Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR – KI1502

Implementasi Artificial Ant Colony Algorithm (AACA) pada Pembangkit Labirin Dinamis untuk Game 2D berbasis Android 'Ant Cave'

Andi Akram Yusuf
NRP. 5113 100 173

Dosen Pembimbing 1
Imam Kuswardayan, S.Kom., MT.

Dosen Pembimbing 2
Wijayanti Nurul Khotimah, S.Kom., M.Sc.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

TUGAS AKHIR – KI1502

Implementasi Artificial Ant Colony Algorithm (AACA) pada Pembangkit Labirin Dinamis untuk Game 2D berbasis Android 'Ant Cave'

Andi Akram Yusuf
NRP. 5113 100 173

Dosen Pembimbing 1
Imam Kuswardayan, S.Kom., MT.

Dosen Pembimbing 2
Wijayanti Nurul Khotimah, S.Kom., M.Sc.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

(Halaman ini sengaja dikosongkan)

FINAL PROJECT – KI1502

ARTIFICIAL ANT COLONY ALGORITHM (AACA) IMPLEMENTATION IN DYNAMIC LABYRINTH GENERATOR FOR 2D GAME ANDROID BASE 'ANT CAVE'

Andi Akram Yusuf
NRP. 5113 100 173

Supervisor 1
Imam Kuswardayan, S.Kom., MT.

Supervisor 2
Wijayanti Nurul Khotimah, S.Kom., M.Sc.

DEPARTMENT OF INFORMATICS
Faculty of Information Technology
Sepuluh Nopember Institute of Technology
Surabaya 2017

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

Implementasi Artificial Ant Colony Algorithm (AACA) pada Pembangkit Labirin Dinamis untuk Game 2D berbasis Android 'Ant Cave'

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:
Andi Akram Yusuf
NRP. 5113100173

Disetujui oleh Pembimbing Tugas Akhir:

1. Imam Kuswardayan, S.Kom., M.T. (Pembimbing 1)
(NIP. *197612152003121001*)

2. Wijayanti Nurul K., S.Kom., M.Sc. (Pembimbing 2)
(NIP. *198603122012122004*)

SURABAYA
Juni, 2017

(Halaman ini sengaja dikosongkan)

Implementasi Artificial Ant Colony Algorithm (AACA) pada Pembangkit Labirin Dinamis untuk Game 2D berbasis Android 'Ant Cave'

Nama : **Andi Akram Yusuf**
NRP : **5113100173**
Jurusan : **Teknik Informatika**
Fakultas Teknologi Informasi ITS
Dosen Pembimbing I : **Imam Kuswardayan, S.Kom., MT.**
Dosen Pembimbing II : **Wijayanti Nurul Khotimah, S.Kom., M.Sc.**

ABSTRAK

Bermain game adalah kegiatan yang menyenangkan. Seringkali seseorang bahkan memainkan game yang sama secara berulang-ulang dalam waktu yang singkat. Bermain game secara berulang dab dalam waktu yang panjang dapat mengakibatkan kebosanan, terlebih lagi bila game tersebut memiliki rancangan peta atau rancangan jalan yang sama setiap tingkatnya.

Salah satu cara untuk menciptakan peta yang berbeda adalah dengan menggunakan Pembangkit Labirin Otomatis. Pembangkit Labirin Otomatis adalah membuat labirin secara otomatis. Ada banyak algoritma yang dipakai pada Pembangkit Labirin Otomatis yang sudah ada.

Tujuan dari pengerjaan Tugas Akhir ini adalah dapat menghasilkan game yang memiliki peta yang dinamis. Yang dimaksud dengan dinamis adalah peta yang berbeda untuk tiap tingkatnya. Untuk itu penulis mencoba menggunakan Artificial Ant Colony Algorithm untuk menciptakan peta yang dinamis pada game 'Ant Cave'. Dengan pengujian fungsionalitas dapat

disimpulkan aplikasi telah mengimplementasikan Artificial Ant Colony Algorithm dan dapat membangkitkan labirin yang berbeda-beda setiap levelnya.

Kata kunci: Game, Pembangkit Labirin Dinamis, Unity, Ant Colony Algorithm.

ARTIFICIAL ANT COLONY ALGORITHM (AACA) IMPLEMENTATION IN DYNAMIC LABYRINTH GENERATOR FOR 2D GAME ANDROID BASE 'ANT CAVE'

Name : **Andi Akram Yusuf**
NRP : **5113100173**
Department : **Department of Informatics**
Faculty of Information Technology ITS
Supervisor I : **Imam Kuswardayan, S.Kom., MT.**
Supervisor II : **Wijayanti Nurul Khotimah, S.Kom., M.Sc.**

ABSTRACT

Playing games is a fun activity. Often a person even plays the same game over and over again in a short time. Playing games repeatedly and over a long period of time can lead to boredom, especially if the game has the same map design or road plan at every level.

One way to create different maps is to use the Dynamic Labyrinth Generator. Automatic Labyrinth Generator is making the maze automatically. There are many algorithms used in existing Dynamic Labyrinth Generators.

The purpose of this final work is to produce games that have a dynamic map. What is meant by dynamic is a different map for each level. For that the author tried to use Artificial Ant Colony Algorithm to create a dynamic map in game 'Ant Cave'. With the functionality testing it can be concluded the application has

implemented Artificial Ant Colony Algorithm and can generate different labyrinth each level.

Key words: Game, Dynamic Labyrinth Generator, Unity, Ant Colony Algorithm.

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Allah SWT karena atas segala karunia dan rahmat-Nya penulis dapat menyelesaikan tugas akhir yang berjudul "Implementasi Artificial Ant Colony Algorithm (AACA) pada Pembangkit Labirin Dinamis untuk Game 2D berbasis Android 'Ant Cave'"

Pengerjaan tugas akhir ini penulis lakukan untuk memenuhi salah satu syarat memperoleh gelar Sarjana Komputer di Program Studi S-1 Jurusan Teknik Informatika Fakultas Teknologi Informasi Institut Teknologi Sepuluh Nopember.

Penulis mengucapkan terima kasih kepada semua pihak yang telah memberikan dukungan baik secara langsung maupun tidak langsung selama proses pengerjaan tugas akhir ini hingga selesai, antara lain:

1. Allah SWT atas segala karunia dan rahmat-Nya yang telah diberikan selama ini.
2. Keluarga penulis, Bapak Abi Zamahsyari, Ibu Andi Tuti Muhammad, kakak Andi Ainul Mardiyah, adik Andi Amaliyah Maryamah, dan juga keluarga yang tidak dapat penulis sebutkan satu per satu yang telah memberi dukungan moral dan material serta doa untuk penulis.
3. Bapak Imam Kuswardayan S.Kom., MT. selaku dosen pembimbing I yang telah memberikan bimbingan dan arahan dalam pengerjaan tugas akhir ini.
4. Ibu Wijayanti Nurul Khotimah S.Kom., M.Sc. selaku dosen pembimbing II yang telah memberikan bimbingan dan arahan dalam pengerjaan tugas akhir ini.
5. Teman-teman angkatan 2013 yang memiliki dosen pembimbing yang sama yang telah memberikan dukungan selama saya menyelesaikan Tugas Akhir ini.

6. Seluruh pihak yang tidak bisa saya sebutkan satu persatu yang telah memberikan dukungan selama saya menyelesaikan Tugas Akhir ini.

Penulis menyadarai bahwa masih terdapat banyak kekurangan dalam tugas akhir ini. Oleh karena itu, penulis menerima dengan rendah hati kritik dan saran untuk pembelajaran dan perbaikan ke depannya. Semoga tugas akhir ini dapat memberikan manfaat yang sebaik-baiknya.

Surabaya, Juni 2017

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

(Halaman ini sengaja dikosongkan)

DAFTAR KODE SUMBER

(Halaman ini sengaja dikosongkan)

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

Bermain *game* adalah kegiatan yang mengasyikkan. Karena itu bermain *game* bisa menghilangkan rasa bosan. Tentunya *game* yang bisa dimainkan dengan cepat, mudah dan tidak membosankan membutuhkan banyak variasi yang dapat dijelajahi lebih jauh. Variasi tersebut bisa berupa *maps*, *item*, musuh, atau hal lainnya. Misalnya setiap kali pemain memasuki tingkat yang sama, *map* yang ditampilkan selalu berbeda, atau adanya *item* langka yang hanya bisa didapatkan apabila kita telah menyelesaikan misi atau syarat tertentu.

Salah satu variasi tersebut, yaitu kebutuhan map yang bervariasi dapat dipenuhi dengan mengimplementasikan pembangkit labirin dinamis dengan menggunakan beberapa algoritma, beberapa diantanya ialah Algoritma Kruskal dan Algoritma Prim[1]. Pada Tugas Akhir ini penulis mencoba menggunakan Artificial Ant Colony Algorithm yang dibahas oleh Dawid Polap untuk game ‘Ant Cave’ [2]. Penggunaan algoritma ini dapat menghasilkan labirin yang bervariasi, sehingga dianggap sesuai untuk diterapkan sebagai Pembangkit Labirin yang bersifat dinamis.

Tujuan dari pengerjaan Tugas Akhir ini adalah mampu menghasilkan aplikasi permainan yang setiap levelnya memiliki labirin yang tidak sama setiap levelnya. Sehingga bermain *game* menjadi lebih menyenangkan.

1.2 Rumusan Masalah

Perumusan masalah yang terdapat pada tugas akhir ini, antara lain:

1. Apakah penggunaan Artificial Ant Colony Algorithm dapat membangkitkan labirin yang berbeda-beda (dinamis) untuk setiap kali pembangkitan dan dengan waktu yang optimal (5 detik) untuk *game* 2D berbasis Android?
2. Bagaimana skenario permainan untuk *game* 2D berbasis Android 'Ant Cave'?
3. Bagaimana aturan permainan untuk *game* 2D berbasis Android 'Ant Cave'?

1.3 Batasan Masalah

Batasan masalah yang terdapat pada tugas akhir ini, yaitu sebagai berikut:

1. Bahasa pemrograman yang akan digunakan adalah bahasa pemrograman C#.
2. Pembangunan aplikasi menggunakan Unity.
3. Level yang diimplementasikan hanya 10 level.

1.4 Tujuan

Tujuan dari pembuatan tugas akhir ini, antara lain:

1. Membuat *game* dengan menggunakan Artificial Ant Colony Algorithm sebagai algoritma untuk Pembangkit Labirin Dinamis.
2. Membuat *game* yang memiliki peta yang berbeda-beda untuk setiap levelnya.

1.5 Manfaat

Tugas Akhir ini diharapkan dapat menciptakan *game* 2D berbasis Android yang memiliki map yang dinamis untuk setiap level dengan menggunakan Artificial Ant Colony Algorithm.

1.6 Metodologi

1. Penyusunan proposal tugas akhir
   Tahap pertama dalam proses pengerjaan tugas akhir ini adalah menyusun proposal tugas akhir. Pada proposal tugas akhir ini diajukan implementasi Artificial Ant Colony Algorithm (AACA) pada Pembangkit Labirin Dinamis untuk Game 2D berbasis Android 'Ant Cave'.

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 desain perangkat lunak
   Perangkat lunak yang akan dibangun merupakan aplikasi untuk perangkat bergerak. Akan ada dua macam tampilan dari aplikasi ini. Tampilan untuk menu dan tampilan permainan.

4. Pengembangan perangkat lunak
   Pembangunan *game* akan dilakukan dengan menggunakan bahasa pemrograman C#, Game Engine Unity, Text Editor MonoDevelop, dan Perangkat Android.

5. Pengujian dan evaluasi
   Pengujian yang akan dilakukan adalah pengujian fungsionalitas. Pengujian ini bertujuan untuk memeriksa apakah *game* sudah terimplementasikan atau belum. Kemudian akan dilakukan evaluasi untuk memeriksa hasil implementasi *game*.

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. Sistematika penulisan buku tugas akhir ini secara garis besar antara lain:
   1. Pendahuluan
      a. Latar Belakang
      b. Rumusan Masalah
      c. Batasan Tugas Akhir
      d. Tujuan
      e. manfaat
      f. Metodologi
      g. Sistematika Penulisan
   2. Tinjauan Pustaka
   3. Desain dan Implementasi
   4. Pengujian dan Evaluasi
   5. Kesimpulan dan Saran
   6. Daftar Pustaka

1.7 Sistematika Penulisan

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 SISTEM

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. PENGUJIAN 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 perangkat lunak 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.

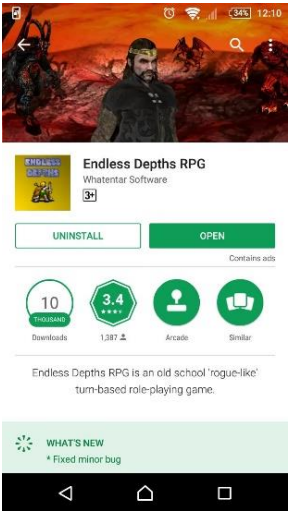
2.1 Permainan Serupa

Dalam pembutan *game* 'Ant Cave' ada 2 *game* yang menjadi inspirasi yaitu Endless Depths dan The Greedy Cave. Kedua permainan ini merupakan permainan dengan tema yang serupa yaitu RPG. Permainan ini memiliki beberapa kesamaan yaitu bertujuan menjelajahi seluruh level yang ada. Levelnya berupa labirin-labirin di mana pemain harus mencari jalan keluar yang berupa tangga.

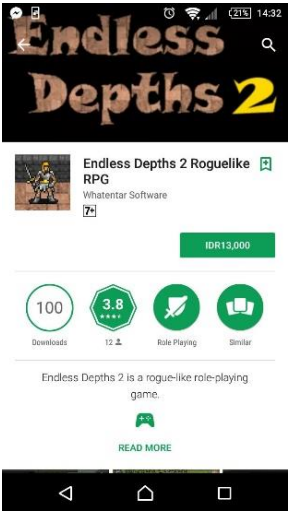
2.1.1 Endless Depths

Endless Depths adalah *game* yang dikembangkan oleh Whatentar Software. *Game* ini adalah *game* lama yang sudah tidak ada lagi di Google Playstore, namun Whatentar Software sudah mengembangkan *game* ini, hasilnya menjadi Endless Depths 2 yang ada di Google Playstore [3]. Untuk yang belum pernah memainkan Endless Depths ini anda sudah tidak bisa lagi mengunduh *game* ini, namun dapat langsung mencoba Endless Depths 2. Terakhir kali Endless Depths mengalami *update* ialah pada tanggal 5 Juni 2014. Untuk Endless Dephts 2 anda harus membeli *game* ini di Google Playstore seharga 13.000 rupiah dan terakhir kali *update* dilakukan pada tanggal 5 Januari 2016. Endless Depths pada Google Playstore dapat dilihat pada Gambar 2.1.1,

sedangkan untuk Endless Depths 2 dapat dilihat pada Gambar 2.1.2.



Gambar 2. 1 Endless Depths pada Google Playstore



Gambar 2. 2 Endless Depths 2 pada Google Playstore

2.1.2 The Greedy Cave

The Greedy Cave adalah *game* yang dikembangkan oleh Avalon-Games. Orang yang menginstal *game* ini sebanyak 1.000.000–5.000.000 orang. *Rating*-nya mencapai 4.2 di Google Playstore [4].

Gambar 2. 3 The Greedy Cave pada Google Playstore

Perbedaan mendasar *game* ini dengan Endless Depth ada pada tampilan *game* yang jauh lebih modern dan juga aturan permainan pada saat pemain bertarung dengan musuh. Cara bertarungnya ialah pemain dan musuh akan menyerang secara otomatis akan tetapi pemain dapat kabur dari pertarungan. Cara bertarung inilah yang menjadi inspirasi cara bertarung pada *game* 'Ant Cave'. The Greedy Cave juga memiliki *item* yang sangat bervariasi, contohnya *game* ini memiliki item unik yang jika dipakai dengan setelan yang sama akan memberikan tambahan untuk atribut khusus.

Gambar 2. 4 Tampilan permainan The Greedy Cave

2.2 Artifical Ant Colony Algorithm

Artificial Ant Colony Algorithm memetakan tingkah laku semut ketika mencari sumber makanan. Semut bergerak secara *random* meninggalkan feromon. Feromon yang tersisa menciptakan jejak untuk mencari sumber makanan. Apabila semut menemukan sumber makanan, maka semut akan meninggalkan feromon yang lebih besar agar semut lain dapat menemukan sumber makanan yang sama [2].

$$f^{t+1}(x_i, x_j) = (1-\rho)f^{t}(x_i, x_j) + \Gamma_i^t \qquad (1)$$

Awalnya feromon di mana-mana bernilai sama. Kemudian setelah memulai iterasi baru feromon di-*update* dengan rumus (1) dimana t adalah iterasi, x_i, x_j adalah kota pada bidang kartesius, f adalah feromon, ρ adalah laju evaporasi, dan Γ_i^t adalah nilai feromon yang dilepaskan oleh semut yang dapat dihitung dengan rumus (2) di mana n adalah jumlah semut

dan L_{ij} adalah jarak yang ditempuh oleh semut dari kota sebelumnya.

$$\Gamma_i^t = \sum_{1}^{n} \frac{1}{L_{ij}^t} \tag{2}$$

Untuk menghitung jarak yang ditempuh oleh semut dapat menggunakan rumus (3) dimana x_i dan x_j adalah titik pada bidang kartesius, k adalah nilai x dan nilai y pada bidang kartesius.

$$L_{ij} = \|x_i - x_j\| = \sqrt{\sum_{k=1}^{2} (x_{i,k} - x_{j,k})^2} \tag{3}$$

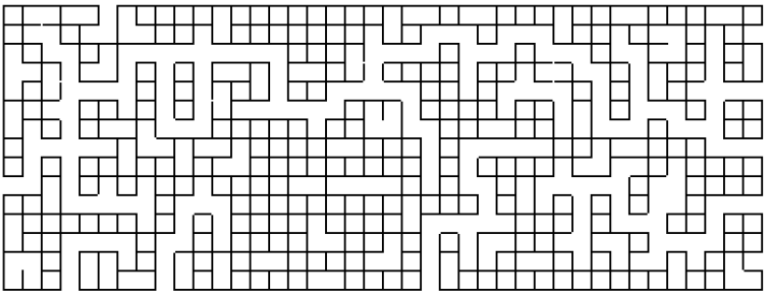
Pada setiap iterasi setiap semut akan memilih kota yang akan dituju berikutnya. Kota tersebut adalah kota dengan nilai probabilitas tertinggi. Probabilitas setiap kota dapat dihitung dengan rumus (4) dimana $\sum_{\alpha \in N_i^k}$ merupakan daftar kota yang belum dikunjungi oleh semut k tapi menuju ke i, sedangkan α dan β adalah koefisien dan f adalah feromon.

$$p^t(x_i, x_j) = \frac{\left[f^t(x_i, x_j)\right]^\alpha \left[\frac{1}{L_{ij}^t}\right]^\beta}{\sum_{\alpha \in N_i^k} \left(\left[f^t(x_i, x_j)\right]^\alpha \left[\frac{1}{L_{ij}^t}\right]^\beta \right)} \tag{4}$$

Untuk jumlah semut yang digunakan haruslah memenuhi syarat pada rumus (5) dimana n adalah jumlah

semut dan S adalah ukuran peta. Contoh gambar menurut Dawid Polap pada papernya dapat dilihat pada Gambar 2. 5 [2].

$$n^2 < S \tag{5}$$



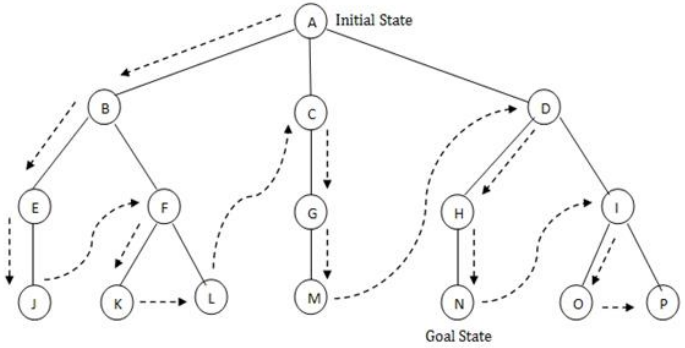
Gambar 2. 5 Contoh labirin 15 x 40 Hasil Ant Colony

2.3 Depth First Search

Depth First Search adalah algoritma yang pada dasarnya digunakan untuk melakukan pencarian. Umumnya berbentuk *tree*. Sesuai namanya *depth first*, yaitu mengutamakan kedalaman dari *tree* yang berarti algoritma ini akan mencari hingga ke bawah terlebih dahulu baru kemudian ke samping.

Penjelasan sederhana dari Gambar 2. 6 adalah algoritma ini akan memeriksa *state* pertama yang ditemukannya, yaitu A, setelah itu algoritma ini akan menuju *state* pertama yang berada di bawah dari *state* A, yaitu B. Hingga pada *state* J algoritma ini tidak menemukan *state* lain dan belum menemukan apa yang ia cari, yaitu N. Maka algoritma ini akan kembali ke *state* sebelumnya lalu mengecek apakah ada *state* ini memiliki percabangan lain. Pada *state* B algoritma ini menemukan percabangan lain, yaitu *state* F, maka algoritma ini akan menuju ke *state* F dan melanjutkan

pencarian. Algoritma ini akan terus mengulangi langkah-langkah ini hingga menemukan apa yang dicari.



Gambar 2. 6 Contoh sederhana Algoritma DFS

2.4 Unity

Unity adalah sebuah *game engine* yang dikembangkan oleh Unity Technologies. Beberapa *platform* yang didukung oleh Unity adalah : Android, Apple TV, BlackBerry 10, iOS, Linux, Nintendo 3DS line, OS X, PlayStation 4, PlayStation Vita, Unity Web Player (termasuk Facebook), Wii, Wii U, Windows Phone 9, Windows, Xbox 360, dan Xbox One [5].

2.5 Bahasa Pemrograman C#

C# merupakan sebuah bahasa pemrograman yang berorientasi objek yang dikembangkan oleh Microsoft sebagai bagian dari inisiatif kerangka .NET Framework. Bahasa pemrograman ini dibuat berbasiskan bahasa C++ yang telah dipengaruhi oleh aspek-aspek ataupun fitur bahasa yang terdapat pada bahasa-bahasa pemrograman lainnya seperti Java,

Delphi, Visual Basic, dan lain-lain dengan beberapa penyederhanaan [6].

BAB III
PERANCANGAN PERANGKAT LUNAK

Bab ini membahas mengenai perancangan dan pembuatan perangkat lunak. Perangkat lunak yang dibuat pada tugas akhir ini adalah *game* dengan peta yang berbeda-beda setiap levelnya.

3.1 Analisis Sistem

Bermain *game* adalah kegiatan yang menyenangkan. Namun jika dilakukan secara berulang-ulang maka akan timbul rasa bosan. Terlebih lagi bila *game* tersebut memiliki jalan yang sama untuk setiap levelnya.

Game ini dibangun agar pemain dapat merasakan jalan yang berbeda setiap kali memainkan *game* ini. *Game* ini juga dibangun agar pemain tidak bosan dalam menjelajahi setiap levelnya karena setiap levelnya jalan labirin yang terbentuk akan berbeda-beda. Untuk desain level akan dijelaskan lebih lanjut nantinya.

Dalam membangun *game* ini penulis menggunakan aplikasi Unity versi 5.4.2f2. Untuk grafis dan animasi penulis akan mengambil grafis yang ada di internet dan bebas digunakan. Aplikasi ini diharapkan dapat berjalan di Android agar dapat dimainkan dengan mudah, di mana saja dan kapan saja.

3.2 Perancangan Permainan Ant Cave

Pada subbab ini akan dibahas skenario permainan dan aturan main pada *game* Ant Cave.

3.2.1 Skenario Permainan Ant Cave

Game ini adalah *game* 2D dengan tema petualangan. *Game* ini dimainkan oleh satu orang saja. Pada setiap level target dari *game* ini adalah menemukan tangga turun tanpa kehilangan darah

yang berupa *health point*. Pada setiap level juga terdapat musuh yang juga memiliki darah berupa *health point*. Pemain dapat melewati musuh tanpa bertarung dengan musuh tersebut untuk menghindari berkurangnya *health point* yang tidak perlu. Untuk melewati jalan tertentu pemain perlu menghancurkan musuh dengan cara bertarung terlebih dahulu. Pemain dan musuh akan bertarung secara otomatis, tetapi pemain dapat memilih untuk kabur dari pertarungan. Akan ada 10 level di mana setiap level akan ada tangga untuk turun ke level selanjutnya. Letak awal pemain, musuh dan tangga turun akan diacak pada setiap levelnya.

Pengaturan musuh juga diatur pada tahap ini. Ada 3 jenis musuh, yaitu musuh lemah, musuh sedang, dan musuh kuat. Rancangan musuh dapat dilihat pada Tabel 3. 1.

Tabel 3. 1 Rancangan level

Level	Musuh Lemah	Musuh Sedang	Musuh Kuat
1	1	0	0
2	2	0	0
3	3	0	0
4	1	1	0
5	1	2	0
6	0	3	0
7	1	0	1
8	1	1	1
9	0	2	1
10	1	2	2

3.2.2 Aturan Permainan Ant Cave

Game ini memiliki beberapa aturan main, yaitu:

1. Pada setiap level akan ada musuh yang muncul.

2. Pemain dapat menggerakkan karakter dengan cara menggeser layar.
3. Pemain dapat menyerang musuh dengan cara menggerakkan pemain ke tempat musuh berada.
4. Pemain dapat memilih untuk mengabaikan musuh.
5. Pemain dan musuh akan memiliki atribut *health point*, *speed*, *attack,* dan *defence.*
6. *Health point* adalah darah dari pemain maupun karakter.
7. Apabila musuh kehilangan seluruh *health*-nya maka musuh akan hancur.
8. Apabila pemain kehilangan seluruh *health*-nya maka permainan akan kalah.
9. Atribut *speed* akan menentukan karakter yang akan menyerang terlebih dahulu, karakter yang memiliki *speed* yang lebih tinggi akan menyerang pertama kali.
10. Atribut *attack* dan *defence* akan mempengaruhi berapakah *damage* yang akan dihasilkan, minimal nilai *damage* adalah 1.
11. Rancangan atribut pemain dan musuh dapat dilihat pada Tabel 3. 2.
12. Pemain dan musuh akan melakukan serangan secara otomatis sesuai dengan atribut dari masing-masing karakter.
13. Pemain dapat memilih untuk kabur ketika bertarung, namun pemain dapat kabur setelah selesai musuh dan pemain menyerang atau satu giliran.
14. Pemain harus menemukan tangga turun agar dapat masuk ke level selanjutnya.
15. Letak awal pemain, musuh dan tangga turun akan diacak.

Tabel 3. 2 Rancangan atribut

	Pemain	Musuh Lemah	Musuh Sedang	Musuh Kuat
Health Point	10	2	3	3
Speed	2	1	2	3
Attack	1	1	1	2
Defence	1	1	2	1

3.3 Perancangan Peta Labirin Ant Cave

Pada subbab ini akan dibahas rancangan labirin dan dua algoritma. Algoritma pertama adalah Artificial Ant Colony Algorithm. Algoritma ini digunakan untuk membangun labirin yang akan digunakan pada setiap levelnya. Algoritma kedua adalah Depth First Search (DFS). Algoritma ini akan digunakan untuk menentukan apakah labrin tersebut sudah terbentuk atau belum.

3.3.1 Rancangan Labirin

Labrin yang akan diperlukan oleh *game* Ant Cave adalah labirin yang dapat digambarkan pada bidang kartesius, dapat dilihat pada Gambar 3. 1. Setiap satu nilai x dan y akan melambangkan satu kota yang dapat dilalui oleh semut. Setiap kota ini akan memiliki nilai feromon tersendiri. Apabila nilai feromon melebihi batas maka akan dianggap sebagai jalan yang dapat dilalui. Contoh hasil labirin dapat dilihat pada Gambar 3. 2. Rancangan ukuran peta pada setiap levelnya dapat dilihat pada Tabel 3. 3.

Syarat-syarat terbentuknya labirin ialah:

1. Adanya kota yang dapat dilalui yang terhubung dengan jarak 1 atau berada di keempat sisinya.

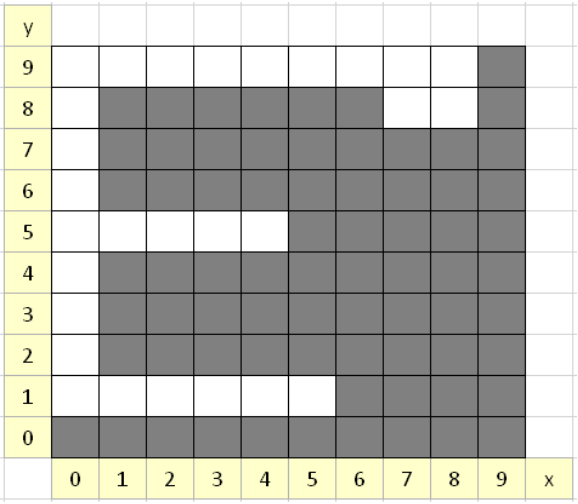
2. Kota yang dapat dilalui adalah kota yang nilai feromonnya melebih batas, yaitu 0.3.
3. Jumlah kota yang saling terhubung minimal 30% dari total kota yang ada dan maksimal 75% dari total kota yang ada.
4. Letak pintu masuk dan pintu keluar akan diacak.

y											
9	0,9	1,9	2,9	3,9	4,9	5,9	6,9	7,9	8,9	9,9	
8	0,8	1,8	2,8	3,8	4,8	5,8	6,8	7,8	8,8	9,8	
7	0,7	1,7	2,7	3,7	4,7	5,7	6,7	7,7	8,7	9,7	
6	0,6	1,6	2,6	3,6	4,6	5,6	6,6	7,6	8,6	9,6	
5	0,5	1,5	2,5	3,5	4,5	5,5	6,5	7,5	8,5	9,5	
4	0,4	1,4	2,4	3,4	4,4	5,4	6,4	7,4	8,4	9,4	
3	0,3	1,3	2,3	3,3	4,3	5,3	6,3	7,3	8,3	9,3	
2	0,2	1,2	2,2	3,2	4,2	5,2	6,2	7,2	8,2	9,2	
1	0,1	1,1	2,1	3,1	4,1	5,1	6,1	7,1	8,1	9,1	
0	0,0	1,0	2,0	3,0	4,0	5,0	6,0	7,0	8,0	9,0	
	0	1	2	3	4	5	6	7	8	9	x

Gambar 3. 1 Bidang Kartesius

Tabel 3. 3 Ukuran peta berdasarkan level

Level	Ukuran Peta
1	10 x 10
2	10 x 10
3	10 x 10
4	11 x 11
5	11 x 11
6	11 x 11
7	12 x 12
8	12 x 12
9	12 x 12
10	12 x 12



Gambar 3. 2 Contoh hasil yang diinginkan

3.3.2 Artificial Ant Colony Algorithm

Game ini menggunakan Artificial Ant Colony Algorithm untuk membangkitkan labirin yang digunakan untuk stiap levelnya. Labirin yang dibangkitkan ini adalah labirin dinamis yang berarti labirin yang berbeda-beda pada setiap levelnya. Rancangan algoritma ini dapat dilihat pada Gambar 3. 3.

3.3.2.1 Inisiasi

Tahap inisiasi ini adalah tahap untuk menentukan ukuran peta (kolom x baris), jumlah semut, nilai evaporasi, koefisien alpha, koefisien beta, batas minimal feromon sebagai jalan, nilai awal feromon. Nilai-nilai yang akan digunakan oleh penulis ialah:

1. Kolom dan baris dapat dilihat pada Tabel 3. 3.
2. Jumlah semut dilihat pada Tabel 3. 4.
3. Nilai evaporasi ialah 0.1.
4. Koefisien alpha ialah 0.3.

5. Koefisien beta ialah 0.5.
6. Batas minimal feromon ialah 0.3.
7. Nilai awal feromon ialah 1.

Tabel 3. 4 Rancangan jumlah Semut

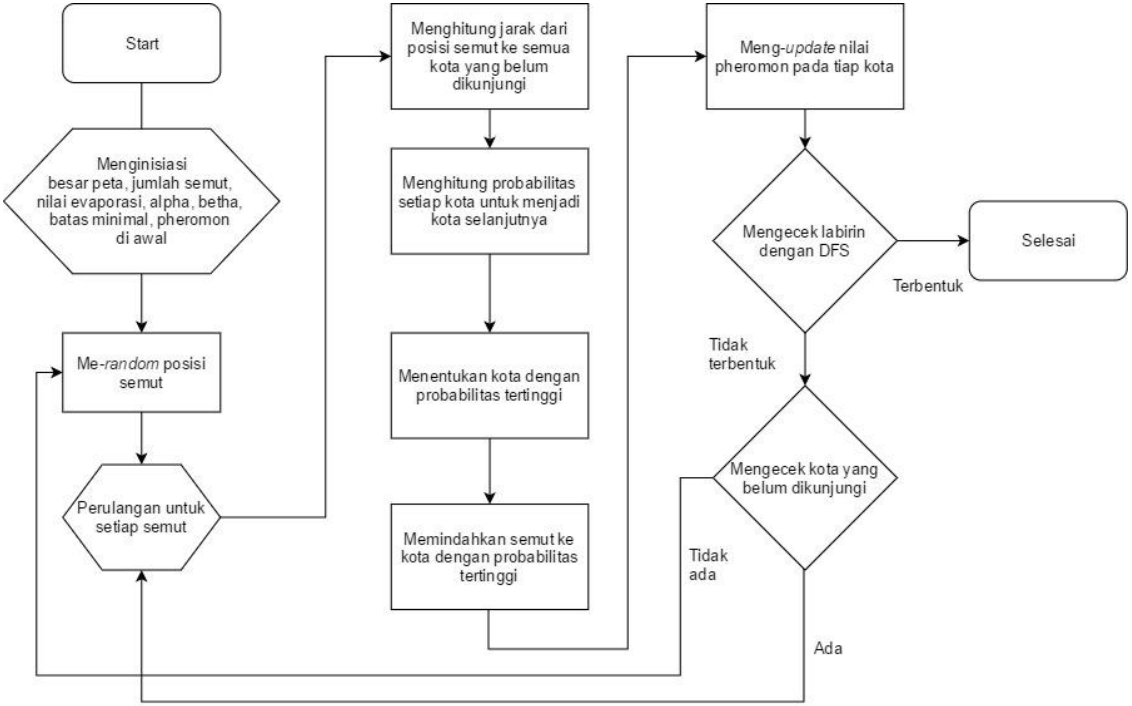
Ukuran Peta	Semut
10 x 10	9
11 x 11	10
12 x 12	11

3.3.2.2 *Random* Posisi Semut

Posisi semut akan dirandom menggunakan fungsi bawaan dari unity yaitu *Random.Range()*. Untuk setiap semut nilai x dan nilai y yang menentukan posisi awal semut akan diacak dengan *Random.Range()*.

3.3.2.3 Perulangan

Perulangan yang dimaksud adalah setiap semut akan memiliki jarak tersendiri, daftar kota yang belum dikunjungi tersendiri, dan daftar probabilitas tersendiri, namun ketika meng-*update* feromon pada setiap kota akan dilakukan satu kali untuk semua semut. Setelah meng-*update* nilai feromon, peta yang ada akan diperiksa apakah sudah terbentuk labirin dengan algoritma DFS apabila telah terbentuk maka Artificial Ant Colony Algorithm ini telah selesai. Apabila belum terbentuk labirin dan masih ada kota yang belum dikunjungi maka semut akan mengulang perulangan ini. Tetapi bila belum terbentuk labirin dan semua kota telah dikunjungi maka posisi semut akan di-*random* ulang.



Gambar 3. 3 Rancangan Artificial Ant Colony Algorithm

3.3.2.4 Menghitung Jarak

Menghitung jarak yang dimaksud adalah menghitung jarak antar semut dan kota yang belum dikunjungi oleh semut. Untuk menghitung jarak digunakan rumus (3) yang telah dijelaskan pada Subbab 2.2.

3.3.2.5 Menghitung Probabilitas

Menghitung probabilitas yang dimaksud adalah menghitung probabilitas setiap kota akan dipilih oleh semut. Apabila ada dua kota yang memiliki nilai feromon dan jarak yang sama maka kota yang dipilih akan diacak. Untuk menghitung probabilitas digunakan rumus (4) yang telah dijelaskan pada Subbab 2.2.

3.3.2.6 Memindahkan Semut

Memindahkan semut berarti semut tersebut akan berjalan menuju kota dengan probabilitas tertinggi. Diasumsikan waktu untuk berjalan ialah 1 kali perulangan walaupun ada semut yang menempuh jarak yang lebih jauh. Ini menandakan semut bisa saja berjalan dari ujung peta ke ujung peta yang lain, walaupun kemungkinan ini sangatlah kecil karena ketika menghitung probabilitas dengan rumus (4) jarak adalah pembagi yang berarti semakin besar jaraknya semakin kecil pula hasil baginya.

3.3.2.7 Meng-*update* Nilai Pheromon

Untuk meng-*update* feromon digunakan rumus (1). Nilai n pada rumus (2) adalah jumlah semut yang berada di kota x_i, x_j. Apabila tidak ada semut di kota tersebut maka nilai Γ_i^t ialah 0 karena tidak ada semut yang melepaskan feromon di kota tersebut. Apabila ada lebih dari satu semut maka nilai Γ_i^t

ialah jumlah feromon yang dilepaskan oleh semut-semut tersebut.

3.3.2.8 Mengecek Labirin dengan DFS

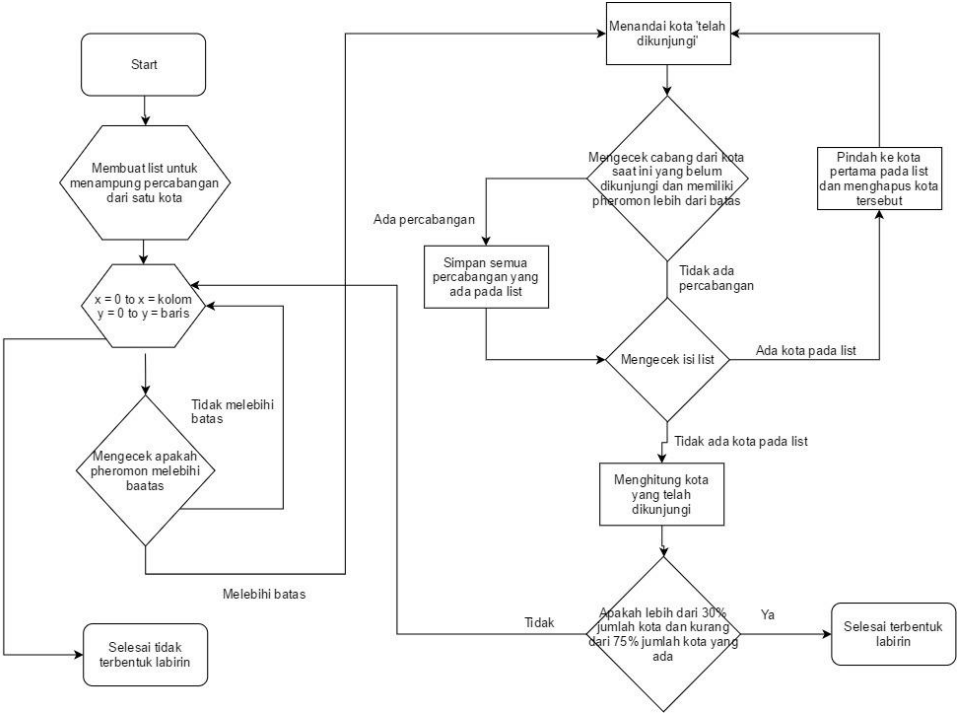
Mengecek labirin dengan algoritma DFS bertujuan untuk mengecek apakah labirin telah terbentuk atau belum, yang dimaksud dengan labirin yang telah terbentuk dapat dilihat pada Subbab 3.3.1 dengan contoh gambarnya pada Gambar 3. 2. Apabila labirin telah terbentuk maka selesailah algoritma ini. Untuk pembahasan lebih mendalam dari algoritma DFS dapat dilihat pada Subbab 3.3.3.

3.3.2.9 Mengecek Kota yang Belum Dikunjungi

Mengecek kota yang belum dikunjungi bertujuan untuk menentukan kota yang akan dikunjungi berikutnya yang mana juga merupakan kota yang belum dikunjungi oleh semut. Apabila semua kota telah dikunjungi oleh semut maka akan dimulai perulangan baru dengan memori baru. Memori ini berisi jarak, probabilitas, dan juga data kota yang belum dikunjungi.

3.3.3 Depth First Search (DFS)

Algoritma ini digunakan untuk menentukan apakah labirin telah terbentuk atau belum. Apabila labirin telah terbentuk maka Artificial Ant Colony Algorithm telah selesai. Algoritma ini disebut *depth first* karena akan langsung memeriksa jalan pertama yang temukan, bukan mencari mana sajakah jalan yang ada pada suatu kota. Rancangan sederhana dari algoritma ini dapat dilihat pada Gambar 3. 4. Untuk menentukan labirin sudah terbentuk atau belum haruslah memenuhi beberapa syarat yang telah dijelaskan pada Subbab 3.3.1.



Gambar 3. 4 Rancangan Algoritma DFS

3.4 Contoh Pembangkitan Labirin

Pada subbab ini akan dibahas mengenai contoh pembangkitan labirin. Contoh yang dimaksud ialah contoh menggunakan Artificial Ant Colony Algorithm dalam peta yang berukuran kecil.

Pada awalnya semua nilai ditetapkan terlebih dahulu. Nilai-nilai yang digunakan:

1. Kolom dan baris ialah 4.
2. Jumlah semut ialah 3.
3. Nilai evaporasi ialah 0.3.
4. Koefisien alpha ialah 0.3.
5. Koefisien beta ialah 0.5.
6. Batas minimal feromon ialah 0.3.
7. Nilai awal feromon ialah 1.

Porses pembangkitan yaitu:

1. Posisi semut di-*random* terlebih dahulu.
2. Posisi semut (x1, x2, dan x3) dapat dilihat pada Tabel 3. 5.
3. Nilai awal feromon dapat dilihat pada Tabel 3. 6.
4. Menambahkan kota yang telah dikunjungi oleh masing-masing semut. Untuk kota yang telah dikunjungi semut satu dapat dilihat pada Tabel 3. 7, untuk kota yang telah dikunjungi semut dua dapat dilihat pada Tabel 3. 8, dan untuk kota yang telah dikunjungi semut tiga dapat dilihat pada Tabel 3. 9 dimana 1 berarti telah dikunjungi dan 0 berarti belum dikunjungi.
5. Menghitung jarak kota lain pada masing-masing semut dengan cara yang telah dijelaskan pada Subbab 3.3.2.4, untuk jarak kota lain pada semut satu dapat dilihat pada Tabel 3. 10, untuk jarak kota lain pada semut dua dapat

dilihat pada Tabel 3. 11, dan untuk jarak kota lain pada semut tiga dapat dilihat pada Tabel 3. 12.

6. Menghitung probabilitas kota yang akan dikunjungi selanjutnya dengan cara yang telah dijelaskan pada Subbab 3.3.2.5. Untuk probabilitas semut satu dapat dilihat pada Tabel 3. 13, untuk probabilitas semut dua dapat dilihat pada Tabel 3. 14 dan untuk probabilitas semut tiga dapat dilihat pada Tabel 3. 15.
7. Memindahkan semut ke kota dengan probabilitas tertinggi. Jika terdapat lebih dari satu kota dengan nilai probabilitas tertinggi maka kota yang dipilih akan di-*random*. Posisi semut yang baru dapat dilihat pada Tabel 3. 16.
8. Menghitung nilai feromon yang baru dengan cara yang telah dijelaskan pada Subbab 3.3.2.5. Nilai feromon yang baru dapat dilihat pada Tabel 3. 17.
9. Mengecek apakah labirin telah terbentuk atau belum dengan menggunakan algoritma DFS. Contoh sederhana dari algoritma ini telah dijelaskan pada Subbab 2.3. Hasil pengecekan kota yang melebih batas atau dapat dilalui dan saling terhubung dapat dilihat pada Tabel 3. 18.
10. Hasil pengecekan kota yang dapat dilalui ada 16 dan itu berarti 100% kota yang ada pada peta dapat dilalui. Karena salah satu syarat terbentuknya labirin ialah kota yang dapat dilalui kurang dari 70%, maka hasilnya ialah labirin belum terbentuk.
11. Perulangan akan dilakukan kembali hingga labirin terbentuk atau semut kehabisan kota yang dapat dikunjungi. Jika kota yang dapat dikunjungi telah habis maka semut akan mengulang perulangan mulai me-*random* posisi semut kembali dan menghapus daftar kota yang telah dikunjungi.

Tabel 3. 5 Contoh posisi semut awal

x1			
		x3	
x2			

Tabel 3. 6 Contoh nilai awal feromon

1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1

Tabel 3. 7 Contoh daftar kota yang telah dikunjungi semut satu

1	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

Tabel 3. 8 Contoh daftar kota yang telah dikunjungi semut dua

0	0	0	0
0	0	0	0
0	0	0	0
1	0	0	0

Tabel 3. 9 Contoh daftar kota yang telah dikunjungi semut tiga

0	0	0	0
0	0	0	0
0	0	1	0
0	0	0	0

Tabel 3. 10 Contoh jarak kota yang akan dikunjungi semut satu

0	1	2	3
1	1.414	2.236	3.162
2	2.236	2.828	3.606
3	3.162	3.606	4.243

Tabel 3. 11 Contoh jarak kota yang akan dikunjungi semut dua

3	3.162	3.606	4.243
2	2.236	2.828	3.606
1	1.414	2.236	3.162
0	1	2	3

Tabel 3. 12 Contoh jarak kota yang akan dikunjungi semut tiga

2.8284	2.236	2	2.236
2.2361	1.414	1	1.414
2	1	0	1
2.2361	1.414	1	1.414

Tabel 3. 13 Contoh probabilitas kota yang akan dikunjungi semut satu

0	0.099947	0.070673	0.057704
0.099947	0.084045	0.066838	0.056204
0.070673	0.066838	0.059429	0.052636
0.057704	0.056204	0.052636	0.048523

Tabel 3. 14 Contoh probabilitas kota yang akan dikunjungi semut dua

0.057704	0.056204	0.052636	0.048523
0.070673	0.066838	0.059429	0.052636
0.099947	0.084045	0.066838	0.056204
0	0.099947	0.070673	0.057704

Tabel 3. 15 Contoh probabilitas kota yang akan dikunjungi semut tiga

0.049355	0.055509	0.058694	0.055509
0.055509	0.069799	0.083006	0.069799
0.058694	0.083006	0	0.083006
0.055509	0.069799	0.083006	0.069799

Tabel 3. 16 Contoh posisi semut yang baru

x1			
			x3
	x2		

Tabel 3. 17 Contoh nilai feromon setelah di-*update*

0.7	0.7	0.7	0.7
1.7	0.7	0.7	0.7
0.7	0.7	0.7	1.7
0.7	1.7	0.7	0.7

Tabel 3. 18 Contoh hasil kota yang dapat dilalui

1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1

3.5 Perancangan Tampilan Antarmuka

Subbab ini membahas bagaimana rancangan antarmuka pengguna yang akan digunakan untuk tugas akhir. Dalam *game* ini terdapat beberapa tampilan, yaitu tampilan awal (menu utama), tampilan memasukkan nama, tampilan permainan, tampilan menu *pause*, tampilan pesan menang dan pesan kalah, dan tampilan *minimap*.

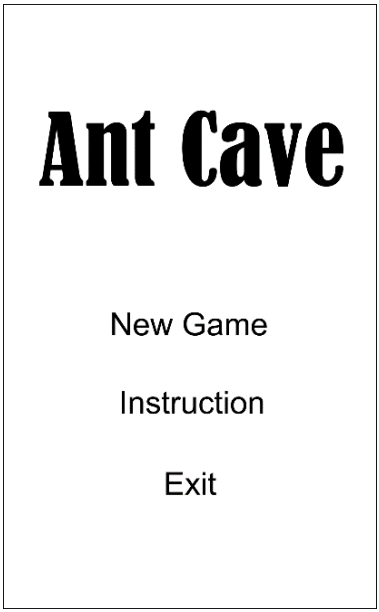
3.5.1 Tampilan Awal

Tampilan awal merupakan halaman yang pertama kali muncul ketika aplikasi dijalankan. Pada halaman ini terdapat tiga tombol, yaitu tombol *new game*, tombol *instruction*, dan tombol *exit*. Rancangan antarmuka tampilan awal dapat dilihat pada Gambar 3. 5.

Fungsi tiga tombol pada halaman awal, yaitu:

1. Tombol *new game* yang berfungsi untuk memulai permainan.

2. Tombol *instruction* yang berfungsi untuk melihat instruksi permainan.
3. Tombol *exit* yang berfungsi untuk keluar dari aplikasi permainan.

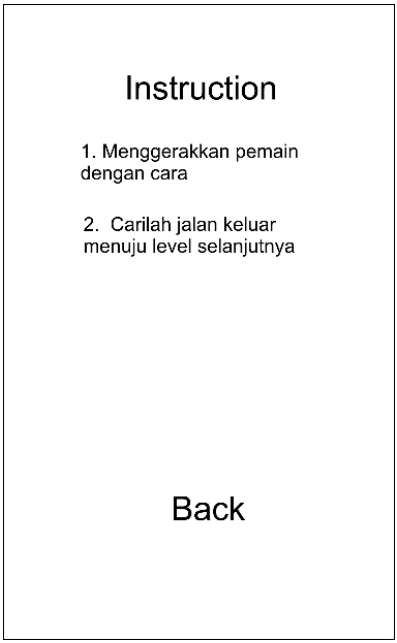


Gambar 3. 5 Rancangan tampilan awal

3.5.2 Tampilan Instruksi

Halaman ini berisikan tentang petunjuk untuk mengendalikan karakter. Petunjuk yang ditampilkan berupa petunjuk untuk menggerakkan karakter dan petunjuk tentang beberapa aturan permainan.

Rancangan tampilan instruksi dapat dilihat pada Gambar 3. 6. Pada halaman ini terdapat satu tombol, yaitu tombol *back.* Tombol ini berfungsi untuk kembali ke halaman awal.



Gambar 3. 6 Rangcangan tampilan instruksi

3.5.3 Tampilan Memasukkan Nama

Setelah menekan tombol *new game* pemain diminta mengisi nama yang akan digunakan oleh pemain. Nama ini akan ditampilkan pada tampilan permainan.

Rancangan antarmuka tampilan memasukkan nama dapat dilihat pada Gambar 3. 7. Terdapat dua tombol pada menu ini, yaitu:

1. Tombol *save* yang berfungsi untuk menyimpan nama pemain.
2. Tombol *back* yang berfungsi untuk kembali ke halaman awal.

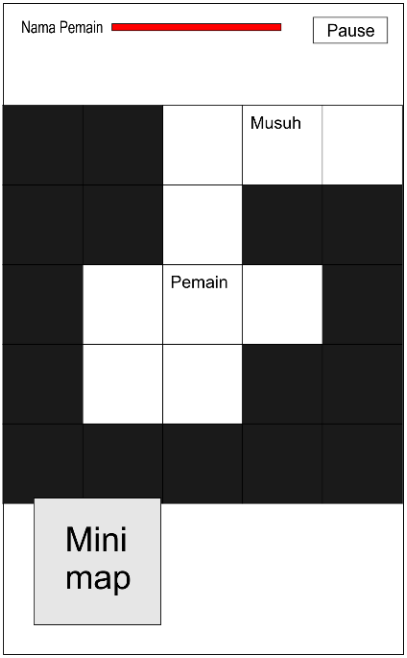
Gambar 3. 7 Rancangan tampilan memasukkan nama

3.5.4 Tampilan Permainan

Tampilan permainan ialah halaman pada saat pemain telah memasukkan nama atau ketika sedang bermain. Rancangan Tampilan Permainan dapat dilihat pada Gambar 3. 8. Pada layar terdapat beberapa hal yaitu:

1. Nama pemain yang terletak di pojok atas kiri.
2. *Slide bar* yang menunjukkan *health point* saat ini, *slide bar* ini terletak di samping bagian nama pemain.
3. Tombol *pause* yang berfungsi untuk menghentikan permainan untuk sementara dan juga untuk keluar dari *game*.
4. Pemain dan musuh yang terletak dalam labirin.

5. *Minimap* yang berfungsi sebagai tombol untuk melihat peta labirin secara keseluruhan.

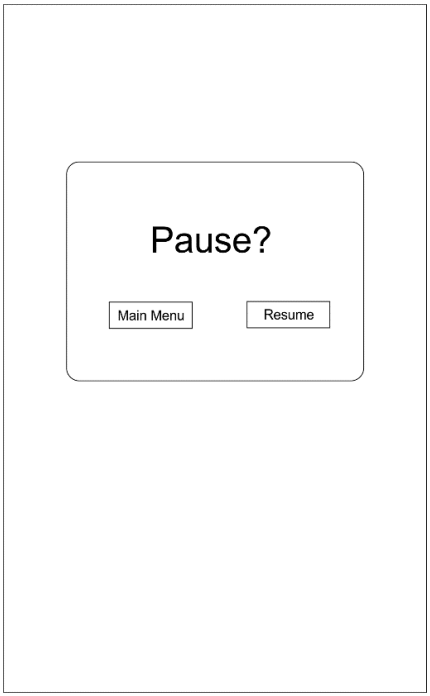


Gambar 3. 8 Rancangan tampilan permainan

3.5.5 Tampilan Menu *Pause*

Tampilan menu pause ialah menu yang akan muncul ketika pemain menekan tombol *pause* pada tampilan permainan. Tombol ini akan menghentikan permainan untuk sesaat.

Rancangan tampilan menu *pause* dapat dilihat pada Gambar 3. 9. Pada menu ini terdapat dua tombol, yang pertama merupakan tombol *main menu* untuk kembali ke menu utama dan yang kedua ialah tombol *Resume* yang berfungsi untuk melanjutkan permainan.



Gambar 3. 9 Rancangan tampilan menu *pause*

3.5.6 Tampilan Pesan Menang dan Kalah

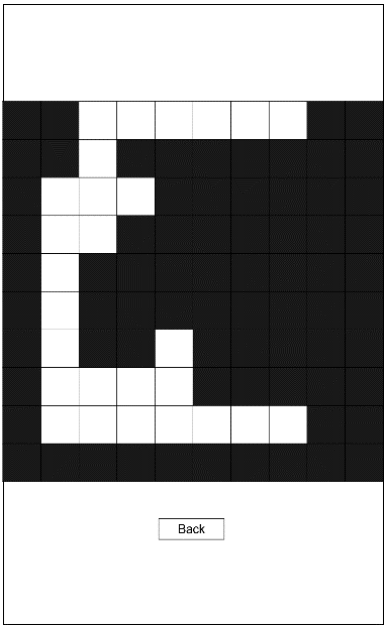
Tampilan pesan menang ialah pesan yang muncul ketika pemain telah mencapai tangga untuk ke level selanjutnya. Tampilan pesan kalah ialah pesan yang muncul ketika pemain *game over* atau kehilangan seluruh *health point*. Tampilan pesan menang dan kalah tidak ada tombol karena secara otomatis akan memulai level selanjutnya untuk pesan menang dan secara otomatis akan kembali ke halaman utama untuk pesan kalah setelah beberapa saat. Rancangan tampilan pesan menang dan kalah dapat dilihat pada Gambar 3. 10 dan Gambar 3. 11.

Gambar 3. 10 Rancangan tampilan pesan menang

Gambar 3. 11 Rancangan tampilan pesan kalah

3.5.7 Tampilan *Minimap*

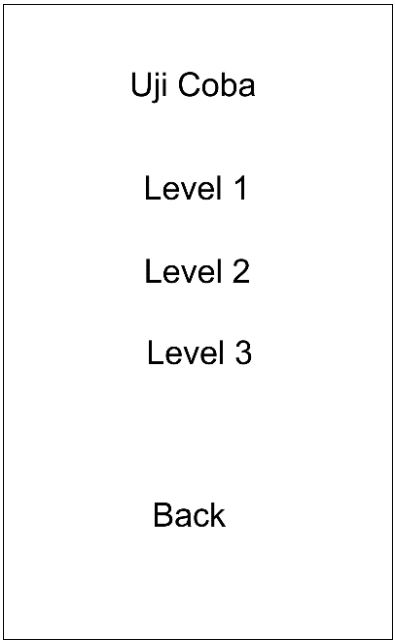
Tampilan *minimap* ialah tampilan ketika *minimap* pada tampilan permainan ditekan. Peta labirin akan terlihat suluruhnya. Pada halaman ini terdapat satu tombol *back* yang berfungsi untuk kembali pada tampilan permainan. Rancangan tampilan minimap dapat dilihat pada Gambar 3. 12.



Gambar 3. 12 Rancangan tampilan minimap

3.5.8 Tampilan Menu Pengujian

Tampilan ini ialah tampilan menu yang dibutuhkan untuk melakukan pengujian. Tampilan ini akan berisi menu-menu khusus untuk menjalankan pengujian. Tampilan menu pengujian dapat dilihat pada Gambar 3. 13.



Gambar 3. 13 Rancangan tampilan menu pengujian

3.6 Perancangan Karakter dan Aset Ant Cave

Pada subbab ini akan dibahas rancangan rancangan karakter dan aset yang digunakan untuk *game* ini. Karakter maupun aset untuk *game* ini penulis ambil dari web yang menyediakan aset secara gratis.

3.6.1 Karakter Pemain Ant Cave

Untuk karakter pemain Ant Cave penulis mengunakan karya Warren Clark pada web itch.io [7]. Karakter ini tersedia dalam bentuk .png sehingga memudahkan penulis untuk menggunakan karakter ini. Karakter pemain dapat dilihat pada Gambar 3. 14.

Gambar 3. 14 Karakter pemain

3.6.2 Karakter Musuh Lemah

Untuk karakter musuh lemah penulis mengunakan karya Henry Software pada web itch.io [8]. Karakter ini tersedia dalam bentuk .png sehingga memudahkan penulis untuk menggunakan karakter ini. Karakter pemain dapat dilihat pada Gambar 3. 15.

Gambar 3. 15 Karakter musuh lemah

3.6.3 Karakter Musuh Sedang

Untuk karakter musuh sedang penulis mengunakan karya Arks pada web itch.io [9]. Karakter ini tersedia dalam bentuk .png sehingga memudahkan penulis untuk menggunakan karakter ini. Karakter pemain dapat dilihat pada Gambar 3. 16.

Gambar 3. 16 Karakter musuh sedang

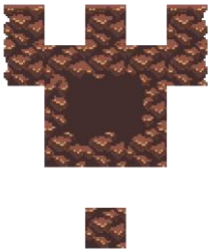
3.6.4 Karakter Musuh Kuat

Untuk karakter musuh kuat penulis mengunakan karya Jesse M pada web itch.io [10]. Karakter ini tersedia dalam bentuk .png sehingga memudahkan penulis untuk menggunakan karakter ini. Karakter pemain dapat dilihat pada Gambar 3. 17.

Gambar 3. 17 Karakter musuh kuat

3.6.5 Aset Labirin

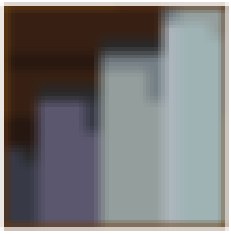
Untuk aset labirin penulis mengunakan karya Petey90 pada web itch.io [11]. Aset ini tersedia dalam bentuk .png sehingga memudahkan penulis untuk menggunakan aset ini. Aset labirin ini dapat dilihat pada Gambar 3. 18.



Gambar 3. 18 Aset labirin

3.6.6 Aset Tangga Turun

Untuk aset tangga turun penulis mengunakan karya Stephen Challener pada web opengameart.org [12]. Aset ini tersedia dalam bentuk .png sehingga memudahkan penulis untuk menggunakan aset ini. Aset tangga turun ini dapat dilihat pada Gambar 3. 19.



Gambar 3. 19 Aset tangga turun

BAB IV
IMPLEMENTASI

Pada bab ini dibahas mengenai implementasi yang dilakukan berdasarkan rancangan yang telah dijabarkan pada bab sebelumnya.

4.1 Lingkungan Implementasi

Lingkungan implementasi merupakan lingkungan dimana *game* 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:

- Tipe : Asus G550JX
- Prosesor : Intel® Core(TM) i7-4720HQ CPU @ 2.60GHz (8 CPUs) ~2.6 GHz
- Memori : 8192 MB RAM

Sedangkan untuk melakukan implementasi, digunakan perangkat bergerak dengan spesifikasi sebagai berikut:

- Tipe : Sony Xperia Z1 Compact
- Prosesor : Qualcomm MSM8974
- Memori : 2048 MB

4.1.2 Perangkat Lunak

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

- Sistem Operasi : Windows 10 Home 64-bit
- *Game Engine* : Unity versi 5.4.2f2
- *Text Editor* : MonoDevelop

Sedangkan untuk melakukan implementasi, digunakan perangkat bergerak dengan spesifikasi perangkat lunak sebagai berikut:

- Sistem Operasi : Android 5.1.1

4.2 Implementasi Labirin

Subbab ini membahas mengenai implementasi rancangan labirin seperti yang sudah dijelaskan pada Subbab 3.3.1. Kode implementasi labirin dapat dilihat pada Kode Sumber 4. 1.

```
void DeclareAll(int level){
    if (level % 10 == 1 || level % 10 == 2 || level %
10 == 3 ) {
        kolom = 10;
        baris = 10;
        jumlahSemut = 9;
    } else if (level % 10 == 4 || level % 10 == 5 ||
level % 10 == 6){
        kolom = 11;
        baris = 11;
        jumlahSemut = 10;
    } else if (level % 10 == 7 || level % 10 == 8 ||
level % 10 == 9 || level % 10 == 0 ){
        kolom = 12;
        baris = 12;
        jumlahSemut = 11;
    } else {
        kolom = 12;
        baris = 12;
        jumlahSemut = 11;
    }
}

void BuatMapsDasar (float phe){

    maps.Clear ();

    for(int x = 0; x < kolom; x++){
        for(int y = 0; y < baris; y++){
            //mulai dari 0.0 sampai kolom.baris
```

```
                maps.Add (new MAP {pos= new Vector3 (x,
y, 0f), pheromon = phe});
            }
        }
}
```

Kode Sumber 4. 1 Kode implementasi labirin

4.3 Implementasi Artificial Ant Colony Algorithm

Subbab ini membahas mengenai implementasi pembangkit labirin seperti yang sudah dijelaskan pada Subbab 3.3.2.

4.3.1 Implementasi Perhitungan Jarak

Pada tahap ini dijelaskan mengenai implementasi dari perhitungan jarak antar kota yang telah dijelaskan pada Bab III. Kode untuk menghitung jarak antar tiap kota dapat dilihat pada Kode Sumber 4. 2.

```
float HitJarak(int x1, int y1, int x2, int y2){
    float jarak;

    float a =(float)Math.Pow((float)x1-(float)x2, 2);
    float b =(float)Math.Pow((float)y1-(float)y2, 2);

    jarak =(float)Math.Sqrt(a + b);

    return jarak;
}
```

Kode Sumber 4. 2 Kode untuk menghitung jarak

4.3.2 Implementasi Probabilitas

Pada tahap ini dijelaskan mengenai implementasi dari perhitungan probabilitas setiap kota seperti yang telah dijelaskan pada Bab III. Kode untuk menghitung probabilitas setiap kota dapat dilihat pada Kode Sumber 4. 3.

```
for (int k = 0; k < kolom * baris; k++) {
```

```
    float jarak = HitJarak (posSemut [i, 0], posSemut
[i, 1], ant [i] [k].posx, ant [i] [k].posy);
    float jarakXphero;

    //kalau belum visited
    if (ant [i] [k].visited==false && jarak!=0) {
        int index = maps.FindIndex (aaaa => aaaa.pos.x
==(float)  ant [i] [k].posx && aaaa.pos.y == (float)
ant [i] [k].posy);

        float a = (float)Math.Pow (maps
[index].pheromon, alpha);

        float b = (float)Math.Pow (1 /jarak, beta);

        jarakXphero = a * b;

    } else
        jarakXphero = 0;

    jarakXpheroTotal = jarakXpheroTotal + jarakXphero;

    ant [i] [k].distXphero = jarakXphero;
}

for (int k = 0; k < kolom * baris; k++) {
    ant [i] [k].prob = ant [i] [k].distXphero /
jarakTotal;
}
```

Kode Sumber 4. 3 Kode menghitung probabilitas

4.3.3 Implementasi Perpindahan Semut

Pada tahap ini dijelaskan mengenai implementasi dari perpindahan semut ke kota dengan probabilitas tertinggi seperti yang telah dijelaskan pada Bab III. Kode untuk mencari probabilitas tertinggi dapat dilihat pada Kode Sumber 4. 4.

```
List<float> tmparray = new List<float>();

for (int k = 0; k < kolom * baris; k++) {
    //probablitas yang telah dihitung sebelumnya
```

```
    tmparray.Add(ant [i] [k].prob);
}

tmparray.Sort();

float nilaiMax = tmparray [tmparray.Count-1];
```

Kode Sumber 4. 4 Kode mencari probabilitas tertinggi

4.3.4 Implementasi Update Pheromon

Pada tahap ini dijelaskan mengenai implementasi dari perhitungan nilai feromon yang baru setelah semut berpindah kota seperti yang telah dijelaskan pada Bab III. Apabila ada lebih dari satu semut pada kota yang sama maka nilai feromon akan bertambah sebanyak jumlah feromon yang dikeluarkan oleh semut tersebut. Jumlah feromon yang dikeluarkan oleh semut ialah 1/jarak semut dari kota sebelumnya. Kode untuk meng-*update* feromon dapat dilhat pada Kode Sumber 4. 5.

```
for (int abc = 0; abc < maps.Count; abc++) {
    //langsung update
    maps [abc].pheromon = (1 - evaporation) * maps
[abc].pheromon;

    for (int j = 0; j < jumlahSemut; j++) {
        if (maps [abc].pos == new Vector3 (posSemut
[j, 0], posSemut [j, 1], 0)) {
            maps [abc].pheromon = maps [abc].pheromon
+ (1 / posSemutJarak [j]);
        }
    }
}
```

Kode Sumber 4. 5 Kode untuk meng-*update* nilai feromon

4.4 Implementasi Algoritma DFS

Pada tahap ini dijelaskan mengenai implementasi dari algoritma DFS seperti yang telah dijelaskan pada Subbab 3.3.3. Algoritma ini digunakan untuk mengecek apakah kota-kota yang

ada telah terhubung sehingga membentuk labirin atau belum. Kode algoritma ini dapat dilihat pada Kode Sumber 4. 6.

```
int CekSelesai(){
    List<MAPTMP> maptmp = new List<MAPTMP>();
    List<int> letaktmp = new List<int> ();

    for (int x = 0; x < maptmp.Count; x++) {
        if (maptmp[x].jalan && !maptmp[x].visited) {
            int xtmp = maptmp [x].posx;
            int ytmp = maptmp [x].posy;

            while (true) {
                int index = maptmp.FindIndex (aaaa =>
aaaa.posx == xtmp && aaaa.posy == ytmp);
                int indexAtas, indexKanan,
indexBawah, indexKiri;
                if (ytmp < baris - 1)
                    indexAtas = maptmp.FindIndex
(aaaa => aaaa.posx == xtmp && aaaa.posy == ytmp + 1);
                else
                    indexAtas = -1;
                if (xtmp < kolom - 1)
                    indexKanan = maptmp.FindIndex
(aaaa => aaaa.posx == xtmp + 1 && aaaa.posy == ytmp);
                else
                    indexKanan = -1;
                if (xtmp > 0)
                    indexBawah = maptmp.FindIndex
(aaaa => aaaa.posx == xtmp - 1 && aaaa.posy == ytmp);
                else
                    indexBawah = -1;
                if (ytmp > 0)
                    indexKiri = maptmp.FindIndex
(aaaa => aaaa.posx == xtmp && aaaa.posy == ytmp - 1);
                else
                    indexKiri = -1;

                int tmppppp = 0;
                if (indexAtas != (-1) && maptmp
[indexAtas].jalan == true && maptmp
[indexAtas].visited == false) {
```

```
                    letaktmp.Insert (tmpppppp, maptmp
[indexAtas].posx);
                     tmpppppp++;
                     letaktmp.Insert (tmpppppp, maptmp
[indexAtas].posy);
                      tmpppppp++;
                }
                if (indexKanan != (-1) && maptmp
[indexKanan].jalan == true && maptmp
[indexKanan].visited == false) {
                     letaktmp.Insert (tmpppppp, maptmp
[indexKanan].posx);
                     tmpppppp++;
                     letaktmp.Insert (tmpppppp, maptmp
[indexKanan].posy);
                     tmpppppp++;
                }
                if (indexBawah != (-1) && maptmp
[indexBawah].jalan == true && maptmp
[indexBawah].visited == false) {
                     letaktmp.Insert (tmpppppp, maptmp
[indexBawah].posx);
                     tmpppppp++;
                    letaktmp.Insert (tmpppppp, maptmp
[indexBawah].posy);
                     tmpppppp++;
                }
                if (indexKiri != (-1) && maptmp
[indexKiri].jalan == true && maptmp
[indexKiri].visited == false) {
                     letaktmp.Insert (tmpppppp, maptmp
[indexKiri].posx);
                     tmpppppp++;
                     letaktmp.Insert (tmpppppp, maptmp
[indexKiri].posy);
                     tmpppppp++;
                }
                if(letaktmp.Count()==0)
                     break;

                xtmp = letaktmp [0];
                letaktmp.RemoveAt (0);
                ytmp = letaktmp [0];
                letaktmp.RemoveAt (0);
```

```
                maptmp [index].visited = true;

                if (letaktmp.Count () == 0)
                    break;
            }
        }
    }
}
```

Kode Sumber 4. 6 Kode Algoritma DFS

4.5 Implementasi Permainan

Subbab ini membahas mengenai implementasi pada permainan. Implementasi yang dimaksud dapat berupa implementasi aturan main, implementasi fungsi, dan implementasi rancangan tampilan awal. Implementasi permainan akan dibahas berdasarkan antarmuka-antarmuka yang ada pada permainan.

4.5.1 Implementasi Tampilan Awal

Hasil implementasi tampilan awal dapat dilihat pada Gambar 4. 1. Fungsi untuk untuk menjalankan halaman ini dapat dilihat pada Kode Sumber 4. 7.

```
public void NewGame(){
    if (newNameCanvas.gameObject.activeSelf)
        newNameCanvas.gameObject.SetActive (false);
    else
        newNameCanvas.gameObject.SetActive (true);
}

public void Instruction(){
    SceneManager.LoadScene ("Instruction");
}

public void Exit (){
    Application.Quit ();
}

public void ForTesting(){
```

```
    SceneManager.LoadScene ("ForTesting");
}
```

Kode Sumber 4. 7 Fungsi pada tampilan awal

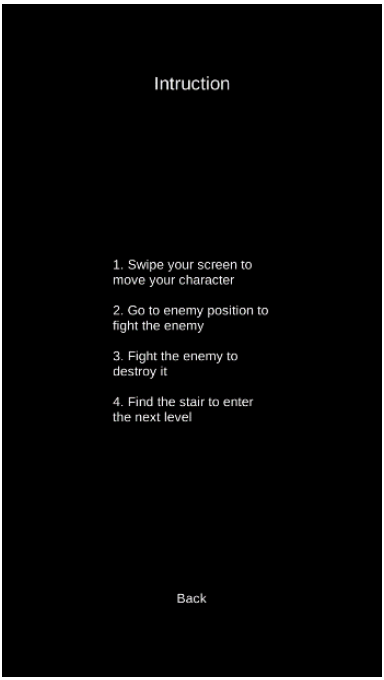
Gambar 4. 1 Tampilan awal

4.5.2 Implementasi Tampilan Instruksi

Hasil implementasi tampilan memasukkan nama dapat dilihat pada Gambar 4. 2. Fungsi untuk untuk menjalankan halaman ini dapat dilihat pada Kode Sumber 4. 8.

```
public void MainMenu(){
    SceneManager.LoadScene ("MainMenu");
}
```

Kode Sumber 4. 8 Fungsi tombol *back*



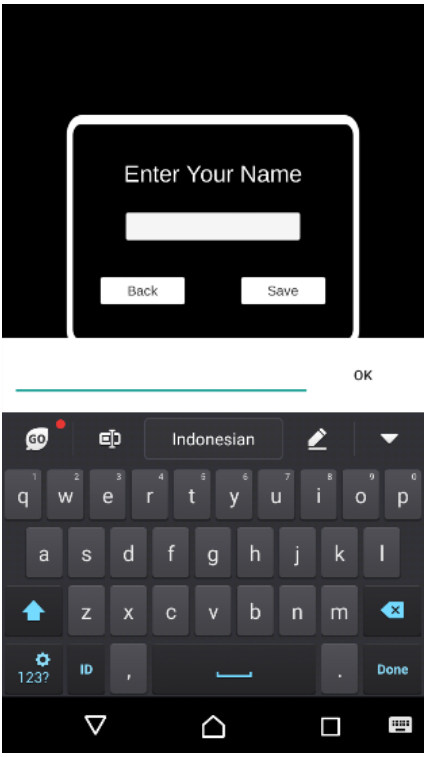
Gambar 4. 2 Tampilan instruksi

4.5.3 Implementasi Tampilan Memasukkan Nama

Hasil implementasi tampilan memasukkan nama dapat dilihat pada Gambar 4. 3. Fungsi untuk untuk menjalankan halaman ini dapat dilihat pada Kode Sumber 4. 9.

```
public void Play(){
    if (iField.text != null) {
         PlayerPrefs.SetString ("Name", iField.text);
         PlayerPrefs.SetInt ("Level", 0);
         SceneManager.LoadScene ("Main");
    }
}
```

Kode Sumber 4. 9 Fungsi tombol *save*



Gambar 4. 3 Tampilan memasukkan nama

4.5.4 Implementasi Tampilan Permainan

Hasil implementasi tampilan permainan dapat dilihat pada Gambar 4. 4. Terdapat beberapa fungsi untuk untuk menjalankan halaman ini, yaitu fungsi untuk mengatur level dapat dilihat pada Kode Sumber 4. 10, fungsi untuk Menggerakkan karakter dapat dilihat pada Kode Sumber 4. 11, fungsi untuk bertarung dapat dilihat pada Kode Sumber 4. 12, dan fungsi untuk jalan keluar dapat dilihat pada Kode Sumber 4. 13.

Gambar 4. 4 Tampilan permainan

```
void SetEnemy(int level){
    if (level == 1)
         LayoutEnemy (enemyTiles [0], 1);
    else if (level == 2)
         LayoutEnemy (enemyTiles [0], 2);
    else if (level == 3)
         LayoutEnemy (enemyTiles [0], 3);
    else if (level == 4) {
         LayoutEnemy (enemyTiles [0], 1);
         LayoutEnemy (enemyTiles [1], 1);
    }
    else if (level == 5) {
         LayoutEnemy (enemyTiles [0], 1);
         LayoutEnemy (enemyTiles [1], 2);
    }
    else if (level == 6) {
         LayoutEnemy (enemyTiles [1], 3);
```

```
    }
    else if (level == 7) {
        LayoutEnemy (enemyTiles [0], 1);
        LayoutEnemy (enemyTiles [2], 1);
    }
    else if (level == 8) {
        LayoutEnemy (enemyTiles [0], 1);
        LayoutEnemy (enemyTiles [1], 1);
        LayoutEnemy (enemyTiles [2], 1);
    }
    else if (level == 9) {
        LayoutEnemy (enemyTiles [1], 2);
        LayoutEnemy (enemyTiles [2], 1);
    }
    else if (level == 10) {
        LayoutEnemy (enemyTiles [0], 1);
        LayoutEnemy (enemyTiles [1], 2);
        LayoutEnemy (enemyTiles [2], 2);
    }
    else  {

    }
}
```

Kode Sumber 4. 10 Implementasi level

```
void Update () {
    GameManager.instance.playerHealthPoints =
healthPoint;
    if(!GameManager.instance.playersTurn) return;
    int horizontal = 0;
    int vertical = 0;

    if (Input.touchCount > 0){
        Touch myTouch = Input.touches[0];

        if (myTouch.phase == TouchPhase.Began){
            touchOrigin = myTouch.position;
        }
        else if (myTouch.phase == TouchPhase.Ended &&
touchOrigin.x >= 0)
        {
            Vector2 touchEnd = myTouch.position;
            float x = touchEnd.x - touchOrigin.x;
            float y = touchEnd.y - touchOrigin.y;
```

```
                touchOrigin.x = -1;

                if (Mathf.Abs(x) > Mathf.Abs(y))
                    horizontal = x > 0 ? 1 : -1;
                else
                    vertical = y > 0 ? 1 : -1;
            }
        }

        if(horizontal != 0 || vertical != 0)
        {
            AttemptMove<Enemy> (horizontal, vertical);
        }
    }
```

Kode Sumber 4. 11 Fungsi untuk menggerakkan karakter

```
public IEnumerator BattleScene (Enemy enemy, Player
player, int arah){
    playersTurn = true;

    canvasController.BattleCanvasSetting (enemy);

    bool playerFirst=false;

    int playerAttackPoint, enemyAttackPoint, difSpeed;

    difSpeed = player.speed - enemy.speed;

    playerAttackPoint = player.attack - enemy.defence;
    if (playerAttackPoint <= 0)
        playerAttackPoint = 1;

    enemyAttackPoint = enemy.attack - player.defence;
    if (enemyAttackPoint <= 0)
        enemyAttackPoint = 1;

    if (difSpeed > 0) {
        playerFirst = true;
    } else if (difSpeed < 0) {
        playerFirst = false;
    }else if (difSpeed == 0) {
        playerFirst = true;
    }
    looping = true;
```

```
    while (looping) {
        if (playerFirst) {
            //player menyerang
            player.AttackAnim(arah);
            yield return new WaitForSeconds (1.2f);
            enemy.healthPoint = enemy.healthPoint -
playerAttackPoint;

            if (enemy.healthPoint>=0) {
                //enemy menyerang
                player.healthPoint =
player.healthPoint - enemyAttackPoint;

                if (player.healthPoint <= 0) {
                    canvasController.GameOver();
                        looping = false;
                }

            } else {
                enemy.Lose ();
                looping = false;
            }
        } else {
            //enemy menyerang
            player.healthPoint = player.healthPoint -
enemyAttackPoint;

            if (player.healthPoint <= 0) {
                canvasController.GameOver();
                looping = false;
            } else {
                player.AttackAnim (arah);
                yield return new WaitForSeconds
(1.2f);
                enemy.healthPoint = enemy.healthPoint
- playerAttackPoint;
                if (enemy.healthPoint<=0) {
                    enemy.Lose ();
                    looping = false;
                }
            }
        }
    }
```

```
    canvasController.BattleCanvasExitSetting ();
    playersTurn = false;
}
```

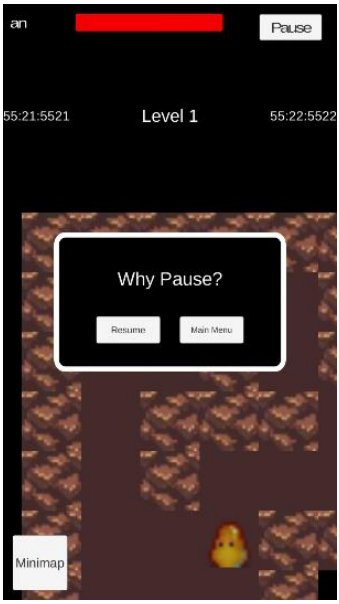
Kode Sumber 4. 12 Fungsi bertarung

```
private void OnTriggerEnter2D(Collider2D other){
    if (other.tag == "Exit") {
         Invoke ("Restart", restartLevelDelay);

         enabled = false;
    }
}
```

Kode Sumber 4. 13 Fungsi jalan keluar

4.5.5 Implementasi Tampilan Menu *Pause*

Hasil implementasi tampilan menu *pause* dapat dilihat pada Gambar 4. 5. Fungsi untuk untuk menjalankan halaman ini dapat dilihat pada Kode Sumber 4. 14.



Gambar 4. 5 Tampilan menu *pause*

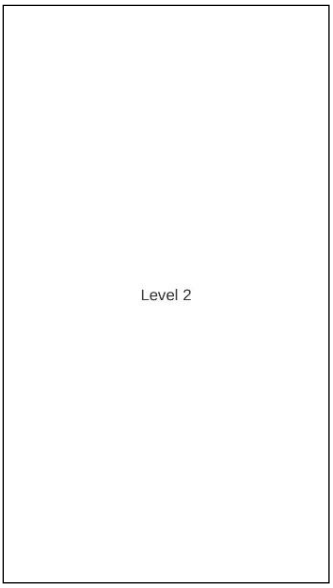
```
public void Pause(){
    Time.timeScale = 0;
    pauseCanvas.gameObject.SetActive (true);
}

public void Resume(){
    Time.timeScale = 1;
    pauseCanvas.gameObject.SetActive (false);
}

public void MainMenu(){
    GameManager.instance.GameOverWait ();
}
```

Kode Sumber 4. 14 Fungsi menu *pause*

4.5.6 Implementasi Tampilan Pesan Menang Dan Kalah

Hasil implementasi tampilan pesan kalah dapat dilihat pada Gambar 4. 6, sedangkan implementasi tampilan pesan menang dapat dilihat pada Gambar 4. 7.

Gambar 4. 6 Pesan kalah



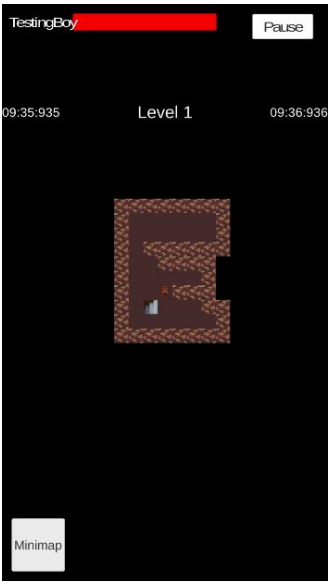
Gambar 4. 7 Pesan menang

4.5.7 Implementasi Tampilan *Minimap*

Hasil implementasi tampilan *minimap* dapat dilihat pada Gambar 4. 8. Fungsi untuk untuk menjalankan halaman ini dapat dilihat pada Kode Sumber 4. 15.

```
public void Minimap(){
    if (minimap) {
         Camera.main.orthographicSize = 5;
         minimap = false;
    }else {
         Camera.main.orthographicSize = 20;
         minimap = true;
    }
}
```

Kode Sumber 4. 15 Fungsi *minimap*



Gambar 4. 8 ***Minimap***

(Halaman ini sengaja dikosongkan)

BAB V
PENGUJIAN DAN EVALUASI

Pada bab ini dijelaskan tentang uji coba dan evaluasi dari implementasi yang telah dilakukan.

5.1 Lingkungan Pengujian

Lingkungan uji coba yang digunakan adalah sebuah *smartphone* dengan spesifikasi sebagai berikut:

1. Perangkat Keras
   - Tipe : Sony Xperia Z1 Compact
   - Prosesor : Qualcomm MSM8974 Snapdragon 800
   - Memori : 2 GB
2. Perangkat Lunak
   - Sistem Operasi : Android 5.1.1

Pada skenario pengujian dijelaskan tentang skenario pengujian yang dilakukan. Pengujian yang akan dilakukan hanya 1 jenis, yaitu pengujian fungsionalitas. Pengujian ini dilakukan untuk mengetahui apakah fungsionalitas *game* telah berjalan sebagai mana mestinya dan untuk mengetahui apakah telah terbentuk labirin.

5.2 Pengujian Fungsionalitas

Pengujian fungsionalitas dilakukan dengan menyiapkan beberapa skenario pengujian sebagai tolok ukur keberhasilan pengujian. Pengujian pertama ialah untuk mengetahui apakah terbentuk labirin yang berbeda setiap levelnya, pengujian kedua adalah untuk mengetahui apakah pemain dapat *game over* dan apakah pemain dapat menyelesaikan 1 level permainan.

5.2.1 Pengujian Labirin

Pengujian terbentuknya labirin merupakan pengujian terhadap aplikasi untuk membentuk labirin-labirin yang berbeda setiap levelnya. Pengujian akan dilakukan sebanyak 9 kali, 3 kali pada level 1 yang miliki ukuran peta 10 x 10, 3 kali pada level 4 yang memiliki ukuran peta 11 x 11, dan 3 kali pada level 7 yang memiliki ukuran peta 12 x 12. Pada setiap pengujian akan dicek apakah waktu untuk membentuk labirin lebih dari 5 detik atau tidak. Hasil pengujian dapat dilihat pada tabel Tabel 5. 1, detail waktu pengujian dapat dilihat pada tabel Tabel 5. 2, dan hasil labirin dapat dilihat pada Gambar 5. 1 Pengujian labirin level 1-1 hingga Gambar 5. 9.

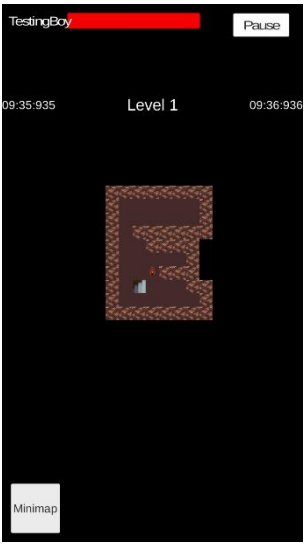
Tabel 5. 1 Hasil pengujian labirin

ID	PF-001
Nama	Pengujian Terbentuknya Labirin
Tujuan uji coba	Terbentuk labirin yang berbeda-beda pada setiap level
Kondisi awal	Pemain berada pada halaman menu pengujian
Skenario 1	***Pemain memilih Level 1 sebanyak tiga kali***
Masukan	Klik tombol Level 1
Keluaran yang diharapkan	Labirin yang berbeda-beda dapat terbentuk dalam waktu yang kurang dari 5 detik
Hasil uji coba	Berhasil
Kondisi Akhir	Terbentuk labirin yang berbeda-beda dan dalam waktu yang kurang dari 5 detik
Skenario 2	***Pemain memilih Level 4 sebanyak tiga kali***
Masukan	Klik tombol Level 4

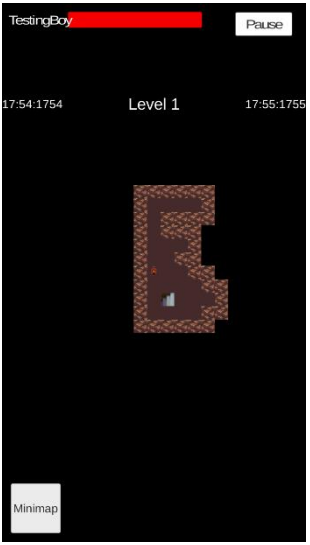
ID	PF-001
Keluaran yang diharapkan	Labirin yang berbeda-beda dapat terbentuk dalam waktu yang kurang dari 5 detik
Hasil uji coba	Berhasil
Kondisi Akhir	Terbentuk labirin yang berbeda-beda dan dalam waktu yang kurang dari 5 detik
Skenario 3	***Pemain memilih Level 7 sebanyak tiga kali***
Masukan	Klik tombol Level 7
Keluaran yang diharapkan	Labirin yang berbeda-beda dapat terbentuk dalam waktu yang kurang dari 5 detik
Hasil uji coba	Berhasil
Kondisi Akhir	Terbentuk labirin yang berbeda-beda dan dalam waktu yang kurang dari 5 detik

Tabel 5. 2 Hasil Waktu pengujian labirin

Level	Waktu
1	1 detik
1	1 detik
1	1 detik
4	1 detik
4	1 detik
4	1 detik
7	3 detik
7	3 detik
7	1 detik



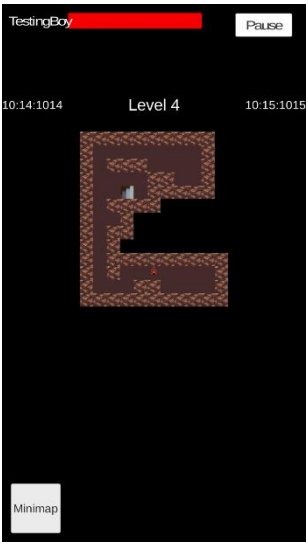
Gambar 5. 1 Pengujian labirin level 1-1



Gambar 5. 2 Pengujian labirin level 1-2

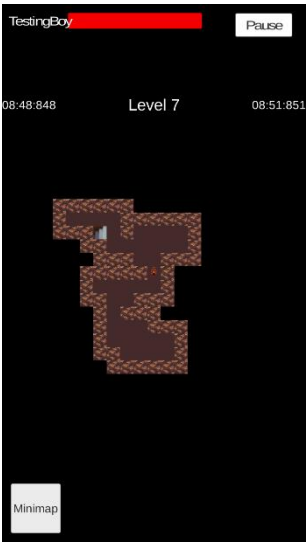
Gambar 5. 3 Pengujian labirin level 1-3

Gambar 5. 4 Pengujian labirin level 4-1



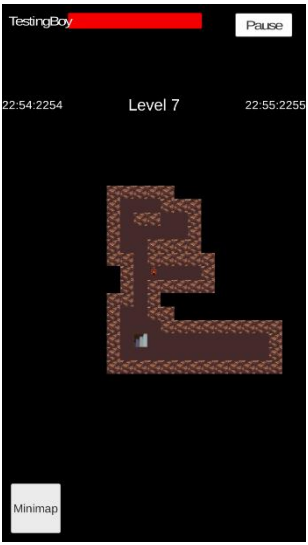
Gambar 5. 5 Pengujian labirin level 4-2

Gambar 5. 6 Pengujian labirin level 4-3



Gambar 5. 7 Pengujian labirin level 7-1

Gambar 5. 8 Pengujian labirin level 7-2



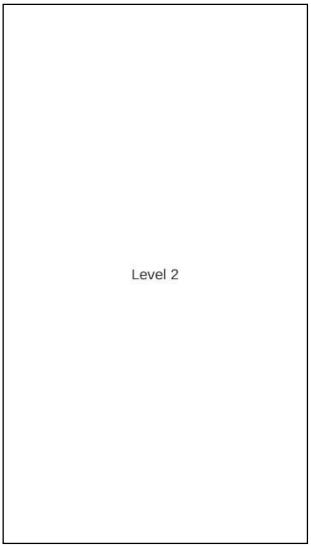
Gambar 5. 9 Pengujian labirin level 7-3

5.2.2 Pengujian Aturan Permainan

Pengujian aturan main merupakan pengujian terhadap *game* untuk melihat apakah pemain dapat menemukan jalan untuk ke level selanjutnya. Skenario pengujian ada tiga, yaitu ketika level selesai karena pemain telah menemukan tangga untuk ke level selanjutnya, ketika pemain bertarung dengan musuh, dan ketika pemain kalah karena kehilangan *health point*. Hasil pengujian aturan main dapat dilihat pada Tabel 5. 3 Hasil pengujian aturan permain. Tampilan ketika pemain menemukan tangga turun dan mendapatkan pesan berupa level selanjutnya dapat dilihat pada Gambar 5. 10, tampilan ketika pemain bertarung dapat dilihat pada Gambar 5. 11 , dan tampilan ketika pemain kalah dapat dilihat pada Gambar 5. 12.

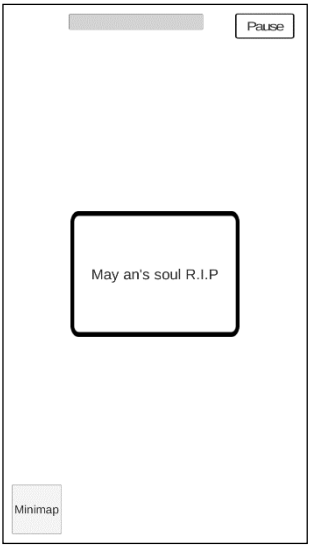
Tabel 5. 3 Hasil pengujian aturan permainan

ID	PF-002
Nama	Pengujian Aturan Main
Tujuan uji coba	Pengguna memainkan *game* sesuai aturan main
Kondisi awal	Pemain berada pada pada *gamei* dan telah memasukkan nama
Skenario 1	***Pemain menemukan tangga turun***
Masukan	Tidak ada
Keluaran yang diharapkan	Pesan untuk level selanjutnya
Hasil uji coba	Berhasil
Kondisi Akhir	Muncul pesan untuk level selanjutnya
Skenario 2	***Pemain bertarung dengan musuh***
Masukan	Pemain menggerakan karakter kea rah musuh
Keluaran yang diharapkan	Darah pemain berkurang
Hasil uji coba	Berhasil
Kondisi Akhir	Darah pemain berkurang
Skenario 3	***Pemain kehilangan seluruh health points***
Masukan	Tidak ada
Keluaran yang diharapkan	Munculnya pesan kalah
Hasil uji coba	Berhasil
Kondisi Akhir	Munculnya pesan kalah



Gambar 5. 10 Pesan menang barupa level pelanjutnya

Gambar 5. 11 Bertarung



Gambar 5. 12 Pesan kalah

5.3 Evaluasi Pengujian Fungsionalitas

Rangkuman hasil pengujian fungsionalitas dapat dilihat pada Tabel 5. 6. Berdasarkan data pada tabel tersebut, dapat disimpulkan bahwa semua skenario pengujian berhasil dijalankan. Sehingga dapat disimpulkan bahwa fungsionalitas dari aplikasi telah bekerja sesuai dengan yang diharapkan.

Tabel 5. 4 Hasil pengujian fungsionalitas

Kode Pengujian	Nama	Hasil
PF001-1	Pengujian Labirin (Skenario 1)	Berhasil
PF001-2	Pengujian Labirin (Skenario 2)	Berhasil
PF001-3	Pengujian Labirin (Skenario 3)	Berhasil
PF002-1	Pengujian Aturan Main (Skenario 1)	Berhasil
PF002-2	Pengujian Aturan Main (Skenario 2)	Berhasil
PF002-3	Pengujian Aturan Main (Skenario 3)	Berhasil

(Halaman ini sengaja dikosongkan)

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, juga diberi saran untuk pengembangan aplikasi kedepannya.

6.1 Kesimpulan

Dari hasil pengamatan selama proses pengerjaan mulai dari proses perancangan, implementasi, dan pengujian yang telah dilakukan terhadap aplikasi, diambil kesimpulan sebagai berikut :

1. Penggunaan Artificial Ant Colony Algorithm dapat digunakan untuk membangkitkan labirin yang berbeda-beda (dinamis) dan dengan waktu yang optimal (5 detik) untuk *game* 2D berbasis Android yang dapat dibuktikan pada Tabel 5. 1 dan Tabel 5. 2.
2. Skenario permainan telah diimplementasikan pada permainan ‘Ant Cave’ berupa level yang terdapat pada permainan.
3. Aturan permainan telah diimplementasikan pada permainan ‘Ant Cave’ dan dapat dibuktikan pada Tabel 5. 3.

6.2 Saran

Adapun saran yang diberikan untuk mengembangkan *game* ini adalah *game* ini masih memerluka pengembangan lebih lanjut, terutama dalam hal pengaturan level atau tingkat kesusahan dan juga dari segi tampilan.

(Halaman ini sengaja dikosongkagn)

DAFTAR PUSTAKA

[1] "Maze generation algorithm," *Wikipedia*. 19-Mei-2017.

[2] D. Powlap, "Designing Mazes for 2D Games by Artificial Ant Colony Algorithm," 2016.

[3] W. Software, *Endless Depths 2 Roguelike RPG*. Whatentar Software, 2016.

[4] Avalon-Games, *The Greedy Cave*. Avalon-Games, 2017.

[5] "Unity (game engine)," *Wikipedia*. 11-Des-2016.

[6] "C sharp," *Wikipedia bahasa Indonesia, ensiklopedia bebas*. 13-Jun-2013.

[7] "4 Directional character by Warren Clark," *itch.io*. [Daring]. Tersedia pada: https://lionheart963.itch.io/4-directional-character. [Diakses: 16-Jun-2017].

[8] "Free Pixel Mob! by Henry Software," *itch.io*. [Daring]. Tersedia pada: https://henrysoftware.itch.io/free-pixel-mob. [Diakses: 16-Jun-2017].

[9] "Witchcraft - Sprite sheet by Arks," *itch.io*. [Daring]. Tersedia pada: https://arks.itch.io/witchcraft-spritesheet. [Diakses: 16-Jun-2017].

[10] "Skeleton Sprite Pack by Jesse M," *itch.io*. [Daring]. Tersedia pada: https://jesse-m.itch.io/skeleton-pack. [Diakses: 16-Jun-2017].

[11] "Canyon Rocks Tiles by Petey90," *itch.io*. [Daring]. Tersedia pada: https://petey90.itch.io/canyon-rocks. [Diakses: 16-Jun-2017].

[12] Redshrike, "16x16 indoor rpg tileset: the baseline," *OpenGameArt.org*, 11-Jan-2011. [Daring]. Tersedia pada: https://opengameart.org/content/16x16-indoor-rpg-tileset-the-baseline. [Diakses: 16-Jun-2017].

(Halaman ini sengaja dikosongkan)

BIODATA PENULIS

Andi Akram Yusuf, lahir pada tanggal 6 Agustus 1996 di Ujung Pandang, Sulawesi Selatan. Hobi yang dimiliki adalah membaca, menonton film, bermain game, dan berolahraga. Penulis menempuh pendidikan mulai dari SD Ujung Pandang Makassar (2001-2007), SMP Ujung Pandang Makassar (2007-2010), SMAN 17 Makassar (20010-2013), dan Teknik Informatika ITS (2013-2017). Di jurusan Teknik Informatika ITS, penulis mengambil bidang minat Interaksi Grafika dan Seni (IGS) dan memiliki ketertarikan dalam eksplorasi teknologi di bidang *game*, *augmented reality*, dan *virtual reality*. Selama perkuliahan, penulis aktif dalam organisasi kemahasiswaan, antara lain Anggota dan Staf UKM ITS Badminton Community, Anggota Panitia National Seminar Of Technology Schematics 2014, Anggota Sie Dana dan Usaha Schematics 2015. Penulis dapat dihubungi melalui surel : andiakram59@yahoo.co.id.