

# IMPLEMENTASI PRIM'S ALGORITHM DAN A\* PATHFINDING PADA GAME ROGUELIKE SEDERHANA

Irawan Maulana<sup>1)</sup>, Joko Christian Chandra<sup>2)</sup>

Program Studi Teknik Informatika, Fakultas Teknologi Informasi, Universitas Budi Luhur.  
Jl. Raya Ciledug, Petukangan Utara, Kebayoran Lama, Jakarta Selatan  
Telp. (021) 5853753, Fax (021) 5866369  
E-mail: [chokoknight@gmail.com](mailto:chokoknight@gmail.com)<sup>1)</sup>, [joko.christian@budiluhur.ac.id](mailto:joko.christian@budiluhur.ac.id)<sup>2)</sup>

## ABSTRAK

*Roguelike* merupakan salah satu *genre* dalam *video game* yang menitikberatkan pada eksplorasi dalam *dungeon* yang dibuat secara acak dan permainan yang kental dengan unsur *RPG (Role Playing Game)*; lengkap dengan pertempuran yang bersifat *turn-based*. *Genre* tersebut akan diimplementasikan secara sederhana dalam sebuah *game* yang diberi judul *RogueTA. Dungeon* yang akan digunakan berbentuk labirin (*maze*) dan akan diisi oleh musuh yang bertugas mengejar pemain. *Dungeon* tersebut akan dibuat secara acak dengan *Prim's Algorithm* sedangkan *A\* Pathfinding* memungkinkan musuh untuk mengejar pemain. Tujuan utama *game* ini adalah mengumpulkan skor sebanyak-banyaknya yang nantinya dapat ditunjukkan pada papan skor. Permainan akan berakhir jika pemain kehabisan nyawa atau pemain mencapai tingkat terakhir dalam permainan.

Kata kunci : Game , Pathfinding , Labirin, Maze, Algoritma A\* , Prim's Algorithm , Roguelike.

## 1. PENDAHULUAN

### 1.1. Latar Belakang

*Genre roguelike* merupakan salah satu contoh sebuah *genre* yang lahir berkat sebuah *video game*. *Genre* ini mengacu pada permainan yang mirip seperti *video game* berjudul *Rogue: Exploring The Dungeon of Doom*. *Rogue* merupakan permainan *dungeon crawling rpg (role playing game)* yang menerapkan eksplorasi pada dunia yang dibuat secara acak dengan dunia yang direpresentasikan dalam bentuk grid; *dungeon crawling* merupakan istilah yang digunakan untuk fitur permainan berupa penjelajahan *dungeon* seperti kastil tua, gua, atau tempat *indoor* lainnya. Biasanya *genre* ini memiliki *stage* yang dibuat secara acak dengan musuh yang menghalangi pemain untuk menyelesaikan permainan. Tujuan *game* dengan *genre* ini umumnya untuk mencapai tingkat terakhir atau mengumpulkan benda tertentu yang ada dalam dunia permainan.

Pada *RogueTA*, definisi dari *genre roguelike* akan diimplementasikan secara sederhana dengan cara menghilangkan beberapa unsur dari *genre* tersebut walau tetap menjaga ciri khas dari *genre* tersebut. Unsur-unsur yang dihilangkan seperti sistem pertempuran ataupun unsur *RPG* akan diganti dengan sistem yang lebih sederhana.

Untuk *stage* yang akan diimplementasikan berupa labirin (*maze*) dengan menggunakan *Prim's Algorithm*. Sedangkan *A\* Pathfinding* akan diimplementasikan pada musuh agar mereka dapat mengejar pemain dan menghalangi mereka untuk menyelesaikan permainan. Pemain akan diberikan nyawa yang akan berkurang jika pemain tertangkap oleh musuh dan akan bertambah jika pemain mengumpulkan *item* berupa hati. Permainan akan berakhir jika nyawa pemain habis atau ketika pemain mencapai tingkat terakhir. Dalam permainan ini juga diimplementasikan sistem skor dimana pemain dapat menambah skor mereka dengan cara mengumpulkan *item* berupa koin ataupun dengan cara naik ke tingkat selanjutnya. Skor tersebut nantinya akan ditampilkan pada papan skor yang dapat diakses melalui menu utama. Tujuan dari permainan ini adalah untuk mengumpulkan skor sebanyak-banyaknya sebelum pemain kehabisan nyawa.

### 1.2. Permasalahan

Berdasarkan uraian dari latar belakang, maka inti yang menjadi permasalahan dari tugas akhir ini adalah bagaimana caranya menciptakan permainan roguelike yang sederhana dan bagaimana caranya mengimplementasikan prim's

algorithm dan A\* pathfinding pada lingkungan permainan.

**1.3. Batasan Masalah**

Untuk menjaga agar penulisan dan pengembangan skripsi ini tetap berada dalam lingkup topik yang diajukan, beberapa batasan yang akan diberikan dalam pengerjaan skripsi ini antara lain:

- a. Permainan akan dikembangkan sebagai aplikasi berbasis desktop, dengan menggunakan *Monogame* sebagai *engine* dalam proses pengembangannya dan ditulis dalam bahasa pemrograman *C#*.
- b. Beberapa *framework* akan digunakan untuk mempermudah pembuatan beberapa fungsi dasar, tetapi implementasi algoritma utama akan diusahakan tidak menggunakan *framework* apapun.
- c. Peta atau *map* yang menjadi dunia dalam permainan ini akan berupa *maze* atau labirin yang akan dibuat dan diatur oleh *prim's algorithm*.
- d. Agar pengembangan dapat tercapai tepat waktu, beberapa penyesuaian akan dilakukan terhadap permainan yang akan dikembangkan seperti mengganti beberapa unsur kunci dalam *genre roguelike* dan menggantinya dengan unsur yang lebih sederhana.
- e. Algoritma yang digunakan dalam permainan ini tidak terbatas kepada kedua algoritma yang diajukan, walau tidak menjadi fokus utama seperti algoritma yang diajukan.

**1.4. Metode Pengembangan**

Metode pengembangan yang digunakan adalah metode *build-and-fix* atau dikenal juga sebagai metode *ad-hoc* yang merupakan metode pengembangan dengan *requirement* paling minim karena tanpa desain pasti dan spesifikasi yang jelas. Metode ini baik digunakan untuk proyek berskala kecil ataupun untuk pengembang yang tidak memiliki pengetahuan yang baik mengenai lingkungan proyek ataupun proyek itu sendiri.

**2. LANDASAN TEORI**

**2.1. Pengertian Graph**

*Graph* memiliki beberapa artian tergantung dari bagaimana penggunaannya, walaupun pada skripsi ini istilah *Graph* mengacu pada struktur

matematis yang digunakan untuk menggambarkan hubungan antara beberapa object. Menurut Keijo Ruohonen (2013), secara konseptual sebuah *Graph* dibentuk oleh titik-titik (*vertices*) dengan sisi (*edges*) yang menghubungkan titik-titik tersebut. Sebagai contoh:



Gambar 2. 1 : Contoh *Graph*

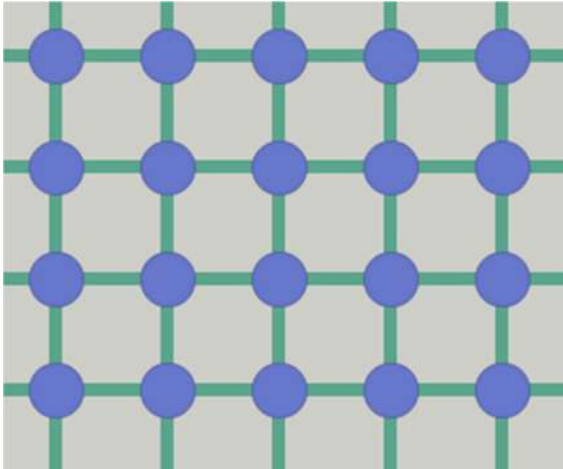
Dan secara formal, sebuah *Graph* merupakan sekumpulan pasangan dari  $(V,E)$ , dimana  $V$  melambangkan sekumpulan titik dan  $E$  melambangkan sekumpulan sisi, yang dibentuk oleh hubungan antar titik. Sisi atau yang dilambangkan dengan  $E$  merupakan *multiset*, dengan kata lain, anggota  $E$  dapat muncul lebih dari satu kali sehingga tiap anggota memiliki sebuah *multiplicity* atau kelipatan.

**2.2. Pengertian Grid**

*Grid* merupakan sebuah struktur dua dimensi yang dibangun dari perpaduan garis horizontal dan vertikal. Bentuk *grid* dipilih karena bentuk ini biasa digunakan dalam merepresentasikan planar pada game berbasis dua dimensi, dan juga karena lebih mudah untuk memanipulasi objek dimana tiap objek ditempatkan berdasarkan koordinat dalam *grid*.

	A	B	C	D	E	F	G	H	I	J
1										
2										
3										
4										
5										
6										
7										
8										
9										
10										

Gambar 2. 2 : Contoh Grid



Gambar 2. 3 : Impemementasi grid pada graph (Amit Pattel, 2014)

### 2.3. Pengertian Pathfinding

*Pathfinding* merupakan metode yang digunakan untuk dapat menemukan jalan terbaik dari satu titik kepada titik tujuan. *Pathfinding* berhubungan erat dengan konsep *Graph* karena pada konsep inilah *pathfinding* dapat digunakan secara optimal melalui bentuk dari *Graph* tersebut yang menyediakan informasi yang diperlukan oleh metode *pathfinding*, yaitu titik dan jalan yang direpresentasikan oleh sisi.

### 2.4. Prim's Algorithm

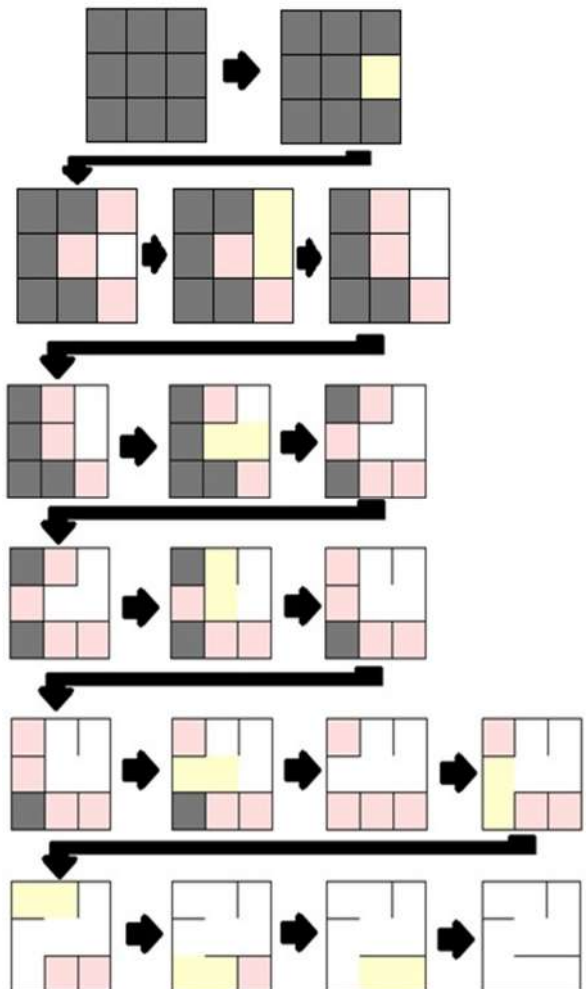
*Prim's algorithm* merupakan sebuah algoritma *greedy* untuk mencari *minimum spanning tree* atau *MST* dalam sebuah *weighted undirected Graph* layaknya *kruskal's algorithm*. *Greedy* merupakan sebutan untuk tipe algoritma yang memecahkan permasalahan heuristik dengan solusi paling optimal di tiap fasenya. Berbeda dengan *kruskal's* yang bekerja dengan memilih sisi secara acak lalu memasukkannya ke dalam set jika mereka membentuk *connected disjoint tree*, *prim's algorithm* dimulai dari satu titik lalu menyebar ke sisi dengan bobot terkecil hingga semua titik dilewati.

Untuk dapat diimplementasikan pada konsep *grid*, terlebih lagi konsep *maze*, diperlukan sedikit perubahan terhadap algoritma ini. Perubahan tersebut bertujuan untuk dapat memilih *cell* selama *cell* tersebut masih dalam "*frontier*". *Frontier* adalah istilah untuk *cell-cell* yang bertetangga dengan dengan *MST*, atau dapat

diibaratkan sebagai garis depan ekspansi *maze*. Berikut ini tahapan *prim's algorithm* yang telah disesuaikan untuk membuat *maze* pada bidang *grid* menurut Jamis Buck (2011):

- 1) Pilih *cell* acak dari *G* (*grid*), dan tambahkan ke *maze*
- 2) Pilih *cell* acak dari *frontier* yang bukan merupakan anggota *maze*.
- 3) Tambahkan *cell* tersebut kedalam *maze*.
- 4) Ulangi langkah 2 dan 3 hingga semua *cell* terjelajahi.

Dari langkah tersebut dapat dilihat bahwa perubahan yang dimaksud adalah memilih *cell* acak dari *frontier* daripada memilih sisi dengan bobot terpendek.



Gambar 2. 4 : Memilih cell acak dari *frontier* daripada memilih sisi dengan bobot terpendek pada *Prim's Algorithm*

## 2.5. A\* Algorithm

A\* merupakan sebuah algoritma yang sering digunakan untuk *pathfinding* pada teori *Graph*. Algoritma ini merupakan modifikasi dari *dijkstra's algorithm* yang merupakan algoritma yang difokuskan untuk pencarian jalan tercepat dari satu titik ke satu titik lainnya. Perbedaannya dengan *dijkstra's algorithm*, dilihat dari fleksibilitas A\* yang mengimplementasi metode heuristik untuk membimbing pencariannya; \* dari nama algoritma ini artinya ada metode heuristik yang digunakan.

Dalam proses kerjanya, A\* menerapkan notasi  $F = G + H$  untuk perhitungan jalan dimana G merupakan jarak dari posisi saat ini ke akhir, dan H merupakan jarak dari titik akhir ke posisi saat ini. Selain itu, algoritma ini menyimpan daftar titik yang telah ditempuh dan titik yang akan ditempuh sebagai *open list* dan *closed list*. *Open list* atau bisa juga yang disebut *frontier*, merupakan daftar dari titik yang bertetangga dengan titik saat ini. *Open list* berarti titik-titik yang dapat dilalui dan dapat menjadi jalur ke tujuan. *Closed list* merupakan titik-titik yang telah dipilih sebagai jalan sehingga tidak perlu memeriksa titik yang berada di daftar ini. Ada juga istilah yang disebut *parent*, yang merupakan titik asal yang menjadi acuan. *Parent* bukan berarti titik awal, melainkan status bantuan untuk menelusuri kembali jalan yang telah dibuat. Anggap saja sebagai sambungan titik sekarang dengan titik sebelumnya.

Adapun tahapan dari cara kerja algoritma A\* yang dijabarkan oleh Patrick Lester (2005); dalam bidang *grid* titik disebut sebagai *square* atau *cell*.

- 1) Masukkan *cell* awal ke *open list*.
- 2) Ulangi tahapan-tahapan ini:
  - a) Cari *cell* dengan nilai F terendah yang ada di *open list*. Nantinya akan menjadi *cell* saat ini.
  - b) Masukkan *cell* tersebut ke *closed list*, yang merupakan kumpulan *cell* yang tidak perlu dicek kembali, dan buang dari *open list*
  - c) Untuk semua *cell* yang bertetangga dengan *cell* tersebut, lakukan ini

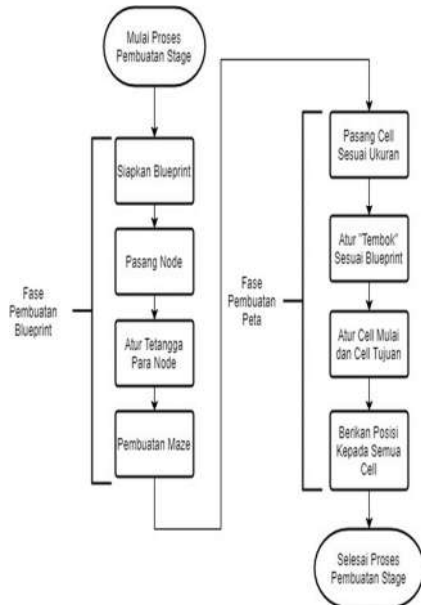
- (1) Jika tidak dapat dilalui atau sudah berada di *closed list*, maka diabaikan. Jika tidak, maka lakukan hal selanjutnya
  - (2) Jika *cell* tersebut belum termasuk di *open list*, maka masukkan ke *open list*. Gunakan *cell* saat ini sebagai *parent* dari *cell* tersebut lalu hitung nilai F, G, dan H untuk *cell* tersebut. Jika sudah termasuk dalam *open list*, maka lakukan hal selanjutnya.
  - (3) Cek apakah jalan dari *cell* saat ini ke *cell* tersebut lebih baik, dengan membandingkan nilai G. Nilai G yang lebih rendah berarti *cell* tersebut lebih baik, maka ganti *parent cell* tersebut ke *cell* saat ini dan hitung ulang nilai F dan G dari *cell* tersebut.
- d) Pengulangan berhenti jika
- (1) Memasukkan *cell* tujuan ke *closed list*, yang berarti telah menemukan jalan ke tujuan.
  - (2) Gagal menemukan *cell* tujuan dan *open list* kosong, berarti tidak ditemukan jalan ke tujuan.
  - (3) Simpan jalan yang telah didapat. Jika mau menelusuri ke titik awal, telusuri tiap *cell* melalui *parent cell* hingga sampai ke *cell* awal.

## 3. RANCANGAN APLIKASI

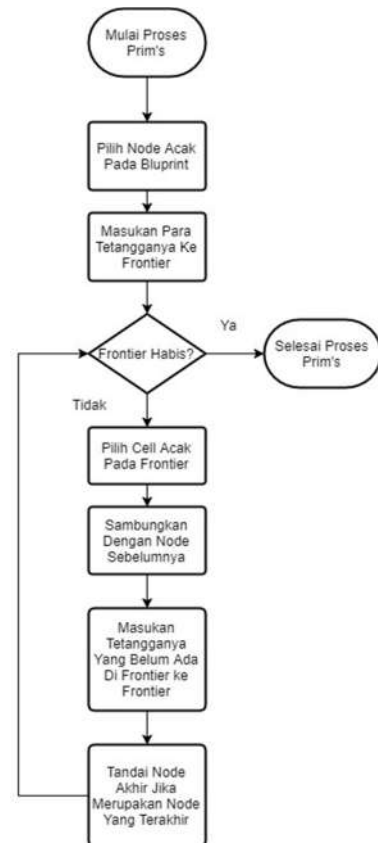
### 3.1. Pembuatan Stage

*Stage* dalam permainan ini berarti sebuah tempat permainan yang tersusun oleh banyak *grid* dan tiap *grid*-nya tersusun oleh banyak *cell*, hal tersebut bertujuan agar tempat permainan menjadi luas sehingga memungkinkan pemain untuk menghindari musuh. *Stage* memiliki *layout* atau *map* yang dibangun berdasarkan dari sebuah *blueprint*. *Blueprint* terdiri dari sekumpulan komponen yang disebut dengan *node* yang saling terhubung, dimana terhubung atau tidaknya *node* menentukan bagian dari *grid* mana yang menjadi tembok dan bagian mana yang tidak. Hal tersebut diterapkan untuk memudahkan proses

implementasi *Prim's Algorithm* pada dunia permainan.



Gambar 3. 1 : Proses Pembuatan *Stage*

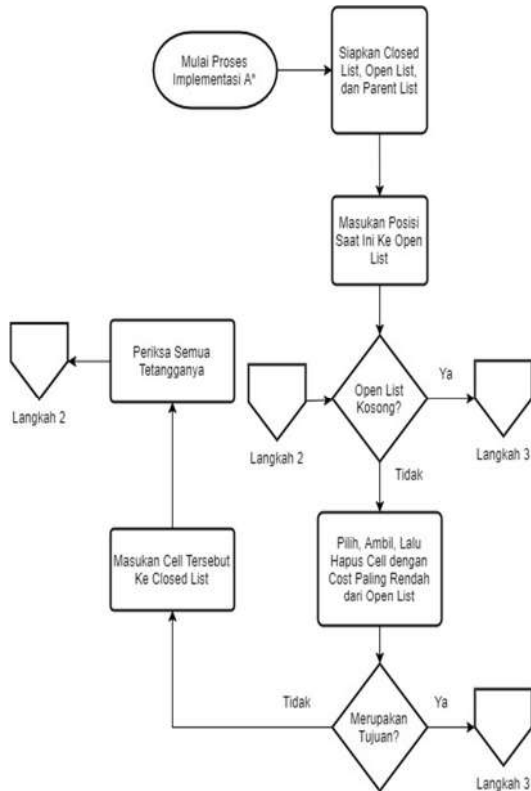


Gambar 3. 2 : Proses Pembuatan *Maze* dengan *Prim's Algorithm*

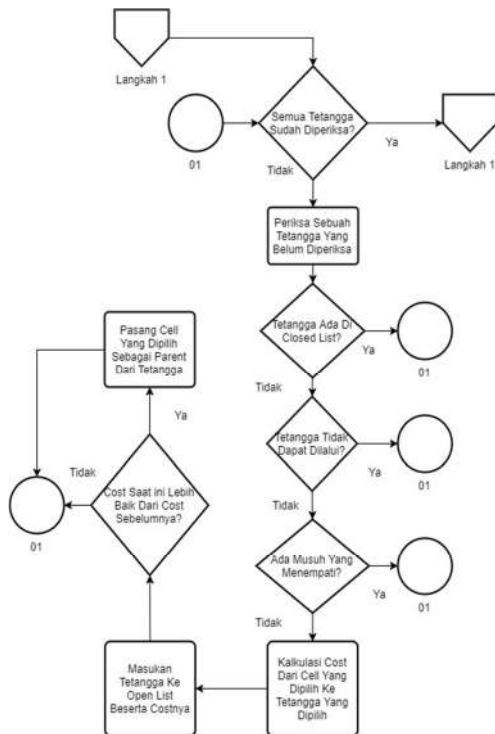
### 3.2. Rancangan AI Musuh

Dalam pengimplementasiannya, kepintaran musuh akan dibagi menjadi tiga fase, yaitu *patrol*, *engaging*, dan *alert*. *Patrol* ketika musuh belum menemukan pemain dan mereka hanya akan menelusuri *maze* saja. *Engaging* ketika musuh menemukan pemain dan mereka akan mengejar pemain. Dan *alert* ketika pemain sudah berada diluar jangkauan pengelihatannya mereka dan mereka akan menelusuri sedikit titik terakhir pemain terlihat sebelum akhirnya kembali ke *patrol*.

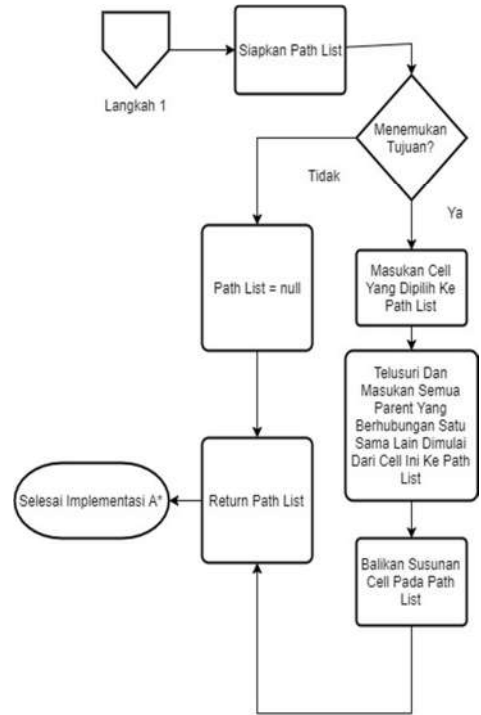
Untuk dapat bergerak dari satu *cell* ke *cell* lain, *A\* Pathfinding* akan diimplementasikan pada musuh. Berikut ini merupakan beberapa *flowchart* yang menggambarkan cara kerja *A\* Pathfinding* yang akan diimplementasikan pada musuh dalam permainan ini.



Gambar 3.3 : Langkah 1 Dari Algoritma A\*



Gambar 3.4 : Langkah 2 Dari Algoritma A\*

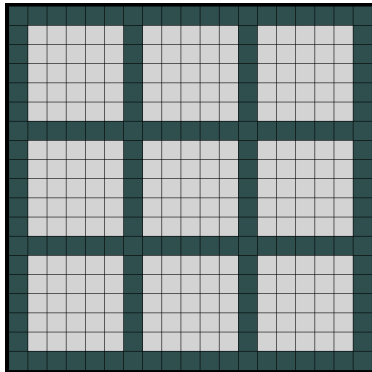


Gambar 3.5 : Langkah 3 Dari Algoritma A\*

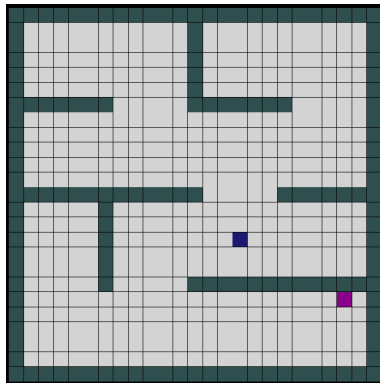
## 4. IMPLEMENTASI DAN UJI COBA PROGRAM

### 4.1. Implementasi Stage Permainan

Berikut ini merupakan hasil dari implementasi *stage* untuk dunia permainan. Pada gambar pertama merupakan *stage* yang dibuat berdasarkan *blueprint* biasa tanpa menggunakan *Prim's Algorithm* dimana semua *node* dalam *blueprint* tersebut tidak terhubung. Gambar kedua merupakan *stage* yang dibangun berdasarkan *blueprint* yang telah diimplementasikan *Prim's Algorithm*.



Gambar 4. 1 : Grid 3x3



Gambar 4. 2 : Sebuah Maze Berukuran 3x3

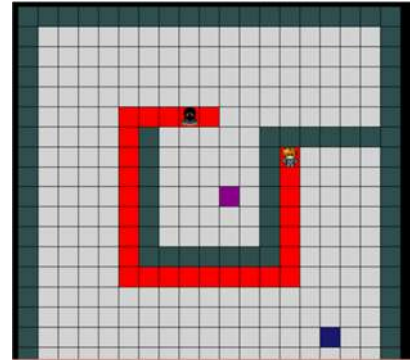
Cell berwarna putih merupakan cell yang dapat dilalui, yang berwarna abu-abu merupakan tembok, yang berwarna biru merupakan *starting point* dimana pemain ditempatkan pertama kali pada saat awal tingkat dan yang berwarna ungu adalah *destination point* merupakan tujuan pemain untuk naik tingkat.

#### 4.2. Implementasi AI Musuh

Berikut ini merupakan hasil dari implementasi  $A^*$  Pathfinding pada musuh.



Gambar 4. 3 : Pathfinding Tanpa Halangan



Gambar 4. 4 : Pathfinding Dengan Halangan

## 5. PENUTUP

### 5.1. Kesimpulan

Berikut ini merupakan beberapa kesimpulan yang dapat ditarik:

- Pengembangan *game* tidak jauh berbeda dengan pengembangan perangkat lunak lainnya, sehingga segala hal yang berkaitan dengan pengembangan perangkat lunak pada umumnya seperti model pengembangan dan desain struktur perangkat lunak juga berlaku untuk *game*.
- Algoritma *prim* dan  $A^*$  merupakan algoritma yang erat kaitannya dengan teori graf, sehingga kedua algoritma ini dapat digunakan dalam berbagai aplikasi yang mengimplementasi teori graf seperti aplikasi pemetaan untuk menemukan jalur terbaik. Karena teori graf juga erat dengan himpunan, maka kedua algoritma tersebut juga dapat diimplementasikan untuk penanganan data, seperti pencarian atau pun sortir data.
- Algoritma  $A^*$  merupakan sebuah algoritma yang flexibel dan kokoh. Karena pada dasarnya algoritma ini merupakan pengembangan dari *dijkstra* yang diberi perhitungan heuristik, maka pola perilaku algoritma ini dapat diubah dengan cara mengubah perhitungan heuristiknya. Sehingga performa algoritma ini sangat bergantung pada nilai heuristik yang diimplementasikan pada algoritma ini.

### 5.2. Saran

Walau game merupakan sebuah aplikasi yang belum memiliki arsitektur standar, dalam perancangan arsitektur game lebih baik mementingkan arsitektur yang bersifat loosely coupled architecture. Hal ini bertujuan agar penambahan dan pengubahan suatu modul tidak melibatkan modul lainnya.

### DAFTAR PUSTAKA

- [1] Hartanto, Toni Rio. 2014. *Implementasi Algoritma A\* Untuk Pergerakan Musuh pada Permainan Corpse Escape*. Jurnal Universitas Budi Luhur.
- [2] Dzikrillah, Ravy. 2016. *Implementasi Algoritma A\* Pada Game First Person Shooter*. Jurnal Universitas Budi Luhur.
- [3] Buck, Jamis. 2015. *Mazes for Programmers: Code Your Own Twisty Little Passages*. Pragmatic Bookshelf.
- [4] Reed, Aaron. 2010. *Learning XNA 4.0*. O'Reilly.
- [5] Nystorm, Robert. 2014. *Game Programming Patterns*. Genever Benning.
- [6] Shvets, Alexander. *Design Patterns Explained Simply* [online], dilihat 29 Oktober 2017, <[https://sourcemaking.com/design\\_patterns](https://sourcemaking.com/design_patterns)>
- [7] Patel, Amit. n.d. *Amit's A\* Pages* [online], dilihat 27 November 2017, <<http://theory.stanford.edu/~amitp/GameProgramming/>>
- [8] Lester, Patrick. 2005. *A\* Pathfinding for Beginners* [online], dilihat 24 Oktober 2017 <<http://homepages.abdn.ac.uk/f.guerin/pages/teaching/CS1013/practicals/aStarTutorial.htm>>
- [9] Jaegers, Kurt. 2012. *XNA 4 3D Game Development by Example Beginner's Guide*, dilihat 7 November 2017, <<http://2.droppdf.com/files/1QVrQ/xna-4-3d-game-development-by-example.pdf>>
- [10] Rouhonen, Keijo. 2013. *Graph Theory*, dilihat 17 Oktober 2017, <[http://math.tut.fi/~ruohonen/GT\\_English.pdf](http://math.tut.fi/~ruohonen/GT_English.pdf)>.