BAB 3
METODOLOGI PENELITIAN

3.1 Metodologi Penelitian

Metodologi penelitian yang digunakan dalam perbandingan level *maze* buatan algoritma *Backtracking, Kruskal, Prim* dan *Eller* dalam game *maze* adalah sebagai berikut:

3.1.1 Studi Literatur

Pada tahap studi literatur, peneliti melakukan riset teori mengenai *maze* dan algoritma-algoritma yang akan digunakan dalam pembuatan *maze*. Teori yang dicari adalah mengenai *Maze*, *Procedural Content Generation, Backtracking Algorithm, Kruskal's Algorithm, Prim's Algorithm, Eller's Algorithm* dan *A-Star Algorithm*.

3.1.2 Perancangan

Pada tahap perancangan, peneliti melakukan perancangan program untuk perbandingan *maze*, alur kerja pembuatan *maze* setiap algoritma, dan alur kerja algoritma A-Star yang akan digunakan untuk membandingkan kompleksitas *maze*.

3.1.3 Implementasi

Pada tahap implementasi, peneliti membuat program dengan algoritma yang sudah dirancang untuk pembuatan *maze*. Peneliti membuat program ini menggunakan *game engine Unity*.

3.1.4 Pengujian

Pada tahap pengujian, peneliti menguji *maze* buatan algoritma menggunakan algoritma A-Star. Algoritma A-Star akan digunakan untuk menyelesaikan berbagai ukuran *maze* yang telah dibuat oleh masing-masing algoritma. Algoritma A-Star akan mengukur jumlah langkah dan waktu yang dibutuhkan untuk menyelesaikan *maze* tersebut.

3.1.5 Evaluasi

Pada tahap evaluasi, peneliti melakukan perbandingan antara jumlah langkah dan waktu yang didapatkan dari tahap pengujian sebelumnya. Hasil dari pengujian masing-masing algoritma akan dibandingkan satu sama lain untuk mengukur perbandingan tingkat kompleksitas *maze* yang dibuat antara algoritma satu sama lain.

3.1.6 Penulisan Laporan

Pada tahap penulisan laporan, dilakukan penulisan laporan yang menjabarkan seluruh proses penelitian dan hasil yang didapatkan dari penelitian ini.

3.2 Perancangan

Perancangan yang dilakukan dalam penelitian ini adalah Perancangan Program dan Perancangan Flowchart.

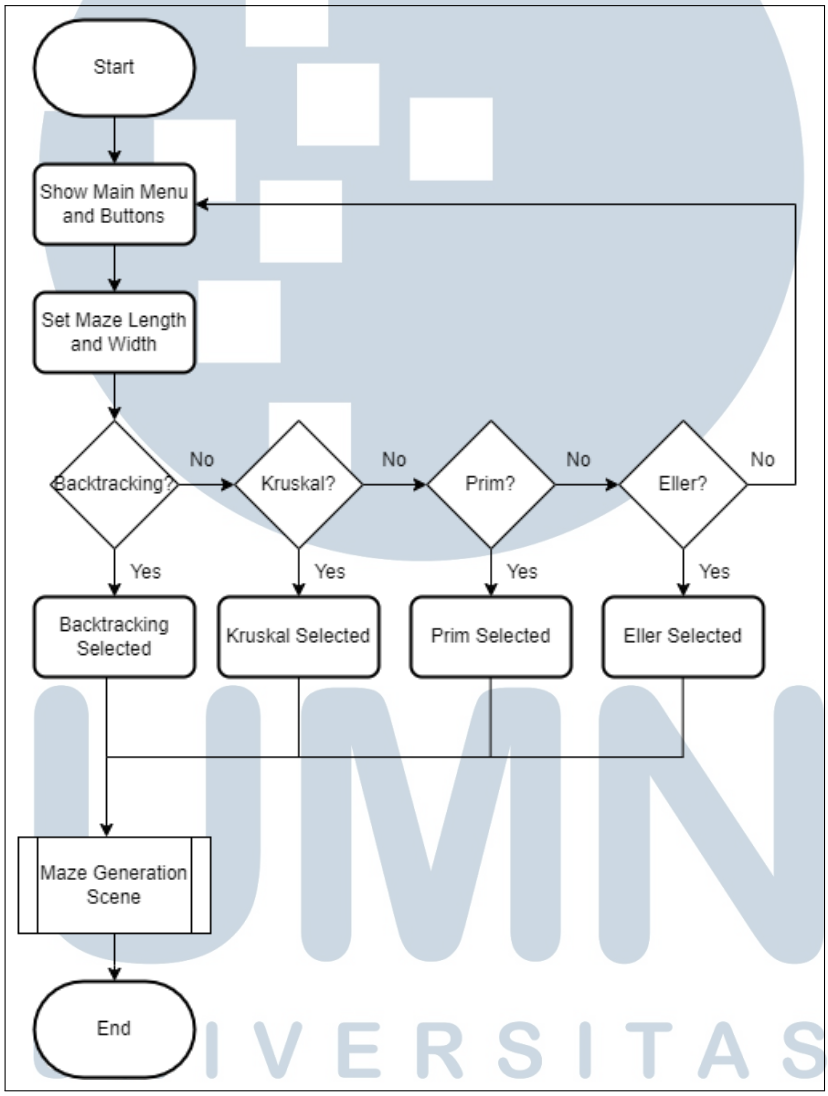
3.2.1 Perancangan Program

Pada penelitian ini, peneliti merancang sebuah program yang digunakan sebagai media bagi algoritma dalam membuat sebuah *maze*. Di dalam program ini, pengguna dapat memilih besar *maze* yang akan dibuat dan algoritma yang akan digunakan dalam pembuatan *maze*. Pengguna juga dapat melihat proses penyelesaian *maze* menggunakan algoritma A-Star dan mendapatkan informasi mengenai waktu penyelesaian yang dibutuhkan *maze* yang dibuat oleh masing-masing algoritma.

3.2.2 Flowchart

Berikut ini adalah *flowchart* dari program yang dipakai dalam penelitian ini:

3.2.3 Flowchart Main Menu

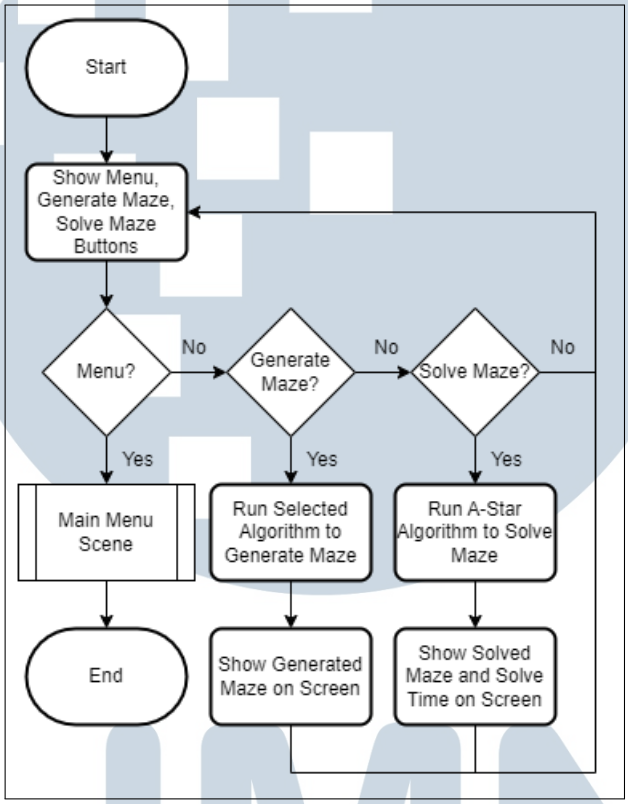


Gambar 3.1. *Flowchart* Main Menu

Flowchart pada Gambar 3.1 menggambarkan alur kerja dari *main menu* yang tampil pada saat menjalankan program. Di dalam *main menu*, program menampilkan tombol untuk memilih algoritma yang akan dipakai dan *input field* untuk menentukan panjang dan lebar *maze* yang akan dibuat. Saat tombol algoritma

ditekan, program akan mencatat algoritma apa yang dipilih dan program akan berpindah ke *scene Maze Generation* dengan algoritma yang dipilih sebelumnya.

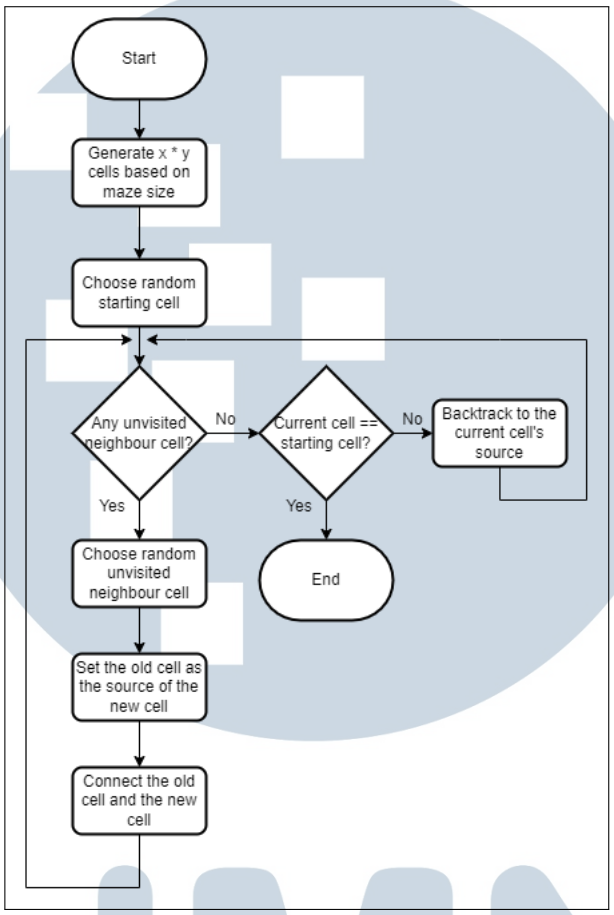
3.2.4 Flowchart Maze Generation Scene



Gambar 3.2. *Flowchart* Maze Generation Scene

Flowchart pada Gambar 3.2 menggambarkan alur kerja dari *scene Maze Generation* yang tampil setelah memilih algoritma pada *main menu*. Di dalam *maze generation scene*, program akan menampilkan tombol *Menu*, tombol *Generate Maze* dan tombol *Solve Maze*. Saat tombol *Generate Maze* ditekan, program akan menjalankan algoritma yang dipilih pada *main menu* sebelumnya untuk membuat *maze* dan menampilkan *maze* itu pada layar. Saat tombol *Solve Maze* ditekan, program akan menjalankan algoritma *A-Star* untuk menyelesaikan *maze* yang ada dan menampilkan *maze* yang sudah diselesaikan dan waktu penyelesaian *maze* di layar.

A Flowchart Algoritma Backtracking

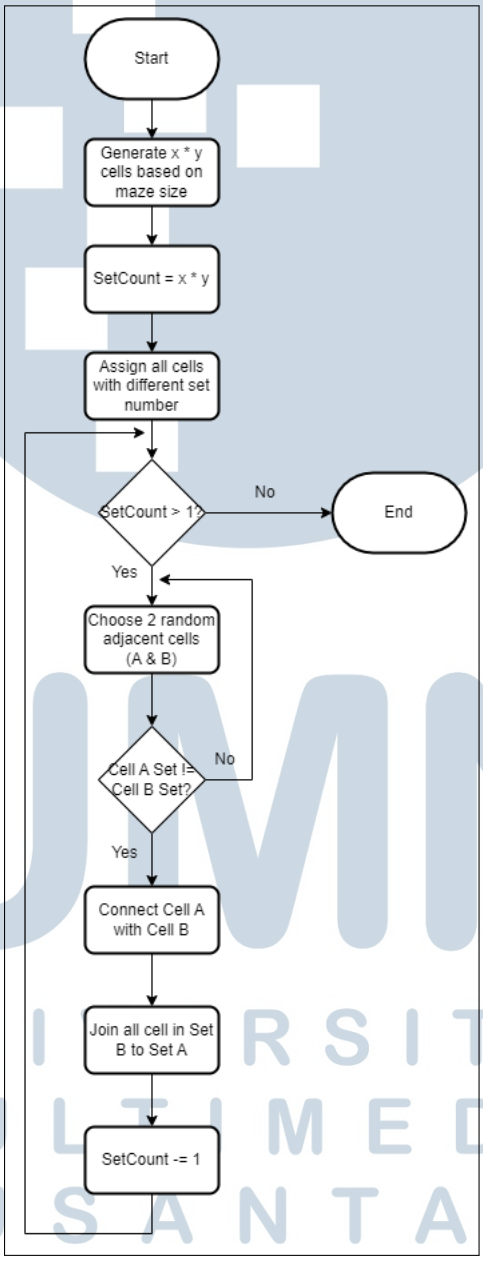


Gambar 3.3. *Flowchart* Algoritma Backtracking yang dimodifikasi untuk Maze Generation

Flowchart pada Gambar 3.3 menggambarkan alur kerja dari algoritma Backtracking. Sebelum algoritma dimulai, program akan membuat sel untuk *maze* berdasarkan ukuran *maze* yang diinginkan. Pertama, algoritma ini akan memilih salah satu sel *random* untuk menjadi sel *Start*. Semua sel yang bersampingan dengan sel yang terpilih akan dicek, jika sel tersebut belum pernah dikunjungi maka sel tersebut akan menjadi calon sel yang dapat dikunjungi. Dari sel-sel yang dapat dikunjungi, algoritma memilih satu sel secara acak untuk dikunjungi. Sel ini akan dihubungkan dengan sel sebelumnya, dan akan mencatat sel sebelumnya sebagai *source*. Proses pengunjungan sel ini akan dilakukan secara berulang sampai ke sel yang tidak memiliki target sel yang dapat dikunjungi lagi. Jika tidak ada sel yang dapat dikunjungi, maka algoritma akan berjalan mundur (*Backtracking*) menuju *source* dari sel sampai ke sel yang memiliki sel yang dapat dikunjungi,

dan melakukan pengunjungan lagi. Jika algoritma *Backtracking* sampai ke sel *Start* dan tidak ada sel yang bisa dikunjungi lagi, berarti semua sel dalam *maze* sudah terhubung dan algoritma selesai.

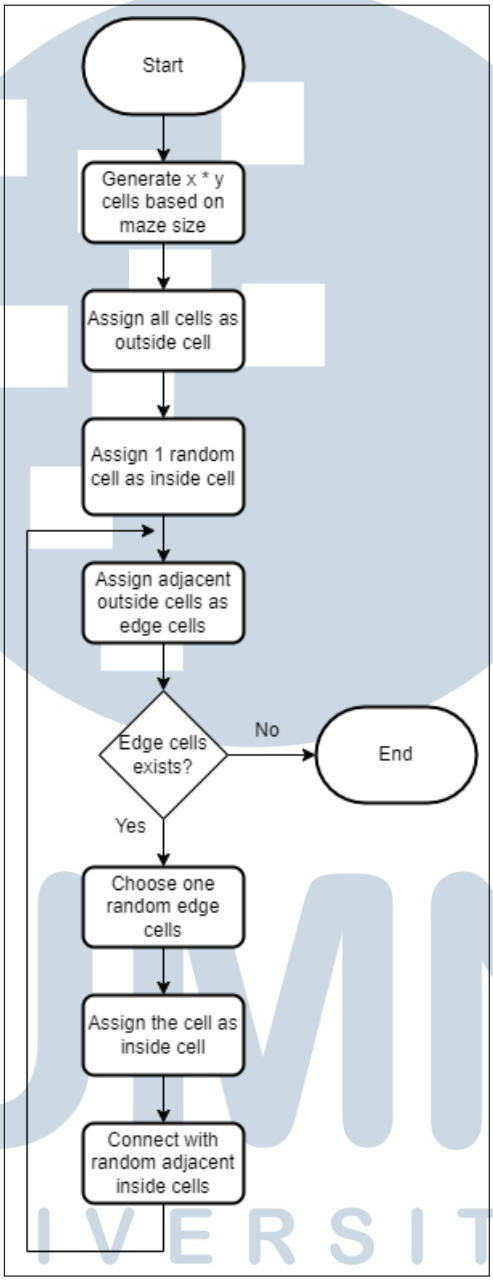
B Flowchart Algoritma Kruskal



Gambar 3.4. Flowchart Algoritma Kruskal yang dimodifikasi untuk Maze Generation

Flowchart pada Gambar 3.4 menggambarkan alur kerja dari algoritma Kruskal. Sebelum algoritma dimulai, program akan membuat sel untuk *maze* berdasarkan ukuran *maze* yang diinginkan. Pertama, algoritma ini akan menetapkan sebuah nomor set yang berbeda-beda pada semua sel dan mencatat jumlah set yang ada. Algoritma akan memilih secara acak dua sel yang bersebelahan. Jika kedua sel tersebut berasal dari set yang berbeda, maka algoritma akan menghubungkan dua sel tersebut dan menggabungkan kedua set tersebut menjadi satu set. Proses pemilihan acak dan penggabungan ini akan dilakukan secara berulang sampai hanya tersisa satu set dalam seluruh *maze*, dan hal itu menandakan bahwa semua sel dalam *maze* sudah terhubung dan algoritma selesai.

C Flowchart Algoritma Prim

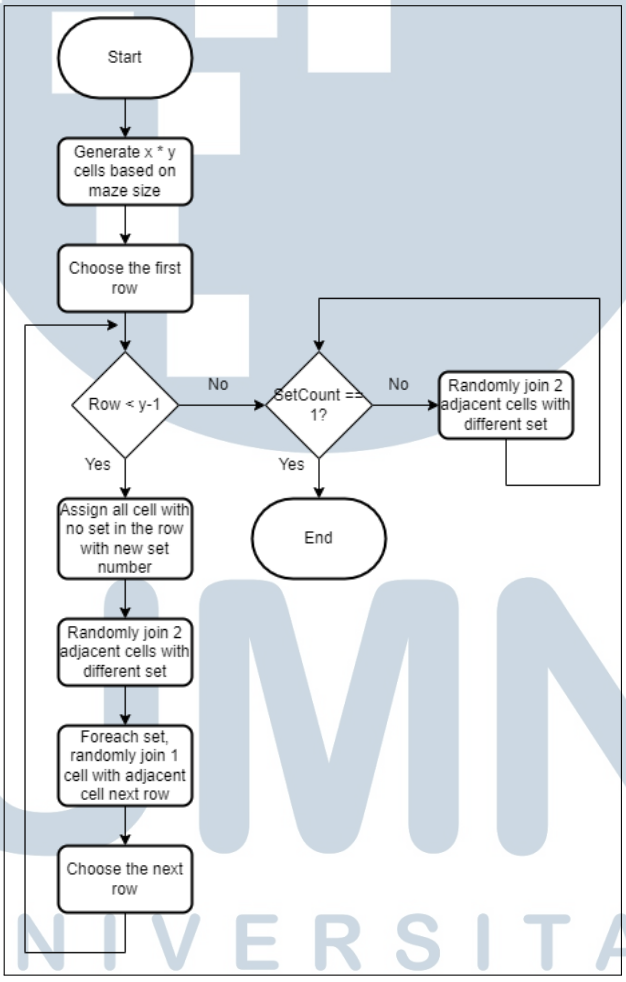


Gambar 3.5. Flowchart Algoritma Prim yang dimodifikasi untuk Maze Generation

Flowchart pada Gambar 3.5 menggambarkan alur kerja dari algoritma Prim. Sebelum algoritma dimulai, program akan membuat sel untuk *maze* berdasarkan ukuran *maze* yang diinginkan. Pertama, algoritma ini akan menetapkan semua sel sebagai *outside cell*. Algoritma akan memilih satu sel secara acak untuk menjadi *inside cell* pertama. Semua *outside cell* yang bersebelahan dengan salah satu *inside*

cell akan diubah menjadi *edge cell*. Algoritma akan memilih satu *edge cell* yang ada untuk dihubungkan dengan *inside cell*, dan mengubah *edge cell* tersebut menjadi *inside cell*. Proses perubahan dan penggabungan *edge cell* ini akan dilakukan secara berulang sampai seluruh *edge cell* habis, dan hal itu menandakan bahwa semua sel dalam *maze* sudah terhubung dan algoritma selesai.

D Flowchart Algoritma Eller

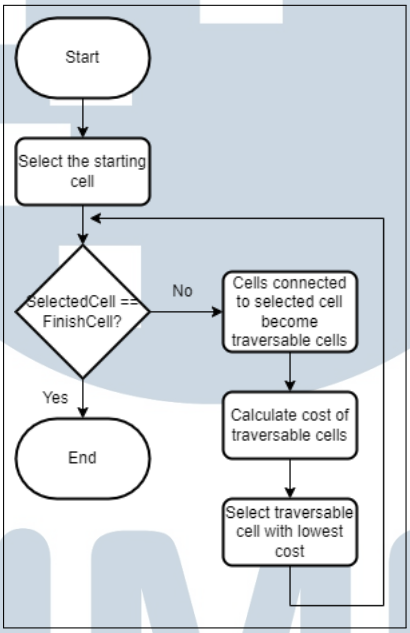


Gambar 3.6. Flowchart Algoritma Eller yang dimodifikasi untuk Maze Generation

Flowchart pada Gambar 3.6 menggambarkan alur kerja dari algoritma Eller. Sebelum algoritma dimulai, program akan membuat untuk *maze* berdasarkan ukuran *maze* yang diinginkan. Pertama, algoritma akan memulai proses dari baris pertama. Algoritma akan menetapkan sebuah nomor set yang berbeda-beda pada baris ini. Lalu akan dilakukan penghubungan secara acak antara dua sel

dengan set yang berbeda dan menggabungkan dua set tersebut. Untuk setiap set yang ada, dilakukan penghubungan salah satu sel dengan sel di baris selanjutnya. Proses penetapan set, penghubungan sel dan penggabungan set ini akan dilakukan secara berulang sampai ke satu baris sebelum baris terakhir. Di baris terakhir, dilakukan penggabungan set terus-menerus sampai hanya tersisa satu set, dan hal itu menandakan bahwa semua sel dalam *maze* sudah terhubung dan algoritma selesai.

E Flowchart Algoritma A-Star



Gambar 3.7. Flowchart Algoritma A-Star yang dimodifikasi untuk Maze Generation

Flowchart pada Gambar 3.7 menggambarkan alur kerja dari algoritma A-Star. Pertama, algoritma dimulai dengan memilih sel *Start*. Semua sel yang terhubung dengan sel yang terpilih akan dicatat sebagai *traversable cell*. Algoritma akan menghitung *cost* dari setiap *traversable cell* yang berasal dari *total cost* sel *Start* menuju *traversable cell* dan *estimated cost traversable cell* menuju sel *Finish*. Algoritma akan memilih *traversable cell* yang memiliki *cost* paling sedikit untuk dilalui. Proses ini akan dilakukan secara berulang sampai tercapai sel *Finish*.