan akan hilang 1 buah nyawa jika diserang NPC. Selain itu, mekanisme berupa kecepatan NPC pada saat mengejar *player* masih mempunyai kondisi yang sama yaitu kecepatan berdasarkan *difficulties*. Perubahan ini terjadi di karenakan kurangnya efesien dan juga kurang cocok untuk penggunaan jumlah nyawa sebagai bentuk *hit point player,* sehingga hal tersebut kurang cocok untuk diterapkan pada *genre* game ini yaitu *survival*.
- Ada perubahan yang sangat besar, di mana dalam *game* ini tidak akan ada *multiple ending. Game* ini hanya akan mempunyai 1 buah *ending* saja, di mana *ending* tersebut hanya dapat muncul pada saat *player* berhasil kabur dari *desert island*. Perubahan ini terjadi di karenakan adanya banyak perubahan pada mekanisme *game* berupa tidak adanya mekanisme tenggelam pada *desert island*. Di karenakan hal tersebut maka dilakukan penyesuaian dengan membuat 1 buah *ending* saja. Penyesuaian tersebut dilakukan agar dapat mempermudah pembuatan *game* dan tidak mempersulit narasi dari alur cerita yang dimiliki oleh *game* ini.

3.2 Game Design

Pada sub bab ini, akan dijelaskan bagaimana cara kerja dan juga detail dari *game* ini. *Game* ini adalah *game* yang berfokus ke *genre Survival,* di mana *player* berusaha kabur dari *Non Player Character* (NPC) yang bertugas mengejar *player* dan membunuhnya. *Game* ini akan terbagi atas 3 tingkat kesulitan, yaitu *Normal, Medium,* dan *Hard*.

3.2.1 *Game Storyline*

Aplikasi *Game* yang akan dibuat Bernama "Island Escapist". *Game* ini menceritakan tentang seseorang (*player*) yang lagi berlibur di tengah laut dengan kapal miliknya yang nantinya akan terjebak di suatu pulau yang terlantar di karenakan ombak yang dahsyat.

Player yang terjebak dalam pulau tersebut kemudian akan mencoba kabur dari pulau tersebut. Dalam perjalanan untuk kabur, *player* akan dihadang dengan berbagai macam tantangan berupa *monster* (*enemy*) yang berkeliaran di sekeliling pulau, dan juga memecahkan teka-teki yang ada di sekitar pulau berdasarkan petunjuk yang diperoleh.

.Jika *player* kemudian berhasil memecahkan teka-teki yang ada dalam pulau tersebut, maka *player* dapat mempunyai kesempatan untuk kabur menggunakan petunjuk yang dimilikinya. Begitu juga dengan sebaliknya, di mana jika *enemy* berhasil menghalangi *player* dengan membunuhnya, maka *player* akan di alihkan ke layar *game over,* yang di mana *player* hanya bisa memilih 2 pilihan menu yaitu *load* dan *title screen*.

3.2.2 *Component* Game *Design*

Ada beberapa *game component* yang dibutuhkan dalam pembuatan aplikasi *game* ini. *Component-component* ini akan dibutuhkan agar dapat Menyusun aplikasi *game* ini. *Component* yang dimaksud adalah:

- *Level*: Tempat di mana *Player* dan NPC (*enemy)* berada. Tema dari *level* ini adalah *desert island.*
- *Level Component*: Benda-benda yang terletak atau berada dalam *level* ini. Benda-benda yang dimaksud adalah pohon, batu-batuan, kapal *yacht* yang terdampar, dan buah-buahan.
- *Level difficulties*: Tingkat kesulitan yang ada pada *game,* di mana tingkat kesulitan yang ada dalam *game* ini ada 3 yaitu *Normal, Medium,* dan *Hard*.

3.2.3 *Player Design*

Di dalam *game* ini, *player* dapat menggerakan karakter mereka di sekeliling pulau terlantar yang terbuka, di mana *player* bermain dari segi *first-person perspective*. *Player* mempunyai wujud sebagai pemuda yang menggenakan jaket *hoodie* dan celana Panjang. Tentunya *player* mempunyai beberapa *component* penting yang dapat membantu *player* pada saat memainkan *game* ini, di antaranya adalah:

- *Hit Point (HP): Hit Point* yang dimiliki oleh *player* ditunjukan dalam bentuk berupa *health bar. Health bar* yang di mana merupakan salah satu *component* yang sangat penting adalah faktor utama yang menjadi penentu hidup atau matinya *player. Health bar* ini hanya bisa dikurangi jika diserang oleh *enemy* atau jatuh dari *fall damage*. *Health bar* tentunya juga dapat dipulihkan melalui makanan berupa buah yang dapat ditemukan di sekeliling pulau dan juga dikonsumsi oleh *player.*

- *Stamina: Stamina* merupakan salah satu *component* penting yang bisa membantu *player* dalam memainkan *game* ini. *Stamina* digunakan agar *player* dapat melakukan *sprint* di dalam *game*. Pada saat *sprint*, stamina otomatis akan berkurang selama *player* melakukan *sprint*. Tentunya juga *stamina* yang telah habis dapat dipulihkan selama *player* memberhentikan karakter yang semenatara melakukan *sprint. Stamina* mempunyai bentuk yang sama seperti *health bar,* di mana pada *game* ini akan di sebut sebagai *Stamina bar*.

- *Armor: Armor* berguna sebagai *item* yang dapat melindung *player* dari *damage* terutama dari serangan *enemy*. Pada saat *player* terkena *damage,* jika *player* menggunakan *armor*, yang berkurang bukanlah *hit point* melainkan *armor* yang digunakan oleh *player*. *Armor* mempunyai bentuk yang sama dengan *hit point* dan *stamina* di mana pada game ini akan di sebut sebagai *armor bar*. Pada saat *armor bar* habis, maka serangan selanjutnya akan mengurangi *hit point,* di karenakan *armor bar* akan menghilang pada saat diserang sampai habis. Agar dapat memperoleh *armor* Kembali, *player* harus membuat *armor* melalui *crafting tools* sehingga *armor bar* akan muncul kembali.

- *Crafting Tools: Crafting Tools* adalah salah satu *component* terpenting dalam game agar *player* dapat selamat pada saat memainkan *game* ini. Dengan menggunakan *Crafting Tools, player* dapat membuat berbagai *items* yang dapat membantu *player* berupa *armor* ataupun *trap.* Tetapi untuk membuat *item* tersebut, *player* harus mengumpulkan *materials* berupa batu dan kayu yang tersebar pada sekeliling pulau. *Material* yang dikumpulkan oleh *player* juga harus memenuhi kriteria untuk pembuatan *item*, jika tidak *player* tidak akan dapat membuat *item* tersebut.

- *Axe: Axe* merupakan salah satu *component* penting agar dapat memperoleh *material*. Tetapi *material* yang bisa diperoleh oleh *axe* hanya satu jenis *material* saja, di mana *material* yang dimaksud adalah kayu. Untuk memperoleh kayu, *player* harus memukul pohon menggunakan *axe* sebanyak 3 kali untuk memperoleh kayu. Kayu yang dapat diperoleh oleh *player* pada saat menebang adalah minimal sebanyak 1 kayu dan maksimal sebanyak 3 kayu.

- *Pickaxe: Pickaxe* merupakan salah satu *component* penting selain *axe* yang dapat memperoleh *material*. Begitu juga dengan *axe* yang hanya dapat memperoleh 1 jenis material berupa kayu, *pickaxe* juga hanya dapat memperoleh 1 jenis *material,* di mana *material* itu adalah batu. Sama dengan cara untuk memperoleh kayu, cara untuk memperoleh batu adalah dengan cara menambang sebuah batu besar sebanyak 3 kali dan batu yang diperoleh hanya minimal sebanyak 1 batu dan maksimal sebanyak 3 batu.

- *Itembox: Itembox* merupakan salah satu *component* penting untuk dimiliki oleh *player. Itembox* mempunyai kegunaan sebagai tempat penyimpan *items* yang akan berguna pada saat memainkan *game*. *Item-item* yang di maksud adalah berupa *tools, usable items,* dan *materials.*

- *Compass: Compass* akan berguna sebagai penuntun arah dalam *in-game*, di mana petunjuk yang diperoleh oleh *player* akan terkadang memberi petunjuk berkaitan dengan arah *compass*.

- *Hotbar: Hotbar* adalah deretan kotak yang terletak pada layar bawah tengah *player*, di mana kotak ini menyimpan *item* dimiliki oleh *player* yang disimpan pada *itembox.* Kegunaan utama dari *hotbar* agar dapat membantu mempercepat *player* dalam melakukan pergantian atau pemakaian dari sebuah *item* tanpa harus pergi ke dalam *itembox.* Agar dapat menggunakan *hotbar,* maka *player* harus menekan *key* tertentu agar dapat mengganti atau memakai *item* tersebut.

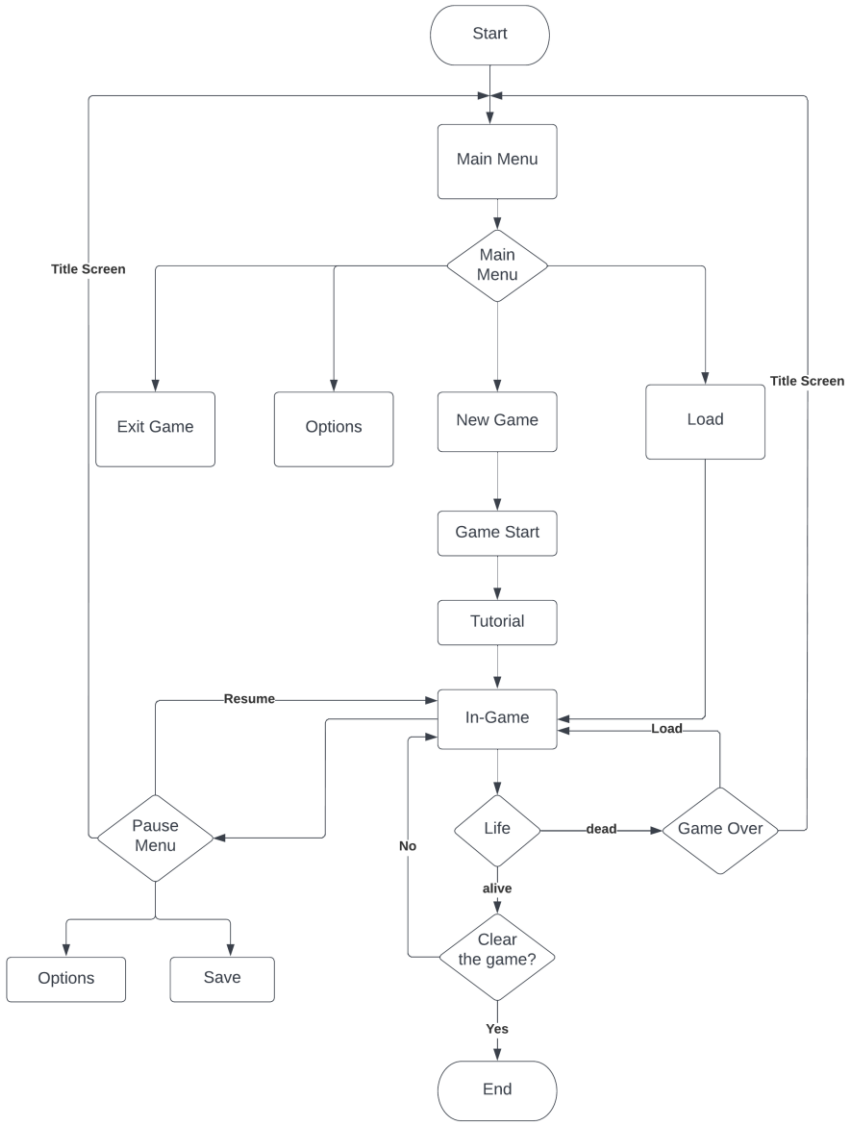
3.2.4 NPC (*enemy*) *Design*

NPC akan bertipe *enemy*, di mana *enemy* yang ada di dalam *game* ini adalah *monster* yang berkeliaran di sekeliling pulau terlantar. Di dalam *game*, *enemy* hanya akan memiliki satu jenis buah tipe *monster* dengan 3 variasi pergerakan berdasarkan *difficulties* yaitu *Normal*, *Medium*, dan juga *Hard.* Selain itu *enemy* juga mempunyai beberapa

component penting, di antaranya adalah:

- *Attacking: Attacking* adalah salah satu kunci *component* penting yang dimiliki oleh *enemy*. *Component* ini berguna agar *enemy* dapat memberikan *damage* atau mengurangi jumlah *Hit Point* yang dimiliki oleh *player*.
- *Sight: Sight* berguna sebagai pengelihatan yang dimiliki oleh *enemy* agar dapat mengetahui posisi *player* jika lagi di depan *enemy.* Tetapi jika *player* berada di daerah yang cukup jauh atau berhasil kabur dari *enemy*, maka ada kemungkinan *enemy* dapat kehilangan posisi *player* dan akan berkeliaran untuk mencari *player*.
- *Chasing*: *Chasing* merupakan *component* yang sangat penting yang ada pada *enemy*. *Component* ini membuat *enemy* dapat mengejar *player* dengan kecepatan berdasarkan *difficulty* yang di pilih oleh *player* pada awal *game. Chasing* hanya bisa dilakukan oleh *enemy* jika *player* lagi berada pada *sight* (pengelihatan) *enemy*. Jika *enemy* sudah kehilangan posisi *player,* maka *player* akan berkeliaran untuk mencari *player*.
- *Searching*: *Searching* merupakan fungsi *component* yang hanya bisa muncul Ketika *enemy* kehilangan letak posisi *player*. Setelah kehilangan posisi dari *player, enemy* akan mulai melakukan *searching* (mencari) keberadaan *player* berdasarkan tempat terdekat dari posisi *enemy* pada saat mencari *player*. Setelah gagal pada saat melakukan pencarian *enemy* kemudian akan Kembali melakukan suatu fungsi yang bisa disebut *patrol*.
- *Patrol*: *Patrol* adalah fungsi *component* yang sangat berguna untuk *enemy* yang hanya bisa aktif Ketika *enemy* tidak melakukan aktifitas apapun. Dengan adanya fungsi *patrol*, *enemy* akan berjalan dengan arah yang spesifik berdasarkan *patrol path* yang sudah diatur dari *game*. *Patrol path* digunakan sebagai tujuan dari arah *enemy* akan berjalan. Fungsi ini hanya akan berhenti pada saat *enemy* mendapat lokasi *player* berdasarkan pengelihatan yang *enemy*.

3.2.5 *Gameplay Design*



Gambar 3.1. *Flowchart Process Gameplay*

Game akan dimulai seperti pada Gambar 3.1. Keterangan yang bisa diambil adalah:

- *Player* akan mulai dari layar *Title screen* di mana akan terdapat beberapa *menu* yang dapat dipilih oleh *player* berupa *New game, Load, options*, dan juga *Exit Game*.
- Jika *player* memilih *New Game*, maka *player* akan mulai memainkan *game* dari keadaan awal. Jika player memilih *Load* maka *player* akan

melanjutkan keadaan *gameplay* yang telah di *save* oleh *player*. Jika *player* memasuki *options* maka *player* dapat mengatur konfigurasi yang dimiliki oleh aplikasi *game*. Jika *player* memilih *Exit game*, maka *player* akan dikeluarkan dari *game*.

- Dilanjutkan dari tahapan *New Game*, *player* kemudian memasuki *Tutorial* yang nantinya akan mengajarkan semua hal berkaitan dengan cara kerja *game*. Setelah menyelesaikan *Tutorial*, *player* kemudian melanjutkan ke bagian *In-game,* di mana *player* akan mulai bermain *game* "Island Escapist".

- Di dalam *In-game, player* dapat melakukan *pause* pada *game* yang dimainkan, dan saat *player* menampilkan *pause menu, player* bisa memilih beberapa pilihan *menu* berupa *Resume, Save, Options*, dan *Title Screen.*

- Pilihan *Resume* berfungsi untuk melanjutkan *game* yang ada dalam keadaan *pause*. Pilihan *Save* berfungsi sebagai sistem yang dapat membantu untuk menyimpan *state gameplay player*, sehingga pada saat *player* ingin memainkan *state gameplay* yang disimpan, maka *player* hanya tinggal melakukan *Load* pada *state gameplay* yang telah di *save.* Pilihan *Options* berfungsi agar *player* dapat mengatur konfigurasi yang berkaitan dengan aplikasi *game*. Pilihan *Title Screen* berfungsi untuk mengembalikan *player* ke bagian *Title Screen* di mana *main menu* berada.

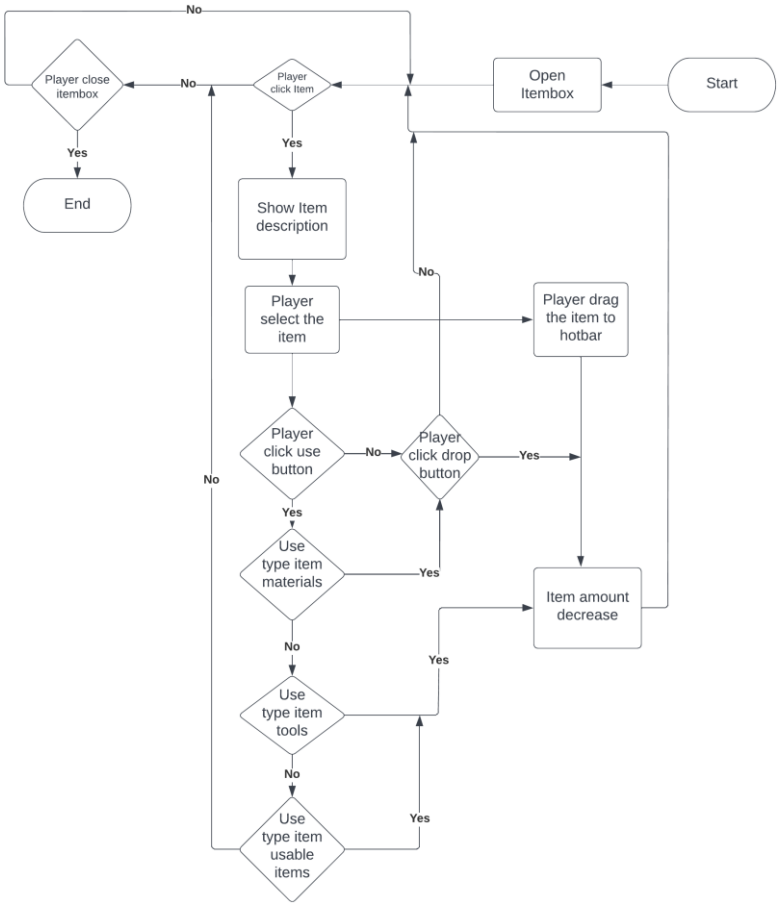
- Ketika *player* sementara berada di dalam *In-game, player* mempunyai *Hit Point* (HP) yang di mana jika berkurang hingga habis, maka karakter yang dikendalikan *player* akan mati dan dialihkan ke layar *Game Over*. Layar *Game Over* nantinya akan menunjukan 2 pilihan yaitu *Load* dan *Title screen*.

- Pilihan *Load* membuat *player* bisa melanjutkan keadaan *gameplay* yang telah di *save* oleh *player.* Untuk pilihan *Title Screen* bisa mengembalikan *player* ke bagian *Title Screen* di mana *main menu* berada.

- Jika *Hit Point* (HP) *player* masih ada dan belum habis, *player* masih dapat melanjutkan *game* "Island Escapist" hingga selesai dengan mencapai *ending* yang dimiliki oleh *game* ini.

3.2.5.1 Proses *Itembox*

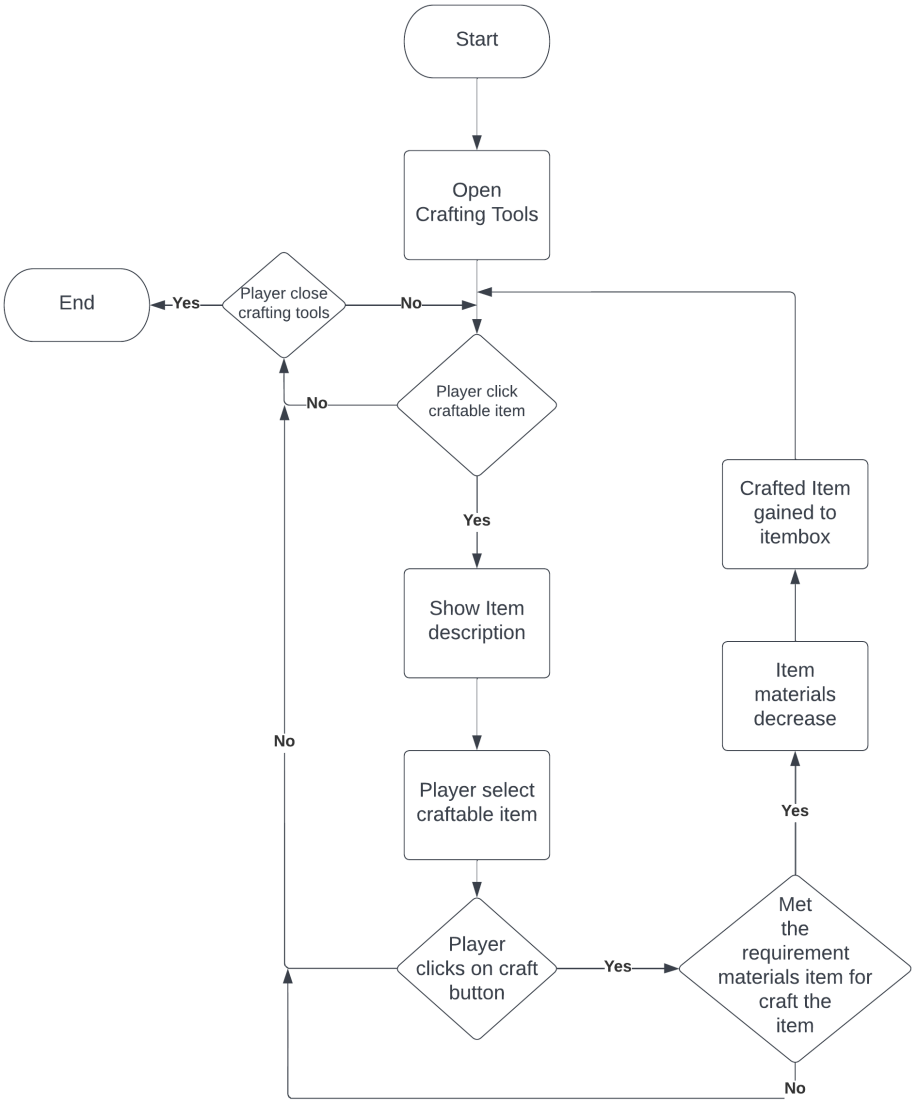
Proses ini dijalankan pada saat *player* ingin memakai atau melihat berbagai macam *item* yang diperoleh. Dengan menjalankan proses ini, tampilan *itembox* akan terbuka pada saat *player* sedang dalam kegiatan *in-game*. *Player* nantinya dapat menganti *item* yang digunakan ataupun dapat juga mengkonsumsi *item* tersebut berdasarkan tipe dan jumlah *item* yang tersedia di dalam *Itembox*.



Gambar 3.2. *Flowchart process itembox*

3.2.5.2 Proses *Crafting Tools*

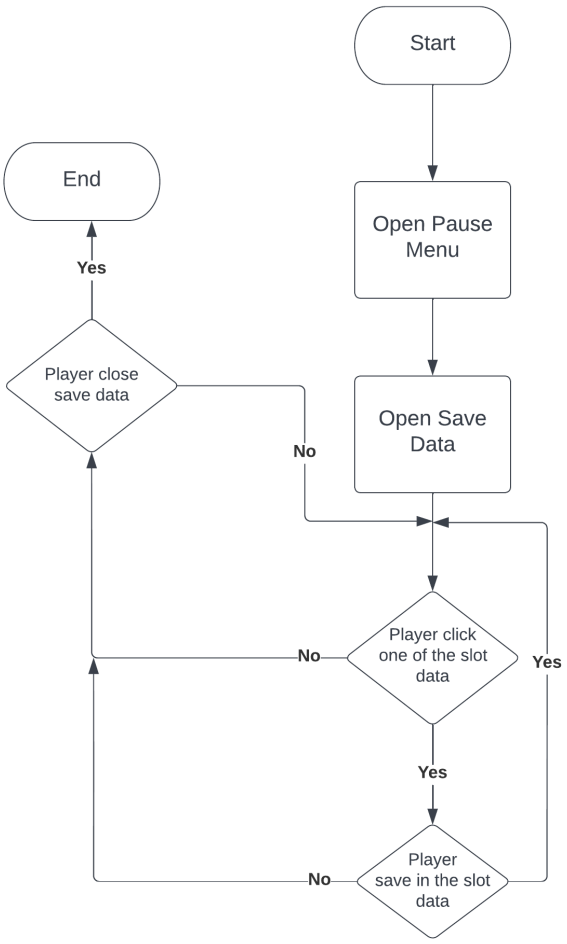
Proses ini dijalankan pada saat *player* akan melakukan *crafting* dengan menggunakan *item* bertipe *material*, di mana *item* yang dikumpulkan tersebut hanya dapat dipakai pada saat *player* akan melakukan *crafting*. Pada saat menjalankan proses ini, tampilan *crafting tools* akan terbuka pada saat *player* sedang dalam kegiatan *in-game*. *Player* nantinya bisa melakukan *crafting* untuk membentuk berbagai macam *item* yang mempunyai fungsi masing-masing dalam membantu *player.*



Gambar 3.3. *Flowchart process Crafting Tools*

3.2.5.3 Proses *Save Data*

Proses ini dijalankan pada saat *player* akan menyimpan *data in-game*. Untuk menyimpan *data*, *player* harus melakukan *pause* terlebih dahulu lalu pilih *menu save data*. Pada saat *save data* telah dibuka, *player* dapat melihat berbagai *slot* yang bisa *player* pilih untuk menyimpan *data*. *Data* tersebut kemudian akan tersimpan dan hanya bisa diakses Ketika *player* melakukan *load* pada *data* tersebut.

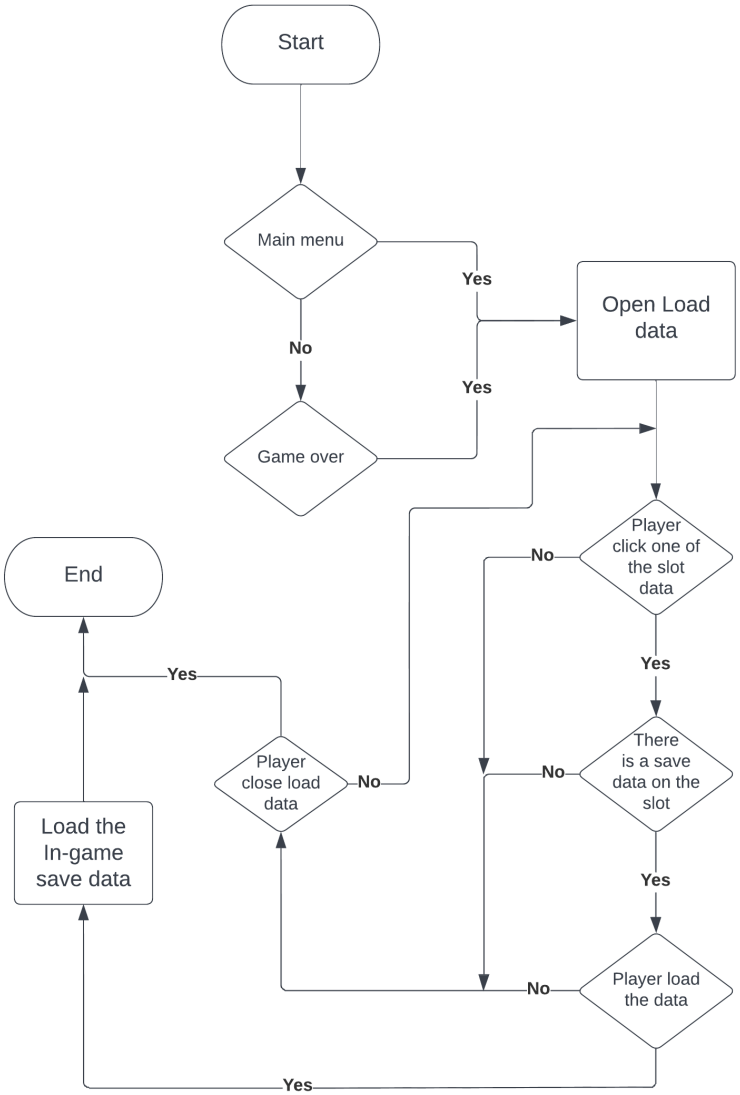


Gambar 3.4. *Flowchart process Save data*

3.2.5.4 Proses *Load Data*

Proses ini dijalankan pada saat *player* mau melanjutkan *progress in-game* yang telah disimpan melalui *save data*. Agar dapat melakukan *load* pada *save data*, *player* dapat membuka *load data* pada 2 *interface* yaitu *main menu* dan *game over*. Pada saat *player* membuka *Load data*, *player* dapat melihat berbagai *slot* yang terdapat pada layar *load data*. *Data-data* tersebut akan berisi dengan 2 tipe *data*, yaitu *data* yang masih

kosong dan *data* yang sudah pernah di *save*. *Data* yang sudah pernah di *save* biasanya mempunyai tanggal terakhir *player* menyimpan *data* tersebut yang terletak di pojok kanan bawah setiap *slot data*.



Gambar 3.5. *Flowchart process Load data*

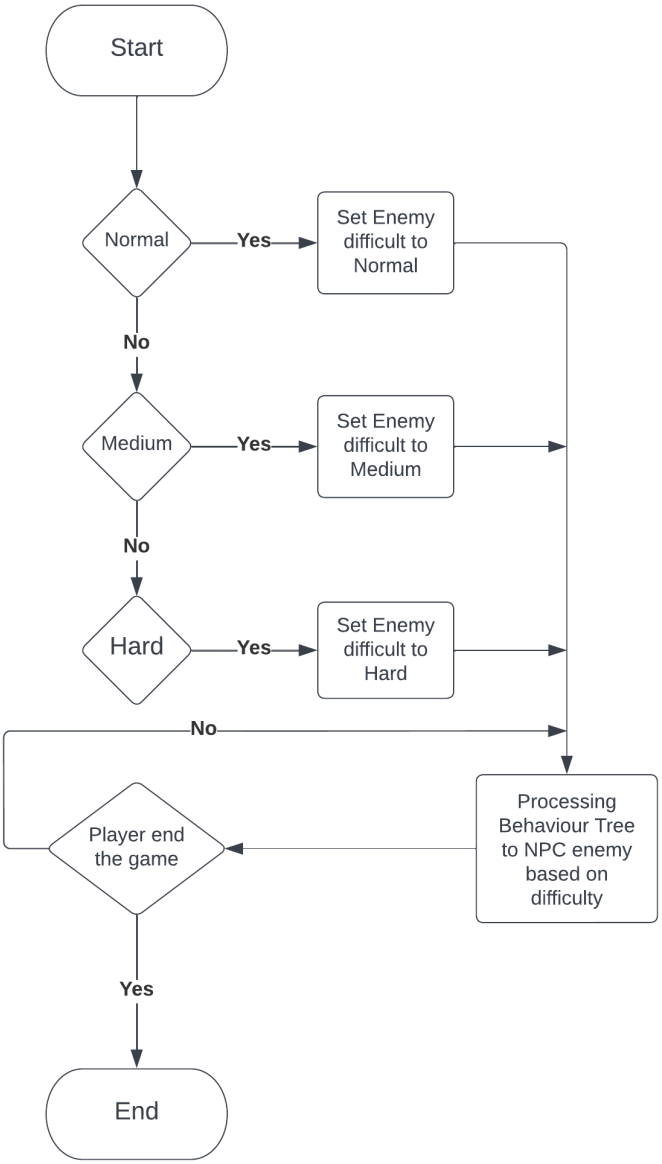
3.3 ***Artificial Intelligence* (AI) Design**

Pada sub bab ini, akan dijelaskan struktur AI yang akan digunakan. Dalam pembuatan *game* ini, AI yang digunakan dalam mengontrol pergerakan NPC *enemy* adalah *Behaviour Tree*. *Behaviour Tree* digunakan sebagai AI yang mengontrol cara perilaku pergerakan dan juga pengelihatan yang bertujuan untuk mengejar dan juga membunuh *player*. Selama *player* masih belum menyelesaikan *game*, maka *Behaviour Tree* dari NPC

akan tetap berjalan hingga *player* menyelesaikan *game* atau mati.

3.3.1 *Behaviour Tree*

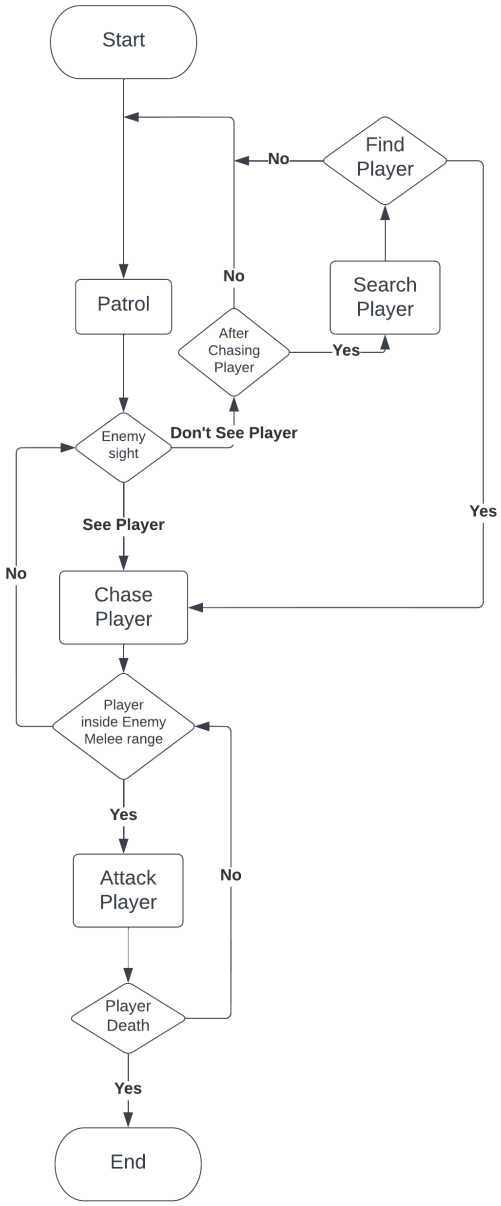
Behaviour Tree adalah salah satu metode yang sering digunakan dalam pembuatan AI yang ada pada *game*. *Behaviour Tree* yang dihasilkan nantinya akan diimplementasikan dalam AI NPC. Proses pembentukan *Behaviour Tree* akan berdasar dari sejumlah elemen-elemen yang ditetapkan yang dapat mengatur AI NPC, yaitu *condition, control flow,* dan juga *task*. Proses *Behaviour Tree* ini akan tetap berjalan selama NPC terdapat pada *game*. Hal ini dikarenakan proses *Behaviour Tree* akan terulang Kembali dari awal pada saat mencapai akar dari *Tree* tersebut, dan akan berpindah ke cabang yang lain jika kondisi dari cabang tersebut terpenuhi terlebih dahulu atau arah cabang tersebut berada pada bagian terkiri pada *Behaviour Tree*. Proses jalan *Behaviour Tree* akan berjalan dari cabang terkiri hingga cabang terkanan dari sebuah *tree.* Selama kondisi sebuah cabang masih dapat terpenuhi, ada kesempatan di mana sebuah *task* akan dilakukan berulang kali di karenakan kondisi dari cabang tersebut masih terpenuhi. Setelah kondisi dari sebuah cabang tidak lagi terpenuhi, maka akan berpindah ke cabang lain yang berada di sebelah kanan cabang sebelumnya hingga mencapai cabang terkanan dari sebuah *tree*. Jika kondisi dari cabang terkanan sudah tidak bisa dipenuhi lagi maka akan berpindah lagi ke cabang terkiri dari sebuah *tree*. Berikut ini adalah implementasi *Behaviour Tree* pada aplikasi *game*:



Gambar 3.6. *Flowchart* implementasi *Behaviour Tree* terhadap *game*

Pada Gambar 3.6. akan dijelaskan cara kerja *Behaviour Tree* terhadap aplikasi *game.* Awalnya *player* akan memilih salah satu dari 3 *difficulties* yang ada dalam *game*, di mana *difficulties* terbagi menjadi *Normal*, *Medium*, dan *Hard*. Setelah *player* memilih *difficulty*, kecepatan dan radius pengelihatan NPC *enemy* akan diatur sesuai *difficulty* yang dipilih dan diproses dengan menggunakan *Behaviour Tree.* Proses *Behaviour Tree* akan berakhir pada saat *player* mengakhiri *game* dengan cara mati atau menyelesaikan *game*.

3.3.2 Implementasi *Behaviour Tree* terhadap AI



Gambar 3.7. *Flowchart* implementasi *Behaviour Tree* terhadap NPC (*enemy*)

Pada Gambar 3.7. akan dijelaskan cara kerja *Behaviour Tree* terhadap *state* yang dimiliki oleh *enemy*. Awalnya *enemy* akan melakukan *patrol* disekeliling pulau. Jika *enemy* berhasil dalam melacak lokasi *player* dari pengelihatan *enemy,* maka *enemy* akan mulai mengejar *player.* Pada saat mengejar, jika *player* berada pada radius menyerang milik *enemy*, *enemy* akan langsung memberi *demage* kepada *player* jika *attack* yang diberikan oleh *enemy* mengenai *player.* Jika *hit point* yang dimiliki oleh *player* habis

dikarenakan *damage* yang diberikan oleh *enemy*, *player* akan langsung dialihkan ke layar *game over* dan *enemy* akan berhenti memberikan *damage* kepada *player*, di karenakan *player* telah mati. Jika seandainya *player* berhasil kabur dari radius penyerangan maupun pengelihatan *enemy,* maka *enemy* akan mulai mencari *player* dari tempat terdekat posisi *enemy*. Tetapi jika *enemy* telah mencari *player* dan tidak menemukan keberadaan lokasi *player,* maka *enemy* akan Kembali lagi pada *state patrol*.

3.4 Interface Design

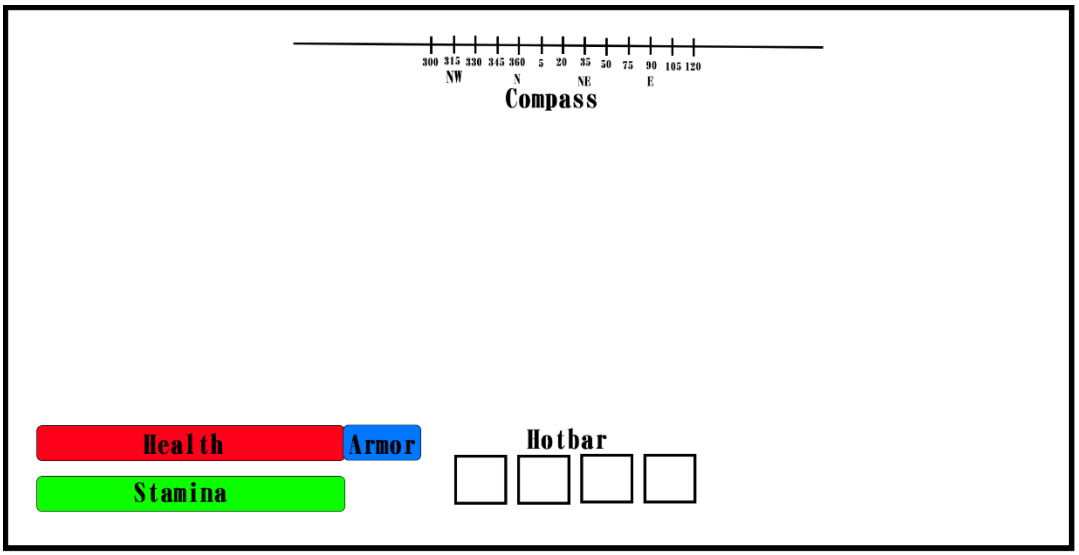
Pada sub bab ini, akan dijelaskan *interface* yang dimiliki oleh aplikasi *game* ini. *Interface* dibagi menjadi 2 modul yaitu *main menu* dan *playing*.

3.4.1 *Main Menu*

Gambar 3.8. Tampilan *Interface Prototype Main Menu Game*

Pada Gambar 3.8. akan terlihat tampilan *main menu* yang mempunyai 4 *menu*. *Menu* pertama yaitu *New Game, menu* kedua adalah *Load, menu* ketiga adalah *Options,* dan *menu* keempat adalah *Exit.* Setelah *player* memilih *button new game, player* akan di alihkan ke layar tampilan selanjutnya yang menunjukan 3 tingkat kesulitan yaitu *Normal, Medium,* dan *Hard*.

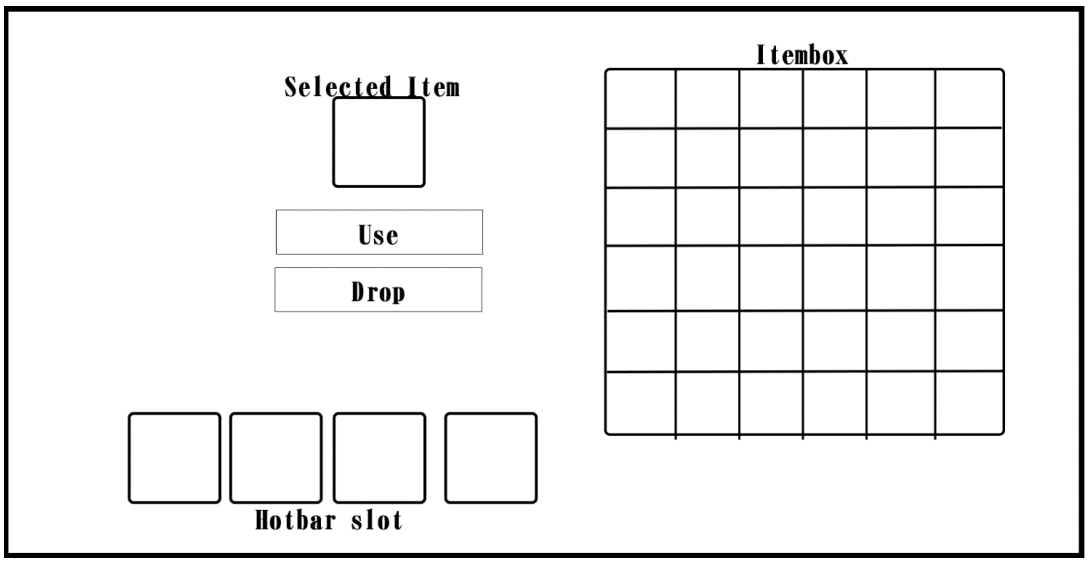
3.4.2 *Playing*



Gambar 3.9. Tampilan *Interface Prototype* Ketika di dalam *in-game*

Ketika *player* memulai *game* dengan menekan *button New Game* pada tampilan menu awal *game,* maka *player* akan di alihkan ke pada tampilan *interface game* seperti yang ada pada Gambar 3.9. Bisa di lihat bahwa *hit point, stamina,* dan *armor* terletak pada bagian pojok kiri bawah. Pada bagian tengah bawah terlihat 4 buah *hotbar*. Sedangkan untuk *compass*, bisa terlihat pada bagian tengah atas.

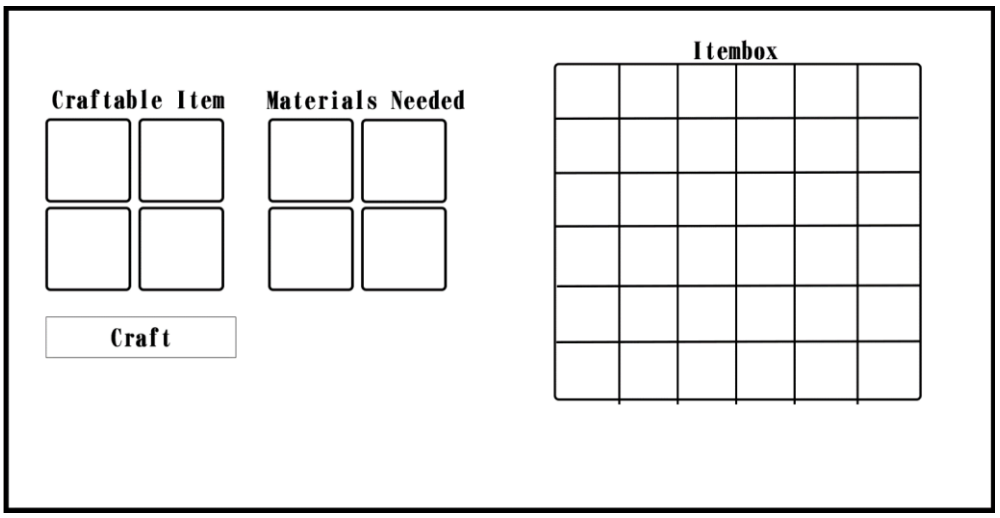
3.4.3 *Itembox*



Gambar 3.10. Tampilan *Interface Prototype Itembox*

Pada saat *player* membuka *itembox*, *player* akan diperlihatkan tampilan seperti pada Gambar 3.10. Di dalam tampilan layar *itembox* terdapat tempat untuk menyimpan berbagai macam *item.* Selain itu *player* juga dapat memegang *item* tersebut di tangan *player* dengan memilih *item* yang ingin dipilih dan tekan *menu use* yang ada di dalam layar tampilan. Dengan menggunakan *menu use, item* yang akan dipegang akan berada ditangan *player. Player* juga dapat membuang *item* yang tidak dibutuhkan dengan cara memilih *item* tersebut dan pilih *menu drop*. *Player* bisa juga memindahkan *item* pada lokasi *hotbar* dengan cara memilih *item* tertentu dan pilih *menu slot 1* untuk mengisi *hotbar*, di mana *player* dapat menggunakan *item* lebih cepat tanpa harus ke *itembox*.

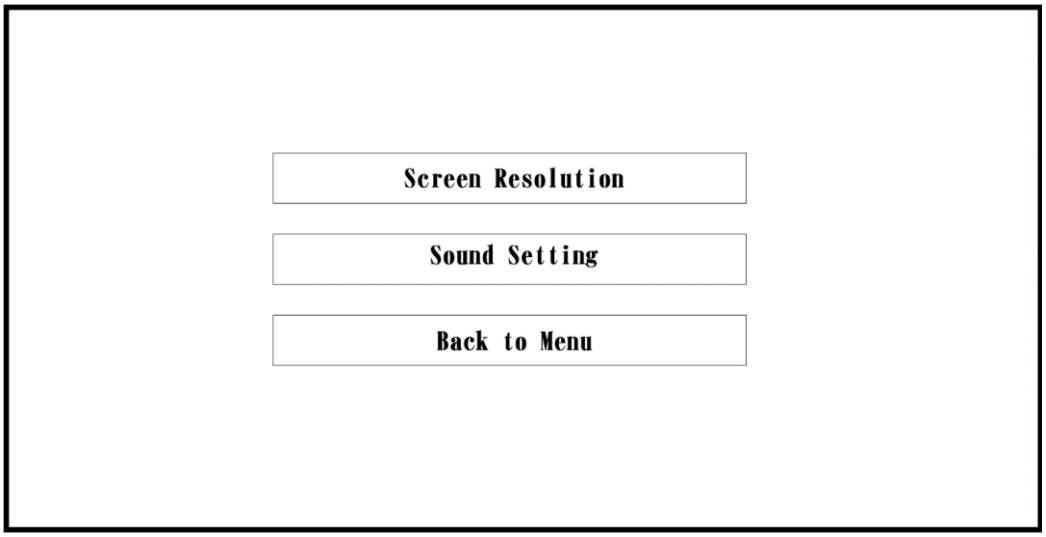
3.4.4 *Crafting Tools*



Gambar 3.11. Tampilan *Interface Prototype Crafting Tools*

Pada saat *player* membuka *Crafting Tools*, *player* akan diperlihatkan tampilan seperti pada Gambar 3.11. Di dalam tampilan layar *crafting tools* terdapat tempat untuk menampilkan berbagai macam *item* yang dapat di *craft.* Selain itu *player* juga dapat melihat berapa jumlah *material item* yang dibutuhkan pada saat melakukan *crafting* pada *item* yang tertentu. *Player* juga dapat melihat jumlah *item* yang tersedia pada *itembox*.

3.4.5 *Options*



Gambar 3.12. Tampilan *Interface Prototype Options*

Pada saat *player* membuka *Options*, *player* akan diperlihatkan tampilan seperti pada Gambar 3.12. Di dalam tampilan layar *Options* terdapat tempat untuk mengatur konfigurasi yang ada pada *game*. Di antaranya ada *screen resolution*, *sound setting*, dan *back to menu*. *Screen resolution* dapat membantu *player* dalam mengatur resolusi dari layar *game* yang dimainkan. *Sound setting* dapat membantu *player* dalam mengatur besar atau kecilnya suara. Dan yang paling terakhir adalah *Back to menu*, di mana dapat membantu *player* Kembali ke layar *menu* utama dari *game*.

3.4.6 *Game Over*

Gambar 3.13. Tampilan *Interface Prototype Game Over*

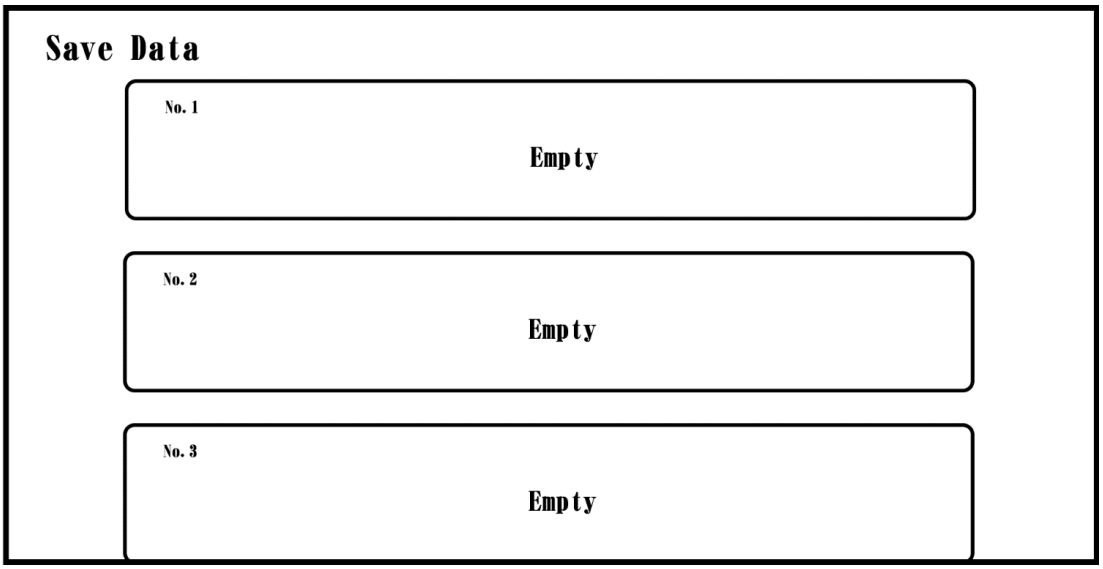
Ketika *player* menewaskan diri atau dibunuh oleh *enemy*, *player* akan diperlihatkan tampilan seperti pada Gambar 3.13. *Player* kemudian akan diberi pilihan untuk melakukan *load* atau *title screen*. Pada saat memilih *Load*, *player* akan menuju ke tampilan *menu load* yang di mana *player* dapat melanjutkan *game* dari hasil *save game* yang dilakukan oleh *player*. Untuk *Title Screen* mempunyai fungsi untuk membuat layar *player* Kembali tampilan layar *main menu.*

3.4.7 *Pause*

Gambar 3.14. Tampilan *Interface Prototype Pause*

Pada Gambar 3.14. adalah *menu pause* yang dimiliki oleh *player.* Pada saat *player* melakukan *pause* di dalam *game,* segala elemen yang ada pada *game* akan berhenti dan *player* akan ditunjukan sebuah *menu.* Di dalam tampilan layar *pause* terdapat beberapa *menu* berupa *resume game*, *save, options*, dan *title screen*. *Resume game* berfungsi untuk membuat segala elemen yang berhenti menjadi jalan Kembali, sehingga *player* dapat meneruskan *game*. *Save* mempunyai fungsi untuk menyimpan *data* agar *player* dapat melakukan *load*, di mana membantu *player* yang ingin menyimpan *progress* dari *in-game. Options* membantu *player* dalam mengatur konfigurasi yang ada pada *game. Title Screen* berfungsi untuk menampilkan layar *main menu.*

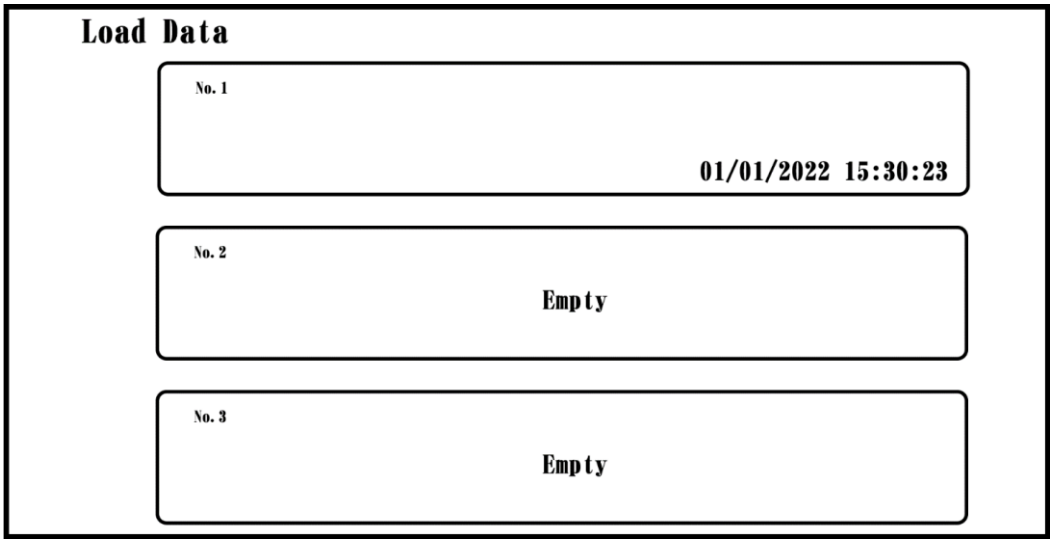
3.4.8 *Save Data*



Gambar 3.15. Tampilan *Interface Prototype Save Data*

Ketika *player* membuka *Save Data*, *player* akan diperlihatkan tampilan seperti pada Gambar 3.15. Di dalam tampilan layar *Save Data*, *player* dapat menyimpan *data* pada saat *player* tidak ingin bermain lagi atau ingin menyimpan *progress in-game. Data* tersebut nantinya dapat berguna jika *player* ingin melanjutkan *progress in-game* mereka melalui *Load Data*.

3.4.9 *Load Data*



Gambar 3.16. Tampilan *Interface Prototype Load Data*

Beda dengan *Save data,* kegunaan *Load data* adalah untuk melanjutkan atau mengakses *progress in-game* yang telah dilakukan *player.* Tampilan *Load data* bisa dilihat seperti pada Gambar 3.16. di mana akan terlihat *data* yang tersimpan mempunyai tanggal tercantum pada bawah pojok kanan *data* yang telah di *save*.