BAB 3
METODOLOGI PENELITIAN

3.1 Metodologi

Metode yang digunakan dalam pembangunan game ini antara lain:

1. Studi Literatur
   Tahap ini dilakukan untuk mengumpulkan data beserta informasi terhadap teori dan konsep mengenai tower defense, pathfinding, algoritma flow field pathfinding, dan algoritma a* pathfinding. Informasi tersebut diperoleh melalui web, jurnal, game, dan lain-lain.

2. Analisis Kebutuhan
   Melakukan analisis terhadap plugin yang digunakan untuk Algoritma Flow Field Pathfinding serta Algoritma A* Pathfinding yang sesuai untuk digunakan ke dalam game tower defense.

3. Perancangan
   Perancangan menggunakan flowchart untuk menggambarkan alur kerja Algoritma Flow Field Pathfinding, Algoritma A* Pathfinding, serta perancangan *map* simulasi.

4. Implementasi
   Membangun *map* simulasi dari permainan tower defense dengan mengimplementasi rancangan serta metode yang telah dibahas.

5. Pengujian
   Untuk pengujian akan melakukan percobaan 10 kali untuk setiap algoritma dan skenario untuk mencari tahu waktu rata-rata yang diperlukan untuk algoritma *pathfinding* menemukan jalur musuh.

6. Evaluasi
   Setelah menemukan waktu rata-rata dari pengujian akan dibandingkan kedua metode algoritma secara waktu dan optimasi jalur yang dibentuk.

7. Dokumentasi
   Dilakukan penulisan laporan oleh penguji untuk mencatat hasil dari penguji serta evaluasi yang telah diberikan terhadap game tersebut. Dokumentasi ini

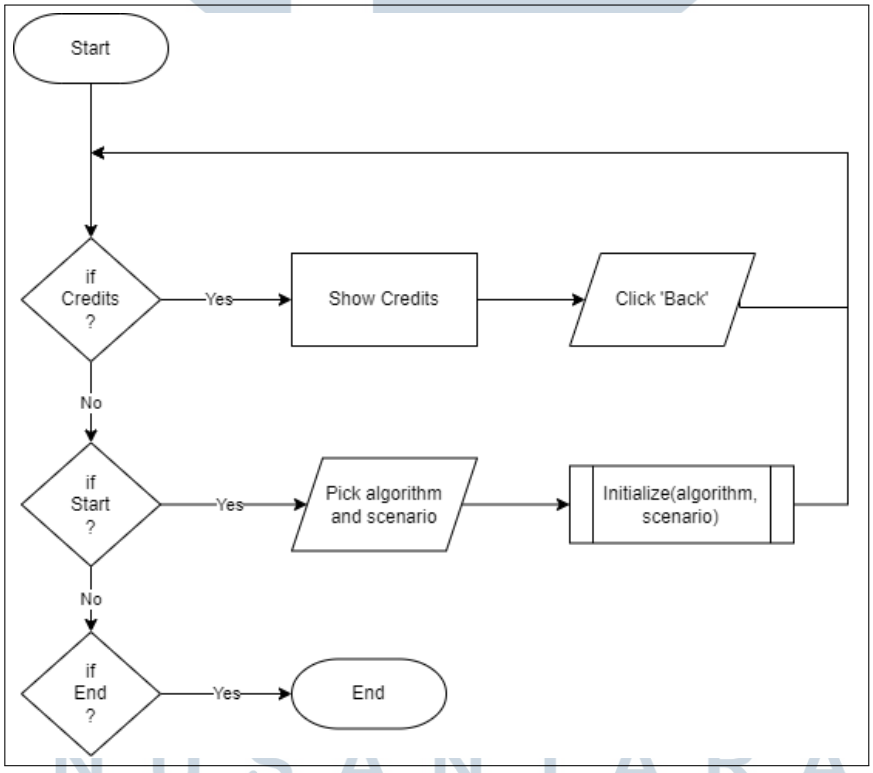
juga berlaku sebagai bukti bahwa perbandingan algoritma telah dilaksanakan dan diselesaikan.

3.2 Perancangan

Dalam penelitian ini, algoritma Flow Field Pathfinding serta algoritma A* Pathfinding merupakan *base code*. Untuk algoritma A* Pathfinding, digunakan *base code* oleh Seb Lague [19], sedangkan untuk algoritma Flow Field Pathfinding, digunakan *base code* dari *Turbo Makes Games* [20] yang kemudian dikembangkan untuk memenuhi kriteria pengujian yang akan dilakukan. Perancangan sistem dilakukan dengan membuat *flowchart* serta perancangan *map*.

3.2.1 Flowchart

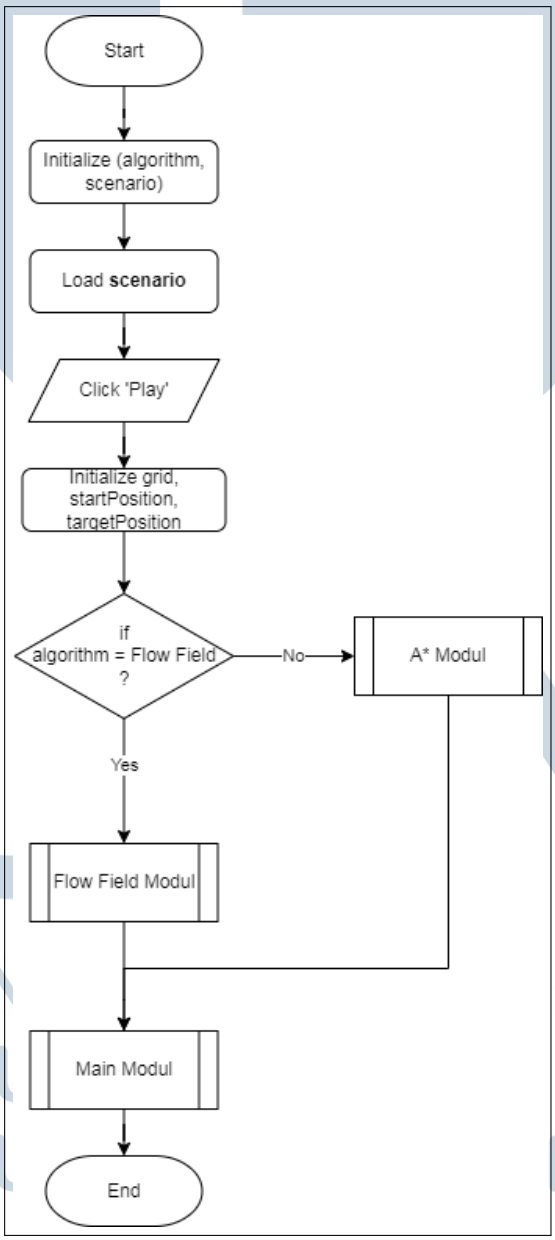
Berikut adalah *flowchart* yang digunakan untuk membuat sistem yang akan dibangun.



Gambar 3.1. Flowchart modul main menu

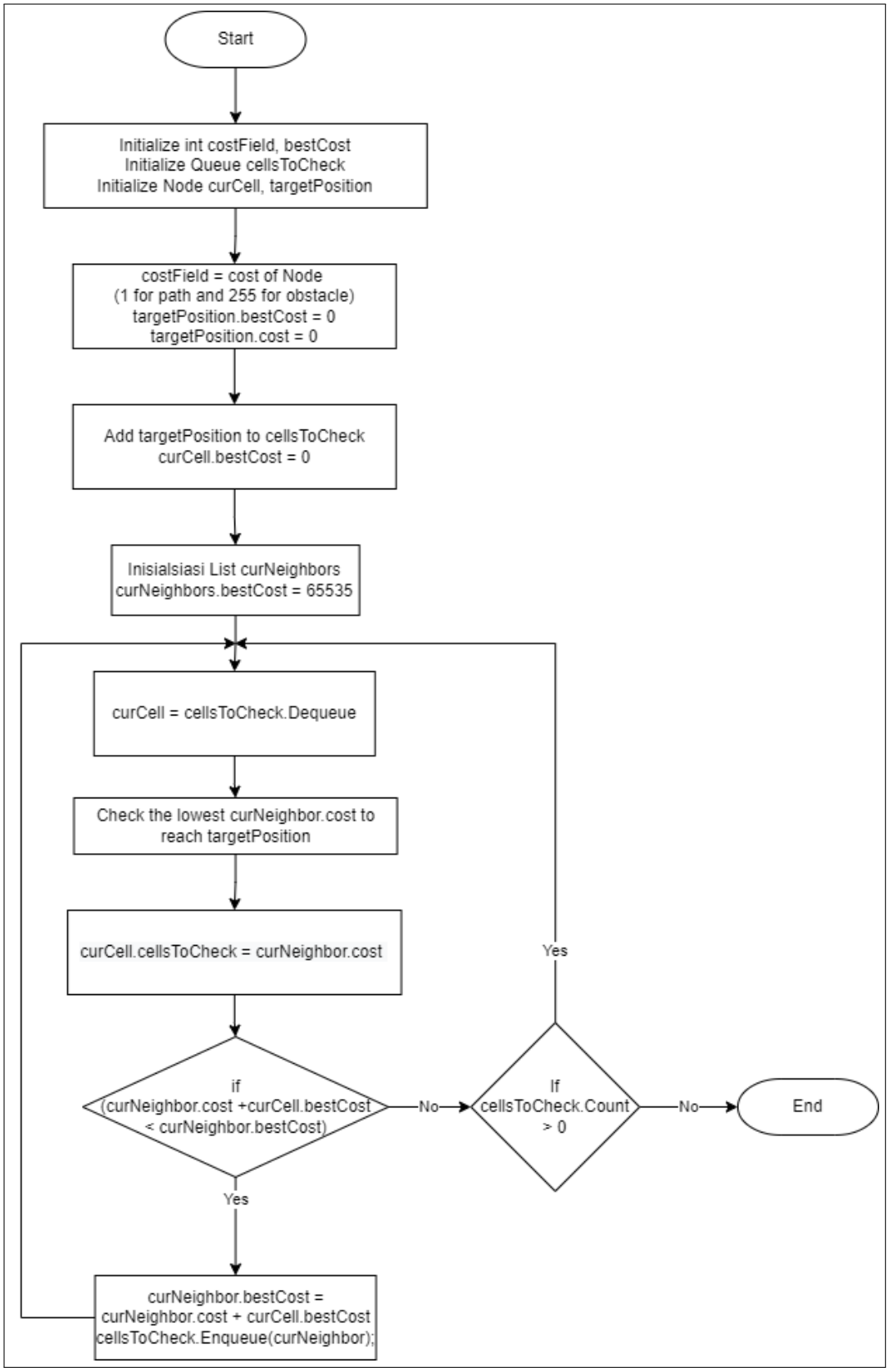
Gambar 3.1 merupakan aluran menu utama yang akan muncul ketika user membuka aplikasi. Terdapat 3 pilihan tombol yaitu ’Start’, ’Credits’, atau ’Exit’.

Jika user memilih ’Credits’, maka credits akan ditampilkan. Untuk kembali ke menu utama, user akan klik tombol ’Back’. Jika user memilih ’Exit’, aplikasi akan ditutup. Setelah user memilih, Jika user memilih ’Start’, akan dimunculkan pilihan untuk memilih algoritma dan skenario. Algoritma yang dapat dipilih yaitu dua: Flow Field Pathfinding dan A* Pathfinding, sedangkan ada 3 skenario untuk setiap algoritma. Setelah user pilih skenario yang diinginkan, proses akan masuk ke modul Initialize di mana dibawa parameter algoritma serta skenario.



Gambar 3.2. Flowchart modul inisialisasi

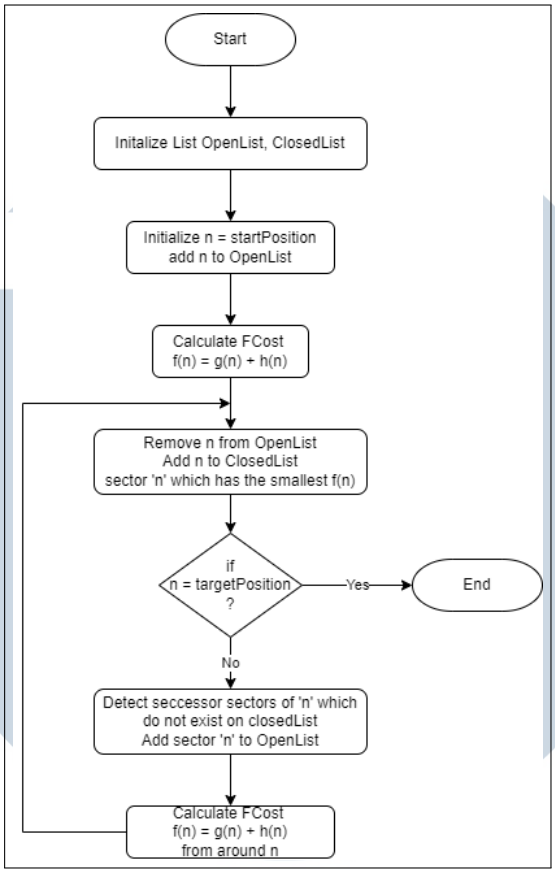
Gambar 3.2 merupakan aluran inisialisasi simulasi. Setelah user telah memilih algoritma serta skenario dari menu utama, proses akan memuat skenario yang dipilih. User akan klik tombol 'Play' untuk menginisialisasi grid, startPosition, serta targetPosition. Algoritma akan berjalan menyesuaikan dengan algoritma yang telah dipilih sebelumnya. Jika algoritma yang dipilih adalah flow field, maka proses akan masuk ke Modul Flow Field dan apabila memilih algoritma A* akan masuk ke Modul A*. Kemudian proses akan memasuki modul utama.



Gambar 3.3. Flowchart modul flow field pathfinding

Gambar 3.3 merupakan aluran dari algoritma Flow Field Pathfinding. Tahap pertama dalam algoritma ini adalah untuk menginisialisasi beberapa hal, terdapat inisialisasi *integer* costField dan bestCost, kemudian inisialisasi *Queue* cellsToCheck, dan inisialisasi *Node* curCell dan targetPosition. Setelah itu nilai costField setiap Node yang terdapat di grid akan di set, nilai tersebut adalah 1 apabila Node tersebut dapat digunakan sebagai jalur, sedangkan apabila Node tersebut tidak dapat dilalui, maka nilai tersebut akan bernilai 255. Node targetPosition juga akan di set targetPosition.bestCost = 0 dan targetPosition.cost = 0, nilai ini berfungsi untuk membedakan nilai Node antara tujuan dengan jalur dan tahap awal untuk mencari jalur terpendek untuk mencapai tujuan. Kemudian menambahkan targetPosition ke *Queue* cellsToCheck dan menigisi nilai curCell.bestCost = 0 sehingga Node tempat tujuan akan di proses paling pertama oleh algoritma.

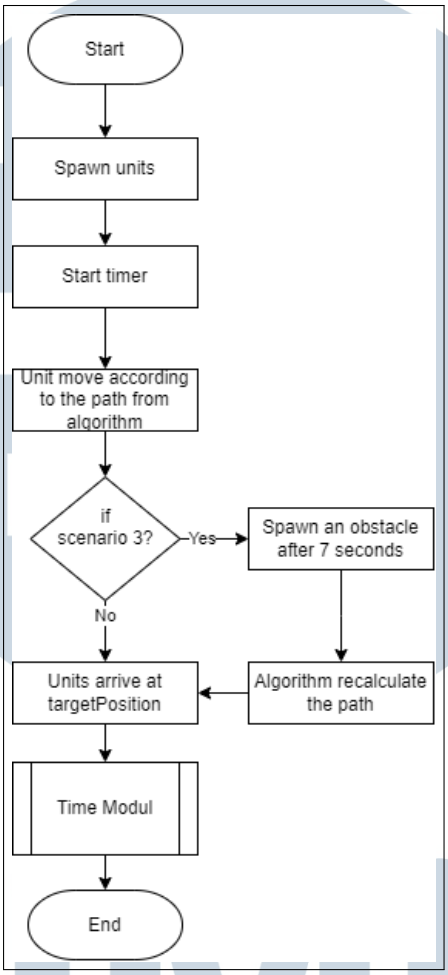
Selanjutnya melakukan inisialisasi List curNeighbors serta mengisi nilai curNeighbors.bestCost dengan nilai yang sangat tinggi (65535) sehinga saat dibandingkan dengan bestCost (jalur) lain yang memiliki nilai yang lebih rendah, jalur tersebut yang akan digunakan. Hal selanjutnya adalah untuk mengisi nilai Node satu per satu dimulai dari curCell, kemudian melakukan pengecekan untuk mencari nilai curNeighbor.cost paling rendah untuk mencapai target, dengan cara membandingkan nilai curNeighbor.cost ditambah curCell.bestCost dengan nilai curNeighbor.bestCost. Apabila nilai lebih kecil maka akan dilanjutkan dengan menentukan nilai Node curNeighbor.bestCost dengan nilai curNeighbor.cost + curCell.bestCost dan menambahkan Node tersebut ke dalam *Queue* yang setelahnya akan melakukan perulangan kembali untuk membaca Node yang terdapat di *Queue*. Apabila nilai tersebut lebih besar akan dilakukan pengecekan apakah masih terdapat *Queue* yang tersisa dari cellsToCheck, apabila iya maka kode akan melakukan perulangan kepada saat algoritma membaca cellsToCheck, apabila *Queue* cellsToCheck sudah kosong hal ini mengartikan bahwa semua Node telah dibaca dan memiliki nilai. Pada tahap ini algoritma *Flow Field* telah dengan berhasil dibuat dan jalur terdekat yang terbuat diantara titik awal dan titik akhir adalah melalui Node dengan total nilai paling rendah.



Gambar 3.4. Flowchart modul A* Pathfinding

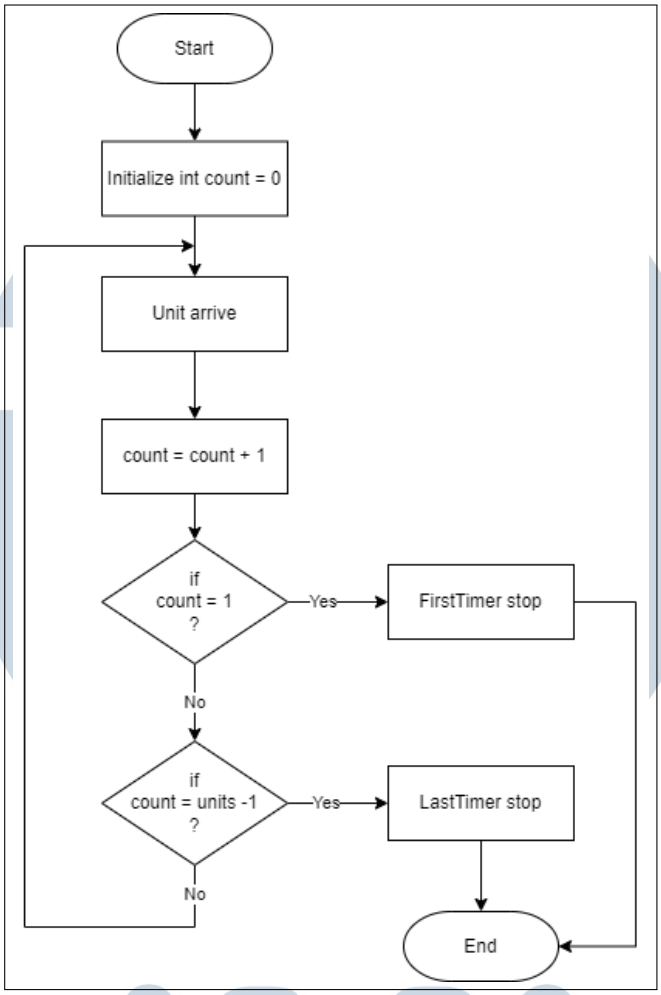
Gambar 3.4 merupakan aluran dari algoritma A* Pathfinding. Algoritma dimulai dengan menginisialisasi List OpenList dan ClosedList, kemudian menginisaliasi n, mengisi nilai n dengan startPosition, dan menambahkan n kepada OpenList. Selanjutnya menghitung nilai FCost dengan rumus f(n) = g(n) + h(n), dimana f(n) merupakan FCost, g(n) merupakan nilai jarak dari Node startPosition dengan Node tujuan, dan h(n) merupakan nilai heuristik, nilai perkiraan yang dibutuhkan dari Node n ke target akhir. Kemudian memindahkan Node n dari OpenList ke ClosedList, hal ini memiliki arti bahwa Node tersebut telah dimasukkan ke dalam jalur yang akan digunakan. Setelah itu algoritma akan mencari Node n yang memiliki nilai f(n) terkecil, tahap berikutnya adalah untuk mencari tahu apakah n merupakan targetPosition, apabila n bukan merupakan targetPosition algoritma akan terus berjalan dengan mencari n yang tidak terdapat dalam ClosedList dan menambahkan Node n kedalam OpenList yang kemudian akan dihitung FCost dan proses perulangan dimulai dengan memindahkan Node n dari OpenList ke ClosedList. Apabila n merupakan targetPosition maka algoritma

telah dengan berhasil dijalankan dan jalur terdekat telah dibuat dengan menelusuri ClosedList yang ada.



Gambar 3.5. Flowchart modul utama

Gambar 3.5 merupakan alur untuk modul utama. Ketika tombol ditekan, entitas sebanyak 10 unit akan muncul dan *timer* akan dimulai. Entitas akan bergerak sesuai dengan jalur yang dibuat oleh algoritma. Jika user memilih skenario 3, setelah 7 detik rintangan akan dimunculkan oleh jalur dan algoritma akan mengkalkulasi jalur baru untuk menghindari rintangan sehingga membuat entitas menghindari rintangan baru yang muncul. Pada akhirnya entitas akan sampai di tujuan dan masuk ke modul waktu. Setelah modul waktu selesai, akan dikalkulasi jarak yang ditempuh oleh entitas pertama yang sampai di tujuan.

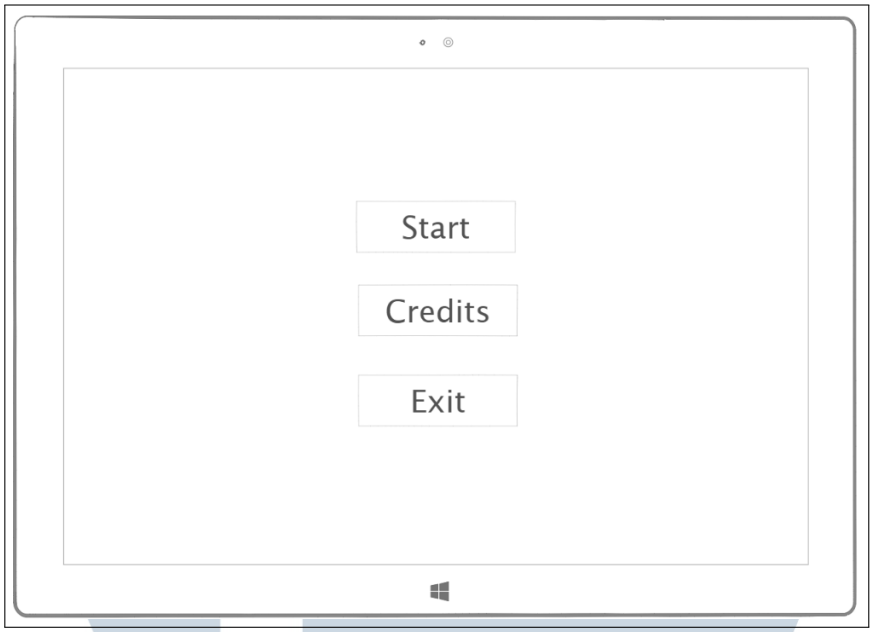


Gambar 3.6. Flowchart modul waktu

Gambar 3.6 merupakan alur untuk modul waktu. Pertama, menginisialisasi *count* untuk menghitung banyaknya *unit* yang dapat terdeteksi. Ketika sebuah *unit* sampai ke tujuan, akan dicek apakah *unit* tersebut adalah *unit* yang pertama (Count = 1). Jika iya, *count* akan bertambah dan waktu akan dicatat. Jika bukan, maka akan dicek apakah *unit* tersebut adalah *unit* terakhir (count = unit - 1). Apabila bukan *unit* terakhir, *count* akan bertambah dan modul waktu akan diulang. Pada saat *unit* adalah *unit* terakhir, timer akan berhenti, waktu akan dicatat, dan modul waktu telah selesai.

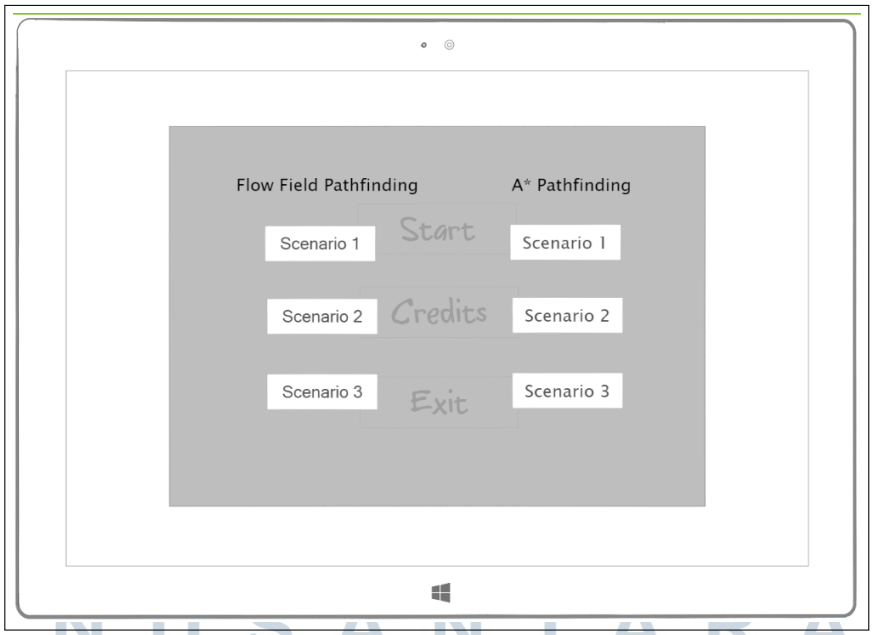
3.2.2 Wireframe

Berikut adalah *wireframe* yang digunakan untuk perancangan simulasi.



Gambar 3.7. Wireframe halaman menu utama

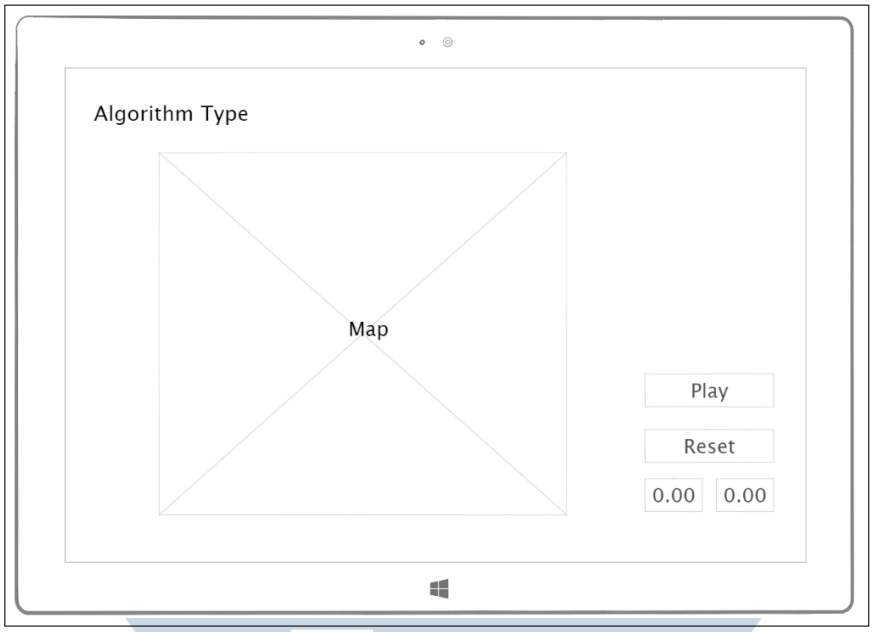
Gambar 3.7 adalah *wireframe* untuk tampilan awal simulasi. Pada halaman ini, user dapat memilih untuk Start, melihat Credits, atau Exit aplikasi. Jika memilih Start, layar akan menampilkan popup menu.



Gambar 3.8. Wireframe popup menu

Gambar 3.8 adalah *wireframe* untuk memilih *scenario* serta jenis *pathfinding* yang akan digunakan dalam simulasi. Setiap *scenario* akan menampilkan *map* yang

berbeda.

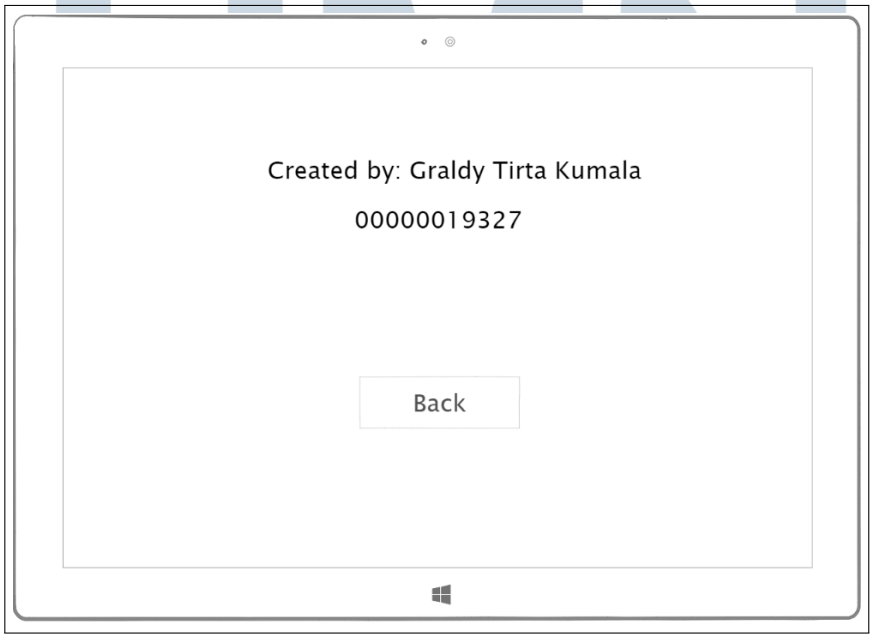


Gambar 3.9. Wireframe simulasi utama

Gambar 3.9 merupakan *wireframe* untuk simulasi utama. Tipe algoritma yang dipakai berada di kiri atas halaman. Dalam tengah halaman terletak *map* yang akan dipakai untuk menjalankan simulasi. *Spawn* merupakan sebuah *area* dan bukan hanya satu titik dan *target* adalah area tujuan akhir unit-unit. Terletak di kanan bawah adalah *timer* untuk menunjukkan waktu yang telah berlalu untuk entitas mencapai tujuan. Dalam perancangan, dimiliki dua *timer* yaitu First Unit Timer untuk menghitung waktu yang diperlukan untuk unit pertama yang telah mencapai target serta Last Unit Timer untuk menghitung waktu yang diperlukan untuk unit terakhir yang mencapai target. Terletak juga tombol 'Play' untuk memulai simulasi dan tombol 'Reset' untuk mengulang simulasi.

Gambar 3.10. Wireframe pause menu

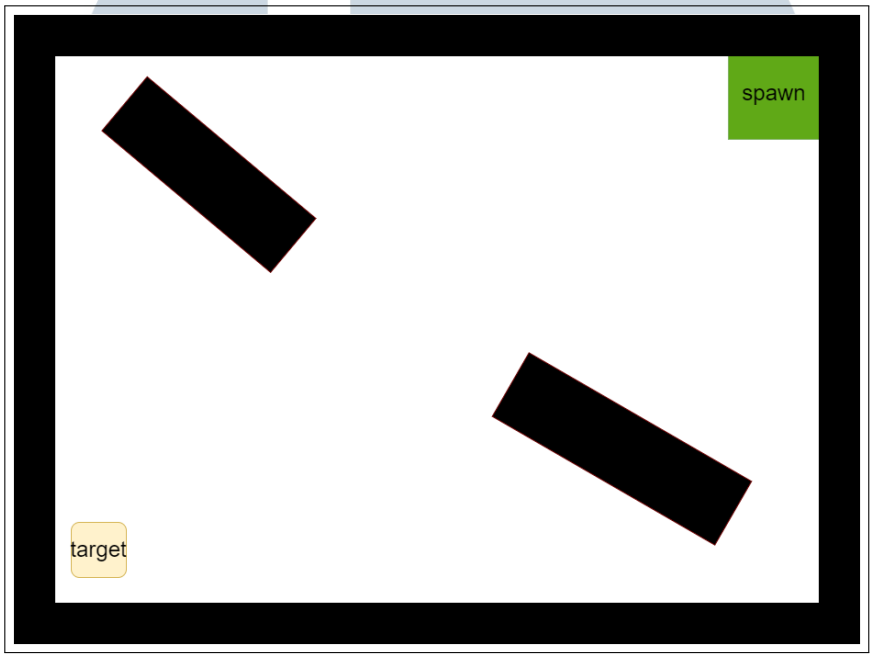
Gambar 3.10 adalah *wireframe* untuk pause menu ketika user menekan tombol 'Escape' dari keyboard. Terdapat 3 pilihan yaitu 'Resume', 'Menu', serta 'Quit'. Jika user menekan tombol 'Resume', simulasi akan berjalan kembali. Jika user menekan tombol 'Menu', maka tampilan akan beralih ke halaman Main Menu. Jika user menekan tombol 'Quit', maka aplikasi akan ditutup.



Gambar 3.11. Wireframe halaman credits

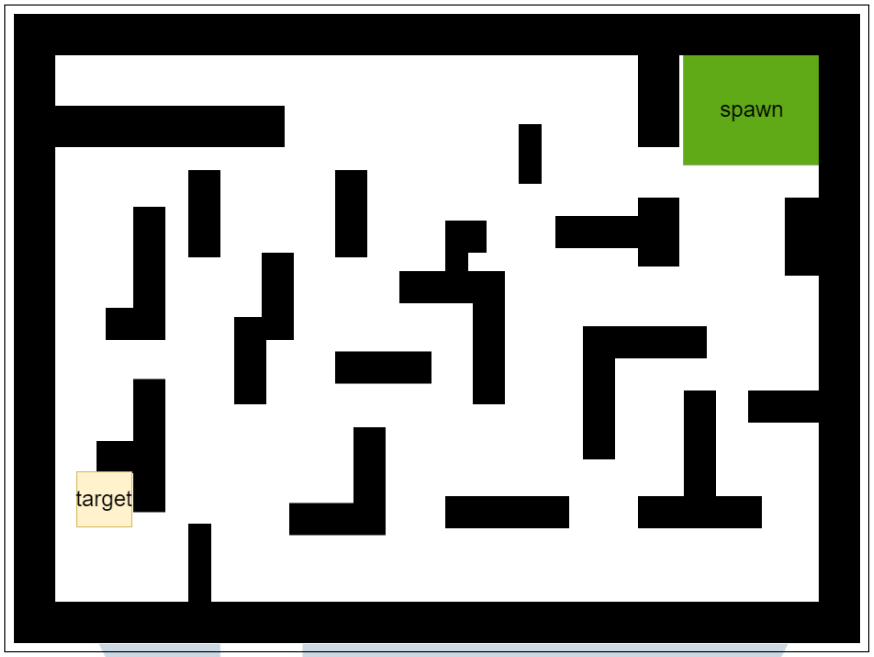
Gambar 3.11 merupakan *wireframe* untuk halaman Credits. Halaman ini memiliki informasi penulis serta tombol 'Back' untuk kembali ke halaman main menu.

3.2.3 Perancangan Map



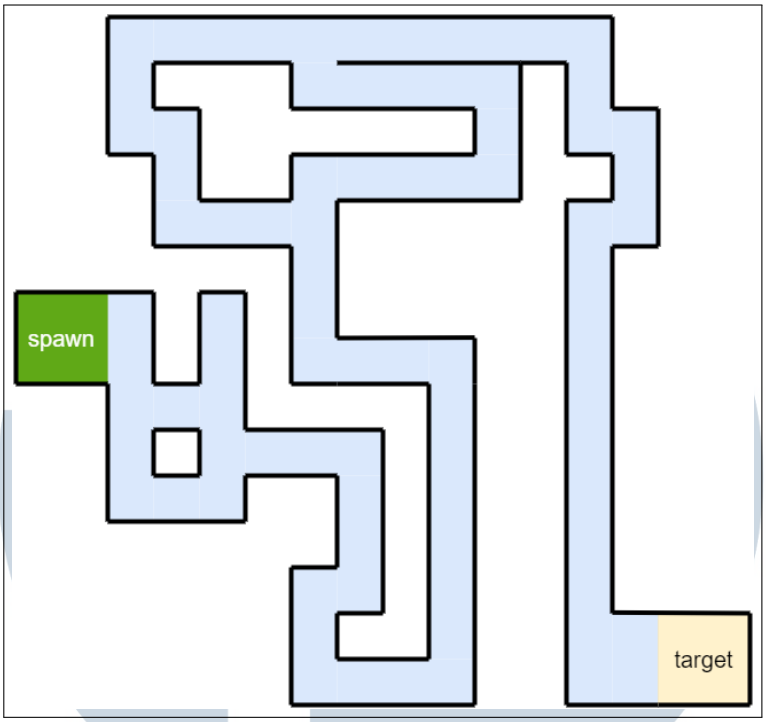
Gambar 3.12. Perancangan map Scenario 1

Gambar 3.12 adalah perancangan *map* untuk skenario 1.

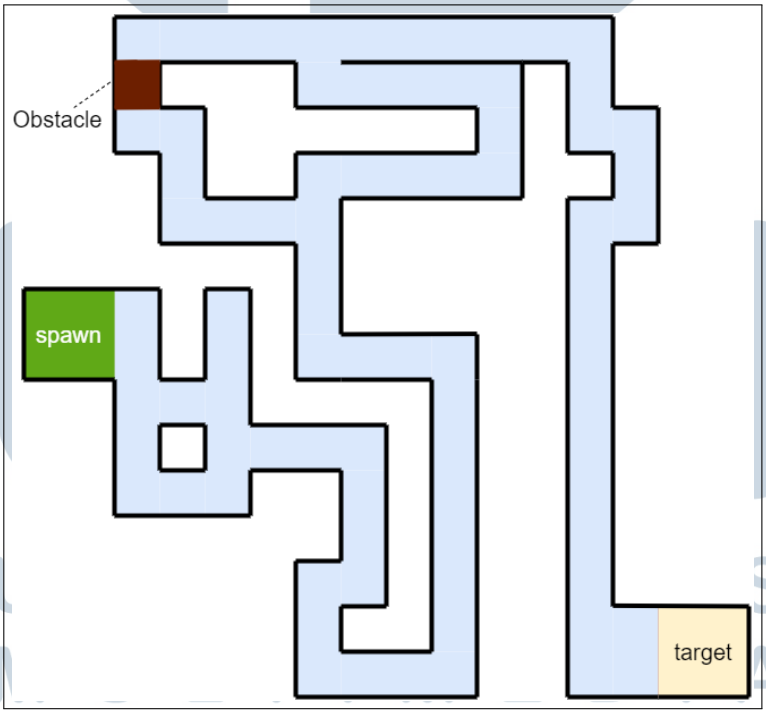


Gambar 3.13. Perancangan map Scenario 2

Gambar 3.13 adalah perancangan *map* untuk skenario 2.



(a) Sebelum rintangan



(b) Sesudah rintangan

Gambar 3.14. Perancangan map simulasi Skenario 3

Gambar 3.14a adalah *map* ketika simulasi awal dimulai dan Gambar 3.14b merupakan rancangan *map* setelah 7 detik simulasi.