BAB III

METODE DAN PERANCANGAN SISTEM

3.1 Metodologi Penelitian

Metodologi penelitian yang akan digunakan untuk perancangan *game First Person Shooter* dengan *Procedural Content Generation* menggunakan teknologi *Virtual Reality* dijelaskan sebagai berikut.

a. Studi Literatur

Pada tahap ini akan dilakukan riset mengenai *teori-teori* yang akan membantu dalam perancangan dan pembangunan *game* Riset ini berupa pendalaman terhadap konsep *Procedural Content Generation* (PCG)*, game design*, dan pembangunan *game* dengan *virtual reality.*

b. Perancangan *Game*

Di tahap perancangan, dilakukan perancangan permainan dan pembuatan *Game Design Document.* Selain itu, pembuatan *flowchart*, desain *grammar*, aset visual, dan audio juga dilakukan pada tahap ini.

c. Pembangunan *Game*

Pada tahap ini dilakukan pembangunan *game* dengan menggunakan *game engine Unity* 2017 dengan target *platform* komputer dan operasi sistem *Windows.* Sesuai dengan rancangan *Game Design Document.*

d. Pengujian game

Pengujian dilakukan dengan *playtesting* oleh sejumlah respoden yang dipilih secara acak dan *professional game tester.* Responden yang dipilih secara acak akan mengisi kuesioner *GUESS* untuk mengumpul data tingkat kepuasan

pemain terhadap *game* sebanyak 30 responden, jumlah minimal untuk mendapat hasil yang baik menurut Sugiyono (2012). Responden juga akan mengisi kuesioner *Game Experience Questionnaire* untuk mengumpulkan data untuk evaluasi *narrative generator.* Selain itu juga akan ada pengujian untuk *game space generator*, dengan men-*generate game space* kemudian mengubahnya menjadi suatu *heatmap* beberapa kali, lalu dengan *expressive range* dapat diperoleh data *generator* untuk evaluasi.

e. Evaluasi Game

Evaluasi *game* akan dilakukan melalui analisa kuesioner *Game User Satisfaction Scale*, kuesioner *Game Experience Questionnaire*, dan *expressive range*. Data yang didapatkan dari *Game User Satisfaction Scale* akan dilakukan evaluasi untuk memberi hasil tingkat kepuasan pemain terhadap *game.* Data yang didapatkan dari *Game Experience Questionnaire* dan *expressive range* akan di evaluasi untuk memberi gambaran hasil *narrative* dan *game space* yang di-*generate.*

3.2 Perancangan Aplikasi

Berikut adalah perancangan untuk aplikasi yang dibangun.

3.2.1 Struktur Permainan

Judul Permainan: DISDAIN

Kata *disdain* berarti tidak merasa senang terhadap diri sendiri atau rasa kesal terhadap diri sendiri.

Formal Elements yang terdapat di dalam permainan dapat dijabarkan sebagai berikut.

1. Players

   Single Player Game: Pemain bermain sendiri.

2. Objectives

   Objektif utama dari *game* adalah pemain harus menyelesaikan semua level dan tantangan di setiap level sampai dengan level terakhir. Tantangan ini berupa 3 macam, yakni 1. Mengalahkan semua musuh di satu level, 2. Mencari pintu keluar di satu level, 3. Mencari semua telepon genggam pada satu level.

3. Procedures

   a. Ketika aplikasi dijalankan dan pemain melewati *splash screen* halaman pertama yang ditampilkan adalah *Main menu* atau halaman utama. Untuk memulai permainan pengguna harus memilih opsi *Campaign.*

   b. Apabila opsi *Campaign* dipilih maka akan muncul dua opsi *New Game* dan *Continue*. *New Game* untuk memulai permainan baru dan *Continue* untuk melanjutkan permainan sebelumnya, jika tidak ada permainan sebelumnya maka opsi ini tak akan bisa dipilih.

   c. Apabila opsi *New Game* dipilih maka pemain akan dibawa ke *scene* permainan. Pemain kemudian akan menerima sebuah cerita menceritakan *premise* dari *game*. Setelah itu pemain akan ditempatkan di suatu level yang di-*generate* dan akan menerima salah satu dari tiga objektif untuk level pertama. Pemain harus menyelesaikan objektif untuk dapat *progress* ke level selanjutnya.

d. Pada setiap level akan ada musuh yang akan menyerang pemain, untuk mengalahkan musuh pemain harus menggunakan senjata senapan dan menembak musuh sampai nyawa musuh habis dan musuh dikalahkan.
e. Jika nyawa pemain sama dengan 0 akan berpindah *scene* dimana akan ada opsi untuk mengulangi level atau keluar dari permainan.
f. Setelah menyelesaikan level terakhir pemain akan diberitahukan data permainan seperti seberapa banyak musuh yang telah dikalahkan dan waktu yang diperlukan pemain untuk menyelesaikan semua *level.* Setelah itu pemain akan dikembalikan ke halaman utama.

4. Rules
   a. Pemain harus memiliki Virtual Reality *Headset* untuk dapat menjalankan permainan.
   b. Pemain dapat menggunakan *keyboard* dan *mouse* atau *motion control* untuk memainkan permainan, pemain dapat mengganti kontrol pada halaman *Settings*.
   c. Setiap level akan memiliki cerita dan objektif yang di-*generate*, pemain harus menyelesaikan objektif yang diberi untuk dapat *progress* ke level selanjutnya.
   d. Ketika bermain ada dua *mode* untuk kontrol karakter yang dimainkan. *Mode* pertama adalah penggerakan, dimana pemain dapat menggerakkan aktor untuk maju, mundur, dan berotasi. *Mode* kedua adalah penembakan, dimana pemain akan mengeluarkan senapannya dan dapat menggunakannya namun tak bisa menggerakkan aktor.

e. Pemain bisa mengalahkan musuh dengan menembak musuh sampai nyawa musuh habis. Musuh juga dapat menyerang pemain dan tiap serangan yang mengenai pemain akan mengurangi nilai nyawa karakter pemain.
f. Pemain akan kalah jika nyawa karakter pemain sama dengan atau di bawah 0.
g. Jika pemain kalah akan berpindah *scene* dimana akan ada opsi untuk mengulangi level atau keluar dari permainan.
h. Pemain dapat berpindah level setelah menyelesaikan objektif pada level dimana pemain sedang berada.

5. Resource
   a. *Player Health:* Menunjukkan nilai nyawa pemain, nyawa akan ditampilkan dengan angka dan bernilai 100 pada awal permainan.
   b. *Ammunition:* Menunjukkan sisa amunisi yang ada pada senjata pemain.
   c. *Mini-map:* Peta yang menunjukkan lokasi pemain saat ini pada *dungeon.*
6. Conflict
   a. Menyelesaikan objektif setiap level dan tetap hidup dengan musuh mencoba untuk membunuh pemain.
7. Boundaries
   a. Ketika bermain pemain akan memiliki dua *mode* yaitu penggerakan dan penembakan. Dalam *mode* penggerakan, pemain hanya bisa menggerakkan karakter dan tak bisa menembak. Sebaliknya dalam *mode* penembakan, pemain hanya bisa menembak tapi tak bisa menggerakkan aktor. Dalam

kedua *mode* ini pemain masih dapat bisa melihat sekelilingnya dan membuka menu jika ingin keluar dari permainan.

b. *Game space* berupa *dungeon* dan pemain akan selalu berada di dalam *dungeon*.

8. Outcome

a. *Total Kills Score:* Pada akhir permainan akan diberikan skor total musuh yang telah dikalahkan oleh pemain.

b. *Time Played:* Waktu yang dibutuhkan pemain untuk menyelesaikan keenam level.

Berikut adalah *Dramatic Elements* yang terdapat di dalam permainan, antara lain.

1. Challenge

a. Mendapatkan *Total Kills* yang banyak.

b. Menyelesaikan *game* dengan cepat dan mendapatkan *Time played* yang rendah.

2. Play

a. *Rule-based play:* Permainan memiliki peraturan, manual (cara bermain), limitasi, dan instruksi.

3. Characters

Karakter yang dimainkan oleh pemain adalah mantan tentara bernama Mike. Mike adalah orang yang lumpuh dan memerlukan bantuan kursi roda untuk berjalan, karena dia kehilangan kedua kakinya.

4. Premise

Mike adalah seorang mantan tentara yang mengalami pengalaman buruk, pada masa aktifnya sebagai tentara tempat dimana dia ditugaskan diserang dengan rudal yang menyebabkan dia kehilangan kedua kakinya dan melihat teman-temannya meninggal. Pengalaman ini menyebabkan Mike tidak stabil dan tiap kali dia tidur dia mengalami mimpi buruk yang seakan-akan nyata.

5. Story

Mimpi buruk yang dialami Mike berbeda setiap kali dia tidur. Pada tiap mimpi yang dialami dia mencari cara untuk bangun dari mimpi buruknya dan di setiap mimpi selalu ada suatu entitas yang membantunya untuk keluar dari mimpi buruknya. Mimpi buruk inilah yang nanti akan di-*generate*. Simbol dan *rules* yang digunakan untuk membuat cerita dapat dilihat pada halaman lampiran dua. Desain simbol dan *rules* cerita bertujuan untuk memberikan pemain rasa terancam dan juga memberitahu pemain mengenai lingkungan *game*. Simbol pembukaan, deskripsi, trauma, dan rasa bertujuan untuk memberikan konteks terhadap lingkungan dan untuk memeberi rasa takut kepada pemain, simbol objektif bertujuan untuk memberikan objektif dari permainan seperti yang tertulis pada bagian *Objective* struktur permainan.

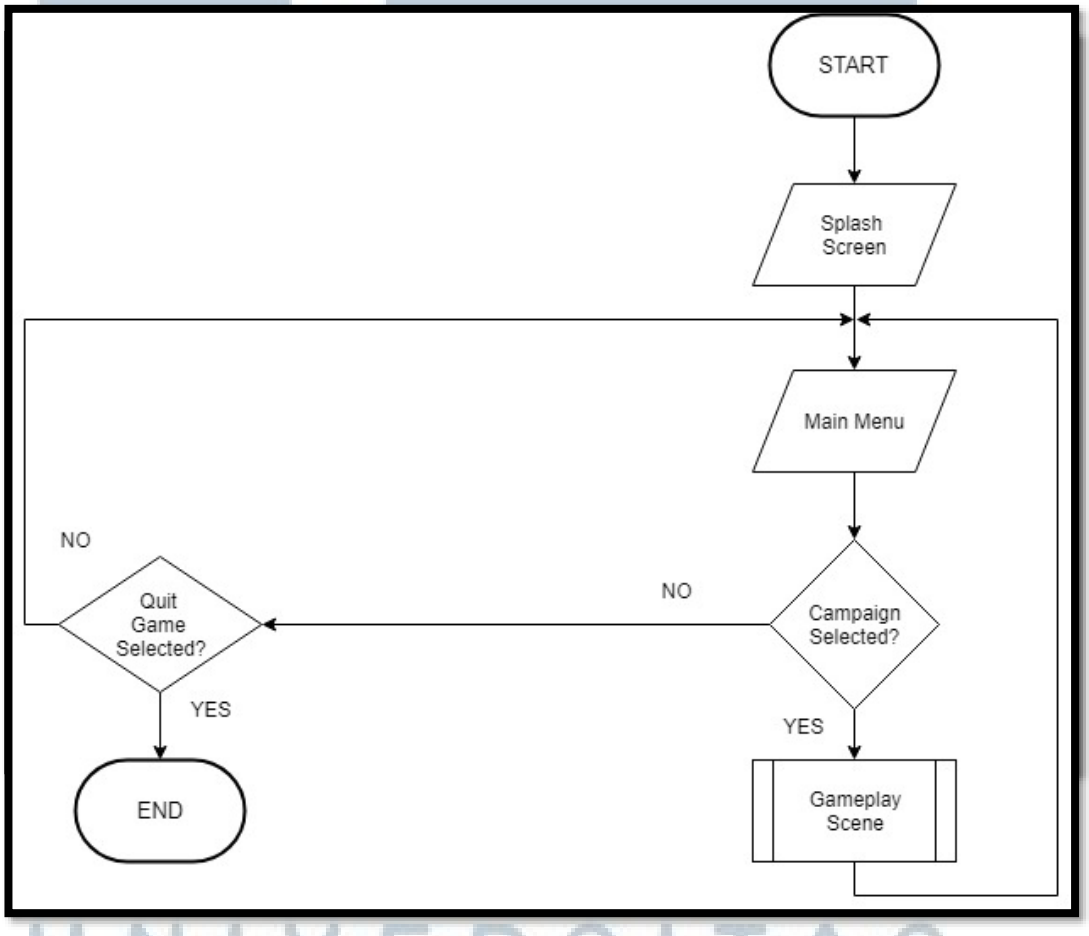
6. World Building

Game akan memiliki tiga tipe *setting*, masing-masing bertujuan untuk memberikan konteks pada cerita dan untuk memberikan rasa takut pada pemain.

1. Rumah sakit
2. Penjara bawah tanah
3. Rumah tua

3.2.2 Flowchart

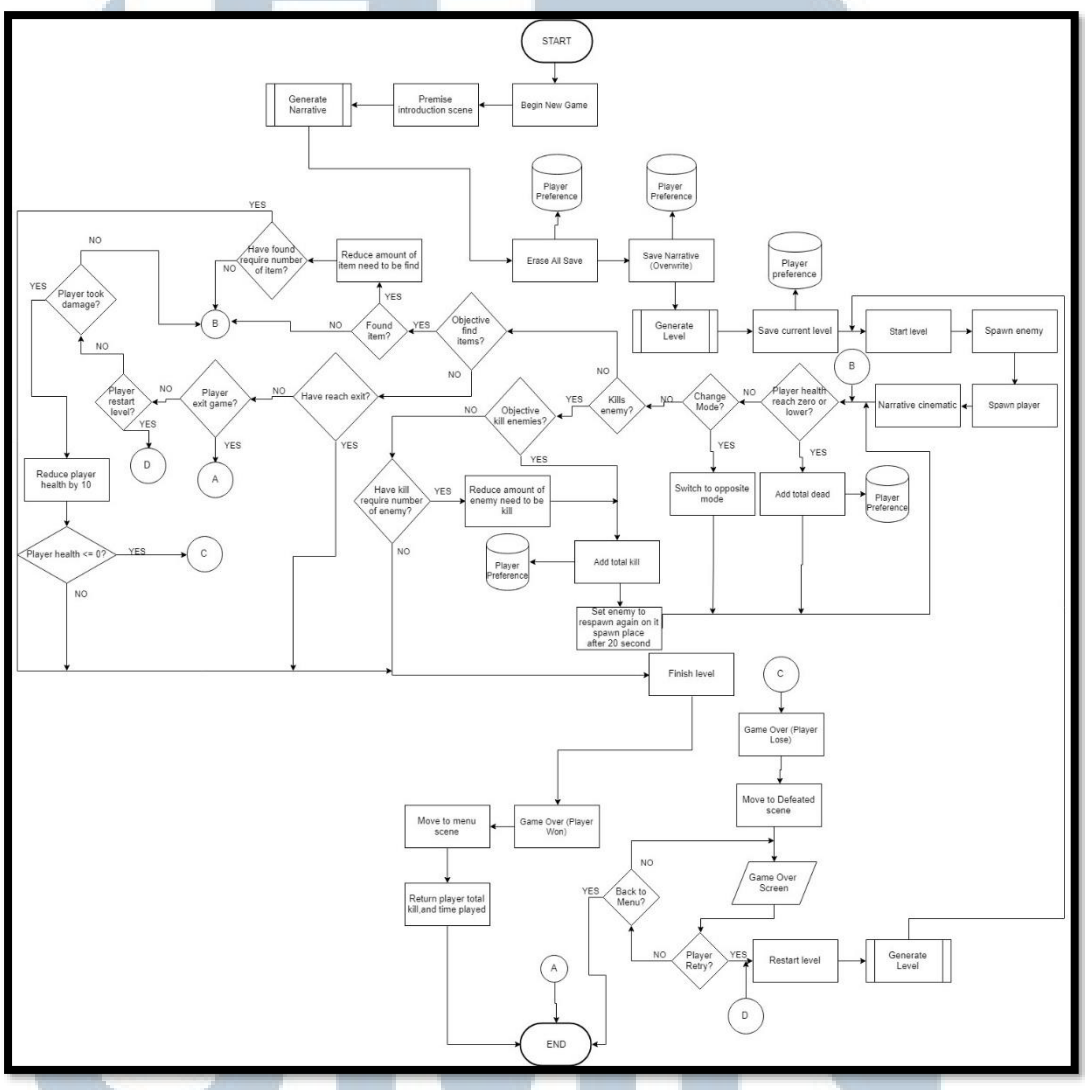
Dirancang *flowchart* sebagai landasan pengerjaan program. Berikut adalah perancangan game secara umum.



Gambar 3.1 Flow Aplikasi Secara Umum

Gambar 3.1 menunjukkan alur dari *game* secara umum, ketika pemain memulai *game* akan diperlihatkan sebuah *splash screen* kemudian pemain akan ditampilkan *main menu*. Di *screen main menu* pemain bisa memilih opsi untuk

memulai permainan dengan memilih *campaign*, pergi ke opsi *game* dengan memilih *setting,* atau keluar dari permain dengan memilih *quit.*



Gambar 3.2 Flowchart Gameplay Scene

Gambar 3.2 menunjukkan *game looping* dari *gameplay* setelah pemain memilih opsi *campaign* di *screen main menu.* Pada *screen* ini pemain bisa memilih opsi *new game* atau *continue*. Untuk *continue* tak akan bisa dipilih jika pemain tidak pernah memulai *campaign*. Ketika pemain memilih *new game* akan dimulai *scene introduction* dimana isi dari *scene* ini adalah menceritakan latar belakang

karakter yang pemain mainkan. Setelah *scene* ini selesai maka pemain akan pergi ke *scene level 1* dimana sebelum karakter pemain di-*spawn* akan di-*generate narrative* dan objektif pada *level* ini, gambaran men-*generate* cerita dapat dilihat pada Gambar 3. 4.

Setelah *narrative* dan objektif dibuat, akan disimpan *narrative* (*narrative* dan objektif dianggap sebagai satu *narrative*) dengan *player preference*. Sebelum itu, *narrative* yang sudah berada di dalam *player preference* akan dibuang terlebih dahulu. Setelah itu barulah *game space* di-*generate.* Gambar 3. 5 menunjukkan cara pembuatan *game space.*

Setelah *game space* dibuat pemain dan musuh akan di-*spawn*. Pemain kemudian akan mendapatkan *narrative cinematic* dimana cerita yang di-*generate* sebelumnya akan diceritakan secara *dialogue*, setelah itu barulah pemain dapat mengontrol karakter di dalam *game.* Dalam kontrol karakter ada dua macam *mode* seperti yang dijelaskan pada *Game Design Document* dan pemain bisa mengganti *mode* kapan saja ketika sedang bermain.

Tiap *level* memiliki objektif yang belum tentu sama, jika pemain berhasil membunuh musuh akan dicek apakah objektifnya adalah membunuh musuh. Jika iya maka *requirement* berapa banyak musuh yang perlu dibunuh akan dikurangi dan jika *requirement* sudah terpenuhi maka pemain akan menyelesaikan *level*. Setiap kali musuh dibunuh, musuh akan disiapkan untuk *spawn* kembali pada tempat asal *spawnnya* dengan jeda waktu 20 detik, juga akan ditambahkan 1 poin ke dalam jumlah musuh yang telah pemain bunuh.

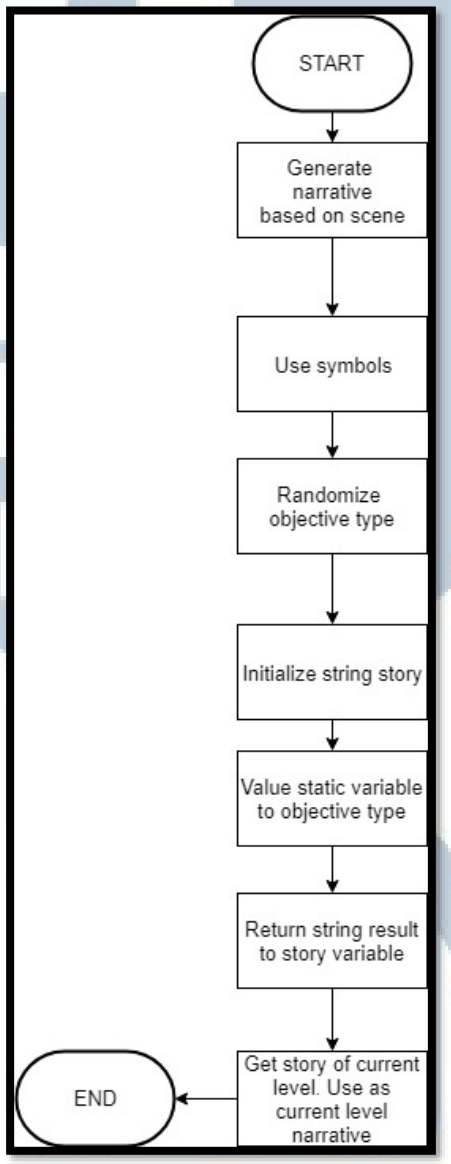
Jika objektif pemain adalah mencari *item* atau benda maka akan dicek apakah pemain telah berhasil menemukan benda atau belum. Jika pemain menemukan benda maka akan di cek *requirement* benda yang dibutuhkan untuk menyelesaikan *level.* Jika sudah memenuhi kebutuhan, maka pemain akan menyelesaikan *level.* Setiap benda yang ditemukan akan mengurangi *requirement* benda yang perlu dicari untuk menyelesaikan *level.*

Jika objektif pemain adalah mencari pintu keluar, akan dicek apakah pemain sudah menemukan pintu keluar. Jika sudah maka pemain akan menyelesaikan *level*. Selebihnya pemain dapat keluar dari *game* atau mengulang *level* kapan saja pemain inginkan.

Jika pemain terserang oleh musuh maka nyawa karakter pemain akan berkurang sebesar sepuluh. Jika jumlah nyawa pemain sudah lebih atau sama dengan nol, maka pemain akan pergi ke *scene game over defeated*, dimana pemain memiliki opsi untuk mengulang *level* dimana kalau dipilih akan melakukan men-*generate game space* yang baru dengan objektif dan *narrative* yang sama. Opsi yang satunya lagi adalah keluar dari permainan dan kembali ke *main menu.*

Namun, jika pemain berhasil menyelesaikan *level* dengan memenuhi syarat objektif maka pemain akan berlanjut ke *level* selanjutnya, tapi kalau pemain berada di *level* terakhir dan sudah menyelesaikan objektif pada *level* itu maka pemain akan pergi ke *scene game over victory* dimana akan dimunculkan statistik jumlah bunuh dan waktu permainan pemain dari awal *level* sampai dengan akhir *level*. Setelah itu pemain dapat pergi ke halaman *main menu* dengan memilih *continue* dimana sebelum pergi ke halaman *main menu* akan dihapus seluruh data permainan

pemain, ini dilakukan agar pemain tak bisa memilih *continue* ketika berada di *screen campaign.*

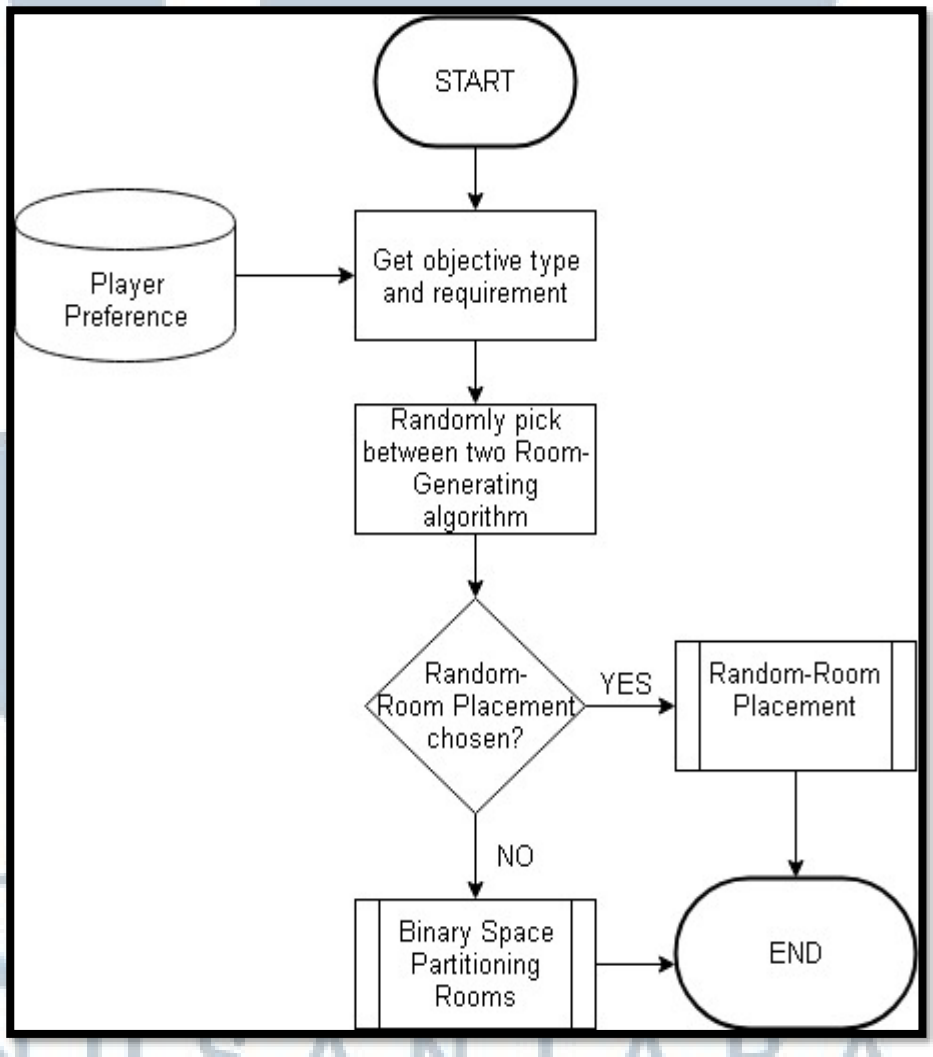


Gambar 3.3 Flowchart Implementasi Grammars

Gambar 3.3 menunjukkan cara kerja dari *Grammars* untuk men-*generate narrative* serta objektif. Pertama dipilih secara *random* antara tiga objektif yang

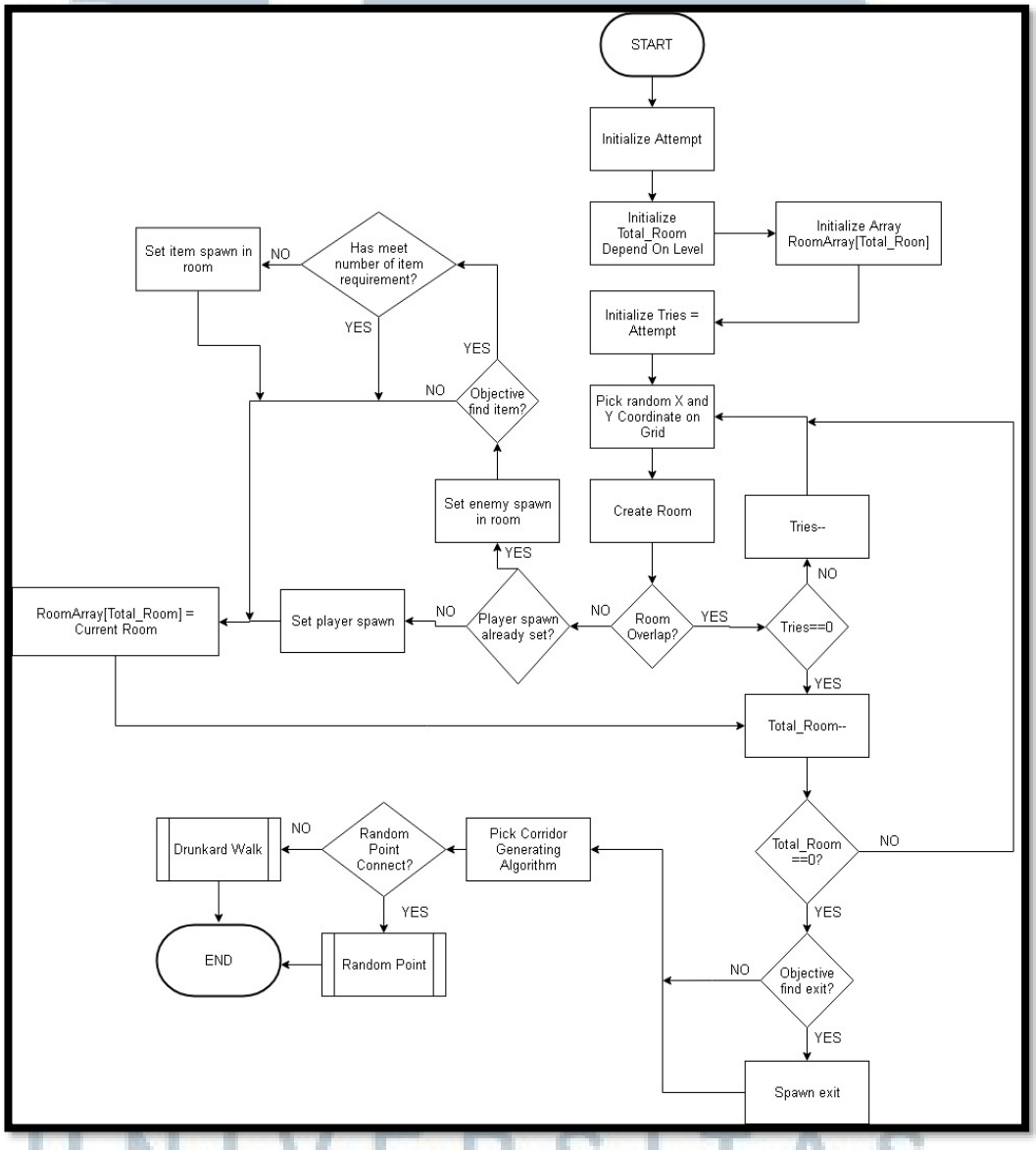
akan dilakukan pada *level*, tergantung objektif apa yang dipilih *requirement* untuk menyelesaikan objektif itu akan berbeda.

Kemudian setelah objektif dibuat, akan di-*generate* cerita pada *level*, tergantung pada *level* mana pemain, *rules* dan *symbols* yang akan digunakan untuk men-*generate* cerita akan berbeda. *Rules* dan *symbols* yang akan digunakan untuk setiap *level* dapat dilihat pada *Game Design Document*. Setelah *narrative* di-*generate* maka akan berlanjut ke proses berikutnya, seperti yang bisa dilihat pada Gambar 3.2.



Gambar 3.4 Flowchart Generate Game Space

Pada tahap men-*generate game space* akan diambil *narrative* yang berserta cerita, objektif, dan objektif *requirement*, kemudian akan dipilih secara acak algoritma mana yang akan digunakan untuk men-*generate Room*.



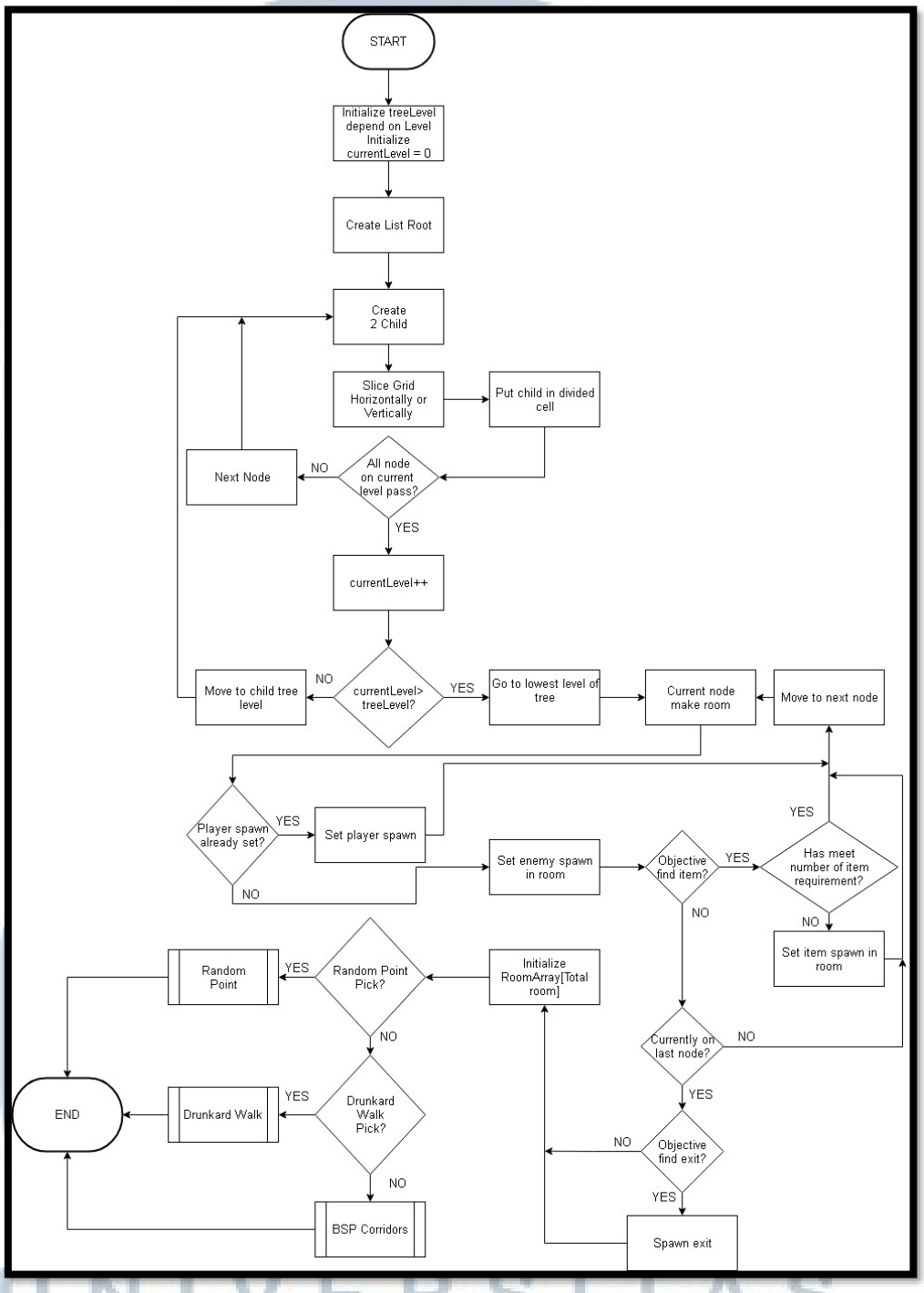
Gambar 3.5 Modul Random-Room Placement

Gambar 3.5 menunjukkan cara kerja dari algoritma *Random-Room Placement* untuk pembuatan ruangan. Secara singkat *room* di-*generate* dengan

memilih titik *random* di *grid* yang merupakan *boundaries* dari besarnya satu *level,* kemudian akan diperiksa jika ruangan yang dibuat bertabrakan dengan ruangan lain. Jika iya akan dicoba mencari posisi lain untuk membuat ruangan. Proses mencari titik, membuat ruangan, mengecek apakah bertabrakan dengan ruangan lain atau tidak, dapat dilakukan beberapa kali, sampai jumlah *tries* untuk satu antrian pembuatan ruangan habis. Jika *tries* habis atau berhasil dibuat maka akan berlanjut ke antrian pembuatan ruangan selanjutnya.

Tapi sebelum pergi ke antrian berikutnya, akan disiapkan beberapa hal yang berkaitan dengan objektif pada *level* saat ini. Pertama akan disiapkan *Player spawn* untuk letak pertama pemain ketika bermain di *level,* jika itu sudah pernah dilakukan akan disiapkan *enemy spawn* tempat musuh akan muncul di ruangan. Jika objektif pemain berupa mencari *item* atau benda maka akan disiapkan juga tempat *spawn* untuk benda namun jika sudah memenuhi *requirement* jumlah *item* yang akan dicari pada *level* ini maka tak akan disiapkan lagi.

Jika antrian ruangan yang akan dibuat sudah sampai pada ruangan terakhir, ketika ruangan terakhir selesai dibuat akan dicek apakah objektif pemain berupa mencari pintu keluar, jika iya maka akan disiapkan *spawn* pintu keluar. Setelah itu selesai akan dipilih secara *random* antara algoritma *corridor generator* apa yang akan digunakan.

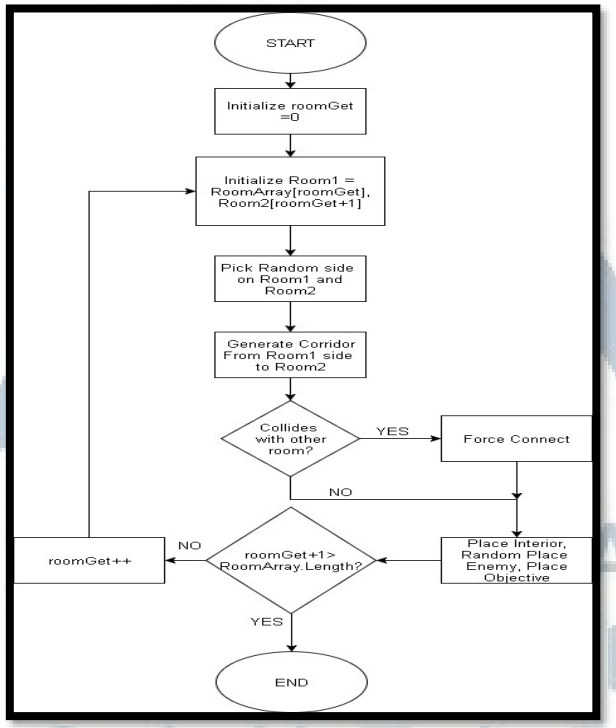


Gambar 3.6 Modul Binary Space Partitioning Rooms

Gambar 3.6 menunjukkan cara kerja *Binary Space Partitioning Rooms,* algoritma yang akan digunakan untuk melakukan men-*generate room.* Secara

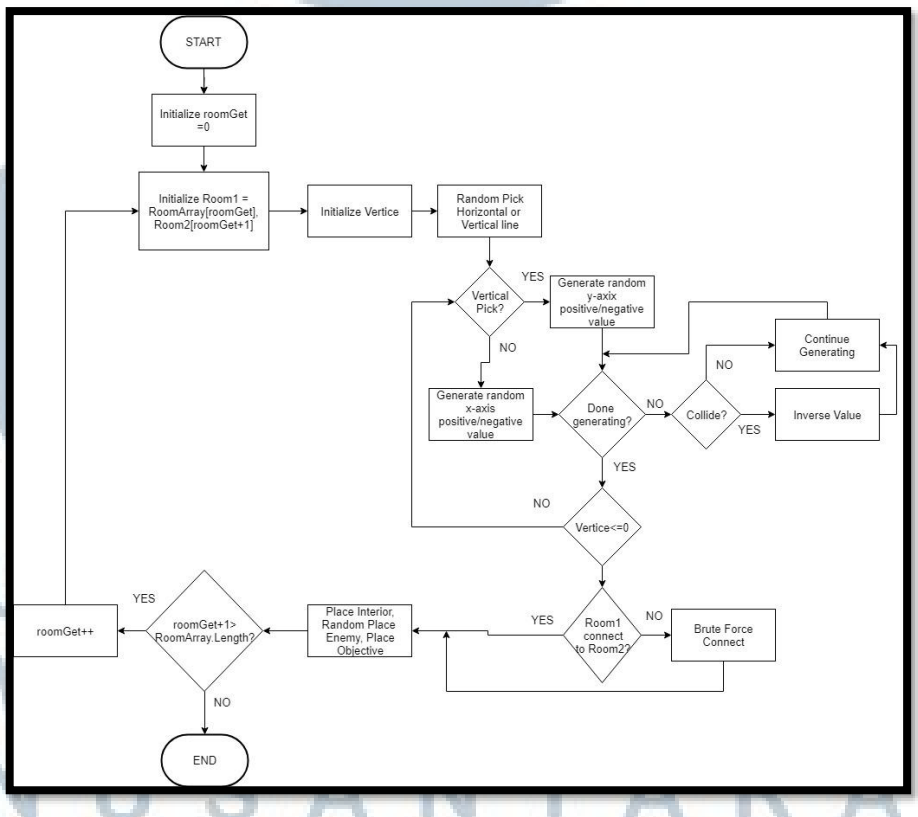
singkat algoritma ini berkerja dengan menganggap satu *grid* sebagai *root tree* dimana kemudian akan dibuat dua *child* dengan membelah *grid* secara horizontal atau vertikal. Setelah itu *child* yang dibuat juga akan memiliki dua *child* dengan membelah wilayah miliknya secara horizontal atau vertikal, *loop* ini akan terus berlanjut sampai *level depth tree* mencapai batas yang ditentukan.

Setelah proses itu selesai, maka akan dibuat ruangan di *child* paling bawah atau *depth tree* terendah. Pembuatan ruangan kemudian juga akan diolah dengan objektif pada *level,* sama dengan yang dilakukan dengan *Random Room* pada Gambar 3. 6, dimana akan disiapkan *player spawn* kalau belum ada, siapkan *enemy spawn*, siapkan *spawn item* jika itu adalah objektif pada *level*, dan siapkan *spawn* pintu keluar di *node* terakhir. Setelah itu akan dipilih algoritma yang akan digunakan untuk menyatukan ruangan-ruangan.



Gambar 3.7 Modul Random Point

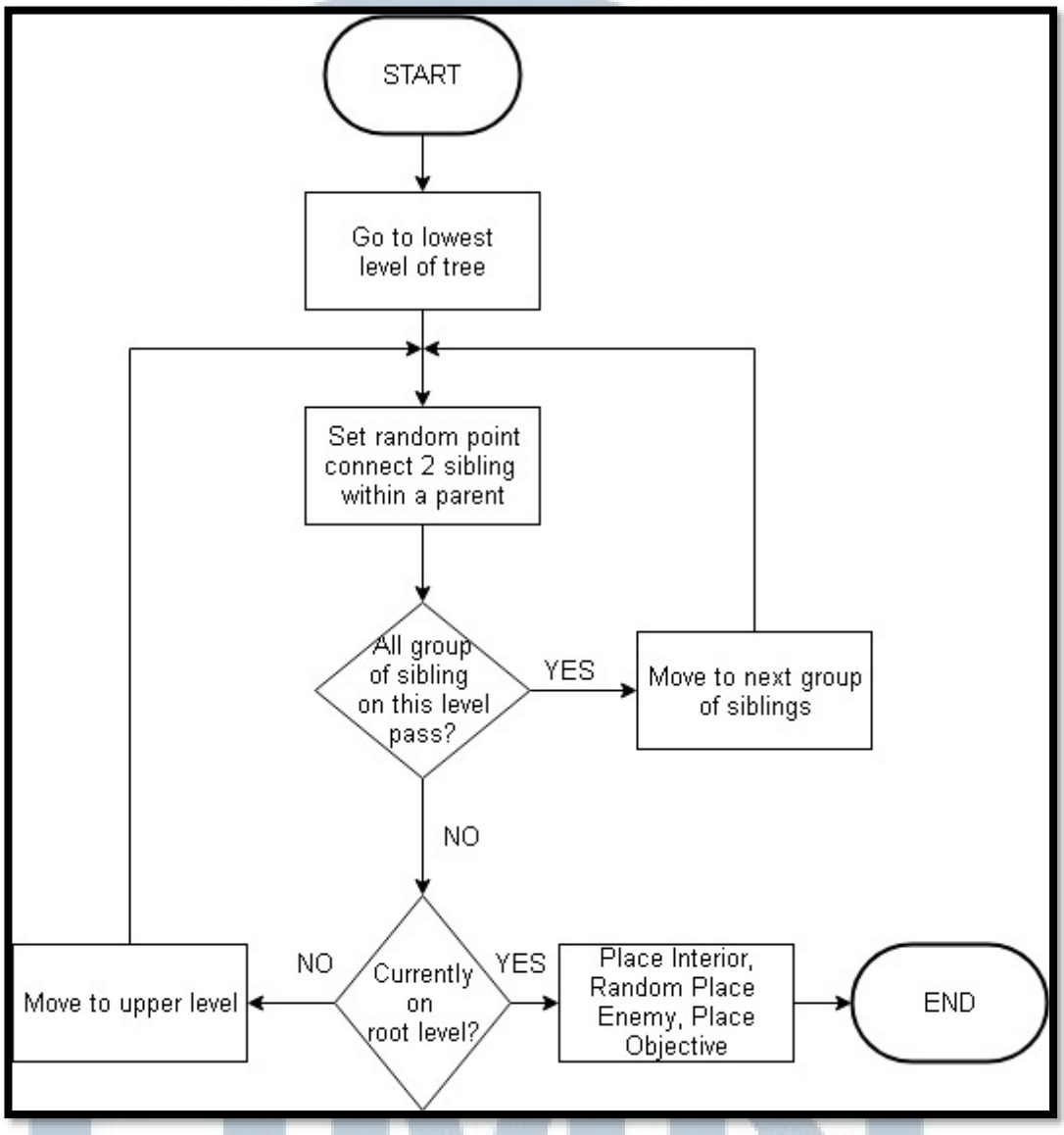
Gambar 3.7 menunjukkan pembuatan *corridors* dengan *Random Point.* Secara singkat, algoritma ini berkerja dengan menyambung dua titik secara *brute force.* Dalam implementasi algoritma ini, akan diambil secara *index* dua *room* secara berurutan, kemudian dari ruangan yang telah dipilih akan dipilih sisi ruangan secara *random*, kemudian sisi ruangan pertama akan membuat *corridor* ke sisi ruangan kedua secara langsung yaitu dengan melihat sejauh mana titiknya di *grid* kemudian memberikan nilai *x-axis* dan *y-axis* yang sesuai untuk menyambung. Jika dalam proses *corridor* bertabrakan dengan ruangan lain, maka secara *brute force* akan disambungkan. Proses ini dilakukan sampai melebihi batas *length array* ruangan. Setelah selesai akan di spawn dekorasi ruangan, posisi *enemy* di-*spawn*, dan posisi objektif di-*spawn.*



Gambar 3.8 Modul Drunkard Walk

Gambar 3.8 menunjukkan kerja algoritma *Drunkard Walk* atau juga biasa dikenal dengan *Random walking*. Dalam implementasi algoritma ini akan diambil *index* dari ruangan yang berhasil dibuat kemudian dipilih secara berurutan pasangannya. Ruangan pertama dan ruangan kedua akan dipilih sisinya secara *random* untuk menyambungkan *corridor*. Pembuatan *corridor* ini dimulai dari ruangan, pertama kali dilakukan adalah menentukan jumlah *vertices*. *Vertices* ini merupakan jumlah seberapa kali *corridor* akan mengubah arah perjalanannya. Sisi ruangan pertama yang akan disambungkan ke ruangan kedua dianggap sebagai *vertice* pertama, dari *vertice* pertama akan dipilih secara *random* arah kemana *corridor* akan pergi berserta jarak perginya. Contohnya di *vertice* pertama didapatkan arah pergi *x-axis* positif atau *east* istilahnya, kemudian jumlah jarak yang akan ditempuhnya, setelah *corridor* di-*generate* sejumlah jarak yang ditentukan maka posisi dimana *corridor* berada dianggap sebagai *vertice* selanjutnya. Prosesnya berulang lagi sampai jumlah *vertice* sudah sampai batas, jika saat pembuatan *corridor* bertabrakan dengan ruangan lain atau *border* maka sisa nilai tempuhnya akan di-*inverse.*

Setelah itu, akan dicek apakah *corridor* sudah mencapai tujuan, jika belum maka secara *brute force* akan disambung posisi *corridor* saat ini dengan tujuan, mirip dengan apa yang dilakukan oleh *Random Point* pada Gambar 3.7. Setelah tersambung maka proses ini akan berulang sebanyak jumlah ruangan yang ada. Setelah selesai maka akan ditempatkan dekorasi, posisi *spawn musuh*, dan posisi *spawn object.*



Gambar 3.9 Modul BSP Corridors

Binary Space Partitioning Corridors merupakan algoritma pembuatan *corridor* yang hanya bisa diimplementasi dengan *pairing* BSP *Room.* Cara kerjanya adalah pergi ke *depth* terendah kemudian semua *child* di satu *parent* akan disambungkan dengan proses yang sama dengan *Random Point*, yaitu dengan memilih sisi kedua ruangan secara *random* kemudian secara *brute force* menyambungkannya.

Jika semua *child* sudah terpasang, maka *level* akan berganti dan proses yang sama akan terulang lagi, yaitu dengan memilih sisi dari salah satu ruangan di satu *node* ke sisi salah satu ruangan di *node sibling* dan menyambungkannya secara *brute force.*

3.2.3 Penggunaan Asset

Tabel 3.1 Asset Game

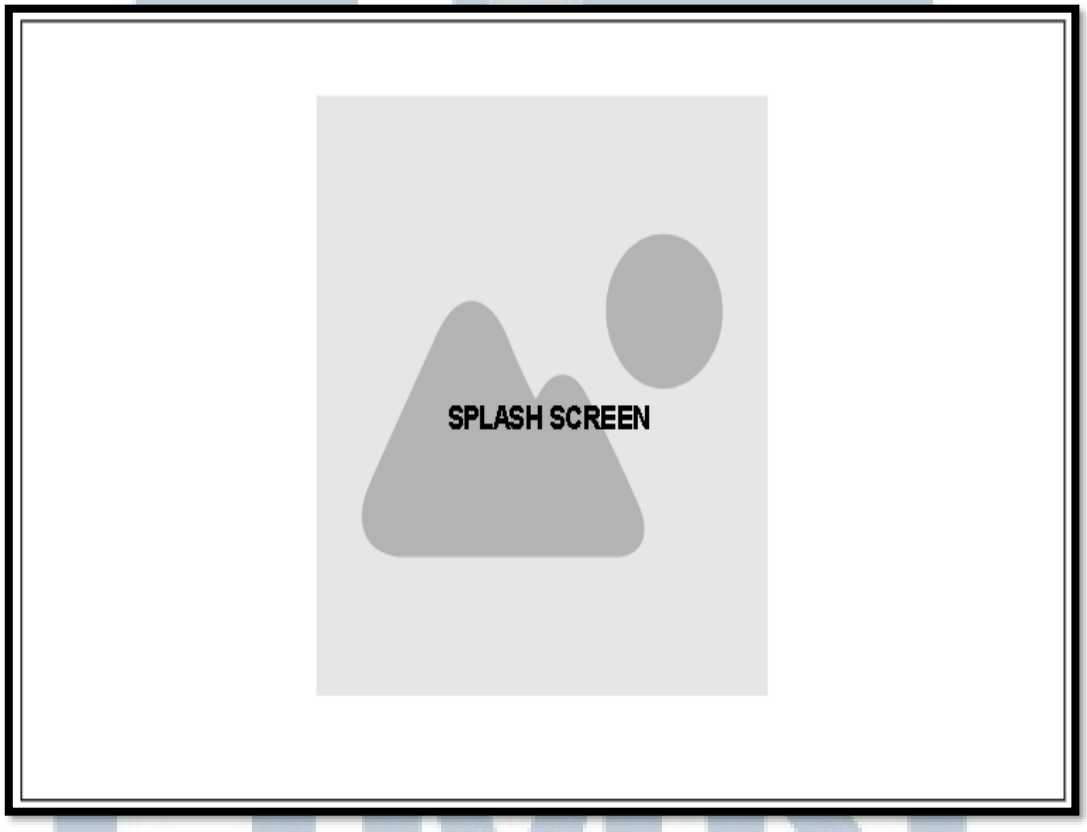
1. 3D model Kursi roda	Author: Slande, 2014 Source: https://free3d.com	
2. 3D model karakter Zombie	Author: Pxltiger, 2015 Source: https://www.assetstore.unity3d.com	
3. 3D model karakter tentara	Author: Unknown Artist Source: www.mixamo.com	
4. 3D model karakter monster	Author: Unknown Artist Source: www.mixamo.com	
5. Enviroment Hospital	Author: FoeJofMay Source: https://www.assetstore.unity3d.com	

Tabel 3.2 Asset Game (Lanjut)

6. Enviroment Dungeon	Author: Prodigious Creations Source: https://www.assetstore.unity3d.com	
7. Interior Objects	Author: Gesta2 Source: https://www.assetstore.unity3d.com	
8. 3D model senjata rifle	Author: 7XF Design Source: https://www.assetstore.unity3d.com	
9. 3D model telepon	Author: Vertext Studio Source: https://www.assetstore.unity3d.com	
10. Efek dan Musik	Seluruh music berasal dari freesound.org dan freemusicarchive.org dan dapat digunakan untuk kepentingan riset atau komersial	

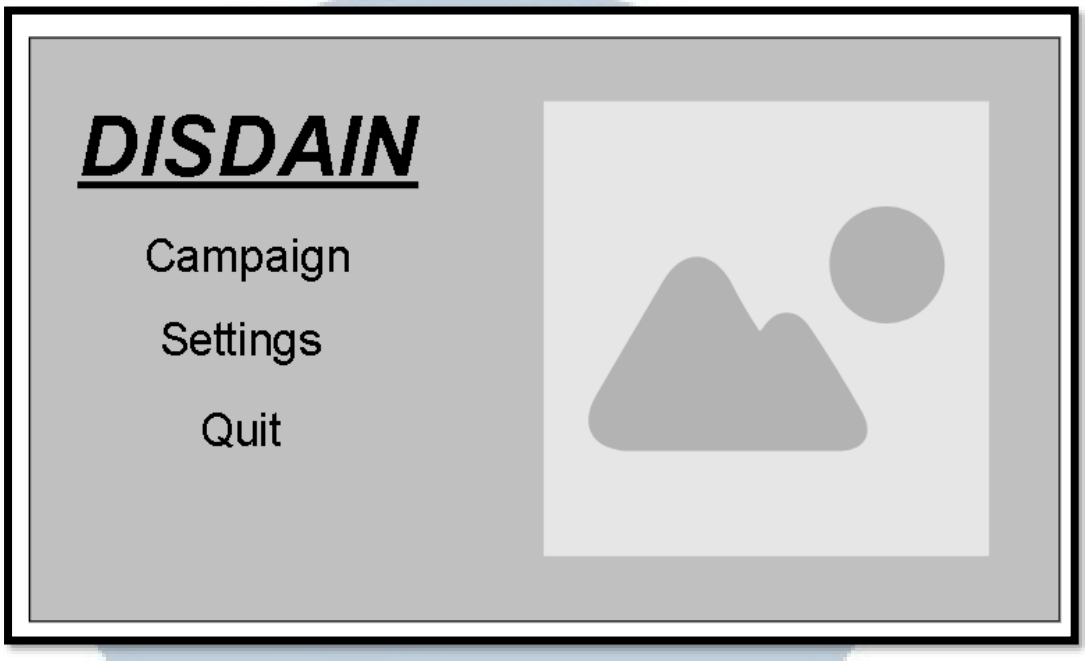
3.2.4 Desain Mockup

Berikut adalah tampilan desain *Mockup* untuk aplikasi *game.* Gambar 3.12 menunjukkan *splash screen* pada awal aplikasi ketika *game* baru dimulai atau baru dijalankan.



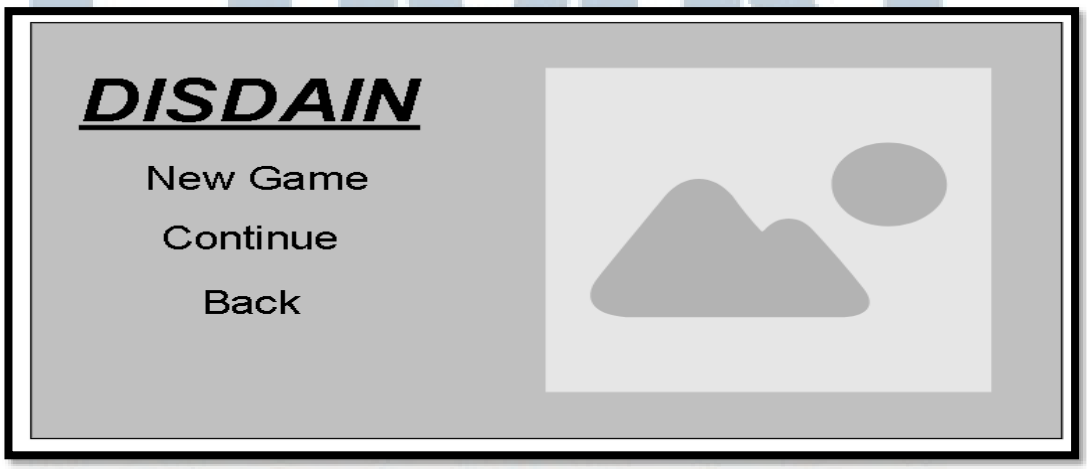
Gambar 3.10 Tampilan Splash Screen

Gambar 3.10 adalah desain *mockup* untuk tampilan utama *game*. Dari sini pemain bernavigasi pertama kali dan di sinilah gambaran dari proses *flowchart* pada Gambar 3.1.



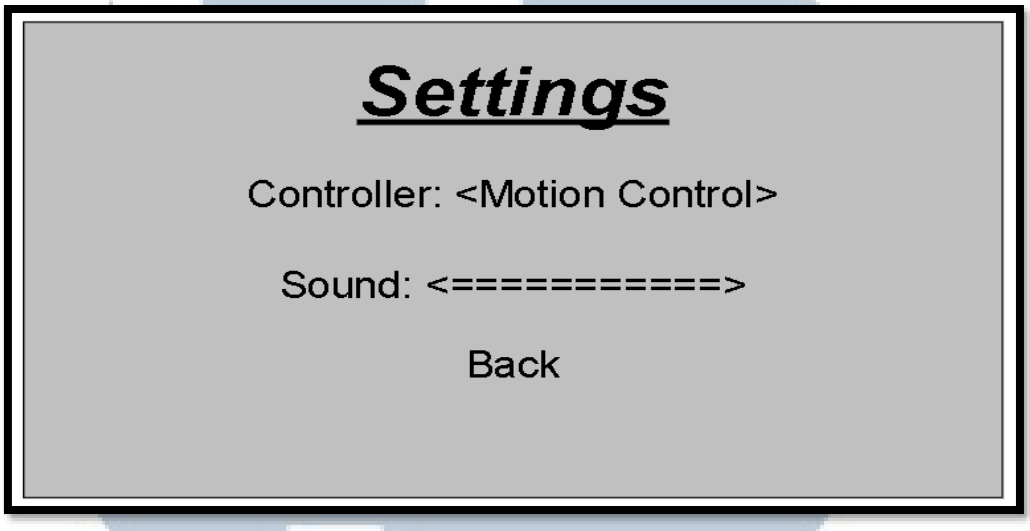
Gambar 3.11 Tampilan *Main Menu*

Gambar 3.11 adalah desain *mockup game* ketika pilihan *Campain* dipilih, dimana seperti pada *flowchart* Gambar 3.2 pemain akan ditampilkan opsi untuk memulai *game* baru atau melanjutkan *game* sebelumnya. Opsi *continue* atau melanjutkan *game* tak akan bisa diklik atau dihitamkan jika tak ada data mengenai *game* sebelumnya.



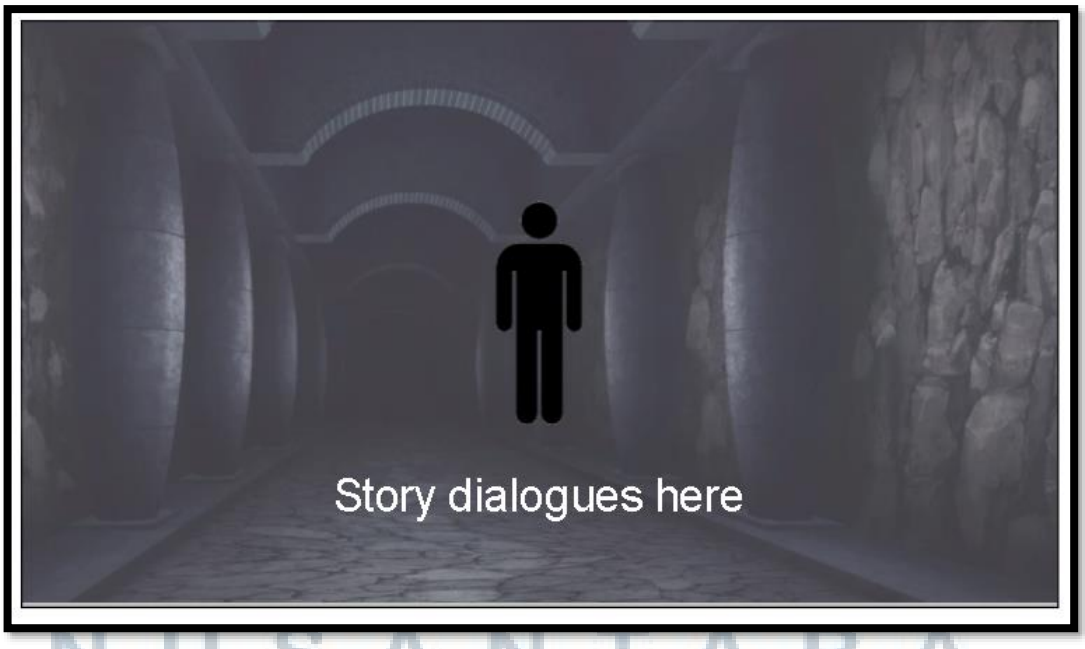
Gambar 3.12 Tampilan *Campaign Menu*

Gambar 3.13 adalah desain *mockup* tampilan *game* ketika pilihan *Settings* di *Main Menu* dipilih seperti pada Gambar 3. 3.



Gambar 3.13 Tampilan *Settings Menu*

Dari sini adalah contoh gambaran dari permainan ketika pemain dalam sesi permainan seperti yang bisa dilihat pada Gambar 3. 2.



Gambar 3.14 *Story-telling* Di Dalam *Game*

Gambar 3.14 ada *mockup* tampilan dari *Narrative Cinematic* seperti yang ditulis pada *flowchart* Gambar 3.2, dari sinilah hasil cerita yang di-*generate* dengan metode *Grammars* akan diceritakan dan ditampilkan kepada pemain. Untuk *scene introduction* atau cerita latar belakang *game* ketika pemain memulai permainan akan mengikuti format yang sama.

Gambar 3.15 Gameplay Scene

Gambar 3.15 adalah tampilan dari permainan, ketika pemain dalam sesi bermain inilah tampilan yang akan pemain lihat. Di sinilah *loop game* terjadi, seperti yang dijelaskan pada *flowchart* Gambar 3.2. *Health* memperlihatkan jumlah dari nyawa pemain, jika mencapai nilai nol maka *game* akan berakhir dan pemain kalah. *Objective* pada Gambar 3.15 menunjukkan objektif yang harus dilakukan pada *level* ini, sesuai dengan objektif apa yang harus diselesaikan akan ditampilkan juga *requirement* penyelesaian. *Map* menunjukkan peta dari *game space* yang telah di-*generate*. *Ammunition* dan jumlah peluru di sebelah kanan bawah juga bisa

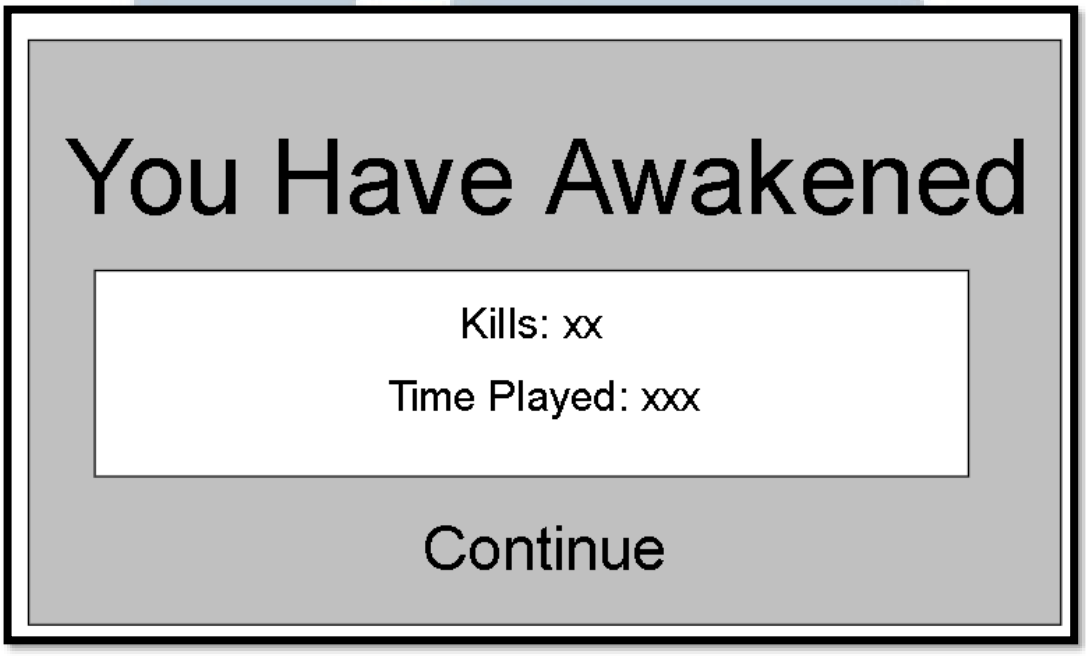
dilihat. Pada permainan inilah, *game space* dimana pemain akan bermain merupakan hasil *generator* dari *Room-Generating Algorithm* dan *Corridor-Generating Algorithm*. *Map* pada gambar juga akan menunjukkan lokasi pemain saat ini pada *game space* yang telah di-*generate.*

Gambar 3.16 In-Game Pause Screen

Gambar 3.16 menunjukkan *mockup* dari tampilan yang keluar ketika pemain memunculkan *menu* dalam *game*, dari sini pemain bisa mengulang *level* atau keluar dari permainan.

Gambar 3.17 Tampilan ketika Pemain Kalah

Gambar 3.17 menunjukkan *mockup* ketika nyawa pemain turun sampai nol dan pemain kalah serta dimunculkan *game over defeated screen* seperti pada penjelasan Gambar 3.2. Seperti apa yang dijelaskan pada Gambar 3.2, pemain akan memiliki opsi untuk keluar dari permainan atau mengulangi *level.*



Gambar 3.18 Tampilan ketika Pemain Menyelesaikan *Game*

Jika pemain menyelesaikan semua *level* dan memenangkan permainan maka pemain akan diperlihatkan *game over victory screen* seperti pada Gambar 3.18. Ketika pemain memilih *continue* maka pemain akan kembali ke *main menu* seperti pada Gambar 3.12.