TUGAS AKHIR – KI141502

RANCANG BANGUN PEMBANGKIT WORLD DINAMIS MENGGUNAKAN ALGORITMA RECURSIVE BACKTRACKING PADA GAME 2D PLATFORMER "MINE MEANDER"

FAISHAL AZKA JELLYANTO
NRP 5112100061

Dosen Pembimbing I
Imam Kuswardayan, S.Kom., M.T.

Dosen Pembimbing II
Dr.Eng. Nanik Suciati, S.Kom., M.Kom.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2016

UNDERGRADUATE THESES – KI141502

DYNAMIC WORLD GENERATOR USING RECURSIVE BACKTRACKING ALGORITHM FOR 2D PLATFORMER GAME "MINE MEANDER"

FAISHAL AZKA JELLYANTO
NRP 5112100061

Supervisor I
Imam Kuswardayan, S.Kom., M.T.

Supervisor II
Dr.Eng. Nanik Suciati, S.Kom., M.Kom.

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2016

LEMBAR PENGESAHAN

RANCANG BANGUN PEMBANGKIT WORLD DINAMIS MENGGUNAKAN ALGORITMA RECURSIVE BACKTRACKING PADA GAME 2D PLATFORMER "MINE MEANDER"

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Rumpun Mata Kuliah Interaksi Grafika dan Seni
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh
FAISHAL AZKA JELLYANTO
NRP : 5112 100 061

Disetujui oleh Dosen Pembimbing Tugas Akhir:

1. Imam Kuswardayan, S.Kom., M.T. .....
   NIP: 19761215 200312 1 001 (Pembimbing 1)

2. Dr.Eng. Nanik Suciati, S.Kom., M.Kom. .....
   NIP: 19710428 199412 2 001 (Pembimbing 2)

SURABAYA
JULI, 2016

RANCANG BANGUN PEMBANGKIT WORLD DINAMIS MENGGUNAKAN ALGORITMA RECURSIVE BACKTRACKING PADA GAME 2D PLATFORMER "MINE MEANDER"

Nama Mahasiswa	**: Faishal Azka Jellyanto**
NRP	**: 5112100061**
Jurusan	**: Teknik Informatika FTIF-ITS**
Dosen Pembimbing 1	**: Imam Kuswardayan S.Kom., M.T.**
Dosen Pembimbing 2	**: Dr.Eng. Nanik Suciati S.Kom., M.Kom.**

Abstrak

Saat ini perkembangan pada dunia game sudah sangat pesat. Perkembangan ini meliputi genre dan aturan main yang baru.Salah satu genre game yang cukup banyak diminati adalah genre 2D Platformer.

2D Platformer adalah genre game dimana pemain harus menggerakkan karakternya menyusuri map untuk menyelesaikan misi tertentu. Map pada game 2D Platformer biasanya tersusun dari daratan (platform), tangga, dan beberapa rintangan. Dalam perkembangan game 2D Platformer, kebanyakan game memiliki jenis map yang bersifat statis. Statis dalam hal ini artinya lingkungan atau area permainan akan tetap sama setiap dimainkan kembali. Dengan jenis map yang statis ini, tentunya bisa membuat beberapa orang merasa cepat bosan.

Untuk mengatasi masalah tersebut, game Mine Meander mencoba menghadirkan sesuatu yang berbeda dengan fitur Dynamic World Generator. Dalam game ini, pemain bisa menjumpai bentuk world map yang berbeda dan juga menjumpai platform atau rintangan yang letaknya selalu berubah setiap kali dimainkan kembali. Game ini menggunakan Algoritma Recursive Backtracking dalam

proses pembentukan world map-nya dan menggunakan metode tertentu dalam peletakan platform atau tangga sehingga game ini selalu mempunyai jalan keluar untuk dapat menyelesaikan level.

Hasil pengujian game ini menunjukkan bahwa world map yang dibentuk pasti memiliki jalan untuk dapat diselesaikan. Selain itu penerapan tingkat kesulitan pada setiap level, hal tersebut akan membuat game ini menjadi lebih dinamis dan menarik.

Kata kunci: 2D Platformer Game, Platform,* Algoritma *Recursive Backtracking,* dan *Dynamic World Generator.

DYNAMIC WORLD GENERATOR USING RECURSIVE BACKTRACKING ALGORITHM FOR 2D PLATFORMER GAME "MINE MEANDER"

Name	**:**	**Faishal Azka Jellyanto**
NRP	**:**	**5112100061**
Department	**:**	**Department of Informatics Faculty of Information Technology ITS**
Supervisor 1	**:**	**Imam Kuswardayan, S.Kom., M.T.**
Supervisor 2	**:**	**Dr.Eng. Nanik Suciati S.Kom., M.Kom.**

Abstract

Nowadays the development in game world have improved a lot. This improvement include genre and new type of gameplay. One of the genre that most people like is 2D Platformer.

2D Platformer is a genre where player have to move their character exploring the map to clear a certain mission. Usually, map in 2D Platformer game consist of platform, ladder, and some obstacles. In the development process of 2D Platformer game, most of the games have static map. Static map means that the environment or the game area will still be the same when being played over and over. With this kind of map, some of the players might become bored eventually.

To overcome this problem, game called Mine Meander try to presents something new by Dynamic World Generator feature. In this game, player will see different world map and see the position of platform or obstacle keep changing everytime the player restart. This game use Recursive Backtracking Algorithm to generate the world map and use certain method in placing platform or ladder so that the game will always has a route to clear the level.

The testing result of this game showed that the generated map will always have a route so that the level can be cleared. Furthermore with the implementation of difficulty in every levels, it will make this game more dynamic and more interesting.

Keywords: 2D Platformer Game, Platform, Recursive Backtracking Algorithm,and Dynamic World Generator.

KATA PENGANTAR

بِسۡمِ ٱللَّهِ ٱلرَّحۡمَٰنِ ٱلرَّحِيمِ

Alhamdulillahirabbil'alamin, segala puji bagi Allah SWT, yang telah melimpahkan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul **"RANCANG BANGUN PEMBANGKIT WORLD DINAMIS MENGGUNAKAN ALGORITMA RECURSIVE BACKTRACKING PADA GAME 2D PLATFORMER MINE MEANDER".** Dengan pengerjaan Tugas Akhir ini, penulis bisa belajar lebih banyak untuk memperdalam dan meningkatkan apa yang telah didapatkan penulis selama menempuh perkuliahan di Teknik Informatika ITS. Dengan Tugas Akhir ini penulis juga dapat menghasilkan suatu implementasi dari apa yang telah penulis pelajari.

Selesainya Tugas Akhir ini tidak lepas dari bantuan dan dukungan beberapa pihak. Sehingga pada kesempatan ini penulis mengucapkan syukur dan terima kasih kepada:

1. Allah SWT dan Nabi Muhammad SAW.
2. Keluarga penulis yang telah memberikan dukungan moral dan material serta do'a yang tak terhingga untuk penulis. Serta selalu memberikan semangat dan motivasi pada penulis dalam mengerjakan Tugas Akhir ini.
3. Bapak Imam Kuswardayan, S.Kom., M.T. selaku pembimbing I yang telah membantu, membimbing, dan memotivasi penulis dalam menyelesaikan Tugas Akhir ini dengan sabar.
4. Ibu Dr. Eng. Nanik Suciati, S.Kom., M.Kom**.** selaku pembimbing II yang juga telah membantu, membimbing, dan memotivasi kepada penulis dalam mengerjakan Tugas Akhir ini.
5. Bapak Dr. Darlis Herumurti, S.Kom., M.Kom. selaku Kepala Jurusan Teknik Informatika ITS.

6. Bapak Radityo Anggoro, S.Kom., M.Sc. selaku koordinator Tugas Akhir, dan segenap dosen Teknik Informatika yang telah memberikan ilmunya.
7. Teman-teman angkatan 2012 yang yang telah membantu, berbagi ilmu, menjaga kebersamaan, dan memberi motivasi kepada penulis.
8. Serta semua pihak yang yang telah turut membantu penulis dalam menyelesaikan Tugas Akhir ini.

Penulis menyadari bahwa Tugas Akhir ini masih memiliki banyak kekurangan. Sehingga dengan kerendahan hati, penulis mengharapkan kritik dan saran dari pembaca untuk perbaikan ke depannya.

Surabaya, Juli 2016

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR KODE SUMBER

BAB I
PENDAHULUAN

Bagian ini akan dijelaskan hal-hal yang menjadi latar belakang, permasalahan yang dihadapi, batasan masalah, tujuan dan manfaat, metodologi dan sistematika penulisan yang digunakan dalam pembuatan tugas akhir ini.

1.1 Latar Belakang

Saat ini perkembangan dunia *game* semakin meningkat secara pesat, mulai dari *genre* dan berbagai aturan main (*gameplay*) yang baru. Salah satu genre *game* yang cukup banyak diminati adalah genre *2D Platformer*. Pada genre *game* ini, pemain harus menggerakkan karakternya menyusuri map yang tersusun dari daratan (*platform*), tangga, dan beberapa rintangan. Contoh *game platformer* yang banyak dikenal antara lain Terraria, Donkey Kong, Space Panic, dan Mario Bros. Namun kebanyakan *game platformer* saat ini cenderung memiliki bentuk *world map* yang statis dimana posisi dan letak *platform* akan selalu tetap sama pada satu level. Dengan bentuk map yang statis seperti ini tentunya akan membuat beberapa pemain cepat merasa bosan.

Untuk dapat mengatasi hal tersebut, diperlukan bentuk *world map* yang dinamis dimana *world map* dapat terus berubah ketika pemain mengulang kembali suatu level. Salah satu Algortima yang bisa digunakan untuk membuat *world map* dinamis adalah Algortima *Recursive Backtracking*. Algoritma Recursive Backtracing pada tugas akhir ini digunakan untuk menentukan *layout* utama dari *world map*. Dari *layout* utama yang sudah dibentuk secara random, barulah diletakkan *platform*, tangga, dan rintangan. Saat pemain mengulangi kembali level tersebut, program akan me-*generate* ulang *layout* utama.

Usulan tugas akhir ini adalah membuat suatu *game 2D Platformer* dengan fitur *world map* yang dinamis menggunakan Algoritma *Recursive Backtracking*. Dengan adanya fitur yang

baru ini, diharapkan pemain akan menjadi lebih tertarik untuk bermain *game 2D Platformer* dan tidak cepat merasa bosan.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam Tugas Akhir ini adalah sebagai berikut:

1. Bagaimana merancang *world map* yang dinamis dalam bentuk *platformer* yang memungkinkan untuk diselesaikan dengan menggunakan Algoritma *Recursive Backtracking* ?
2. Bagaimana merancang aturan main, skenario, dan tingkat kesulitan pada *game* Mine Meander ?
3. Bagaimana mengimplementasikan rancangan dengan *framework* LibGDX ?

1.3 Batasan Masalah

Batasan dalam Tugas Akhir ini, yaitu:

1. Algoritma yang digunakan untuk pembuatan *layout* utama dari *world map* adalah Algoritma *Recursive Backtracking*
2. Batas maksimal *room* dari *world map* adalah 64 x 64.

1.4 Tujuan

Tujuan dari pembuatan tugas akhir ini adalah menerapkan fitur atau inovasi baru yang berupa rancangan *world map* yang dinamis menggunakan Algoritma *Recursive Backtracking* dalam *game 2D Platformer*.

1.5 Manfaat

Manfaat dari hasil pembuatan tugas akhir ini adalah adanya inovasi baru berupa bentuk *world map* yang dinamis pada *game 2D platformer* yang bisa membuat pemain merasa lebih terhibur dan tidak cepat bosan saat bermain.

1.6 Metodologi

Tahapan-tahapan yang dilakukan dalam pengerjaan Tugas Akhir ini adalah sebagai berikut:

1. Penyusunan proposal Tugas Akhir.
   Tahap pertama dalam proses pengerjaan tugas akhir ini adalah menyusun proposal tugas akhir. Pada proposal tugas akhir ini diajukan *game 2D Platformer* yang mampu menghasilkan word map yang dinamis.

2. Studi literatur
   Pada tahap ini, akan dicari studi literatur yang relevan untuk dijadikan referensi dalam pengerjaan tugas akhir. Studi literatur ini didapatkan dari buku, internet, dan materi-materi kuliah yang berhubungan dengan metode yang akan digunakan.

3. Analisis dan perancangan perangkat lunak
   Tahap ini meliputi perancangan sistem berdasarkan studi literatur. Tahap ini mendefinisikan alur metode dari implementasi. Langkah-langkah yang dikerjakan juga didefinisikan pada tahap ini.

4. Implementasi perangkat lunak
   Implementasi perangkat lunak merupakan tahap membangun rancangan aplikasi yang telah dibuat. Pada tahap ini akan direalisasikan mengenai rancangan apa saja yang telah didefiniskan pada tahap sebelumnya. Pembangunan aplikasi akan dilakukan menggunakan bahasa pemrograman Java dengan bantuan IDE Android Studio dan Map Editor Tiled.

5. Pengujian dan evaluasi
   Pengujian akan dilakukan oleh beberapa orang pengguna. Pengguna akan diminta mengisi lembar kuesioner untuk

memberikan umpan baliknya. Kemudian akan dilakukan evaluasi untuk memeriksa apakah *world map* yang dihasilkan benar-benar dinamis dan apakah pengguna mampu menyelesaikan setidaknya satu level.

6. Penyusunan buku Tugas Akhir.
   Pada tahap ini dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam tugas akhir ini. Pada tahap ini juga disertakan hasil dari implementasi metode dan algoritma yang telah dibuat.

1.7 Sistematika Penulisan Laporan Tugas Akhir

Buku tugas akhir ini disusun dengan sistematika penulisan sebagai berikut:

Bab I Pendahuluan
Bab ini berisi mengenai latar belakang, tujuan, dan manfaat dari pembuatan tugas akhir. Selain itu, permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penulisan juga merupakan bagian dari bab ini.

Bab II Tinjauan Pustaka
Bab ini berisi penjelasan secara detail mengenai dasar-dasar penunjang dan teori-teori yang digunakan untuk mendukung pembuatan tugas akhir ini.

Bab III Analisis dan Perancangan Perangkat Lunak
Bab ini membahas tahap analisis permasalahan dan perancangan dari sistem yang akan dibangun. Analisis permasalahan membahas permasalahan yang yang diangkat dalam pengerjaan tugas akhir.

Bab IV Implementasi

Bab ini membahas implementasi dari desain yang telah dibuat pada bab sebelumnya. Bab ini berisi proses implementasi dari setiap kelas pada semua modul.

Bab V Uji Coba Dan Evaluasi

Bab ini menjelaskan kemampuan perangkat lunak dengan melakukan pengujian kebenaran dan pengujian kinerja dari sistem yang telah dibuat.

Bab VI Kesimpulan Dan Saran

Bab ini merupakan bab terakhir yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan dan saran untuk pengembangan aplikasi ke depannya.

BAB II
TINJAUAN PUSTAKA

Bab ini berisi penjelasan teori-teori yang berkaitan dengan metode yang diajukan pada pengimplementasian perangkat lunak. Penjelasan ini bertujuan untuk memberikan gambaran secara umum terhadap sistem yang dibuat dan berguna sebagai penunjang dalam pengembangan perangkat lunak.

2.1 *2D Platformer Game*

2D Platformer Game adalah salah satu *genre video game* yang memiliki *gameplay* meliputi perjalanan antar *platform* [6]. *Platform* dalam konteks ini adalah objek berupa daratan yang berfungsi sebagai tempat berpijak karakter dalam permainan [7]. Elemen-elemen lain yang seringkali dimasukkan pada genre ini antara lain berlari, melompat, menaiki atau menuruni tangga, dan menghindari rintangan. Contoh dari *game 2D platformer* yang terkenal adalah : Donkey Kong, Space Panic, dan Mario Bros.

2.2 LibGDX

LibGDX adalah suatu *framework* untuk pengembangan *game* yang sebagian besar ditulis dalam bahasa pemrograman Java. LibGDX bersifat gratis dan *cross-platform* dimana *game* yang dihasilkan dapat dijalankan di beberapa *platform* seperti Windows, Linux, Mac OS X, Android, iOS, dan HTML [2].

Proses pengembangan dari LibGDX secara umum menggunakan bahasa pemrograman Java dan pengguna disarankan untuk menggunakan IDE Eclipse, meskipun pada penerapannya dimungkinkan untuk menggunakan IDE lain seperti Netbeans atau Android Studio. Hampir sama seperti Unity, meskipun LibGDX tersedia untuk berbagai macam *platform* namun kelebihan LibGDX ini adalah proses *development* dan

debugging dapat dilakukan hanya dengan menggunakan komputer *desktop* tanpa memerlukan perangkat lain. Setelah *debugging* siap maka pengguna dapat melakukan proses kompilasi dan uji coba sesuai dengan *platform* yang diinginkan. Hal ini sangat memudahkan karena proses kompilasi dan proses berjalannya program menjadi lebih cepat.

Untuk menangani masalah *math* dan *physics* , LibGDX menyediakan ekstensi Box2D. Pada LibGDX juga tersedia ekstensi lain seperti Controllers, AI, Tools (2D/3D editor), dan lain-lain.

Beberapa fitur lainnya adalah LibGDX mampu menangani berbagai jenis masukan seperti *keyboard*, *mouse*, *touch screen*, bahkan lengkap dengan *gesture detector*. Untuk masalah penyimpanan progres *game* , LibGDX sudah menyediakan semacam kelas yang berfungsi sebagai *file manager*. *Output file* yang disediakan dapat berupa *file* json maupun xml.

2.3 Android Studio

Android Studio merupakan sebuah *Integrated Development Environment* (IDE) khusus untuk membangun aplikasi yang berjalan pada *platform* android. Android studio ini berbasis pada IntelliJ IDEA [5], sebuah IDE untuk bahasa pemrograman Java. Bahasa pemrograman utama yang digunakan adalah Java, sedangkan untuk membuat tampilan atau *layout*, digunakan bahasa XML. Android studio juga terintegrasi dengan Android *Software Development Kit* (SDK) untuk *deploy* ke perangkat android.

Sebagai pengembangan dari Eclipse, Android Studio mempunyai banyak fitur-fitur baru dibandingkan dengan Eclipse. Berbeda dengan Eclipse yang menggunakan Ant, Android Studio menggunakan Gradle sebagai lingkungan *build*-nya. Fitur-fitur lainnya adalah sebagai berikut :

- Menggunakan *Gradle-based build system* yang fleksibel

- Bisa menghasilkan multiple APK (Android Application Package)
- *Template* mendukung Google Services dan berbagai macam tipe perangkat
- Mendukung Google Cloud Platform, sehingga mudah untuk integrasi dengan Google Cloud Messagin dan App Engine.

2.4 Tiled

Tiled adalah suatu map editor yang bertujuan khusus dalam pembuatan map berbasis tile [3]. Aplikasi ini menyediakan *tool* gratis untuk membuat *layout* map dalam *game*. Tiled mampu mengolah objek-objek seperti *background map*, *sprites*, *collision area*, hingga posisi *spawn* musuh. Semua data ini kemudian disimpan ke dalam format .tmx .

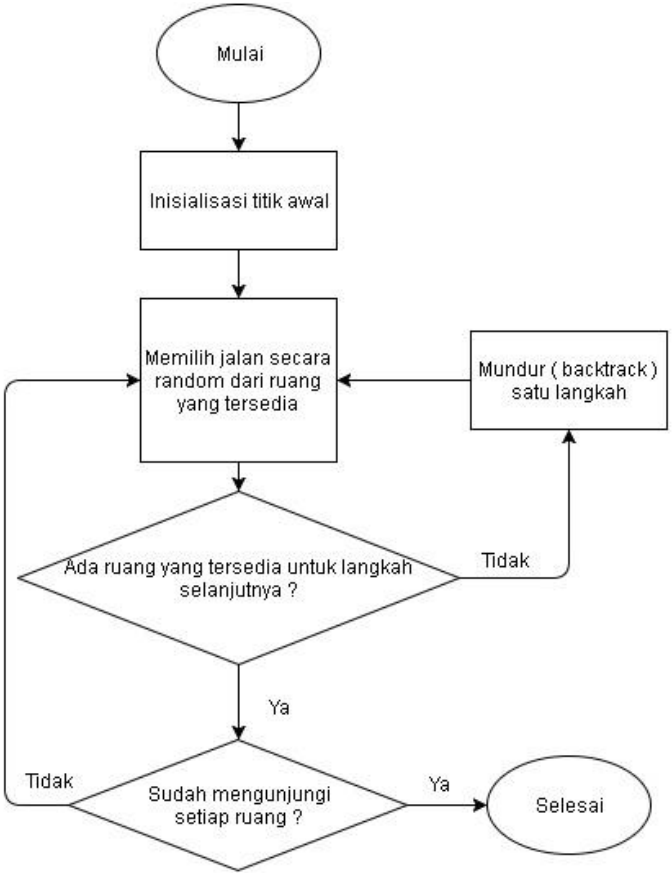
Selain berfungsi sebagai map editor dalam pengolahan *tiled-based* map, Tiled juga bisa digunakan sebagai level editor. Dengan Tiled, pengguna dapat menentukan ukuran-ukuran dari tiap *tile* pada gambar sehingga pengguna dapat membuat map tanpa dibatasi oleh ukuran tetap dari gambar.

2.5 Algoritma *Recursive Backtracking*

Algoritma *Backtracking* pertama kali diperkenalkan oleh D.H. Lehmer pada tahun 1950. Algoritma ini cukup praktis untuk digunakan dalam beberapa penyelesaian masalah dan juga untuk memberikan kecerdasan buatan dalam *game*. Beberapa *game* seperti sudoku, maze (labirin), atau catur juga bisa diimplementasikan dengan Algoritma *Backtracking* [1].

Pada proses pembentukan Maze, Algoritma ini awalnya memilih jalan secara random dari ruang-ruang yang tersedia. Apabila pada langkah tertentu proses pemilihan jalan tidak bisa dilanjutkan dan masih terdapat ruang yang belum dikunjungi, maka akan mundur (*backtrack*) ke langkah sebelumnya dan

akan memilih jalan lain yang belum pernah dikunjungi [4]. Proses akan berhenti apabila telah mengunjungi setiap ruang. Alur Algoritma dapat dilihat pada Gambar 2.1 .



Gambar 2.1 Alur Algoritma Recursive Backtracking

BAB III
ANALISIS DAN PERANCANGAN PERANGKAT LUNAK

Bab ini membahas tahap analisis permasalahan dan perancangan dari sistem yang akan dibangun. Analisis permasalahan membahas permasalahan yang yang diangkat dalam pengerjaan tugas akhir. Dalam tugas akhir ini akan dibuat sebuah *game* dengan jenis *2D Platformer*.

3.1 Analisis

Tahap analisis dibagi menjadi beberapa bagian antara lain analisis permasalahan, analisis kebutuhan, deskripsi umum aplikasi, dan kasus penggunaan.

3.1.1 Analisis Permasalahan

Kemajuan teknologi informasi menjadikan perkembangan dunia *game* menjadi semakin bermacam-macam jenisnya baik dalam hal *genre*, *gameplay*, serta kualitas grafis yang diimplementasikan. Di lingkungan pasar *game*, terdapat banyak sekali *game* yang memiliki genre *2D Platformer*. *Game 2D Platformer* ini tentunya memiliki fitur dan keunikan masing-masing.

Game ini dibangun dengan tujuan memberikan inovasi dan hiburan kepada pemain *game* khususnya pecinta *game 2D Platformer*. *Game* ini juga diharapkan dapat meningkatkan ketertarikan seseorang pada *game* genre *2D Platformer*.

Penulis menggunakan *framework* LibGDX dan IDE Android Studio dengan bahasa pemrograman Java untuk memfasilitasi pengembangan aplikasi ini. Penulis juga menggunakan perangkat lunak Tiled Map Editor untuk mengatur *asset 2D* yang akan dipakai.

3.1.2 Analisis Kebutuhan

Kebutuhan utama dalam aplikasi ini adalah pemain dapat memainkan permainan *2D Platformer* yang bentuk *world map*-nya dinamis. Kebutuhan fungsional dapat dilihat pada Tabel 3. 1.

Tabel 3. 1 Kebutuhan fungsional aplikasi

Kode Kebutuhan	Kebutuhan Fungsional	Deskripsi
F-0001	Memilih level	Pemain dapat memilih level yang diinginkan
F-0002	Melihat instruksi permainan	Pemain dapat melihat bagaimana aturan main dan cara memainkan permainan.
F-0003	Memainkan Permainan	Pemain masuk ke dalam level yang dipilih dan memainkan permainan.
F-0004	*Generate* map secara dinamis	Aplikasi dapat melakukan *generate* map secara dinamis pada setiap level.

3.1.3 Deskripsi Umum Aplikasi

Aplikasi yang akan dibuat dalam tugas akhir ini adalah sebuah permainan berjenis *2D Platformer*. Permainan ini hanya bisa dimainkan oleh satu orang. Fokus utama dari permainan ini terletak pada bentuk *world map* yang dinamis. Pada setiap level pemain akan menjumpai bentuk *world map* yang berubah-ubah. *World map* yang dinamis ini dibentuk dengan menggunakan Algoritma *Recursive Backtracking*. Setiap level dari permainan ini juga memiliki tingkat kesulitan yang berbeda-beda. Semakin tinggi level yang dipilih maka tingkat kesulitan yang dihadapi pemain untuk menyelesaikan level tersebut menjadi semakin sulit. Pemain dinyatakan menang apabila mampu menemukan portal atau pintu keluar.

3.1.4 Karakteristik Pengguna

Berdasarkan deskripsi umum di atas, maka dapat diketahui karakteristik pengguna dari pemain. Karakteristik pengguna tercantum dalam Tabel 3.2 .

Tabel 3.2 Karakteristik Pengguna

Nama Aktor	Tugas	Hak Akses Aplikasi	Kemampuan yang harus dimiliki
Pemain	Pihak yang menggunakan	Menjalankan segala fungsionalitas dalam permainan.	Tidak ada

3.2 Perancangan Aplikasi

Tahap perancangan dibagi menjadi beberapa bagian antara lain perancangan antarmuka, perancangan aturan main, perancangan *world map*, serta perancangan skenario dan tingkat kesulitan.

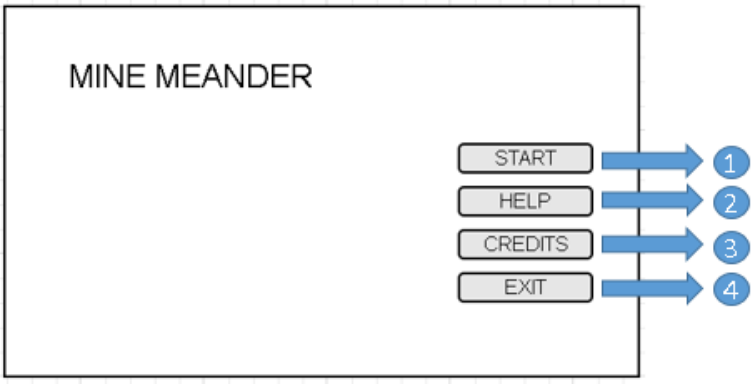
3.2.1 Perancangan Tampilan Antarmuka

Sub bab ini membahas bagaimana rancangan antarmuka pengguna yang akan digunakan pada tugas akhir. Dalam aplikasi ini terdapat beberapa tampilan, yaitu tampilan halaman awal, tampilan halaman instruksi, tampilan halaman pilihan level, dan tampilan area permainan.

3.2.1.1 Tampilan Halaman Awal

Tampilan halaman awal adalah tampilan yang pertama kali muncul ketika aplikasi dijalankan. Pada tampilan awal terdapat 3

tombol, yaitu tombol Start, tombol Help, tombol Credits dan tombol Exit.



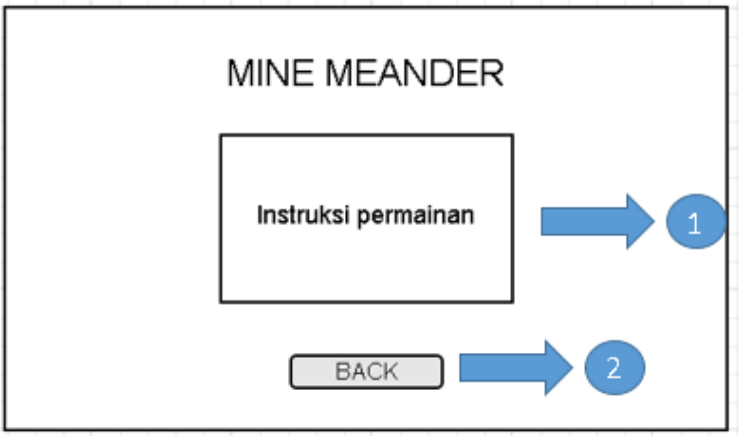
Gambar 3. 1 Rancangan antarmuka halaman awal

Seperti yang terlihat pada Gambar 3.1 :

1. Tombol **Start**, berfungsi untuk masuk ke halaman pilihan level.
2. Tombol **Help**, berfungsi untuk melihat instruksi permainan.
3. Tombol **Credits,** berfungsi untuk melihat sumber dari *asset* yang dipakai.
4. Tombol **Exit**, berfungsi untuk keluar dari aplikasi.

3.2.1.2 Tampilan Halaman Instruksi

Tampilan halaman instruksi merupakan halaman yang akan menampilkan penjelasan cara bermain dan aturan main dari permainan ini.



Gambar 3.2 Rancangan antarmuka halaman instruksi

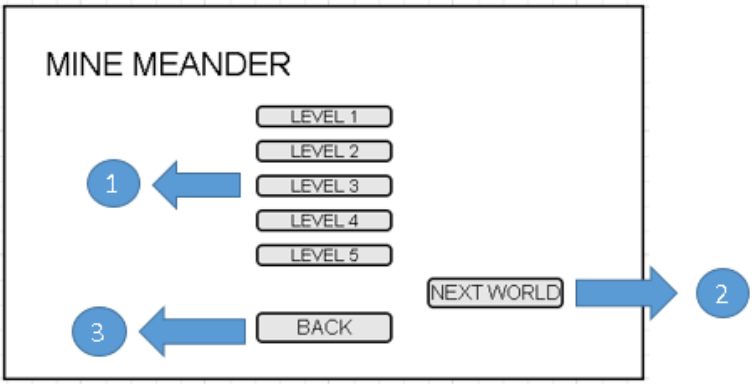
Seperti yang terlihat pada Gambar 3.2 :

1. Konten instruksi permainan berisi informasi mengenai cara memainkan permainan
2. Tombol **Back**, berfungsi untuk kembali ke antarmuka halaman utama.

3.2.1.3 Tampilan Halaman Pilihan Level

Tampilan halaman pilihan level adalah halaman yang akan menampilkan pilihan level. Pemain bisa memilih level dan world yang ingin dimainkan. Rancangan antarmuka halaman pilihan level dapat dilihat pada Gambar 3.3 , keterangannya adalah :

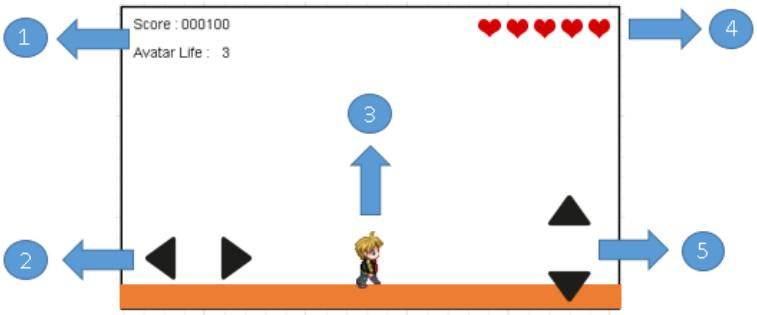
1. Tombol **Level** , berfungsi untuk memilih level dan berlanjut ke halaman permainan.
2. Tombol **Next World**, berfungsi untuk berganti ke sub halaman world selanjutnya.
3. Tombol **Back**, berfungsi untuk kembali ke antarmuka halaman utama.



Gambar 3.3 Rancangan antarmuka halaman pilihan level

3.2.1.4 Tampilan Area Permainan

Tampilan area permainan adalah tampilan ketika pemain telah memilih level dan masuk ke dalam area permainan. Rancangan antarmuka area permainan bisa dilihat pada Gambar 3.4



Gambar 3. 4 Rancangan antarmuka area permainan

Seperti yang terlihat pada Gambar 3.4 :

1. Info Karakter, berfungsi untuk menampilkan score dan jumlah nyawa yang dimiliki.

2. *Controller* 1, berfungsi untuk mengatur pergerakan karakter ke kanan dan ke kiri.
3. *Heart*, berfungsi untuk menampilkan jumlah *Heart* yang dimiliki karakter.
4. Karakter, merepresentasikan bentuk karakter yang dikontrol oleh pemain.
5. *Controller* 2, berfungsi untuk melompat, memanjat, dan menuruni tangga.

3.2.2 Perancangan Aturan Main

Aturan main secara umum dari permainan ini adalah pemain bisa menggerakkan karakternya secara horizontal, melompat, dan bisa menaiki atau menuruni tangga. Selama permainan berlangsung, pemain akan menemukan kristal-kristal di dalam map. Kristal-kristal tersebut berfungsi untuk menambah perolehan skor dari pemain.

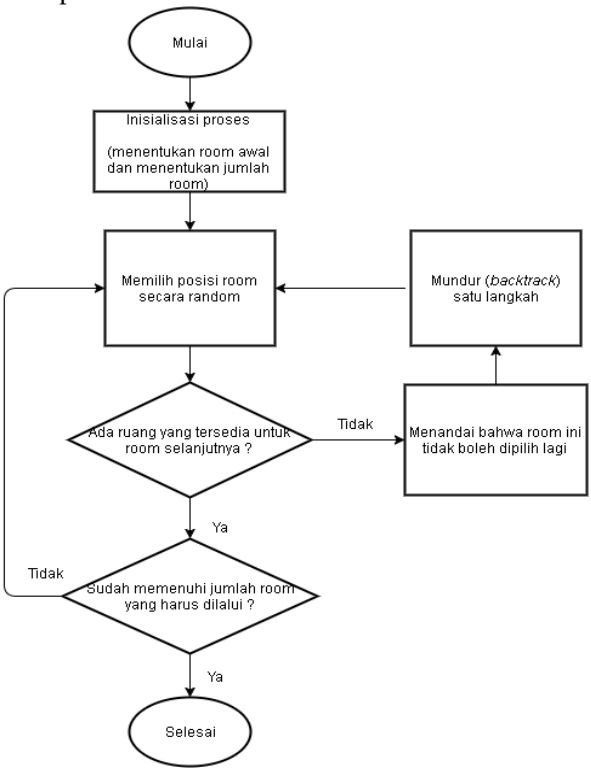
Di dalam map juga terdapat beberapa rintangan dimana apabila karakter dari pemain bersentuhan dengan rintangan tersebut, maka *Heart* karakter akan berkurang. Rintangan yang ada di dalam *world map* antara lain *spike* (duri), *dangerous flower*, *spider*, dan *zombie*. *Spike* adalah jenis rintangan yang diam atau tidak bergerak. Apabila karakter melompat dan jatuh di atasnya, maka *Heart* karakter akan berkurang. *Dangerous flower* adalah jenis rintangan yang juga diam dan tidak bergerak. Namun apabila karakter bersentuhan baik dari samping atau atas/bawah, maka *Heart* karakter akan berkurang. *Spider* adalah jenis rintangan yang bisa bergerak apabila karakter mendekat ke tempat *spider* berada. *Heart* karakter juga akan berkurang apabila bersentuhan dengannya. *Zombie* juga merupakan jenis rintangan yang bisa bergerak secara horizontal dan bisa menaiki atau menuruni tangga. Apabila bersentuhan dengan *zombie*, *Heart* karakter akan berkurang.

Untuk dapat memenangkan level dalam permainan ini, pemain harus bisa menemukan dan masuk ke dalam portal atau

pintu keluar yang berada pada *world map*. Apabila karakter kehabisan *life* (nyawa) sebelum bisa masuk ke dalam pintu keluar, maka permainan akan berakhir (Game Over).

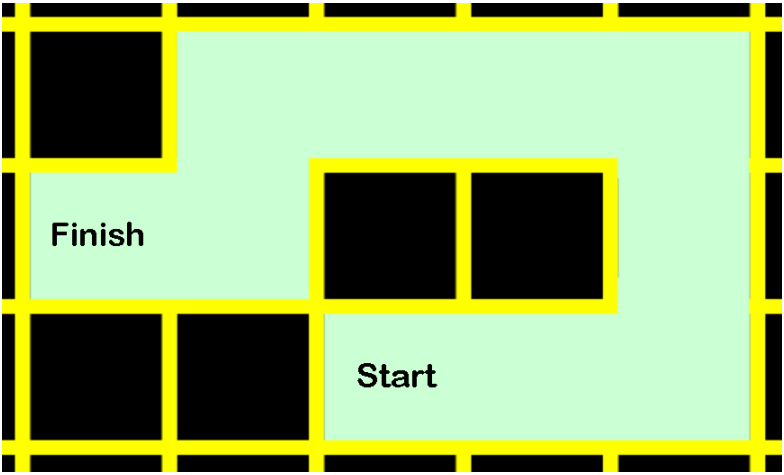
3.2.3 Perancangan *World Map*

World map merupakan area permainan dimana pemain menggerakkan karakternya. Pada permainan ini tiap satu level permainan memiliki satu *world map* yang dinamis atau bisa berubah-ubah. Pembuatan *world map* ini akan menggunakan Algoritma *Recursive Backtracking*. Diagram alur dari Algoritma bisa dilihat pada Gambar 3.5 .



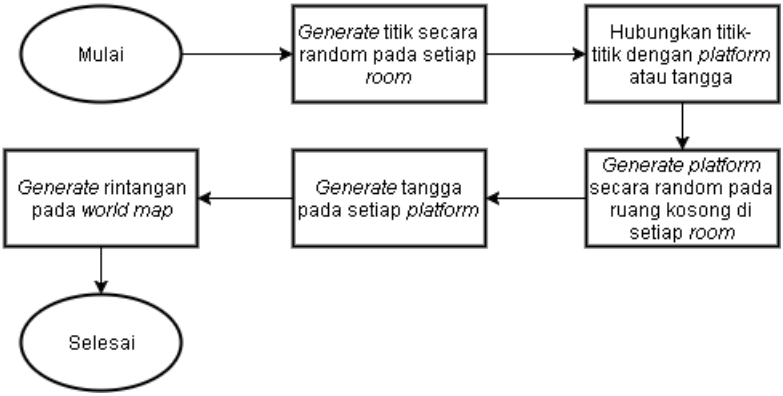
Gambar 3.5 Diagram alur perancangan world map

Setiap *world map* yang dibentuk pasti memiliki beberapa jumlah *room*. *Room* dalam hal ini adalah suatu ruang kecil yang berfungsi sebagai tempat menampung objek seperti karakter, *platform*, rintangan, serta objek-objek lain. Pembuatan jalur utama dari *world map* ini bergantung pada jumlah *room* yang harus dilalui. Setelah *room* awal selesai dibuat, Algoritma akan memilih secara random *room* mana yang akan dipilih pada langkah selanjutnya. Apabila menemui jalan buntu dan belum mencapai jumlah minimal *room* yang harus ditempuh, maka room saat ini ditandai supaya pada langkah selanjutnya tidak akan dipilih lagi. Kemudian dilakukan *backtrack* satu langkah sehingga akan kembali ke posisi *room* yang sebelumnya.. Contoh ilustrasi dari proses pembentukan *layout* utama bisa dilihat pada Gambar 3.6.



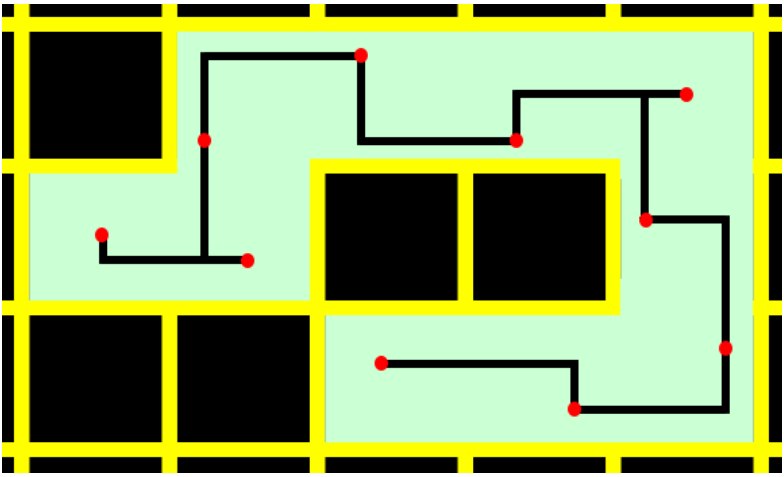
Gambar 3.6 Contoh ilustrasi pembentukan layout utama

Setelah *layout* utama dari *world map* telah terbentuk, langkah selanjutnya adalah mengisi tiap *room* dengan objek-objek. Diagram alur dari proses ini bisa dilihat pada Gambar 3.7.

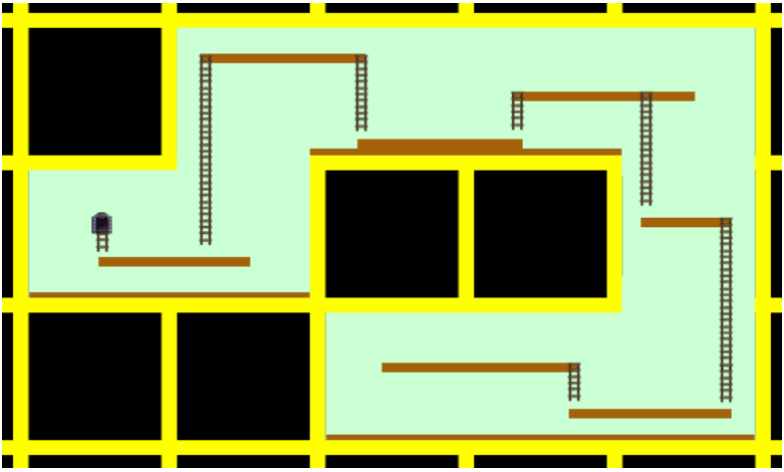


Gambar 3. 7 Diagram alur proses pembentukan objek-objek di dalam world map

Dari diagram alur di atas, proses pertama yang dilakukan ada me-*generate* titik-titik yang posisinya diatur secara random pada setiap *room*. Titik-titik ini berfungsi sebagai acuan jalan penghubung antar *room* dimana jalan ini bisa berupa *ladder* (tangga) atau *platform*. *Ladder* adalah salah satu objek pada *world map* dimana karakter bisa memanjat serta menuruni objek tersebut. *Platform* akan di-*generate* sepanjang selisih jarak horizontal dari *room* awal ke *room* selanjutnya. Sedangkan tangga akan di-*generate* sepanjang selisih jarak vertikal antara *room* awal dengan *room* selanjutnya. Khusus untuk titik pada *room* terakhir dari *world map*, akan diletakkan sebuah portal atau pintu keluar. Portal atau pintu keluar ini berfungsi sebagai syarat untuk dapat menyelesaikan level dimana pemain harus menggerakkan karakternya ke portal tersebut. Setelah langkah ini selesai, maka bisa dipastikan bahwa level atau *world map* ini memungkinkan untuk diselesaikan oleh pemain. Ilustrasi dari hasil proses ini bisa dilihat pada Gambar 3.8 dan Gambar 3.9.



Gambar 3.8 Contoh ilustrasi world map yang sudah diberi titik penghubung antar room

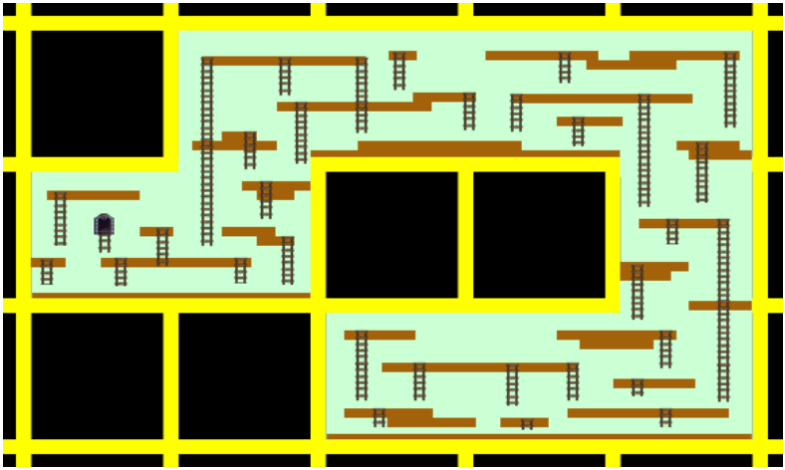


Gambar 3.9 Contoh ilustrasi world map yang sudah diberi platform dan tangga sebagai jalan penghubung antar room

Setelah jalur utama terbentuk, pada setiap *room* yang mempunyai bagian yang kosong akan diisi dengan *platform* dan

tangga. Dalam penentuan posisi atau koordinat dari *platform* ini, akan bergantung pada proses randomisasi. Artinya pada setiap titik x,y dalam satu *room*, ada kemungkinan bahwa platform akan di-*generate* atau tidak. Apabila di-*generate*, kemudian secara random ditentukan panjang dari *platform* tersebut. Platform-platform yang di-generate ini juga harus dipastikan agar peletakannya tidak berdekatan dengan jalur utama. Hal itu bertujuan supaya jalur utama tidak tertutup oleh *platform-platform* lain.

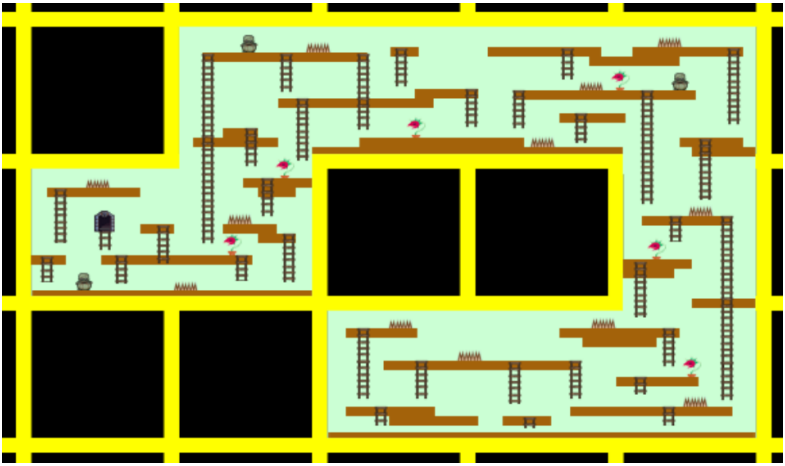
Langkah selanjutnya setelah selesai me-*generate platform* adalah meletakkan *ladder* atau tangga. Dalam satu *room* pada *world map*, pasti terdapat beberapa *platform*. Dari tiap *platform* tersebut kemudian diletakkan satu buah tangga. Panjang tangga ini akan menyesuaikan pada seberapa jauh selisih antara *platform* satu dengan *platform* yang ada di bawahnya. Hasil dari proses ini diilustrasikan pada Gambar 3.10.



Gambar 3.10 Contoh world map yang telah diisi dengan platform dan tangga

Dari platform yang telah dibentuk, langkah berikutnya adalah meletakkan rintangan-rintangan pada world map. Sebelum

meletakkan satu rintangan, pertama ditentukan dulu secara random platform mana yang akan dipilih sebagai tempat peletakkan rintangan. Setelah salah satu platform dipilih, kemudian rintangan diletakkan pada platform tersebut. Langkah ini diulangi kembali sampai mencapai jumlah rintangan yang telah ditentukan. Hasil dari proses ini diilustrasikan pada Gambar 3.11.



Gambar 3.11 Contoh world map yang telah diisi dengan rintangan

3.2.4 Perancangan Skenario dan Tingkat Kesulitan

Seperti yang sudah dijelaskan pada sub bab sebelumnya, pemain harus menggerakkan karakternya menyusuri map dan menemukan portal atau pintu keluar untuk dapat menyelesaikan satu level. Selama permainan berlangsung, karakter mempunyai 5 *Heart* dan 3 nyawa. Apabila karakter bersentuhan dengan salah satu rintangan, maka Heart akan berkurang 1. Lalu apabila karakter kehabisan *Heart*, maka nyawa akan berkurang 1 dan karakter akan kembali ke titik awal (*start point*). Setelah nyawa berkurang 1, *Heart* dari karakter akan kembali menjadi 3. Namun apabila karakter kehabisan nyawa, maka permainan akan berakhir (Game Over).

Permainan ini mempunyai 15 level dimana tiap level akan memiliki tingkat kesulitan yang berbeda. Faktor-faktor penentu tingkat kesulitan antara lain adalah jumlah *room* yang harus dilewati, jumlah *Dangerous Flower*, jumlah *Spider*, dan jumlah *Zombie*. Jumlah minimum & maksimum serta nilai bobot dari faktor-faktor ini bisa dilihat pada Tabel 3.3 .

Tabel 3.3 Faktor penentu tingkat kesulitan

Faktor Penentu	Kode Faktor	Nilai Minimum	Nilai Maksimum	Nilai Bobot
Jumlah *Room*	JR	4	12	0.4
Jumlah *Dangerous Flower*	JF	8	24	0.1
Jumlah *Spider*	JS	0	24	0.2
Jumlah *Zombie*	JZ	0	12	0.3

Nilai bobot dari masing-masing faktor penentu tingkat kesulitan tersebut kemudian dimasukkan ke tiap level. Hasil dari penentuan faktor tingkat kesulitan pada semua level dapat dilihat pada Tabel 3.4. Untuk menghitung total bobot dalam satu level, digunakan rumus sebagai berikut :

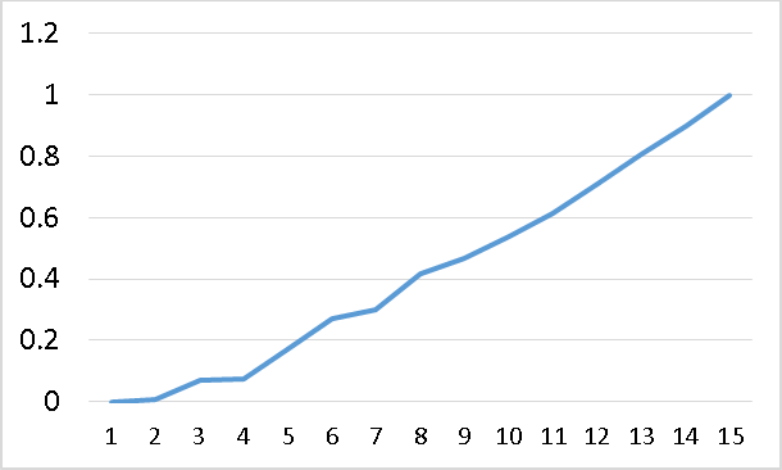
$$\text{Total bobot} = \frac{Nilai\ JR\ pada\ level - Nilai\ Minimum\ JR}{Nilai\ Maksimum\ JR - Nilai\ Minimum\ JR} \times Nilai\ Bobot\ JR +$$
$$\frac{Nilai\ JF\ pada\ level - Nilai\ Minimum\ JF}{Nilai\ Maksimum\ JF - Nilai\ Minimum\ JF} \times Nilai\ Bobot\ JF +$$
$$\frac{Nilai\ JS\ pada\ level - Nilai\ Minimum\ JS}{Nilai\ Maksimum\ JS - Nilai\ Minimum\ JS} \times Nilai\ Bobot\ JS +$$
$$\frac{Nilai\ JZ\ pada\ level - Nilai\ Minimum\ JZ}{Nilai\ Maksimum\ JZ - Nilai\ Minimum\ JZ} \times Nilai\ Bobot\ JZ$$

Tabel 3.4 Tingkat kesulitan pada tiap level

Level	JR	JF	JS	JZ	Total Bobot
1	4	8	0	0	0
2	4	4	4	0	0.008333333

3	5	6	4	0	0.070833333
4	5	4	6	0	0.075
5	6	6	4	2	0.170833333
6	7	6	4	4	0.270833333
7	7	7	7	4	0.302083333
8	8	16	8	4	0.416666667
9	8	16	8	6	0.466666667
10	9	18	9	6	0.5375
11	9	18	9	9	0.6125
12	10	20	10	10	0.708333333
13	11	22	11	11	0.804166667
14	11	22	22	11	0.895833333
15	12	24	24	12	1

Total bobot pada setiap level merepresentasikan tingkat kesulitan. Apabila nilai total bobot semakin besar, maka dipastikan level tersebut semakin sulit. Peningkatan tingkat kesulitan ini bisa dilihat pada grafik pada Gambar 3.12.



Gambar 3.12 Grafik peningkatan tingkat kesulitan

BAB IV
IMPLEMENTASI

Pada bab ini akan dibahas mengenai implementasi yang dilakukan berdasarkan rancangan yang telah dijabarkan pada bab sebelumnya. Sebelum penjelasan implementasi akan ditunjukkan terlebih dahulu lingkungan untuk melakukan implementasi.

4.1 Lingkungan Implementasi

Lingkungan implementasi merupakan lingkungan dimana aplikasi akan dibangun. Lingkungan implementasi dibagi menjadi dua, yaitu lingkungan implementasi berupa perangkat keras dan lingkungan implementasi berupa perangkat lunak.

4.1.1 Perangkat Keras

Perangkat keras yang digunakan dalam pengembangan aplikasi ini adalah komputer dengan spesifikasi sebagai berikut :

- Prosesor : Intel® Core(TM) i5-4460 CPU @ 3.20GHz (4 CPUs) ~3.2 GHz
- Memori : 8192 MB DDR3

4.1.2 Perangkat Lunak

Perangkat lunak yang digunakan dalam pengembangan aplikasi ini adalah sebagai berikut :

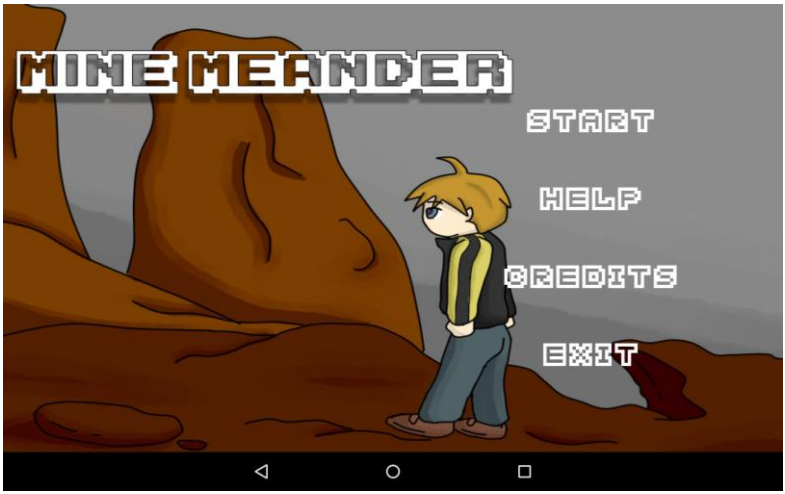
- Sistem Operasi : Windows 7 Ultimate
- IDE : Android Studio
- *Map Editor* : Tiled

4.2 Implementasi Antarmuka

Pada bagian ini dijelaskan mengenai implementasi antarmuka dari *game*. Implementasi antarmuka dari *game* ini meliputi antarmuka halaman awal (*Main Menu*), antarmuka halaman instruksi (*Help Menu*), antarmuka halaman pilihan *Level* (*Level Select Menu*), dan antarmuka area permainan.

4.2.1 Antarmuka Halaman Awal (Main Menu)

Antarmuka halaman awal (Main Menu) adalah tampilan pertama saat aplikasi dibuka. Tampilan ini memiliki 4 tombol yaitu "START", "HELP", "CREDITS", dan "EXIT". Tampilan dari antarmuka ini ditunjukkan pada Gambar 4.1.



Gambar 4.1 Implementasi antarmuka halaman awal (Main Menu)

Tombol "START" digunakan untuk masuk ke antarmuka halaman pilihan level (Level Select Menu). Implementasi kode sumber untuk tombol "START" ditunjukkan pada Kode Sumber 4.1.

```
TextButton button = new TextButton("START", skin);
button.addListener(new ChangeListener(){
    @Override
    public void changed(ChangeEvent event, Actor actor){
        MainMenu.this.transitionTo(new LevelSelectScreen());
    }
})
```

Kode Sumber 4.1 Tombol "START"

Tombol "HELP" berfungsi untuk masuk ke antarmuka halaman instruksi. Sedangkan tombol "CREDITS" berfungsi untuk masuk ke antarmuka halaman Credit. Implementasi kode sumber untuk tombol "HELP" dan tombol "CREDITS" ditunjukkan pada Kode Sumber 4.2 dan Kode Sumber 4.3.

```
TextButton button2 = new TextButton("HELP", skin);
button2.addListener(new ChangeListener(){
    @Override
    public void changed(ChangeEvent event, Actor actor){
        MainMenu.this.transitionTo(new HelpScreen());
    }
});
```

Kode Sumber 4.2 Tombol "HELP"

```
TextButton button3 = new TextButton("CREDITS", skin);
button3.addListener(new ChangeListener() {
    @Override
    public void changed (ChangeEvent event, Actor actor) {
        MainMenu.this.transitionTo(new CreditsScreen());
    }
});
```

Kode Sumber 4.3 Tombol "CREDITS"

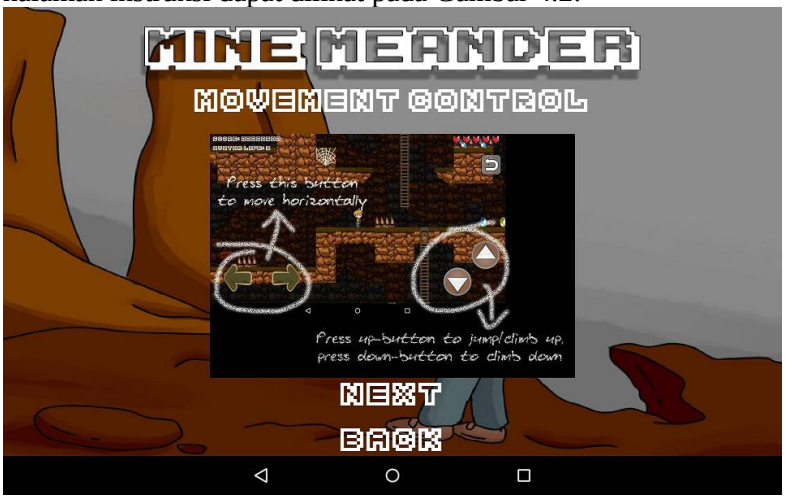
Tombol "EXIT"digunakan untuk keluar dari aplikasi. Implementasi kode sumber untuk tombol "EXIT" ditunjukkan pada Kode Sumber 4.4.

```
TextButton button4 = new TextButton("EXIT", skin);
button4.addListener(new ChangeListener() {
    @Override
    public void changed (ChangeEvent event, Actor actor) {
        Gdx.app.exit();
    }
});
```

Kode Sumber 4.4 Tombol "EXIT"

4.2.2 Antarmuka Halaman Instruksi (Help Menu)

Antarmuka halaman instruksi adalah antarmuka yang muncul setelah tombol "HELP" pada antarmuka halaman awal dipilih. Antarmuka ini digunakan untuk memberi informasi kepada pengguna mengenai instruksi permainan dan bagaimana cara memainkan permainan ini. Tampilan antarmuka dari halaman instruksi dapat dilihat pada Gambar 4.2.



Gambar 4.2 Tampilan antarmuka halaman instruksi (Help Menu)

Pada antarmuka halaman instruksi terdapat 2 tombol, yaitu tombol "NEXT" dan tombol "BACK". Tombol "NEXT"

berfungsi untuk melihat instruksi-instruksi lain. Tombol “BACK” digunakan untuk kembali ke antarmuka halaman awal (Main Menu). Implementasi kode sumber untuk tombol “NEXT” ditunjukkan pada Kode Sumber 4.5 , sedangkan implementasi kode sumber untuk tombol “BACK” ditunjukkan pada Kode Sumber 4.6.

```
TextButton button6 = new TextButton("NEXT", skin);
button6.addListener(new ChangeListener() {
    @Override
    public void changed (ChangeEvent event, Actor actor) {
        HelpScreen.this.transitionTo(new HelpScreen2());
    }
});
```

Kode Sumber 4.5 Tombol "NEXT"

```
TextButton button7 = new TextButton("BACK", skin);
button7.addListener(new ChangeListener() {
    @Override
    public void changed (ChangeEvent event, Actor actor) {
        HelpScreen.this.transitionTo(new MainMenu());
    }
});
```

Kode Sumber 4.6 Tombol "BACK"

4.2.3 Antarmuka Halaman Pilihan Level (Level Select Menu)

Antarmuka halaman pilihan level adalah antarmuka yang menampilkan daftar pilihan level yang tersedia. Dalam permainan ini terdapat 15 level yang bisa dipilih. Pada bagian antarmuka, 15 level ini dibagi menjadi 3. Tampilan antarmuka dari halaman pilihan level dapat dilihat pada Gambar 4.3 , 4.4 , dan 4.5.

Gambar 4.3 Tampilan antarmuka halaman pilihan level

Gambar 4.4 Tampilan antarmuka halaman pilihan level

Gambar 4.5 Tampilan antarmuka halaman pilihan level

Pada antarmuka halaman pilihan level terdapat 4 jenis tombol, yaitu tombol "LEVEL", tombol "NEXT WORLD", tombol "PREVIOUS WORLD", dan tombol "BACK". Tombol "LEVEL" berfungsi untuk memilih level tertentu dan masuk ke dalam area permainan. Kode sumber dari tombol "LEVEL" ditunjukkan pada Kode Sumber 4.7.

```
TextButton button = new TextButton("LEVEL 1", skin);
button.addListener(new ChangeListener(){
    @Override
    public void changed(ChangeEvent event, Actor actor){
        selectlevel = 1;
        fadeOut();
    }
});
TextButton button2 = new TextButton("LEVEL 2", skin);
button2.addListener(new ChangeListener(){
    @Override
    public void changed(ChangeEvent event, Actor actor){
        selectlevel = 2;
        fadeOut();
```

```
    }
});
TextButton button3 = new TextButton("LEVEL 3", skin);
button3.addListener(new ChangeListener(){
    @Override
    public void changed(ChangeEvent event, Actor actor){
        selectLevel = 3;
        fadeOut();
    }
});
TextButton button4 = new TextButton("LEVEL 4", skin);
button4.addListener(new ChangeListener(){
    @Override
    public void changed(ChangeEvent event, Actor actor){
        selectLevel = 4;
        fadeOut();
    }
});
TextButton button5 = new TextButton("LEVEL 5", skin);
button5.addListener(new ChangeListener(){
    @Override
    public void changed(ChangeEvent event, Actor actor){
        selectLevel = 5;
        fadeOut();
    }
});
```

Kode Sumber 4.7 Tombol "LEVEL"

Tombol "NEXT WORLD" berfungsi untuk berpindah ke halaman *World* selanjutnya, sedangkan tombol "PREVIOUS WORLD" berfungsi untuk berpindah ke halaman *World* sebelumnya. Tombol "BACK" berfungsi untuk kembali ke antarmuka halaman awal (Main Menu). Implementasi dari ke-3 tombol ini ditunjukkan pada Kode Sumber 4.8.

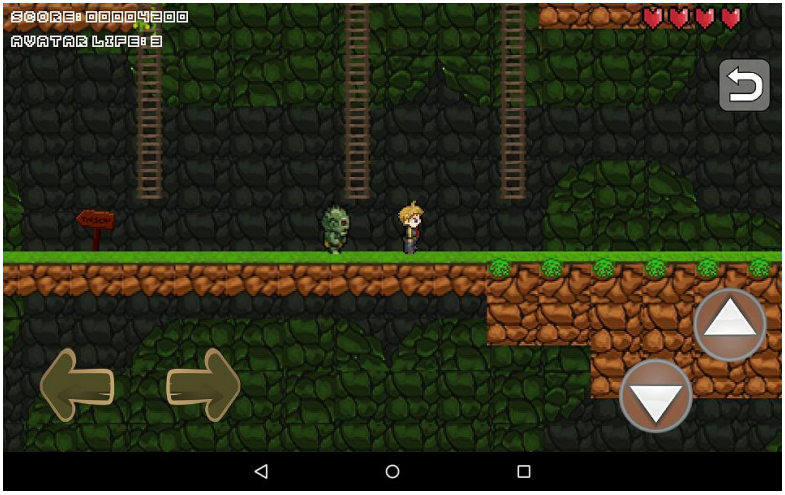
```
TextButton button6 = new TextButton("PREVIOUS WORLD", skin);
button6.addListener(new ChangeListener(){
    @Override
    public void changed(ChangeEvent event, Actor actor){
```

```
        LevelSelectScreen2.this.transitionTo(new
LevelSelectScreen());
    }
});

TextButton button8 = new TextButton("NEXT WORLD", skin);
button8.addListener(new ChangeListener(){
    @Override
    public void changed(ChangeEvent event, Actor actor){
        LevelSelectScreen.this.transitionTo(new
LevelSelectScreen2());
    }
});

TextButton button10 = new TextButton("BACK", skin);
button10.addListener(new ChangeListener() {
    @Override
    public void changed (ChangeEvent event, Actor actor) {
        LevelSelectScreen2.this.transitionTo(new MainMenu());
    }
});
```

Kode Sumber 4.8 Tombol "PREVIOUS WORLD", "NEXT WORLD", dan "BACK"

4.2.4 Antarmuka Area Permainan

Antarmuka area permainan adalah antarmuka yang muncul setelah tombol "LEVEL" pada halaman pilihan level dipilih. Antarmuka ini merupakan tempat dimana pengguna bisa mulai memainkan permainan. Selain terdiri dari *world map*, pada antarmuka area permainan juga terdapat informasi status karakter, skor, dan juga menyediakan tombol kontrol. Tampilan antarmuka area permainan dapat dilihat pada Gambar 4.6.



Gambar 4.6 Antarmuka area permainan

4.3 Implementasi Aturan Main

Pada bagian ini dijelaskan mengenai aturan main saat permainan berlangsung. Implementasi ini meliputi tentang implementasi pergerakan karakter, implementasi interaksi karakter dengan objek lain, implementasi perolehan skor, dan implementasi kondisi menang atau kalah.

4.3.1 Implementasi Pergerakan Karakter

Karakter yang dikontrol oleh pemain bisa berlari secara horizontal, melompat, dan juga menaiki atau menuruni tangga. Saat berlari, melompat, ataupun melalui tangga, Karakter memiliki nilai *default* yang merepresentasikan seberapa cepat saat berlari atau seberapa jauh saat dia melompat. Nilai *default* ini ditunjukkan pada Kode Sumber 4.9.

```
public static float WALK_POWER = 3;
public static float JUMP_POWER = 700;
public static float CLIMB_POWER = 50;
```

Kode Sumber 4.9 Nilai default kecepatan saat berlari, melompat, dan memanjat

Saat pemain menggerakkan karakternya ke kiri atau ke kanan, akan ada fungsi yang menerima input dari pemain. Fungsi ini kemudian akan me-*return* dengan nilai WALK_POWER. Kode sumber dari fungsi ini ditunjukkan pada Kode Sumber 4.10.

```
// Berjalan ke kiri
public float getLeftThrust() {
   if(Gdx.input.isKeyPressed(Input.Keys.A) ||
LevelScreen.controller.isLeftPressed()){
      return WALK_POWER;
   }
   else {
      return 0f;
   }
}

// Berjalan ke kanan
public float getRightThrust() {
   if(Gdx.input.isKeyPressed(Input.Keys.D) ||
LevelScreen.controller.isRightPressed()) {
      return WALK_POWER;
   }
   else {
      return 0f;
   }
}
```

Kode Sumber 4.10 Fungsi yang dipanggil ketika pemain menggerakkan karakternya ke kiri atau ke kanan

Dari return value yang didapat, nilai tersebut kemudian digunakan untuk menggerakan objek karakter. Proses ini ditunjukkan pada Kode Sumber 4.11.

```
float goRight = getRightThrust();
float goLeft = getLeftThrust();

if (goRight > 0) {
   switch(ladderStatus) {
   case GameObject.LADDER:
      body.applyLinearImpulse(forceVector.set(4.4f*goRight,
0.0f), FORCE_APPLICATION_POINT, true);break;
   case GameObject.LADDER + GameObject.LADDER_BELOW:
     body.applyLinearImpulse(forceVector.set(1.4f*goRight,
0.0f), FORCE_APPLICATION_POINT, true);break;
   default:
     body.applyLinearImpulse(forceVector.set(4.4f*goRight,
0.0f), FORCE_APPLICATION_POINT, true);break;
   }
}

else if (goLeft > 0) {
   switch(ladderStatus) {
   case GameObject.LADDER:
      body.applyLinearImpulse(forceVector.set(-4.4f*goLeft,
0.0f), FORCE_APPLICATION_POINT, true);break;
   case GameObject.LADDER + GameObject.LADDER_BELOW:
      body.applyLinearImpulse(forceVector.set(-1.4f*goLeft,
0.0f), FORCE_APPLICATION_POINT, true);break;
   default:
      body.applyLinearImpulse(forceVector.set(-4.4f*goLeft,
0.0f), FORCE_APPLICATION_POINT, true);break;
   }
}
```

Kode Sumber 4.11 Proses ketika menggerakkan objek karakter ke kiri atau ke kanan

Kemudian saat pemain menggerakkan karakternya untuk melompat atau menuruni tangga, akan ada fungsi yang bertugas menangani input dari pemain. Fungsi ini ditunjukkan pada Kode Sumber 4.12.

```
// Melompat
public float goUp() {
   if(Gdx.input.isKeyJustPressed(Input.Keys.W) ||
LevelScreen.controller.isUpPressed()) {
      jump();
      return 10f;
   }
   else {
      return 0f;
   }
}

// Turun
public float goDown() {
   if(Gdx.input.isKeyPressed(Input.Keys.S) ||
LevelScreen.controller.isDownPressed()) {
      wasDraggingDown = true;
      return 10f;
   }
   else {
      return 0f;
   }
}
```

Kode Sumber 4.12 Fungsi yang dipanggil ketika pemain menggerakkan karakternya ke atas atau ke bawah

Fungsi *goUp()* akan dipanggil ketika pemain menekan tombol Arah Atas pada *controller*. Fungsi tersebut kemudian akan menjalankan fungsi *jump()* dimana fungsi ini akan menggerakkan objek karakter ke atas. Fungsi *jump()* ditunjukkan pada Kode Sumber 4.13.

```
public void jump() {
   if (wasClimbing) {
         if (isTopOfTheLadder()) {
            body.applyLinearImpulse(jumpVector.set(0f,
CLIMB_POWER*2), FORCE_APPLICATION_POINT, true);
         }
         else {
          body.applyLinearImpulse(jumpVector.set(0f,
CLIMB_POWER), FORCE_APPLICATION_POINT, true);
         }
   }
   else {
      Vector2 linearVelocity = body.getLinearVelocity();
      if (Math.abs(linearVelocity.y) < 0.0001f) {
         body.applyLinearImpulse(jumpVector.set(0f,
JUMP_POWER), FORCE_APPLICATION_POINT, true);
      }
   }
   wasDraggingDown = false;
}
```

Kode Sumber 4.13 Fungsi jump()

Fungsi goDown() akan dipanggil saat pemain menekan tombol Arah Bawah pada controller. Fungsi ini akan mengganti nilai *variable* "*wasDraggingDown*" ke *true* yang nantinya apabila karakter sedang berada di tangga, maka karakter akan bisa menuruni tangga. Proses ini ditunjukkan pada Kode Sumber 4.14.

```
@Override
public void preSolve(Contact contact, Manifold oldManifold) {
   Avatar avatar = objectManager.getAvatar();

   Fixture fixtureA = contact.getFixtureA();
   Fixture fixtureB = contact.getFixtureB();
   Body bodyA = fixtureA.getBody();
   Body bodyB = fixtureB.getBody();
   GameObjectData userDataA = (GameObjectData)
bodyA.getUserData();
```

```
   GameObjectData userDataB = (GameObjectData)
bodyB.getUserData();
...
if ((userDataA.collisionCategory == CollisionCategory.AVATAR
&& userDataB.collisionCategory ==
CollisionCategory.TRAVERSABLE_PLATFORM)
      || (userDataB.collisionCategory ==
CollisionCategory.AVATAR && userDataA.collisionCategory ==
CollisionCategory.TRAVERSABLE_PLATFORM)) {

   if (avatar.wasDraggingDown || avatar.isClimbing()) {
      contact.setEnabled(false);
   }
}
...
}
```

Kode Sumber 4.14 Proses ketika karakter menuruni tangga

4.3.2 Implementasi Interaksi Karakter dengan Rintangan

Ketika karakter bersentuhan secara langsung dengan objek rintangan, maka karakter akan kehilangan *Heart* sebanyak 1. Contoh objek rintangan adalah *Spike*, *Dangerous Flower*, *Spider*, dan *Zombie*. Keempat objek tersebut dimasukkan ke dalam kategori "Enemy". Kode yang bertugas mendeteksi adanya sentuhan antara karakter dengan rintangan dapat dilihat pada Kode Sumber 4.15.

```
@Override
public void beginContact(Contact contact) {
   Fixture fixtureA = contact.getFixtureA();
   Fixture fixtureB = contact.getFixtureB();
   Body bodyA = fixtureA.getBody();
   Body bodyB = fixtureB.getBody();
   GameObjectData userDataA = (GameObjectData)
bodyA.getUserData();
```

```
   GameObjectData userDataB = (GameObjectData)
bodyB.getUserData();
   if ((userDataA.collisionCategory ==
CollisionCategory.AVATAR && userDataB.collisionCategory ==
CollisionCategory.ENEMY)) {

   Avatar avatar = objectManager.getAvatar();
   if (!avatar.isInvicible()) {
      Vector2 positionA = bodyA.getPosition();
      Vector2 positionB = bodyB.getPosition();
      Vector2 b2a = positionA.sub(positionB).nor();
      bodyA.applyLinearImpulse(b2a.scl(200f), new Vector2(32,
32), true);
      avatar.onHit();
   }
}
   else if ((userDataB.collisionCategory ==
CollisionCategory.AVATAR && userDataA.collisionCategory ==
CollisionCategory.ENEMY)) {

   Avatar avatar = objectManager.getAvatar();
   if (!avatar.isInvicible()) {
      Vector2 positionA = bodyA.getPosition();
      Vector2 positionB = bodyB.getPosition();
      Vector2 b2a = positionA.sub(positionB).nor();
      bodyB.applyLinearImpulse(b2a.rotate(180f).scl(200f), new
Vector2(32, 32), true);
      avatar.onHit();
   }
}
...
}
```

Kode Sumber 4.15 Fungsi yang mendeteksi adanya sentuhan antara karakter dengan rintangan

Dari kode sumber di atas ketika telah terjadi sentuhan antara karakter dan rintangan (Enemy), maka akan memanggil fungsi onHit(). Fungsi onHit() berfungsi untuk mengurangi

jumlah Heart sebanyak 1. Saat karakter bersentuhan dengan rintangan,karakter akan berada dalam keadaan *invicible* selama beberapa detik. Kode dari fungsi tersebut ditunjukkan pada Kode Sumber 4.16.

```
public void onHit() {

   if (!invicible) {
      if (heart > 0) {
         heart--;
      }

      if (heart > 0) {
         grantInvisibilityOnNextRender = 2;
         invicible = true;
      }

      else {
         onDeath();
      }
   }
}
```

Kode Sumber 4.16 Fungsi yang dijalankan setelah karakter bersentuhan dengan rintangan

4.3.3 Implementasi Perolehan Skor

Saat karakter bersentuhan dengan kristal, maka karakter akan mendapatkan skor atau poin. Implementasi dari proses ketika karakter bersentuhan dengan kristal dapat dilihat pada Kode Sumber 4.17.

```
@Override
public void beginContact(Contact contact) {
   Fixture fixtureA = contact.getFixtureA();
   Fixture fixtureB = contact.getFixtureB();
   Body bodyA = fixtureA.getBody();
```

```
    Body bodyB = fixtureB.getBody();
    GameObjectData userDataA = (GameObjectData)
bodyA.getUserData();
    GameObjectData userDataB = (GameObjectData)
bodyB.getUserData();

...

    else if ((userDataB.collisionCategory ==
CollisionCategory.AVATAR && userDataA.collisionCategory ==
CollisionCategory.COLLECTABLE)) {
    Art.coinSound.play();
    if (!userDataA.hidden) {
       Avatar avatar = objectManager.getAvatar();

avatar.onItemCollected((Collectable)objectManager.get(userDataA
.id));
    }
    userDataA.hidden = true;
}

    else if ((userDataA.collisionCategory ==
CollisionCategory.AVATAR && userDataB.collisionCategory ==
CollisionCategory.COLLECTABLE)) {
    Art.coinSound.play();
    if (!userDataB.hidden) {
       Avatar avatar = objectManager.getAvatar();

avatar.onItemCollected((Collectable)objectManager.get(userDataB
.id));
    }
    userDataB.hidden = true;
}
...
}
```

Kode Sumber 4.17 Implementasi ketika karakter bersentuhan dengan kristal

Kristal yang didapat memiliki skor atau poin tertentu. Nilai *constant* dari kristal disimpan pada *class enum* JewelType, kode sumber dapat dilihat pada Kode Sumber 4.18.

```
public enum JewelType {
    BLUE,
    YELLOW;

    public int getScoreValue() {
        return 100;
    }
}
```

Kode Sumber 4.18 Enum Class "JewelType"

Selain terdapat *class enum* JewelType, juga terdapat *interface* Collectable. Pada interface ini terdapat *method* getScoreValue() yang nantinya akan diimplementasi pada *class* BlueJewel dan *class* YellowJewel. Kemudian setiap kali karakter mendapatkan kristal Blue atau Yellow, nilai atau poin yang didapat akan dijumlahkan dengan skor yang dimiliki sebelumnya. Kode untuk melakukan penghitungna skor ini dapat dilihat pada kode sumber 4.19.

```
public void onItemCollected(Collectable collectable) {

    incScore(collectable.getScoreValue());
}

public void incScore(int delta) {
    score+=delta;
}
```

Kode Sumber 4.19 Penjumlahan dari skor yang didapat

4.3.4 Implementasi Kondisi Menang atau Kalah

Di dalam permainan, karakter dinyatakan menang apabila telah menemukan dan masuk ke dalam pintu keluar. Implementasi dari bagian ini ditunjukkan pada Kode Sumber 4.20.

```
private void checkCollisions() {
   Vector2 position = body.getPosition();
   int tileX = (int) (position.x/Constant.METERS_PER_TILE);
   int tileY = (int) (position.y/Constant.METERS_PER_TILE);

   TiledMapTileLayer ladderLayer =
(TiledMapTileLayer)level.tiledMap.getLayers().get(Constant.LAD
DER_LAYER);
   Cell cell = ladderLayer.getCell(tileX, tileY);

   if (cell!=null) {
      ...
      else if
(CommonTile.EXIT.name().equals(cell.getTile().getProperties().
get("id")))
      {
         if(level.getWorldId() == 15)
         {
         level.screen.transitionTo(new
FinalLevelScreen(score));
         }
         else
         {
            level.onCompletion();
         }
      }
   }
}
```

Kode Sumber 4.20 Proses ketika karakter masuk ke pintu keluar

Karakter dinyatakan kalah apabila kehabisan *life* (nyawa). Saat kondisi kalah, aplikasi akan berganti antarmuka ke antarmuka Game Over. Implementasi dari bagian ini ditunjukkan pada Kode Sumber 4.21.

```
public void onDeath() {
   body.applyLinearImpulse(jumpVector.set((float) ((-
1+Math.random()*2)*WALK_POWER), CLIMB_POWER*8),
FORCE_APPLICATION_POINT, true);
```

```
    dead = true;
    invicible = true;
    grantInvisibilityOnNextRender = 10;
    life--;
    if (life == 0) {
        level.screen.transitionTo(new GameOverScreen(score));
    }
    else {
        level.reset();
    }
}
```

Kode Sumber 4.21 Fungsi yang dipanggil ketika karakter kehabisan life (nyawa)

4.4 Implementasi *Dynamic World Generator*

Pada subbab ini akan dijelaskan mengenai implementasi pembuatan *world map* yang dinamis. Tahap pembuatan *world map* ini meliputi implementasi *layout* utama dan implementasi *platform* dan tangga.

4.4.1 Implementasi *Layout* Utama

Pada Tugas Akhir ini, pembuatan *world map* menggunakan Algoritma *Recursive backtracking*. Implementasi dari Algoritma *Recursive backtracking* dilakukan pada *class* MMLevel*Layout*. Pada *class* MMLevel*Layout*, pencarian arah dilakukan dengan menggunakan *enum* Direction yang merepresentasikan 4 arah (*NORTH*, *SOUTH*, *EAST*, dan *WEST*). Kode *enum* ditunjukkan pada Kode Sumber 4.22.

```
public static enum Direction {
    NORTH, SOUTH, EAST, WEST
}
```

Kode Sumber 4.22 Enum Direction

Kemudian dibuat *class* Orientation yang berfungsi sebagai tempat untuk menyimpan Direction *room*. Setelah *class* selesai

dibuat, kemudian dilakukan instansiasi. Implementasi dapat dilihat pada Kode Sumber 4.23.

```
public static class Orientation {
   public Direction previous;
   public Direction current;
   public Orientation(Direction previous, Direction current) {
      super();
      this.previous = previous;
      this.current = current;
   }
}
public static Orientation[][] directions = new
Orientation[64][64];
```

Kode Sumber 4.23 class Orientation

Untuk memulai pembuatan *layout*, ditentukan terlebih dahulu posisi *room* pertama. Pada bagian ini, *room* pertama akan selalu diletakkan di tengah. Dari *room* pertama ke *room* kedua akan diatur supaya selalu mengarah ke kanan (*EAST*). Kode untuk membuat *room* awal ditunjukkan pada Kode Sumber 4.24.

```
public static MMLevelLayout random(int nbRooms) {
   int directionGridCenter = directions.length/2;
   do{
      failure = false; stateSearching = true;
      clearDirectionGrid();
      Direction lastDirection = Direction.EAST;
      tempDirectionWhenFail = Direction.EAST;
      cursorX = directionGridCenter;
      cursorY = directionGridCenter;
      directions[directionGridCenter][directionGridCenter] =
new Orientation(null, Direction.EAST);
      searchingPath(nbRooms-1, cursorX, cursorY,
lastDirection);}
while(failure);
```

Kode Sumber 4.24 Pembuatan room pertama

Setelah *room* pertama selesai, pencarian arah pada *room* kedua dan seterusnya akan menggunakan Algoritma *Recursive Backtracking*. Algoritma ini diletakkan pada fungsi searchingPath dimana fungsi ini memiliki 4 parameter. Parameter tersebut antara lain sisa *room* yang harus dilewati (nbRooms), koordinat *room* (cursorX dan cursorY), serta arah yang dituju (lastDirection).

Dari parameter lastDirection, akan didapat kemana arah yang akan dituju selanjutnya. Apabila lastDirection menunjuk ke arah *NORTH*, maka nilai cursorY akan ditambah 1 sehingga koordinat *room* yang dituju berada di atas *room* sebelumnya. Lalu apabila lastDirection menunjuk ke arah *SOUTH*, maka nilai cursorY akan dikurangi 1 sehingga koordinat *room* yang dituju berada di bawah *room* sebelumnya. Apabila lastDirection menunjuk ke arah *EAST*, maka nilai cursorX akan ditambahkan dengan 1 sehingga *room* yang dituju berada di samping kanan *room* sebelumnya. Kemudian apabila lastDirection menunjuk ke arah *WEST*, maka nilai cursorX akan dikurangkan dengan 1 sehingga *room* yang dituju berada di samping kiri *room* sebelumnya. Kode dari proses ini ditunjukkan pada Kode Sumber 4.25.

```
...
if(lastDirection != null){
   switch(lastDirection){
      case NORTH:
         cursorY++;break;
      case SOUTH:
         cursorY--;break;
      case EAST:
         cursorX++;break;
      case WEST:
         cursorX--;break;
   }
}
...
```

Kode Sumber 4.25 Proses mencari room yang dituju berdasarkan Direction

Setelah mendapatkan koordinat dari *room* yang dituju, kemudian dilakukan pengecekan apakah ada *room* yang tersedia di sekitar *room* yang dituju tersebut. Sebelum dilakukan pengecekan, pertama kita harus membuat suatu ArrayList yang bertugas sebagai tempat penyimpanan Direction yang memungkinkan untuk dipilih. Deklarasi ArrayList ini dilakukan di luar fungsi searchingPath dan diberi nama possibleDirections. Pengecekan dilakukan dengan melihat apakah ada *room* yang menempati posisi *NORTH*, *EAST*, *SOUTH*, dan *WEST* dari *room* yang dituju. Apabila ada, maka hapus Direction tersebut dari ArrayList possibleDirections. Kode dari proses pengecekan ketersediaan *room* di sekitar *room* yang dituju ini ditunjukkan pada Kode Sumber 4.26.

```
...
possibleDirections.clear();
possibleDirections.addAll(directionList);
if(cursorX == 29 || directions[cursorX-1][cursorY] != null){
   possibleDirections.remove(Direction.WEST);
}
if(cursorY == 35 || directions[cursorX][cursorY+1] != null){
   possibleDirections.remove(Direction.NORTH);
}
if(cursorY == 29 || directions[cursorX][cursorY-1] != null){
   possibleDirections.remove(Direction.SOUTH);
}
if(cursorX == 35 || directions[cursorX+1][cursorY] != null){
   possibleDirections.remove(Direction.EAST);
}

size = possibleDirections.size();
...
```

Kode Sumber 4.26 Proses pengecekan ketersediaan room disekitar ruang yang dituju

Dari possibleDirections yang didapat, apabila jumlah Direction-nya lebih dari nol maka secara random dipilih salah

satu. Setiap kali suatu *room* baru dan Direction dipilih, koordinat XY dari *room* dan juga Direction nya harus disimpan di dalam *stack*. Hal itu agar kita bisa melakukan *backtracking* ke langkah sebelumnya apabila dalam proses pencarian *path* menemui jalan buntu. Setelah memilih Direction dari possibleDirections, Direction tersebut akan menjadi lastDirection di proses rekursi yang selanjutnya. Kode dari proses ini dapat dilihat pada Kode Sumber 4.27.

```
...
// Merandom dan memilih salah satu Directions dari
possibleDirections
random = Math.random();
index2 = (int)(random*size);
if( lastDirection != null){
   // Menyimpan koordinat XY dan Directions ke stack
   lastoDirections.push(lastDirection.toString());
   lastoCursorX.push(cursorX); lastoCursorY.push(cursorY);
}
// Assign koordinat dan Directions room yang baru
newDirection = possibleDirections.get(index2);
directions[cursorX][cursorY] = new Orientation(lastDirection,
newDirection);
if(lastDirection == null){
   stringDirection = lastoDirections.arrayTop();
   lastDirection = Direction.valueOf(stringDirection);
   directions[cursorX][cursorY] = new
Orientation(lastDirection, newDirection);
}
lastDirection = newDirection;
stateSearching = true;

// Melanjutkan ke proses rekursi yang selanjutnya
searchingPath(nbRooms-1, cursorX, cursorY, lastDirection);
```

Kode Sumber 4.27 Proses pemilihan Direction untuk room selanjutnya secara random

Kemudian apabila dari possibleDirections yang didapat hasilnya nol atau tidak ada yang memungkinkan untuk dipilih, maka harus dilakukan proses *backtracking*. Sebelum proses *backtracking* dimulai, kita harus menyimpan variabel lastDirection dengan tujuan untuk menandai agar Direction tersebut tidak akan dipilih lagi. Kemudian nilai cursorX dan cursorY diatur supaya kembali ke *room* pada langkah yang sebelumnya. Setelah itu dilakukan pengecekan apakah ada room yang bisa dipilih disekitar posisi cursorX dan cursorY. Apabila ada maka, pilih *room* tersebut dan dilanjutkan ke proses rekursi berikutnya. Apabila tidak ada, maka dilakukan *backtracking* sehingga nilai cursorX dan cursorY diubah ke langkah sebelumnya lagi. Kode dari proses ini ditunjukkan pada Kode Sumber 4.28.

```
...
if(size == 0){

   //Menyimpan direction yang tidak boleh dipilih lagi
   tempDirectionWhenFail = lastDirection;

   //Set nilai cursorX dan cursorY dan mengosongkan room pada
cursorX dan cursorY
   cursorX = lastoCursorX.arrayTop();
   cursorY = lastoCursorY.arrayTop();
   directions[cursorX][cursorY] = null;

   //menyimpan top stack dari Direction
   stringDirection = lastoDirections.arrayTop();
   Direction topDirection =
Direction.valueOf(stringDirection);
   cursorX = lastoCursorX.arrayTop(); cursorY =
lastoCursorY.arrayTop();

   //Mengecek ketersediaan ruang di sekitar
   possibleDirections.clear();
   possibleDirections.addAll(directionList);
   if(directions[cursorX-1][cursorY] != null ||
```

```
tempDirectionWhenFail == Direction.WEST)
        possibleDirections.remove(Direction.WEST);
    if(directions[cursorX+1][cursorY] != null ||
tempDirectionWhenFail == Direction.EAST)
        possibleDirections.remove(Direction.EAST);
    if(directions[cursorX][cursorY-1] != null ||
tempDirectionWhenFail == Direction.SOUTH)
        possibleDirections.remove(Direction.SOUTH);
    if(directions[cursorX][cursorY+1] != null ||
tempDirectionWhenFail == Direction.NORTH)
        possibleDirections.remove(Direction.NORTH);
    size = possibleDirections.size();
    stateSearching = false;
    nbRooms += 1;
    // Jika tidak ada ruang kosong yang tersedia, backtrack
    if( size == 0 )
    {
        lastoDirections.pop();
        lastoCursorX.pop(); lastoCursorY.pop();
        searchingPath2(nbRooms, cursorX, cursorY, null);
        return false;
    }
    // Jika terdapat room kosong, pilih room tsb
    else
    {
        random = Math.random();
        index2 = (int)(random*size);
        lastDirection = possibleDirections.get(index2);
        directions[cursorX][cursorY] = new
Orientation(topDirection, lastDirection);

        searchingPath(nbRooms-1, cursorX, cursorY,
lastDirection);
         return false;
}
```

Kode Sumber 4.28 Proses ketika backtracking

Fungsi tersebut dilakukan berulang sampai variabel nbRooms mencapai nilai nol. Koordinat *room* dan Direction yang sudah tercatat tersebut kemudian digunakan sebagai *layout* utama dari *world map*.

4.4.2 Implementasi *Platform* dan Tangga

Dari *room* atau *layout* utama yang telah terbentuk, langkah selanjutnya adalah mengisi *room* tersebut dengan *platform* dan tangga. Langkah awal dari pembuatan tangga dan *platform* ini adalah membuat titik-titik yang diletakkan secara random pada setiap *room*. Seperti yang sudah dijelaskan pada bab sebelumnya, titik-titik ini berfungsi sebagai acuan jalan utama penghubung antar *room*. Karena untuk memastikan setidaknya ada satu jalan penghubung antar *room*, maka dibuatlah jalan utama ini. Khusus untuk titik pada *room* terakhir, akan langsung diletakkan portal atau pintu keluar yang berfungsi sebagai tujuan akhir dari karakter pemain. Proses pembuatan titik-titik ini ditunjukkan pada Kode Sumber 4.29.

```
private void buildLevelPath(MMLevelLayout levelLayout) {
   GridPoint2 wpStartPosition = null;
   GridPoint2 wpTargetPosition = null;
   // Meletakkan titik di setiap room secara random
   for (Room room : levelLayout) {
      if (wpStartPosition == null) {
wpStartPosition = levelLayout.randomPositionInRoom(room, rng);
         this.startPosition = wpStartPosition;
      }
      Room nextRoom = levelLayout.nextRoom(room);
      if (nextRoom == null) {
         continue; }
      wpTargetPosition =
levelLayout.randomPositionInRoom(nextRoom, rng);
      buildLevelPathBetween(wpStartPosition, wpTargetPosition,
room.orientation.current);
      wpStartPosition = wpTargetPosition;
```

```
    }
spriteLayer.setCell(startPosition.x, startPosition.y,
CommonTile.AVATAR.toCell(commonTileSet));
ladderLayer.setCell(endPosition.x, endPosition.y,
CommonTile.EXIT.toCell(commonTileSet));
}
```

Kode Sumber 4.29 Proses ketika meletakkan titik-titik penghubung antar room

Titik-titik yang telah terbentuk tersebut kemudian dijadikan acuan untuk meletakkan *platform* dan tangga. Pada dua titik yang menghubungkan 2 *room*, pasti terdapat selisih jarak pada sumbu X dan sumbu Y. Apabila selisih jarak pada sumbu X lebih besar daripada sumbu Y, maka *platform* akan diletakkan sepanjang selisih jarak pada sumbu X tersebut. Kemudian karena yang tersisa tinggal selisih jarak pada sumbu Y, maka diletakkan tangga setinggi selisih jarak pada sumbu Y. Lalu apabila sebaliknya selisih jarak pada sumbu Y lebih besar daripada sumbu X, maka diletakkan terlebih dahulu tangga setinggi selisih jarak pada sumbu Y. Kemudian diletakkan tangga sepanjang selisih jarak pada sumbu X. Kode dari proses ini ditunjukkan pada Kode Sumber 4.30.

```
...
while(cursor.x != targetPosition.x || cursor.y !=
targetPosition.y)
{
   dx = targetPosition.x-cursor.x;
   dy = targetPosition.y-cursor.y;
   if (Math.abs(dx) > Math.abs(dy)) {
      int sign = (int)Math.signum(dx);
      for (int pi = 0; pi < Math.abs(dx) + 1; pi++)
      {
         int platformX = cursor.x+(pi*sign);
         int platformY = cursor.y-1;
         setPlatformTile(platformX, platformY,
WorldTile.DIRT);
```

```
        }
        cursor.x+=dx;
    }
    else
    {
      int sign = (int)Math.signum(dy);
      if (dy<=2 && dy > 0)
      {
         if (last != 1)
         {
           clearLadderTile(cursor.x, cursor.y);
         }
         for (int pi = 0; pi <= Math.abs(dy)-1; pi++) {
            setPlatformTile(cursor.x, cursor.y+(pi*sign),
WorldTile.DIRT);
         }
      }
      else
      {
         int startLadderIndex = (last == 1 ? 1 : 0);
         for (int pi = startLadderIndex;pi<=Math.abs(dy);pi++)
         {
            setLadderTile(cursor.x, cursor.y+(pi*sign), false);
         }
      }
      cursor.y+=dy;
   }
}
...
```

Kode Sumber 4.30 Peletakkan platform dan tangga sebagai jalan utama

Setelah jalan utama terbentuk, kemudian diletakkan *platform* dan tangga lain secara random. Peletakkan *platform* ini dimulai dari bagian kiri-atas dari setiap *room*. Dalam proses *generate platform* ini, panjang *platform* ditentukan secara random. Panjang minimum adalah 0 (tidak me-*generate* apa-apa) dan panjang maksimum adalah 8. Ketika meletakkan platform, harus

dipastikan bahwa platform ini tidak menutupi jalan utama yang telah dibuat pada langkah sebelumnya. Kode proses *generate platform* ini ditunjukkan pada Kode Sumber 4.31.

```
...
for (int y = roomHeight - 3; y > 0; y--)
   {
   if (Math.random() <= 0.7) {
      for (int x = 1; x < roomWidth - 2; x++)
         {
         if (Math.random() < 0.2) {

            // Mengatur panjang platform
            double nextGaussian = rng.nextGaussian() + 1;
            if (nextGaussian < 0) {
               nextGaussian = 0;
            }
            if (nextGaussian > 2) {
               nextGaussian = 2;
            }
            int platformLength = (int) (nextGaussian * 4);
            if (x + platformLength > roomWidth) {
               platformLength = roomWidth - x;
            }

            int i = 0;
            if (platformLength > 0) {
               for (i = 0; i < platformLength; i++)
                  {

                  // Jika tidak ada objek lain, letakkan
platform pada titik x,y
                  if (!blockAt(roomOffsetX, roomOffsetY, x +
i, y - 2) && !blockAt(roomOffsetX, roomOffsetY, x + i, y - 1))
                  {
                     setPlatformTile(roomOffsetX, roomOffsetY,
x + i, y);
                  }
```

```
                    else
                    // Jika ada objek lain, break loop
                    {
                        break;
                    }
                }
            }
            // Memberi ruang / sela-sela
            int variableSpaceLength = 1 + (int)
rng.nextDouble() * 8;
            x += i + variableSpaceLength;
        }
    }
}
}
```

Kode Sumber 4.31 Generate random platform

Kemudian dari *platform-platform* yang telah terbentuk, diletakkan 1 tangga. Posisi tangga diletakkan secara random dan bergantung pada seberapa panjang platformnya. Kode proses *generate* tangga dapat dilihat pada Kode Sumber 4.32.

```
private void createRandomLadders(int roomIndex, int
roomOffsetX, int roomOffsetY)
{

   List<Platform> platformList=generatedPlatforms[roomIndex];
   for (Platform platform : platformList) {

      int ladderX = rng.nextInt(platform.length);
      if (!blockAt(platform.x + ladderX, platform.y + 1) &&
!blockAt(platform.x + ladderX, platform.y - 1))
         {
         setLadderTile(platform.x + ladderX, platform.y,
true);
         ladderList.add(new Ladder(platform.x + ladderX,
platform.y, 1));
         }
```

```
    }
}
```

Kode Sumber 4.32 Fungsi generate tangga

4.5 Implementasi Skenario dan Tingkat Kesulitan

Dari skenario yang telah dirancang pada bab 3, karakter yang dikontrol oleh pemain memiliki Heart sebanyak 5 dan life (nyawa) sebanyak 3. Kode implementasi untuk bagian ini ditunjukkan pada Kode Sumber 4.33.

```
public static int life = 3;
public int heart = 5;
```

Kode Sumber 4.33 Status karakter

Tingkat kesulitan pada permainan ini berdasar pada 4 faktor, yaitu jumlah *room* yang harus dilewati, jumlah *dangerous flower*, jumlah *spider*, dan jumlah *zombie*. Implementasi tingkat kesulitan untuk banyaknya *room* yang harus dilewati dapat dilihat pada Kode Sumber 4.34.

```
...

if( worldId == 1 || worldId == 2)
{
   numRooms = 4;
}

else if(worldId == 3 || worldId == 4)
{
   numRooms = 5;
}

else if(worldId == 5)
{
   numRooms = 6;
}
```

```
else if(worldId == 6 || worldId == 7)
{
   numRooms = 7;
}

else if(worldId == 8 || worldId == 9)
{
   numRooms = 8;
}

else if(worldId == 10 || worldId == 11)
{
   numRooms = 9;
}

else if(worldId == 12)
{
   numRooms = 10;
}

else if(worldId == 13 || worldId == 14)
{
   numRooms = 11;
}

else if(worldId == 15)
{
   numRooms = 12;
}

MMLevelLayout mmLevelLayout = MMLevelLayout.random(numRooms);

...
```

Kode Sumber 4.34 Implementasi tingkat kesulitan berdasarkan jumlah room yang harus dilewati

Sedangkan untuk pengaturan tingkat kesulitan berdasarkan jumlah rintangan, implementasi *pseudocode* ditunjukkan pada

Gambar 4.7. Langkah awal dalam peletakan rintangan adalah mengecek terlebih dahulu seberapa banyak *platform* yang ada pada satu *room*. Daftar *platform* yang ada tersebut disimpan di variabel platformList. Setelah itu, kita menentukan berapa banyak rintangan yang akan ditaruh pada satu *room*. Jumlah rintangan ini disimpan pada variabel indexObstacle. Kemudian dari banyaknya rintangan yang sudah ditentukan, kita memilih secara random salah satu *platform* dari platformList. Dari *platform* yang dipilih tersebut, terlebih dahulu dihitung panjangnya. Kemudian secara random dipilih satu titik diantara panjang tersebut. Titik ini berfungsi sebagai posisi dari rintangan yang akan diletakkan. Apabila tidak ada objek pada titik tersebut, maka letakkan rintangan pada posisi itu.

Masukan	-
Keluaran	_

```
1. platformList = list of generated platforms
2. indexObstacle = number of obstacle in each rooms
3. boolean findStatus
4. for( indexObstacle )
5.   findStatus = false
6.   While(findStatus = false)
7.      Randomly choose platform from platformList
8.      Size = platform.size
9.      Choose random place between the size
10.     If there is no object above the placement
11.         findStatus = true
12.     If findStatus = true
13.        Set the value of Flower,Spider,or Zombie
```

Gambar 4.7 Pseudocode tingkat kesulitan berdasarkan jumlah rintangan

BAB V
UJI COBA DAN EVALUASI

Pada bab ini akan dijelaskan mengenai rangkaian uji coba dan evaluasi yang dilakukan. Proses pengujian berdasarkan skenario yang telah ditentukan dan pengujian dilakukan dengan survei langsung ke pengguna.

5.1 Lingkungan Uji Coba

Lingkungan pelaksanaan uji coba meliputi perangkat keras dan perangkat lunak yang akan digunakan pada sistem ini. Spesifikasi perangkat keras dan perangkat lunak yang digunakan dalam rangka uji coba perangkat lunak ini ditunjukkan pada Tabel 5.1.

Tabel 5.1 Lingkungan pengujian perangkat lunak

Perangkat Keras	**Komputer Desktop** Prosesor : Intel® Core(TM) i5-4460 CPU @ 3.20GHz (4 CPUs) ~3.2 GHz Memori : 8192 MB DDR3 **Perangkat Bergerak** Tipe : Asus Google Nexus 7 (2013) Prosesor : Quad-core 1.5 GHz Krait Memori : 2 GB RAM
Perangkat Lunak	**Komputer Desktop** Sistem Operasi : Microsof Windows 7 Ultimate 64-bit Perangkat Pengembang : Android Studio 2.1.1 **Perangkat Bergerak** Sistem Operasi : Android 5.0.2

5.2 Skenario Pengujian

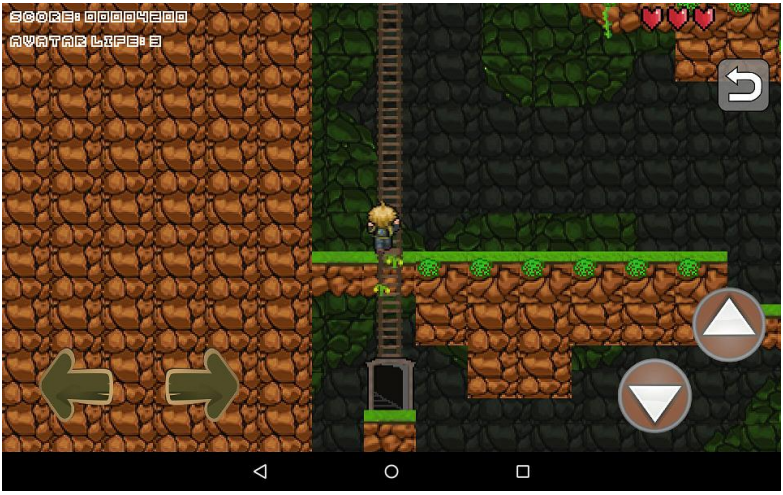
Skenario pengujian ini dilakukan untuk menguji apakah fungsionalitas yang diidentifikasi pada tahap kebutuhan benar-benar diimplementasikan dan bekerja semestinya. Metode pengujian akan mengacu pada *black-box testing*. Black-box testing adalah pengujian yang dilakukan hanya mengamati hasil eksekusi melalui data uji dan memeriksa fungsionalitas dari perangkat lunak. Pengujian akan dibagi menjadi dua bagian, yaitu pengujian fungsionalitas dan pengujian aplikasi terhadap pengguna. Pengujian ini dilakukan untuk mengetahui apakah fungsionalitas sistem telah berjalan sebagaimana mestinya.

5.2.1 Pengujian Fungsionalitas

Pengujian fungsionalitas dilakukan dengan menyiapkan beberapa skenario pengujian sebagai tolok ukur keberhasilan pengujian. Metode pengujian yang digunakan pada pengujian fungsionalitas adalah metode *black-box.*

5.2.1.1 Pengujian Alur Permainan

Skenario pengujian alur permainan digunakan untuk mengetahui apakah permainan sudah dapat dimainkan dan pengguna bisa menyelesaikan setidaknya satu level. Dalam pengujian alur permainan terdapat 2 skenario. Skenario yang pertama adalah pemain harus bisa mencapai keadaan menang dalam permainan. Sedangkan skenario yang kedua adalah pemain harus bisa mencapai keadaan kalah dalam permainan. Skenario ini ditunjukkan pada Tabel 5.2. *Screenshot* hasil pengujian dapat dilihat pada Gambar 5.1, Gambar 5.2, dan Gambar 5.3



Gambar 5.1 Hasil pengujian alur permainan ketika pengguna berhasil menemukan pintu keluar

Gambar 5.2 Hasil pengujian alur permainan ketika pengguna telah menyelesaikan satu level

Gambar 5.3 Hasil pengujian alur permainan ketika pengguna mencapai kondisi kalah

Tabel 5.2 Pengujian alur permainan (Skenario 1)

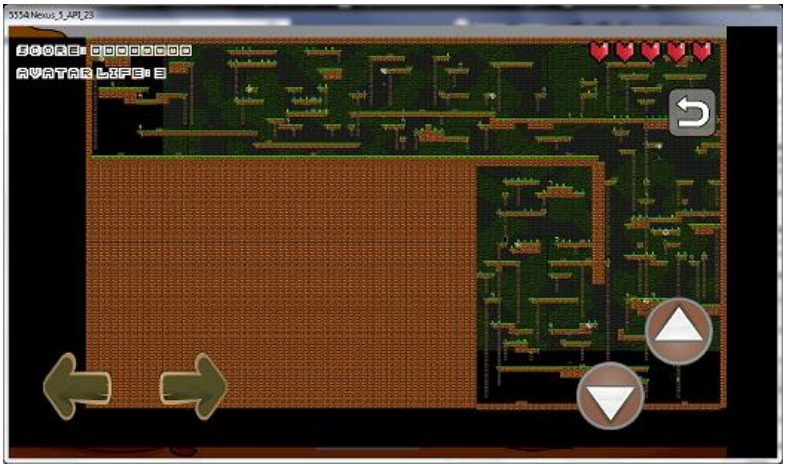
Nomor	SP-001-01
Nama	Alur permainan
Tujuan	Mengecek apakah aplikasi dapat menampilkan area permainan dan pengguna dapat mencapai kondisi menang atau kalah.
Kondisi Awal	Pengguna sudah berada di Main Menu.
Skenario	Pengguna menekan tombol Start, kemudian memilih salah satu level. Setelah masuk ke dalam area permainan, pengguna menggerakkan karakternya menyusuri map untuk mencari pintu keluar.
Keluaran yang Diharapkan	Pengguna dapat menyelesaikan level atau berakhir pada kondisi kalah.
Kondisi Akhir	Pengguna berhasil menemukan pintu keluar dan menyelesaikan level.
Hasil Pengujian	Berhasil.

Tabel 5.3 Pengujian alur permainan (Skenario 2)

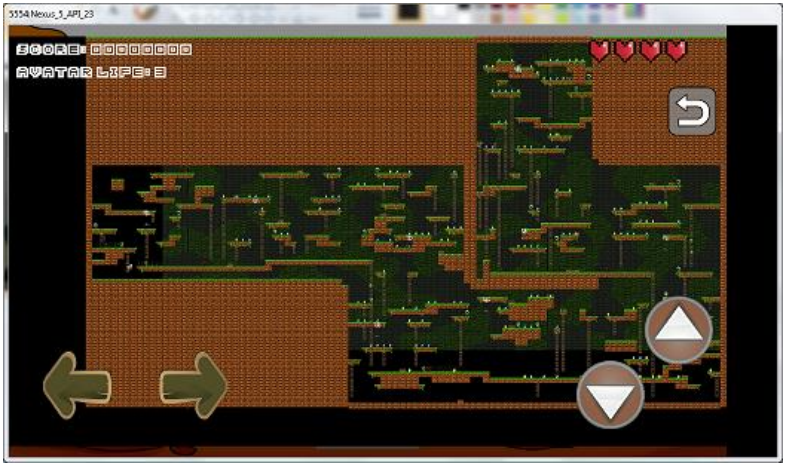
Nomor	SP-001-02
Nama	Alur permainan
Tujuan	Mengecek apakah aplikasi dapat menampilkan area permainan dan pengguna dapat mencapai kondisi menang atau kalah.
Kondisi Awal	Pengguna sudah berada di Main Menu.
Skenario	Pengguna menekan tombol Start, kemudian memilih salah satu level. Setelah masuk ke dalam area permainan, pengguna menggerakkan karakternya menyusuri map untuk mencari pintu keluar.
Keluaran yang Diharapkan	Pengguna dapat menyelesaikan level atau berakhir pada kondisi kalah.
Kondisi Akhir	Pengguna menemui kondisi kalah dalam permainan (Game Over) .
Hasil Pengujian	Berhasil.

5.2.1.2 Pengujian *Dynamic World Generator*

Skenario pengujian *dynamic world generator* digunakan untuk mengecek apakah *world map* pada setiap level bisa berubah-ubah. Skenario yang dilakukan adalah pengguna diminta untuk memilih salah satu level. Setelah mencoba memainkan pada level tersebut, pengguna kemudian kembali ke menu pilihan level dan memilih level itu lagi. Skenario ini ditunjukkan pada Tabel 5.4. Pengujian dinyatakan berhasil apabila *world map* yang dihasilkan berbeda dibanding dengan yang sebelumnya. *Screenshot* hasil pengujian dapat dilihat pada Gambar 5.4 dan 5.5.



Gambar 5.4 Tampilan hasil pengujian dynamic world generator pada level tertentu



Gambar 5.5 Tampilan hasil pengujian dynamic world generator pada level yang sama

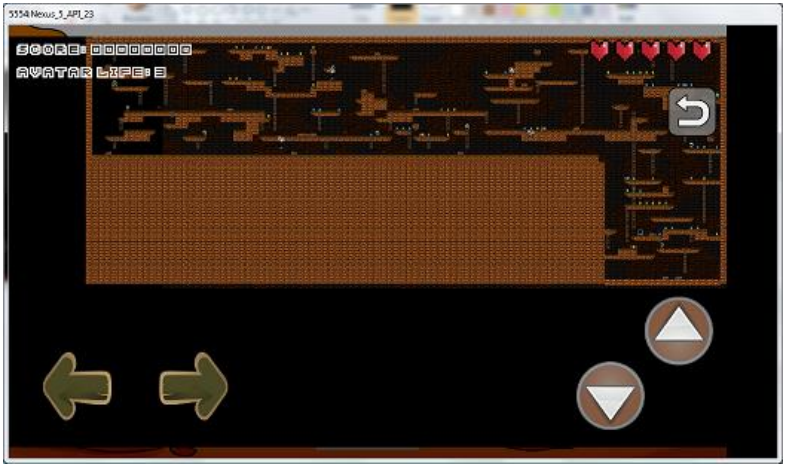
Tabel 5.4 Skenario pengujian dynamic world generator

Nomor	SP-002
Nama	*Dynamic World Generator*
Tujuan	Mengecek apakah aplikasi dapat menghasilkan *world map* yang dinamis.
Kondisi Awal	Pengguna memilih level tertentu dari menu pilihan level.
Skenario	Pada menu pilihan level, pengguna memilih salah satu level. Setelah masuk ke area permainan, pengguna mencoba memainkan level tersebut. Setelah beberapa saat, pengguna diminta menekan tombol Back untuk kembali ke menu pilihan level. Kemudian pengguna masuk kembali level yang tadi dipilih. Setelah masuk ke area permainan, pengguna diminta mencoba memainkan level tersebut untuk mengecek apakah *world map* telah berubah.
Keluaran yang Diharapkan	*World map* yang dihasilkan berbeda dibanding sebelumnya.
Kondisi Akhir	Pengguna mengkonfirmasi bahwa *world map* yang dihasilkan bisa berbeda dibanding sebelumnya.
Hasil Pengujian	Berhasil.

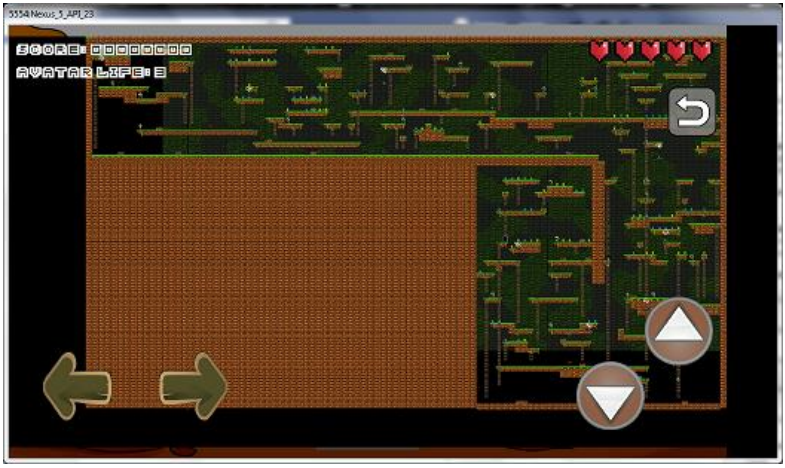
5.2.1.3 Pengujian Tingkat Kesulitan

Pengujian tingkat kesulitan digunakan untuk mengetahui apakah tingkat kesulitan dari tiap level semakin meningkat mulai dari level rendah sampai level tinggi. Skenario yang dilakukan adalah pengguna diminta untuk memainkan salah satu level dari masing-masing World. Dalam permainan ini terdapat 3 macam World sehingga pemain diharuskan memainkan 3 level. Skenario pengujian ditunjukkan pada Tabel 5.5. Pengujian dinyatakan

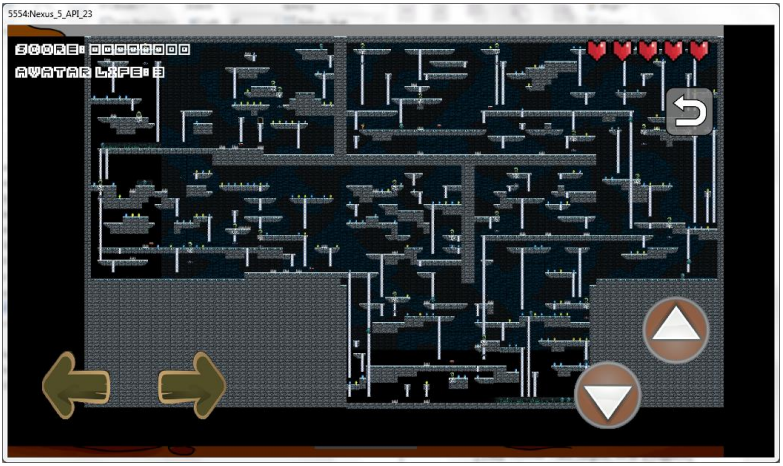
berhasil apabila faktor penentu tingkat kesulitan dari World 1 sampai World 3 nilainya semakin meningkat. Faktor yang diperhatikan adalah jumlah room dan banyaknya rintangan pada level.



Gambar 5.6 Screenshot pengujian tingkat kesulitan pada World 1



Gambar 5.7 Screenshot pengujian tingkat kesulitan pada World 2



Gambar 5.8 Screenshot pengujian tingkat kesulitan pada World 3

Tabel 5.5 Skenario pengujian tingkat kesulitan

Nomor	SP-003
Nama	Tingkat kesulitan
Tujuan	Mengecek apakah permainan memiliki tingkat kesulitan yang semakin meningkat.
Kondisi Awal	Pengguna berada pada menu pilihan level.
Skenario	Pada menu pilihan level, terdapat 3 World yang berbeda. Pada langkah awal, pengguna memainkan salah satu level pada World 1. Setelah selesai, kemudian pengguna memainkan salah satu level pada World 2. Setelah itu pengguna memainkan salah satu level pada World 3.
Keluaran yang Diharapkan	Tingkat kesulitan dari masing-masing World berbeda dan terasa semakin sulit.
Kondisi Akhir	Pengguna mengkonfirmasi bahwa tingkat kesulitan dari World 1 sampai World 3 semakin sulit.
Hasil Pengujian	Berhasil.

5.2.2 Pengujian Aplikasi terhadap Pengguna

Pengujian pada aplikasi yang dibangun tidak hanya dilakukan pada fungsionalitas yang dimiliki, tetapi juga pada pengguna untuk percobaan secara langsung. Pengujian ini berfungsi sebagai pengujian subjektif yang bertujuan untuk mengetahui tingkat keberhasilan aplikasi yang dibangun dari sisi pengguna. Hal ini dapat dicapai dengan meminta penilaian dan tanggapan dari pengguna terhadap sejumlah aspek aplikasi yang ada.

Pengujian dilakukan oleh beberapa orang yang diminta oleh penulis dan bersedia untuk melakukan pengujian. Pengguna akan diminta untuk mengoperasikan aplikasi. Setelah selesai, pengguna diminta untuk mengisi kuisioner yang telah disediakan. Kuisioner dapat dilihat pada Tabel 5.6. Tiap kolom akan memiliki bobot penilaian tersendiri dengan ketentuan sebagai berikut :

- Sangat Setuju = 4
- Setuju = 3
- Tidak Setuju = 2
- Sangat Tidak Setuju = 1

Tabel 5.6 Daftar pertanyaan kuisioner

No	Pertanyaan	Pilihan Jawaban			
		Sangat Setuju	**Setuju**	**Tidak Setuju**	**Sangat Tidak Setuju**
1	Apakah tampilan aplikasi sudah sesuai dengan tema dan				

	nyaman untuk digunakan ?				
2	Apakah kontrol dalam permainan mudah untuk digunakan ?				
3	Apakah fitur *dynamic world generator* memberi nilai hiburan yang lebih pada permainan ?				

5.3 Evaluasi

Tahap evaluasi dibagi menjadi dua bagian, yaitu evaluasi pengujian fungsionalitas dan evaluasi pengujian aplikasi terhadap pengguna. Tahap evaluasi ini berdasarkan hasil yang didapat dari proses pengujian sebelumnya.

5.3.1 Evaluasi Pengujian Fungsionalitas

Rangkuman mengenai hasil pengujian fungsionalitas dapat dilihat pada Tabel 5.7. Berdasarkan data pada tabel tersebut, dapat disimpulkan bahwa semua skenario dalam pengujian fungsionalitas telah berhasil dijalankan. Sehingga dapat disimpulkan bahwa fungsionalitas dari aplikasi telah bekerja sesuai dengan yang diharapkan.

Tabel 5.7 Rangkuman hasil pengujian fungsionalias

ID	Nama	Hasil
SP-001-01	Alur permainan (skenario 1)	Berhasil
SP-001-02	Alur permainan (skenario 2)	Berhasil
SP-002	*Dynamic World Generator*	Berhasil
SP-003	Tingkat kesulitan	Berhasil

5.3.2 Evaluasi Pengujian Aplikasi terhadap Pengguna

Rentang umur pengguna adalah 20 sampai 22 tahun. Semua pengguna berjenis kelamin laki-laki dan sedang menempuh pendidikan sebagai mahasiswa. Jawaban kuisioner dari masing-masing pengguna dapat dilihat pada Lampiran A.

Tabel 5.8 Rangkuman hasil kuisioner

No.	Pertanyaan	Penilaian				Rata-Rata
		1	2	3	4	
1	Apakah tampilan aplikasi sudah sesuai dengan tema dan nyaman untuk digunakan ?	0	0	4	1	3,2
2	Apakah kontrol dalam permainan mudah untuk digunakan ?	0	1	4	0	2.8
3	Apakah fitur dynamic world generator memberi nilai hiburan yang lebih pada permainan ?	0	0	2	3	3,6
Nilai Akhir						**3,2**

Berdasarkan tabel 5.8 , dapat diketahui bahwa nilai rata-rata adalah . Nilai rata-rata yang didapatkan sudah melebihi angka 3 yang memiliki arti “Setuju” , sehingga dapat disimpulkan bahwa aplikasi ini sudah nyaman untuk dimainkan dan fitur *dynamic world generator* sudah memberi nilai hiburan lebih kepada pengguna.

LAMPIRAN

```
private void leveldesignWorld1(int roomIndex)
{
   List<Platform> platformList =
generatedPlatforms[roomIndex];
   if (platformList == null) {
      return;
   }

   int obstacle = 2;
   int loop = 1;
   boolean findStatus;

   for (int i = 0; i < obstacle; i++) {
      findStatus = false;
      while( findStatus == false )
      {
         Platform platform =
platformList.get(rng.nextInt(platformList.size()));
         int randomPlacement = 0;
         if(platform.length <= 1)
         {
            findStatus = false;
         }
         else
         {
            randomPlacement = rng.nextInt(platform.length);
            if(!blockAt(platform.x+randomPlacement,
platform.y+1))
            {
               findStatus = true;
            }
            else
            {
               findStatus = false;
            }
         }
```

```
            if ( findStatus == true ) {
               spriteLayer.setCell(platform.x+randomPlacement,
platform.y+1, CommonTile.FLOWER.toCell(commonTileSet));
            }
         }
      }
      roomNum++;
}
```

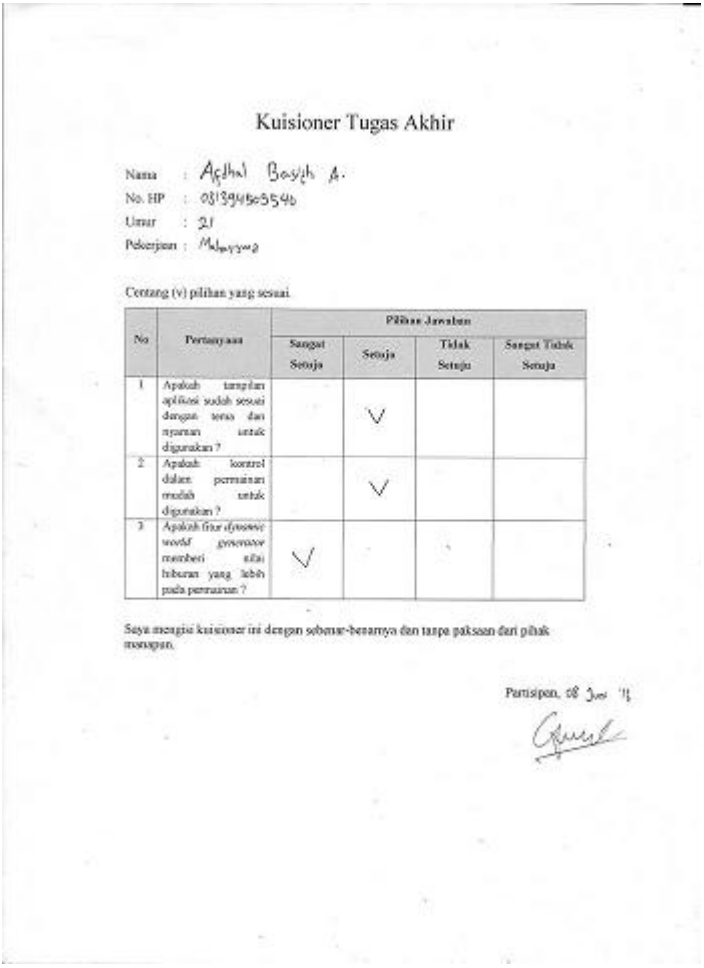
Kode Sumber A.1 Pengaturan tingkat kesulitan level 1

```
private void leveldesignWorld15(int roomIndex)
{
   List<Platform> platformList =
generatedPlatforms[roomIndex];
   if (platformList == null) {
      return;
   }

   int obstacle = 3;
   int loop = 1;
   boolean findStatus;

   for (int i = 0; i < obstacle; i++) {
      findStatus = false;
      while( findStatus == false )
      {
         Platform platform =
platformList.get(rng.nextInt(platformList.size()));
         int randomPlacement = 0;
         if(platform.length <= 1)
         {
            findStatus = false;
         }
         else{
            randomPlacement = rng.nextInt(platform.length);
            if(!blockAt(platform.x+randomPlacement,
```

```
platform.y+1))
            {
               findStatus = true;
            }
            else
            {
               findStatus = false;
            }
         }

         if ( findStatus == true )
         {
            if( i != 2 )
            {

spriteLayer.setCell(platform.x+randomPlacement,
platform.y+1, CommonTile.FLOWER.toCell(commonTileSet));
            }
            else if( i == 2 )
            {
               if(!blockAt(platform.x+randomPlacement+1,
platform.y+1)){

spriteLayer.setCell(platform.x+randomPlacement+1,
platform.y+1, CommonTile.ESKIMO.toCell(commonTileSet));
               }
               else if(!blockAt(platform.x+randomPlacement-
1, platform.y+1)){

spriteLayer.setCell(platform.x+randomPlacement-1,
platform.y+1, CommonTile.ESKIMO.toCell(commonTileSet));
               }
            }
            if( i != 2 )
            {

ladderLayer.setCell(platform.x+randomPlacement, platform.y,
CommonTile.SPIDER.toCell(commonTileSet));
```

```
spriteLayer.setCell(platform.x+randomPlacement, platform.y,
CommonTile.SPIDER.toCell(commonTileSet));
                }
            }
        }
    }
    roomNum++;
}
```

Kode Sumber A.2 Pengaturan tingkat kesulitan level 15



Kuisioner Tugas Akhir

Nama : Afdhal Basith A.
No. HP : 0813945e554b
Umur : 21
Pekerjaan : Mahasiswa

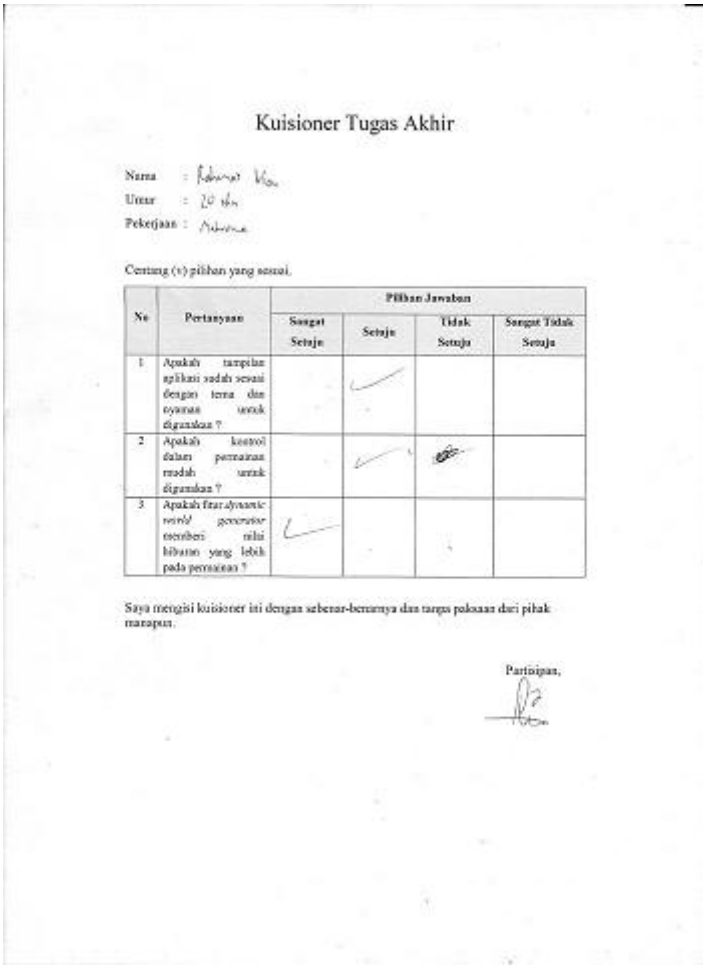
Centang (v) pilihan yang sesuai.

No	Pertanyaan	Pilihan Jawaban			
		Sangat Setuju	Setuju	Tidak Setuju	Sangat Tidak Setuju
1	Apakah tampilan aplikasi sudah sesuai dengan tema dan nyaman untuk digunakan ?		V		
2	Apakah kontrol dalam permainan mudah untuk digunakan ?		V		
3	Apakah fitur dynamic world generator memberi nilai hiburan yang lebih pada permainan ?	V			

Saya mengisi kuisioner ini dengan sebenar-benarnya dan tanpa paksaan dari pihak manapun.

Partisipan, 08 Juni '21

Gambar A.1 Kuisioner Afdhal Basith A.



Kuisioner Tugas Akhir

Nama : Rahmat Irfan
Umur : 20 thn
Pekerjaan : Mahasiswa

Centang (v) pilihan yang sesuai.

No	Pertanyaan	Pilihan Jawaban			
		Sangat Setuju	Setuju	Tidak Setuju	Sangat Tidak Setuju
1	Apakah tampilan aplikasi sudah sesuai dengan tema dan nyaman untuk digunakan ?		✓		
2	Apakah kontrol dalam permainan mudah untuk digunakan ?		✓	✓	
3	Apakah fitur *dynamic enemy generator* memberi nilai hiburan yang lebih pada permainan ?	✓			

Saya mengisi kuisioner iti dengan sebenar-benarnya dan tanpa paksaan dari pihak manapun.

Partisipan,

Gambar A.2 Kuisioner Rahmat Irfan

Kuisioner Tugas Akhir

Nama : Iskandar
Umur : 22 tahun
Pekerjaan : Pelajar

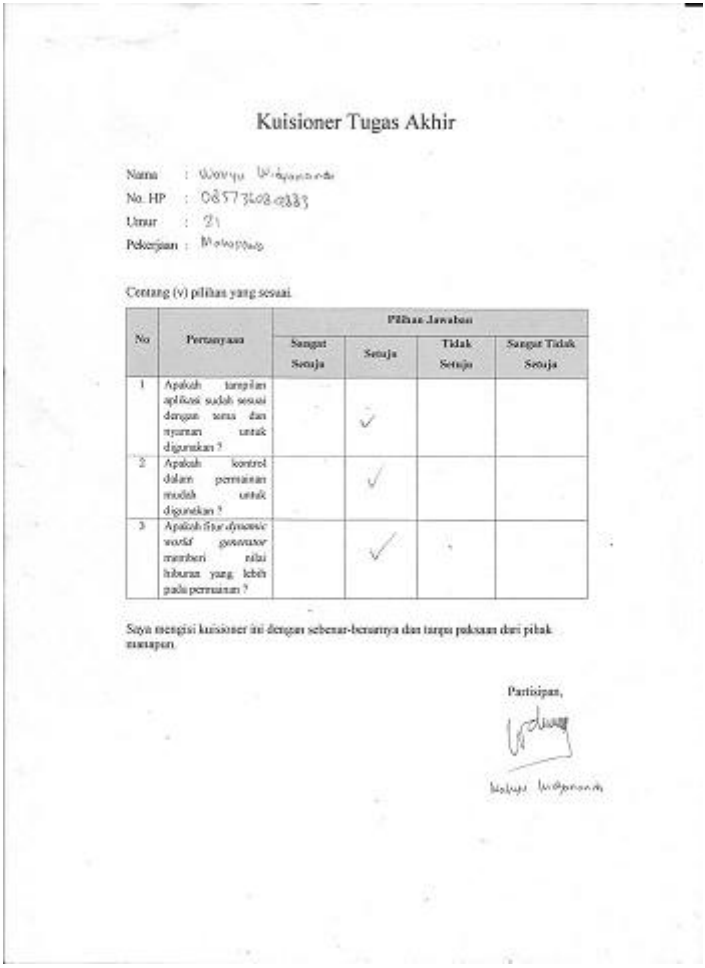
Centang (v) pilihan yang sesuai.

No	Pertanyaan	Pilihan Jawaban			
		Sangat Setuju	Setuju	Tidak Setuju	Sangat Tidak Setuju
1	Apakah tampilan aplikasi sudah sesuai dengan tema dan nyaman untuk digunakan ?		✓		
2	Apakah kontrol dalam permainan mudah untuk digunakan ?		✓		
3	Apakah fitur *dynamic world generator* memberi nilai hiburan yang lebih pada permainan ?		✓		

Saya mengisi kuisioner ini dengan sebenar-benarnya dan tanpa paksaan dari pihak manapun.

Partisipan,

Gambar A.3 Kuisioner Iskandar



Kuisioner Tugas Akhir

Nama : Wahyu Widyananda
No. HP : 0857308[illegible]
Umur : 21
Pekerjaan : Mahasiswa

Centang (v) pilihan yang sesuai.

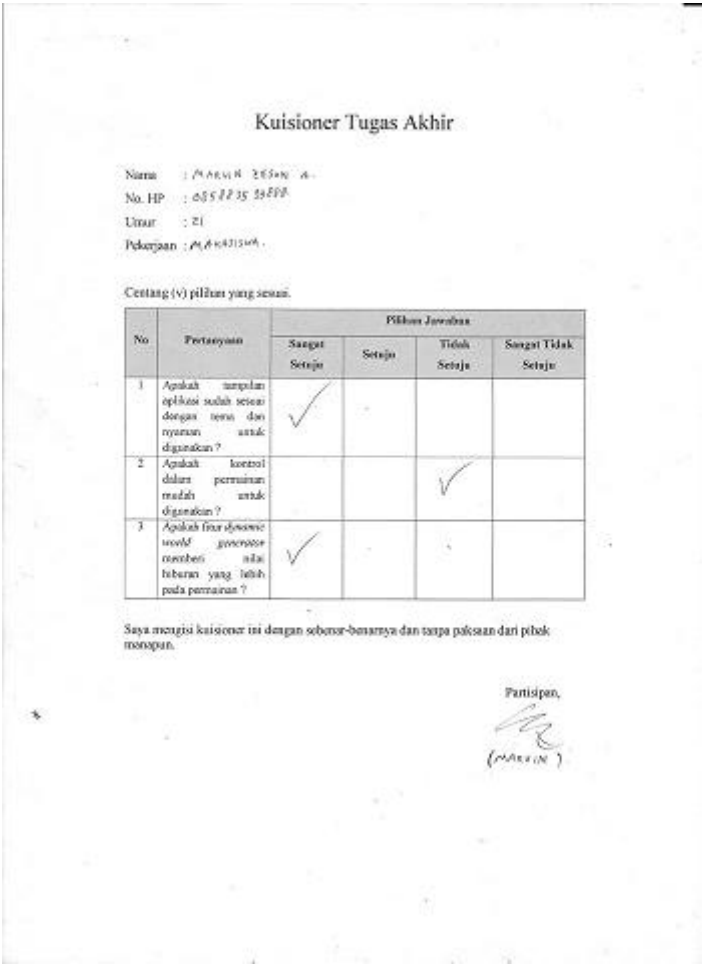
No	Pertanyaan	Pilihan Jawaban			
		Sangat Setuju	Setuju	Tidak Setuju	Sangat Tidak Setuju
1	Apakah tampilan aplikasi sudah sesuai dengan tema dan nyaman untuk digunakan ?		✓		
2	Apakah kontrol dalam permainan mudah untuk digunakan ?		✓		
3	Apakah fitur *dynamic world generator* memberi nilai hiburan yang lebih pada permainan ?		✓		

Saya mengisi kuisioner ini dengan sebenar-benarnya dan tanpa paksaan dari pihak manapun.

Partisipan,

Wahyu Widyananda

Gambar A.4 Kuisioner Wahyu Widyananda



Kuisioner Tugas Akhir

Nama : MARVIN ZESON A.
No. HP : 0852 2235 5888
Umur : 21
Pekerjaan : MAHASISWA

Centang (v) pilihan yang sesuai.

No	Pertanyaan	Pilihan Jawaban			
		Sangat Setuju	Setuju	Tidak Setuju	Sangat Tidak Setuju
1	Apakah tampilan aplikasi sudah sesuai dengan tema dan nyaman untuk digunakan ?	✓			
2	Apakah kontrol dalam permainan mudah untuk digunakan ?			✓	
3	Apakah fitur *dynamic world generator* memberi nilai keluaran yang lebih pada permainan ?	✓			

Saya mengisi kuisioner ini dengan sebenar-benarnya dan tanpa paksaan dari pihak manapun.

Partisipan,

(MARVIN)

Gambar A.5 Kuisioner Marvin Zeson A.

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini dijelaskan mengenai kesimpulan yang dapat diambil dari hasil pengujian yang telah dilakukan sebagai jawaban dari rumusan masalah yang dikemukakan. Selain kesimpulan, saran juga diberikan dengan tujuan agar pengembangan aplikasi kedepannya bisa lebih baik.

6.1 Kesimpulan

Dalam proses pengerjaan Tugas Akhir ini, mulai dari tahap analisis, perancangan, dan implementasi, hingga pengujian didapatkan kesimpulan sebagai berikut :

1. Dengan menggunakan *framework* LibGDX, kita bisa membuat permainan ber-*genre 2D Platformer* dan mengimplementasikan fitur *dynamic world generator* pada permainan tersebut.
2. Algoritma *Recursive Backtracking* sudah terbukti mampu menghasilkan *world map* yang dinamis. Selain itu, bisa dipastikan bahwa setiap level pada permainan ini memungkinkan untuk diselesaikan. Tingkat kesulitan yang berbeda-beda juga telah diimplementasikan pada setiap level.
3. Berdasarkan hasil pengujian pada pengguna, diketahui bahwa tampilan permainan sudah sesuai dengan tema dan antarmuka permainan terasa mudah ketika digunakan.

6.2 Saran

Berikut ini adalah beberapa saran untuk pengembangan aplikasi di masa yang akan datang berdasarkan pada hasil perancangan, implementasi, dan uji coba yang telah dilakukan :

1. Sebaiknya tampilan grafis dan animasi pada permainan dibuat lebih bagus agar menambah daya tarik dari permainan.

2. Aturan main atau fitur dari permainan perlu ditambah untuk memberikan hiburan lebih banyak kepada pengguna, misalnya seperti menambahkan fitur RPG.
3. Menambahkan alur cerita ke dalam permainan sehingga pengguna merasa lebih terhibur ketika memainkan.

DAFTAR PUSTAKA

[1] J. Edmonds, "Recursive Backtracking," in *How to Think About Algorithms*, Cambridge, Cambridge University Press, 2008, pp. 251-266.

[2] libGDX, "libGDX," [Online]. Available: https://libgdx.badlogicgames.com/. [Diakses 15 Maret 2016].

[3] T. Lindeijer, "Tiled Map Editor," [Online]. Available: http://www.mapeditor.org/. [Diakses 20 Maret 2016].

[4] H. M. Pandey, "Recursive Backtracking," in *Design and Analysis Algorithm*, New Delhi, University Science Press, 2008, pp. 185-187.

[5] "Android Studio," Android Studio, [Online]. Available: https://developer.android.com/studio/index.html. [Diakses 15 Maret 2016].

[6] "What is a Platform Video Game?," [Online]. Available: http://compactiongames.about.com/od/gameindex/a/platformer_def.htm. [Diakses 29 Juni 2016].

[7] G. Smith, M. Cha and J. Whitehead, "A Framework for Analysis of 2D Platformer Levels," in *Sandbox '08: Proceedings of the 2008 ACM SIGGRAPH symposium on video games*, New York, 2008.

BIODATA PENULIS

Faishal Azka Jellyanto, lahir pada tanggal 18 September 1993 di Blitar, Jawa Timur. Hobi yang dimiliki adalah membaca *manga*, menonton *anime*, menggambar atau desain, dan bermain *game*. Penulis menempuh pendidikan mulai dari SDI Kardina Massa Blitar (2000-2006), SMP Negeri 1 Blitar (2006-2009), SMA Negeri 1 Blitar (2009-2012), dan Teknik Informatika ITS (2012-2016). Di jurusan Teknik Informatika ITS, penulis mengambil bidang minat Interaksi Grafika dan Seni (IGS) dan memiliki ketertarikan dalam bidang *game*, animasi 3D, dan perangkat bergerak. Selama perkuliahan, penulis cukup aktif dalam kegiatan kemahasiswaan, antara lain Staf Departemen Media Informasi Himpunan Mahasiswa Teknik Computer-Informatika ITS, Staf Divisi Keamanan dan Perijinan Schematics 2013, dan Staf Divisi Web & 3D.

Kritik dan saran sangat diharapkan guna meningkatkan kualitas penulisan selanjutnya. Untuk itu, silahkan mengirim email ke : **azukineru@gmail.com**