BAB 3
METODOLOGI PENELITIAN

3.1 Metodologi penelitian

Metodologi penelitian yang digunakan dalam proses rancang bangun *MazeGame* menggunakan metode *Procedural Content Generation* berdasarkan *Prim's Algorithm* adalah sebagai berikut:

3.1.1 Studi Literatur

Pada tahap studi literatur, riset mengenai teori dan metode yang dibutuhkan dan digunakan selama perancangan dan pembangunan *game* akan dilakukan. Materi yang dicari adalah pendalaman mengenai pengimplementasian metode *Procedural Content Generation*, konsep *video game*, *Prim's Algorithm*, *formal* dan *dramatic game elements*, *enemy* AI and *behaviours*, dan lainnya.

3.1.2 Analisa dan Perancangan

Pada tahap perancangan *game*, dilakukan tahap-tahap untuk menganalisa dan merangkum semua *game elements* yang dibutuhkan, baik *formal elements* maupun *dramatic elements*. Selain itu, konsep seperti perencanaan *flowchart*, perancangan metode *Procedural Content Generation*, *player behaviour*, *level objective*, *mockup design*, hingga perencanaan *game assets* juga akan dilakukan.

3.1.3 Implementasi dan Pengembangan

Setelah tahap perancangan selesai, tahap pembangunan *game* akan dilakukan sesuai dengan rancangan yang telah dibuat dan direvisi. Pembangunan *game* akan dilakukan dengan menggunakan *Unity Game Engine* dengan target *platform* untuk hasil akhir adalah *game* dengan *platform* PC. Selain *game* yang akan di-*build* dengan tujuan pengujian, dibuat juga suatu *copy of project* untuk melakukan *maze generation* bertujuan untuk pengumpulan data pada implementasi metode Detect Wall Pattern.

3.1.4 Pengujian

Game yang telah selesai dibangun akan diuji di tahap pengujian. Pada tahap ini, akan dilakukan *playtest* kepada sampel yang akan dilakukan dengan metode *Simple Random Sampling*. Responden yang bersedia dan telah melakukan *playtest* akan diberikan kuesioner *online* yang dibuat berdasarkan pertanyaan berdasarkan *Game User Experience Satisfaction Scale* (GUESS). Selain *playtest* dan pencarian hasil GUESS dilakukan, dilakukan pencarian dan penampilan hasil pola labirin yang dibuat dengan melakukan *maze generation* dalam jumlah besar, dan mencari ada atau tidaknya *maze* yang serupa saat *generation* dilakukan. Hasil maze generation akan dibandingkan dengan data jumlah *spanning tree* yang memungkinkan terbentuk pada *n x n grid* untuk dicari apakah hasil maze unik yang dihasilkan akan melebihi batas spanning tree yang memungkinkan terbentuk pada n x n grid tersebut.

3.1.5 Evaluasi

Berdasarkan hasil pengujian, baik dari Kuesioner yang telah dibagikan dan juga melalui pencarian hasil maze unik yang dibuat dengan melakukan *maze generation*, tahap berikutnya yang akan dilakukan adalah evaluasi hasil *game*. Hasil Kuesioner akan dikumpulkan, dihitung, dan disimpulkan untuk menghitung nilai *video game satisfication* dengan *Game User Experience Satisfaction Scales* (GUESS). Selain itu, hasil maze generation akan dicari dan ditampilkan setelah melakukan *generation* dalam jumlah banyak.

3.1.6 Dokumentasi

Dokumentasi dilakukan selama penelitian berlangsung, dimana proses dokumentasi dilakukan dengan cara yang sesuai dan terkait. Hasil dokumentasi dapat berupa potongan *code*, *screenshot* dari hasil perancangan dan pembagunan, *screenshot* hasil gameplay, hasil pencarian, hingga lampiran hasil *survey* GUESS ke *playtester*.

3.2 Perancangan Game

Perancangan Game terbagi menjadi empat subbab, Struktur *Game*, *Game Asset*, *Flowchart* dan *Mockup Design*. Struktur *Game* akan menjabarkan *formal* dan *dramatic element* yang ada pada hasil *game*. *Game Assets* berisikan daftar tabel asset-asset; audio maupun visual, yang digunakan selama pengembangan *game*. *Flowchart* akan menjabarkan proses kerja dari metode-metode yang ada dalam *game*. *Mockup Design* berisikan gambaran desain awal untuk pengembangan *game*.

3.2.1 Struktur Game

Judul Game : MazeGame

MazeGame merupakan *3D arcade game*, dimana *player* harus mengumpulkan poin dengan cara mengumpulkan *pellet* di dalam *maze* sebanyak mungkin dalam sebuah maze sebelum waktu habis. *Formal Elements* yang terdapat dalam *game* adalah sebagai berikut:

1. Player

   Game ini merupakan *single player game*, dimana *player* dapat menggerakan satu *sprite* dalam *gameplay*.

2. Objective

   Player harus mengumpulkan *point* sebanyak-banyaknya dari mengumpulkan *pellet* yang ada di *maze* sebelum *timer* habis. *Timer* akan bertambah setiap kelipatan *score* tertentu dari *score* yang dikumpulkan.

3. Procedures

   - *Player* menjalankan aplikasi *game*, setelah *splash screen* dijalankan akan diarahkan ke *Main Menu*.
   - Pada *Main Menu*, *player* dapat memilih satu dari empat tombol yang ada. Tombol *Start* akan membawa *player* ke *gameplay*. Tombol *Credit* akan menampilkan daftar referensi *asset* yang digunakan pada aplikasi *game*. Tombol *How to Play* akan menampilkan instruksi cara bermain kepada *player*. Tombol *Exit* akan mengarahkan *player* ke *form* GUESS dan menutup aplikasi *game*.

- *Gameplay* akan dimulai dengan *maze generation* untuk *level map*. Setelah *maze* dibuat, akan ada *countdown* sebelum *gameplay* dimulai. *Timer* akan dimulai saat *gameplay* dimulai dan akan mengakhiri *gameplay* begitu *timer* habis.
- *Score* yang dikumpulkan *player* akan muncul saat *gameplay*, dimana setiap kelipatan *score* tertentu akan menambahkan beberapa detik waktu dari *timer*.
- *Pause* dan *Gameover* display akan menampilkan 2 tombol, dimana *player* dapat melanjutkan atau mengulang *gameplay* dan mengarahkan *player* kembali ke *Main Menu*.

4. Rules

   - *Player control* berupa W, A, S, D atau *Arrow Keys* pada *keyboard*. Aksi klik *mouse cursor* pada tombol digunakan untuk berinteraksi dengan tombol yang ada dalam aplikasi *game*.
   - *Gameplay* berakhir begitu *timer* habis, Setiap *gameplay* juga akan dimulai dengan melakukan *maze generation* baru.

5. Resource

   - *Pellets*, merupakan *gameobject* yang harus dikumpulkan oleh *player* untuk meningkatkan *score* akhir.
   - *Timer*, merupakan parameter yang akan mengakhiri *gameplay* begitu waktu habis. Waktu pada *timer* akan bertambah begitu *score* mencapai kelipatan angka tertentu.

6. Conflict

   Player harus mendapatkan *score* sebanyak-banyaknya sebelum waktu habis. Hasil *score* tertinggi akan dicatat sebagai *highscore* dari aplikasi *game*.

7. Boundaries

   Player hanya dapat bergerak di dalam *maze* yang telah di-*generate*. *Gameplay* hanya akan berlangsung selama *timer* belum habis.

8. Outcome

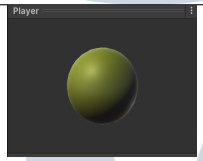
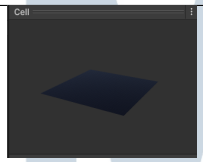
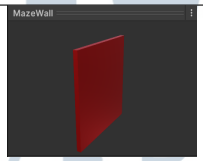
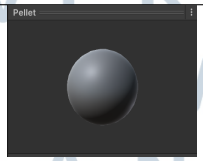
   Score akan dibandingkan dengan *highscore* yang ada, jika *score* melebihi *highscore* maka *score* tersebut akan dicatat menjadi *highscore* baru.

Dikarenakan *game* yang dikembangkan berfokus pada *procedural content generation* dari *level* dan memiliki *genre arcade*, game ini tidak terlalu memiliki elemen yang mendukung *dramatic element* dari Mazegame. Adapun beberapa komponen dari *dramatic element* yang dapat dijabarkan adalah sebagai berikut:

1. *Challenge*, *player* harus berusaha mendapatkan *score* setinggi mungkin sebelum *timer* habis.
2. *Play*, *score* didapat ketika *player* berhasil mengumpulkan *pellet* yang ada di dalam *maze* saat *gameplay* selesai.

3.2.2 Game Assets

Game Asset pada MazeGame terdiri dari *asset* yang dibuat dari jenis *game object* yang disediakan oleh *Unity Game Engine*, dan *asset* yang didapat dari sumber lainnya. Berikut merupakan daftar tabel dari *game assets* pada MazeGame.

Nama	Gambar	Sumber
Player		Unity Resources
MazeCell		Unity Resources
MazeWall		Unity Resources
Pellet		Unity Resources

Tabel 3.1. Tabel Game Asset - Game Objects

Tabel 3.1 diatas merupakan daftar *game object* yang ada di dalam MazeGame, khususnya saat *gameplay*.

Judul (BMG/SFX)	Author	Sumber
Battle (BGM)	gooseninja	https://gooseninja.itch.io/space-music-pack
Cancel 1 (SFX)	jdwasabi	https://jdwasabi.itch.io/8-bit-16-bit-sound-effects-pack
Confirm 1 (SFX)	jdwasabi	https://jdwasabi.itch.io/8-bit-16-bit-sound-effects-pack
Fruit Collect 1 (SFX)	jdwasabi	https://jdwasabi.itch.io/8-bit-16-bit-sound-effects-pack
Loading (BGM)	gooseninja	https://gooseninja.itch.io/space-music-pack
River (BGM)	brevynmusic	https://brevynmusic.itch.io/short-circuits-energetic-8-bit

Tabel 3.2. Tabel Game Asset - Audio

Tabel 3.2 diatas merupakan daftar audio yang ada di dalam MazeGame, baik digunakan sebagai *sound effects* (SFX), ataupun *background music* (BGM).

Nama	Jenis	Sumber
Ardestine	Font	freefonts
Barlow	Font	freefonts
Chopsic	Font	freefonts
Sleek Essential UI Pack	UI Button Element	assetstore.unity.com

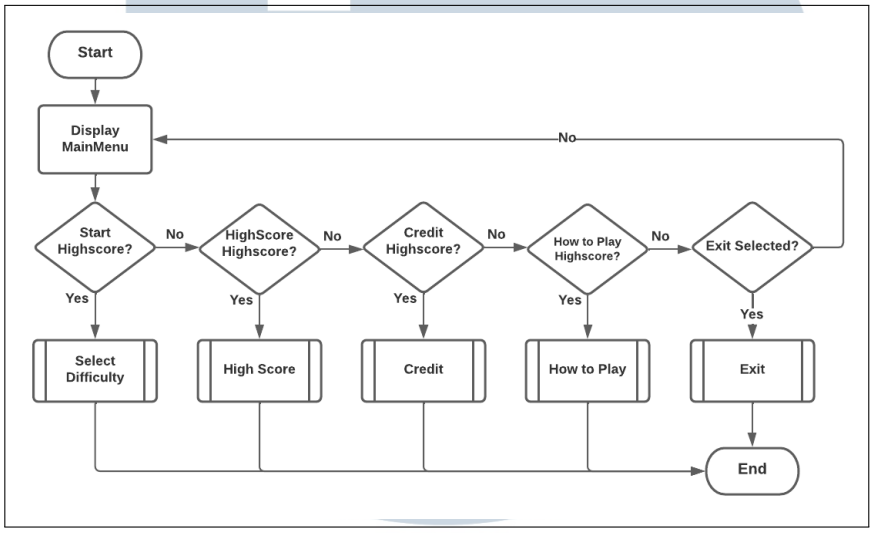
Tabel 3.3. Tabel Game Asset - User Interface

Tabel 3.1 diatas merupakan daftar *asset* yang ada di dalam MazeGame, khususnya untuk tampilan UI, berupa *fonts* dan *UI Asset Package* yang didapat dari *Unity Asset Store*.

3.2.3 Flowchart

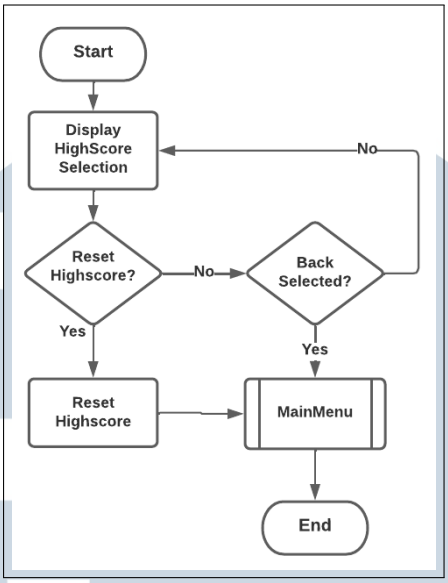
A. Flowchart Game Flow

Alur *game* dari MazeGame dijabarkan menjadi beberapa *flowchart*. Adapun *flowchart* yang menjelaskan beberapa fungsi dalam *game* yang ada adalah sebagai berikut.



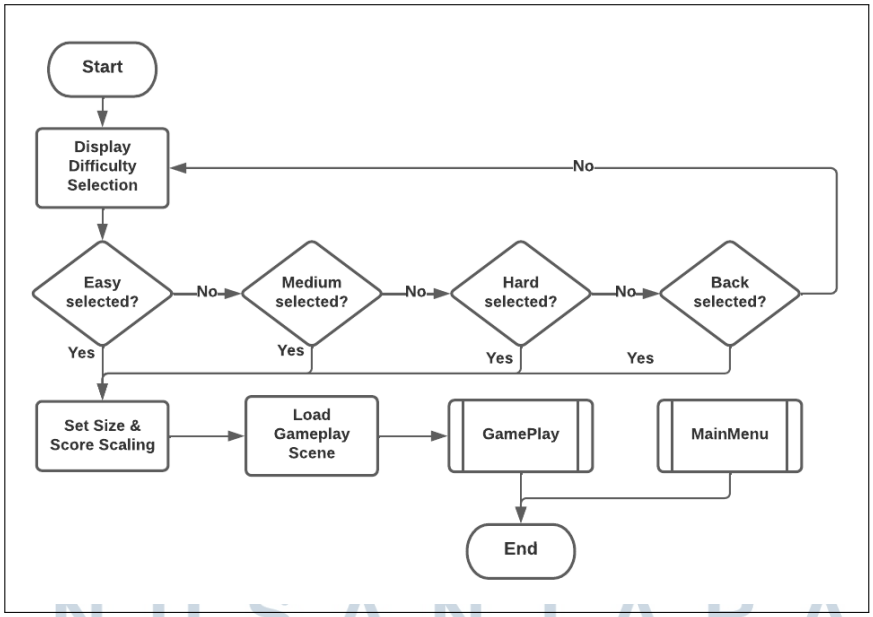
Gambar 3.1. Flowchart Main Menu

Flowchart pada Gambar 3.1 adalah *flowchart* untuk bagian *Main Menu*. Terdapat empat *button* yang dapat diinteraksi oleh *player* dalam *Main Menu, Start, Credit, How to Play*, dan *Exit*. *Start button* akan mengarahkan *player* ke tampilan *Select Difficulty*. *Credit button* akan mengarahkan *player* ke tampilan *credits* untuk *asset* yang digunakan dalam *game*. *How to Play button* akan mengarahkan *player* ke tampilan yang berisikan tulisan mengenai cara bermain dan rangkuman singkat dari *gameplay*. *Exit button* akan mengarahkan ke tampilan *exit*, dimana *player* dapat berinteraksi untuk membuka link kuisioner atau hanya menutup aplikasi.



Gambar 3.2. *Flowchart Highscore*

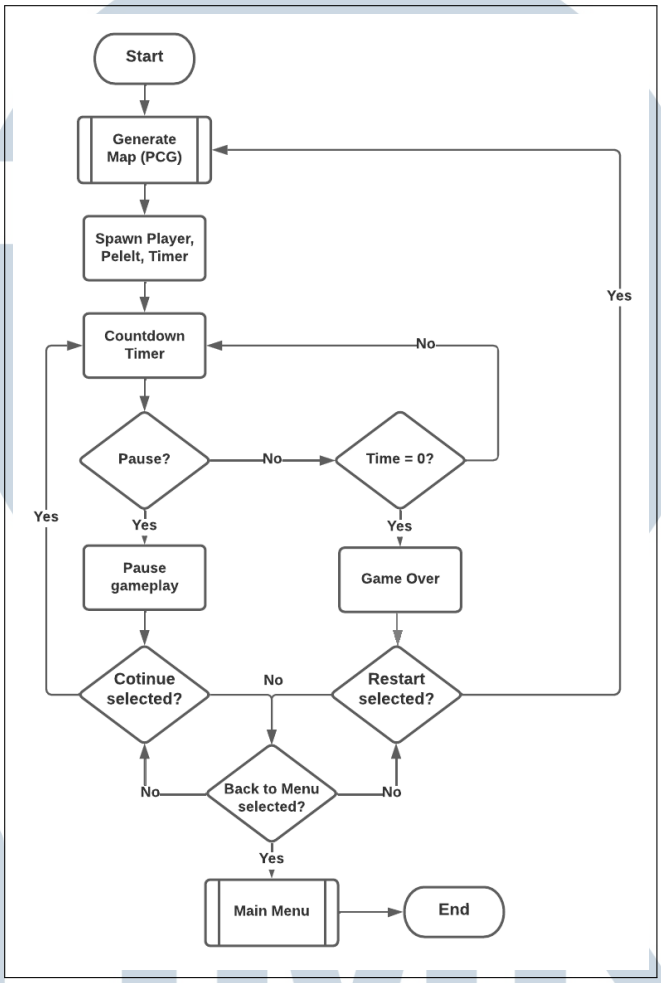
Flowchart pada Gambar 3.2 adalah *flowchart* untuk bagian *Highscore*. Pada halaman ini, *player* dapat melihat daftar *highscore* yang dimiliki pada setiap *difficulty mode* dan dapat melakukan *reset highscore*. *Player* dapat kembali ke *main menu* dari tampilan ini.



Gambar 3.3. *Flowchart Difficulty Selection*

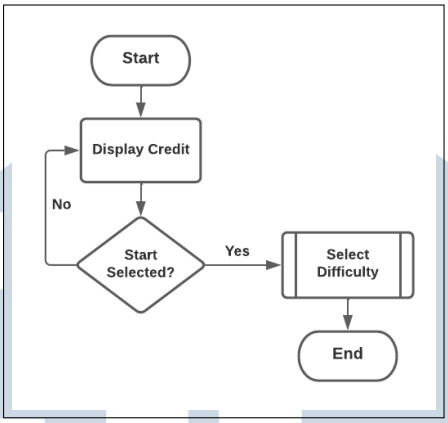
Flowchart pada Gambar 3.3 adalah *flowchart* untuk bagian *Select Difficulty*. Dalam MazeGame, terdapat tiga jenis *Difficulty,*

Easy, Medium, dan *Hard*. *Difficulty* yang dipilih akan mengubah ukuran *maze* dan *score scaling* pada *gameplay*. Selain itu, terdapat *Back button* untuk membawa *user* kembali ke *Main Menu*.



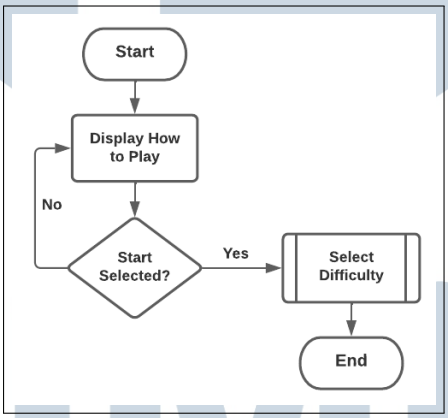
Gambar 3.4. *Flowchart Gameplay*

Flowchart pada Gambar 3.4 adalah *flowchart* untuk *Gameplay*. Pertama, *maze generation* akan dilakukan sebelum *gameplay* dimulai. *Timer* kemudian akan diset, disertai *instantiate* untuk *gameobject player* dan *pellet* di dalam *maze*. Setelah semua komponen siap, *gameplay* akan dimulai. *Timer* akan terus berjalan mundur hingga waktu mencapai 0 detik dan mentrigger *Game Over*. Saat di dalam *gameplay*, *player* dapat melakukan *Pause*, dimana *timer* akan di-*pause*, berserta dengan kontrol *player* yang dihentikan hingga *player* kembali melanjutkan *gameplay* ataupun keluar ke *Main Menu*.



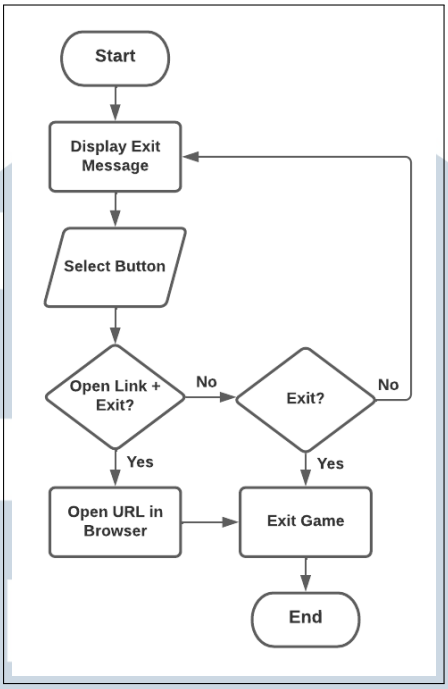
Gambar 3.5. *Flowchart Credit*

Flowchart pada Gambar 3.5 adalah *flowchart* untuk bagian *Credit*. Bagian ini menampilkan sumber referensi untuk *asset* yang digunakan MazeGame,



Gambar 3.6. *Flowchart How to Play*

Flowchart pada Gambar 3.6 adalah *flowchart* untuk bagian *How to Play*. Bagian ini akan menampilkan instruksi cara bermain untuk *player*, dan rangkuman singkat dari *gameplay*.

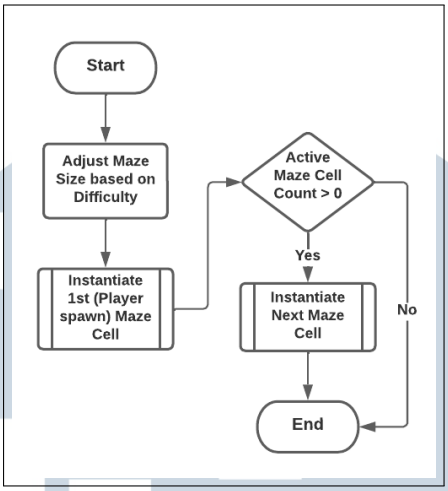


Gambar 3.7. *Flowchart Exit*

Flowchart pada Gambar 3.7 adalah *flowchart* untuk bagian *Exit*. Pada bagian ini, terdapat dua *button*, dimana salah satu *button* akan mengarahkan *player* ke link kuisioner serta menutup aplikasi *game*, dan *button* lainnya akan langsung menutup aplikasi *game* tanpa membuka link kuisioner.

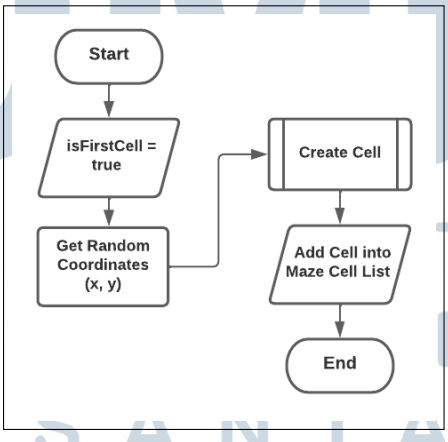
B. Procedural Content Generation (Generate Map)

Pada prosedur *Generate Map* dalam *gameplay*, terdapat beberapa fungsi yang terlibat untuk mengimplementasikan *Procedural Content Generation* berdasarkan *Prim's Algorithm* ke dalam MazeGame. Berikut adalah *flowchart* untuk fungsi *Generate Map* dalam MazeGame.



Gambar 3.8. *Flowchart Maze Generation*

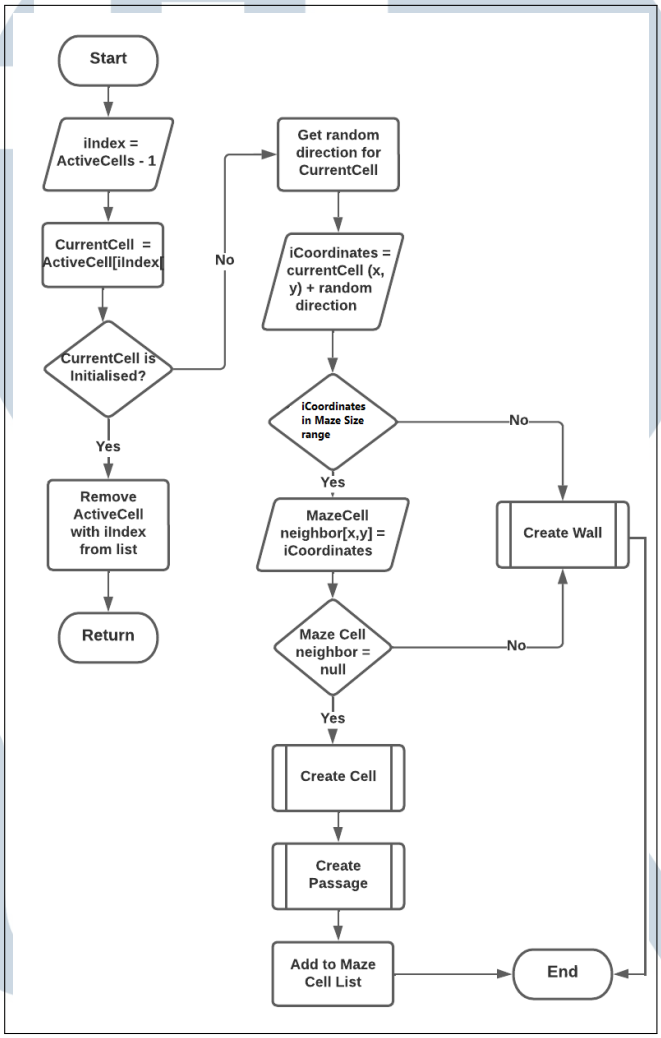
Flowchart pada Gambar 3.8 di atas adalah *flowchart* untuk *Maze Generation*. Fungsi ini dimulai dengan menentukan ukuran dari *maze* yang digenerate, sesuai dengan *difficulty* yang dipilih *player*. Setelah itu, dilakukan *instantiate* untuk *maze cell* pertama, yang nantinya akan menjadi tempat *spawn* dari *gameobject player*. Setelah itu, sesuai dengan jumlah *maze cell* yang dimiliki, *maze cell* akan dibuat melalui fungsi *Instantiate Next Maze Cell*. Setelah *Maze Cell* selesai diinisialisasi beserta *maze wall* dan *maze passage* di dalamnya, *Maze Generation* selesai, dan akan dilanjutkan ke bagian lain dalam *gameplay flowchart*.



Gambar 3.9. *Flowchart First Maze Cell Instantiate*

Flowchart pada Gambar 3.9 di atas merupakan *flowchart* untuk fungsi *Instantiate First Maze Cell* pada *Maze Generation*. Untuk

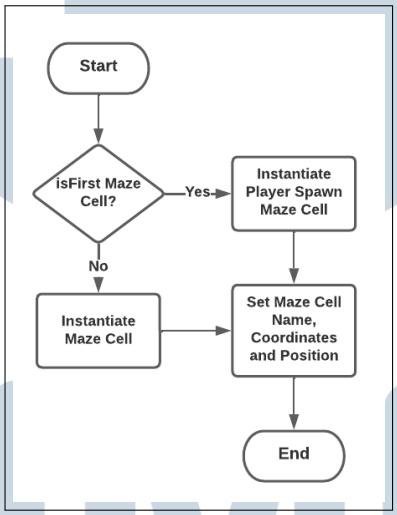
fungsi ini, sebuah variabel *boolean isFirst* akan diset menjadi *true* sebelum sebuah koordinat *random* diambil dan dijalankan fungsi *Create Cell*. *Maze cell* yang sudah dibuat ini akan ditambahkan ke dalam sebuah *List MazeCell* untuk digunakan saat inisialisasi komponen dari *maze* lain seperti *maze wall* dan *maze passage*.



Gambar 3.10. *Flowchart Maze Cells Instantiate*

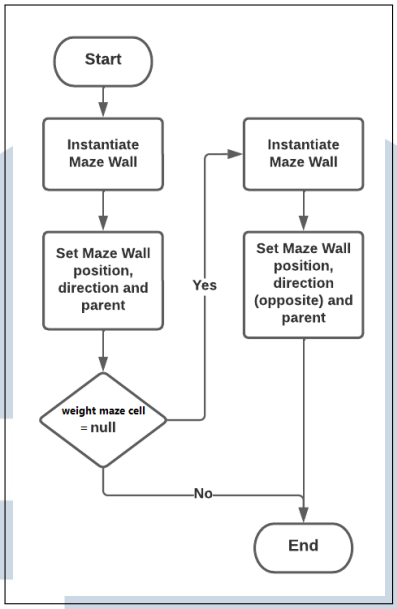
Flowchart pada Gambar 3.11 di atas merupakan *flowchart* untuk fungsi *Maze Cell Instantiate* pada *Maze Generation*. Setelah memilih *maze cell* terakhir dalam list, *maze cell* akan diperiksa apakah telah selesai diinisialisasi posisi *maze wall* ataupun *maze passage* yang memungkinkan di cell tersebut. Jika *maze cell* tersebut sudah selesai diinisialisasi, *maze cell* tersebut akan dihapus dari *ist* dan fungsi ini akan selesai dengan return. Jika *maze cell* belum se-

lesai diinisialisasi, *maze cell* tersebut akan melakukan pengecekan kondisi pada *maze cell neighbor* yang *4-adjacent* dengan *maze cell* tersebut. Akan dibuat sebuah variabel iCoordinate yang berisikan coordinat maze cell yang tambah direction random pada untuk posisi *4-adjacent* dengan *maze cell*, jika icoordinate tersebut berada di luar *maze size*, itu berarti *maze cell* awal berada di tepi *maze*, dan fungsi hanya akan membentuk *maze wall* sebagai *border* dari *maze*. Jika iCoordinate ada di dalam *range maze*, akan diperiksa apakan posisi *maze cell* tersebut sudah terdapat *maze cell* atau belum. Jika *maze cell* baru belum terbentuk, akan dilakukan *Create Cell* dan *Create Maze Passage* pada *maze cell neighbor* tersebut. Jika pada *maze cell* baru sudah terbentuk, akan dilakukan fungsi *Create Wall*.



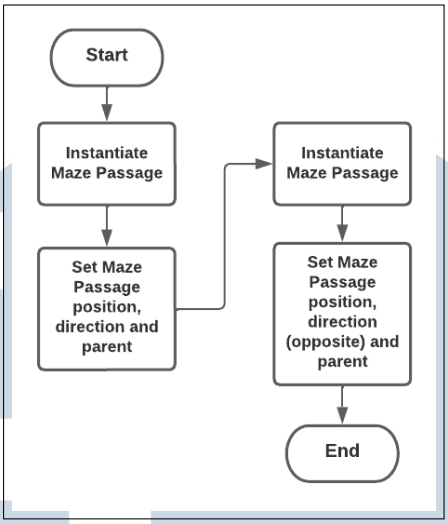
Gambar 3.11. *Flowchart Create Cell*

Flowchart pada Gambar 3.11 di atas merupakan *flowchart* untuk fungsi *Create Maze Cell* pada *Maze Generation*. Untuk fungsi ini, variabel *boolean isFirst* akan dicek apakah mengandung nliai *true* atau *false*. Jika *isFirst* bernilai *true*, maka *maze cell* yang akan diinisialisasi adalah *maze cell* yang mengandung *gameobject player spawner*. Jika *isFirst* bernilai *false*, *maze cell* yang akan diinisialisasi adalah *maze cell* biasa. Setelah *maze cell* diinisialisasi, koordinat, nama dan posisi dari *maze cell* tersebut akan ditentukan sesuai dengan nilai pada variabel yang diberikan.



Gambar 3.12. *Flowchart Create Wall*

Flowchart pada Gambar 3.12 di atas merupakan *flowchart* untuk fungsi *Create Maze Wall* pada *Maze Generation*. Pada fungsi ini *maze wall* akan diinisialisasi pada sebuah *maze cell*, kemudian *position*, *direction* dan *parent* dari *maze wall* tersebut akan ditentukan. Setelah itu akan di cek *weight* atau *mazecell neighbor* dari *maze cell* dengan *4-adjacent* yang ditentukan pada *maze cell* neighbor. Jika *weight* kosong, akan dilakukan insisialisasi *maze wall* baru dan diletakan pada *position* dan *parent* yang sama, namun dengan *direction* sebaliknya.

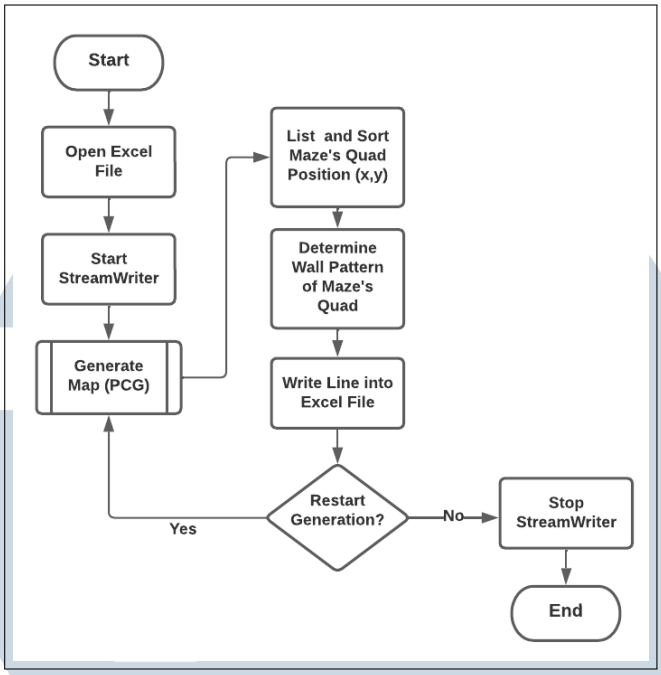


Gambar 3.13. *Flowchart Create Passage*

Flowchart pada Gambar 3.13 di atas merupakan *flowchart* untuk fungsi *Create Maze Passage* pada *Maze Generation*. Maze passage merupakan bagian dari *maze* yang tidak mengandung *maze wall* atau hanya berupa jalan biasa dalam *maze*. Pada fungsi ini *maze passage* akan diinisialisasi pada sebuah *maze cell*, kemudian *position*, *direction* dan *parent* dari *maze passage* tersebut akan ditentukan. Setelah itu akan diinisilaisasi *maze passage* baru di *maze cell* tersebut dan ditentukan *position, direction* dan *parent*, dimana *direction* yang ditentukan kali ini adalah *direction* sebaliknya dari *maze passage* sebelumnya.

C. Prosedur Awal Pencarian Hasil Maze Generation

Selain *game flow flowchart* di atas, terdapat juga beberapa rangkaian fungsi yang dirancang untuk menentukan posisi tembok yang akan dimiliki setiap *cell* dalam *maze*, dimana data tersebut akan disimpan untuk keperluan pencarian hasil maze generation menggunakan *Procedural Content Generation* yang berdasarkan *Prim's Algorithm*.

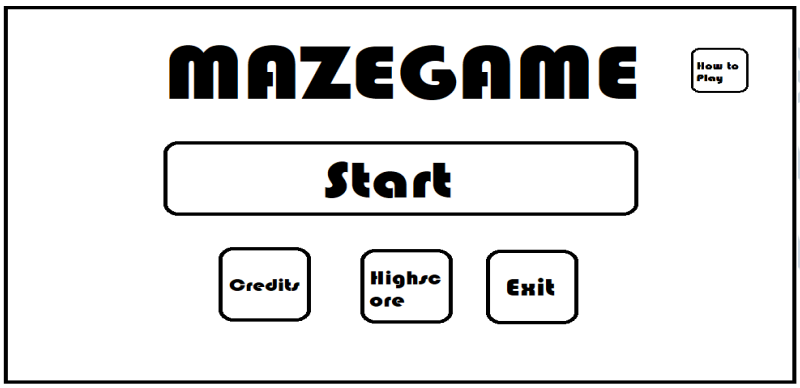


Gambar 3.14. *Flowchart Get Wall Patterns*

Flowchart pada Gambar 3.14 di atas merupakan *flowchart* untuk perintah yang digunakan untuk mencari, menganalisa dan menyimpan data jumlah dan pola posisi dari tembok pada tiap kotak pada hasil *maze generation*. Hasil dari analisa tersebut disimpan ke dalam sebuah *comma-separated values file* (.csv) yang kemudian dapat diolah dalam bentuk *spreadsheet*.

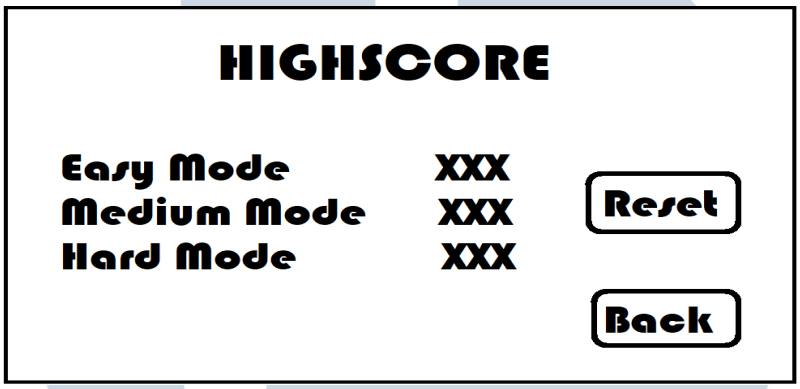
3.2.4 Mockup Design

Berikut merupakan tampilan dan penjabaran rancangan awal berupa *mockup design* dari MazeGame.



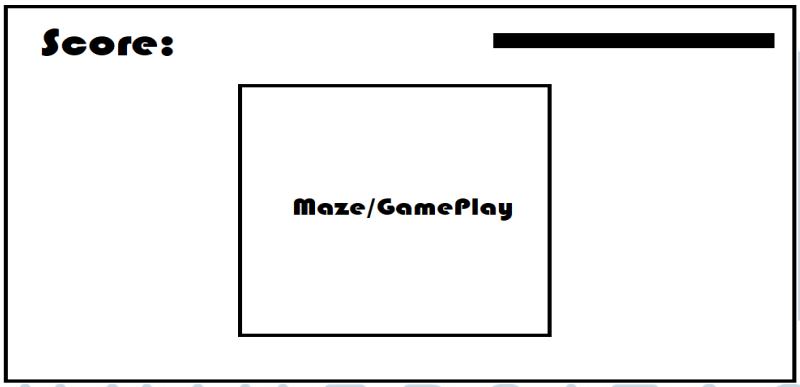
Gambar 3.15. *Mockup* Tampilan *Main Menu*

Gambar 3.15 merupakan desain awal dari tampilan *main menu* dari MazeGame. Dalam tampilan ini, terdapat judul *game*, tombol *Start, Credit, How to Play*, dan *Exit*. *Player* dapat berintetaksi dengan keempat *button* tersebut untuk diarahkan ke tujuan dari tombol masing-masing.



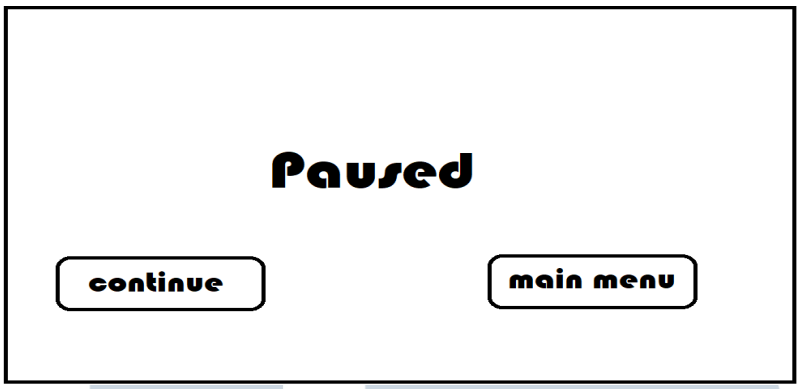
Gambar 3.16. *Mockup* Tampilan *Highscore*

Gambar 3.16 merupakan desain awal dari tampilan *highscore* pada MazeGame. Dalam tampilan ini, terdapat tombol *Reset* untuk mengembalikan nilai *highscore* kembali menjadi 0, dan *Back* untuk mengarahkan player kembali ke *Main Menu*.



Gambar 3.17. *Mockup* Tampilan *Gameplay*

Gambar 3.17 merupakan tampilan dari *gameplay* yang saat *player* menekan *Start* pada *Main Menu*. Pada *gameplay*, terdapat komponen teks yang menampilkan total poin *player*, *maze* hasil *maze generation* sebagai level dari *gameplay*, dan *linear timer* yang akan memulai *countdown* saat *gameplay* dimulai.

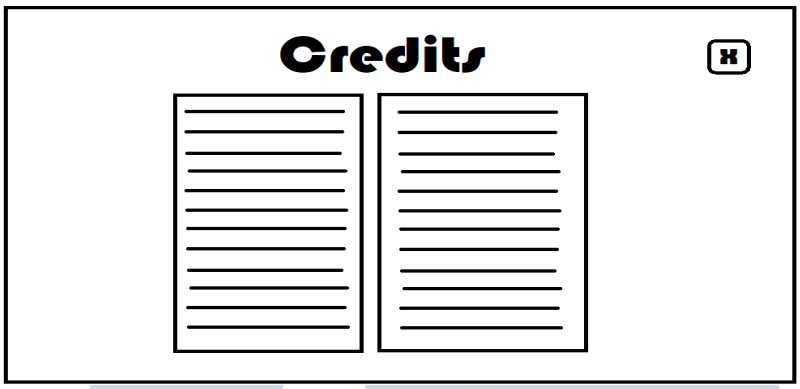


Gambar 3.18. *Mockup* Tampilan *Pause*

Saat *gameplay* berlangsung, *player* dapat menekan *Escape* pada *keyboard* untuk melakukan *pause gameplay*. Tampilan pada Gambar 3.18 akan muncul saat *pause gameplay* diinisialisasi. Pada tampilan ini terdapat dua *button* yang dapat *player* pilih, *Continue* untuk melanjutkan *gameplay*, atau *Main Menu* untuk keluar dari *gameplay* dan kembali ke tampilan *Main Menu*.

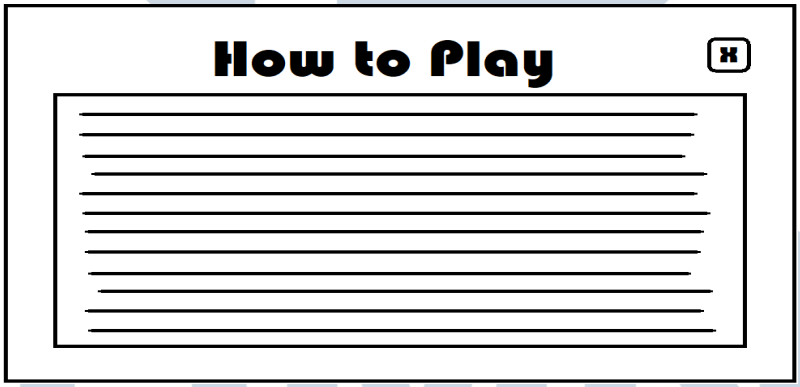
Gambar 3.19. *Mockup* Tampilan *Gameover*

Ketika *timer* habis dan *gameplay* selesai, akan muncul tampilan *Game Over* seperti pada Gambar 3.19. Tampilan *Game Over* ini mirip dengan tampilan *Pause* pada *gameplay*, dimana terdapat dua buah *button*, *Restart* untuk memulai kembali *gameplay* baru dan *Main Menu* untuk kembali ke *Main Menu*.



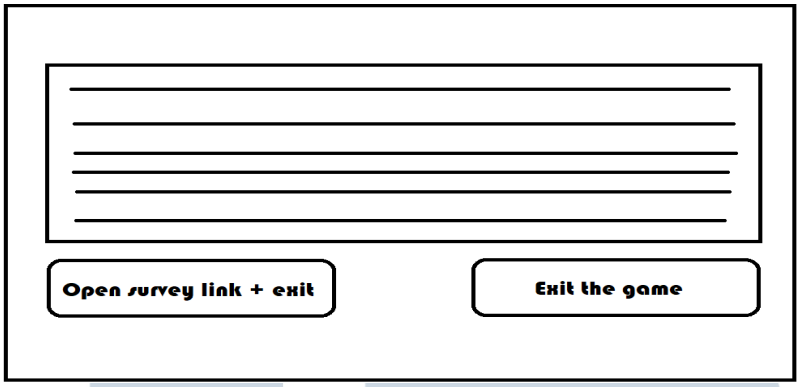
Gambar 3.20. *Mockup* Tampilan *Credits*

Credit button pada tampilan *Main Menu* akan mengarahkan *player* ke tampilan pada Gambar 3.20. Pada bagian *Credit*, akan dicantumkan sumber referensi untuk *asset*; baik visual maupun audio, yang digunakan selama pengembangan MazeGame. Terdapat juga button yang dapat diakses *player* untuk kembali ke *Main Menu*.



Gambar 3.21. *Mockup* Tampilan *How to Play*

How to Play button pada *Main Menu* akan mengarahkan *player* ke tampilan pada Gambar 3.21. Seperti namanya, bagian *How to Play* akan menjabarkan objektif dan kontrol yang akan digunakan pada *gameplay* dari MazeGame. Selain itu, terdapat juga beberapa penjelasan singkat mengenai *game mechanic* dari MazeGame.



Gambar 3.22. *Mockup* Tampilan *Exit*

Exit button akan mengarahkan ke proses penutupan aplikasi *game*. Namun sebelum ditutup, *player* akan diarahkan ke tampilan seperti pada Gambar 3.22. Tampilan ini akan menampilkan dua *button*, menjelaskan dan mengarahkan *player* ke form kuisioner yang telah disiapkan untuk kepentingan penelitian atau langsung keluar dari aplikasi jika *player* sudah pernah mengisi kuisioner.