TUGAS AKHIR – KI141502

Pembangkit *World* dan *Tileset* Dinamis pada *Game* BATTLE Of THRONE dengan Genre *Turn Based Strategy Game* Menggunakan Metode *Midpoint Displacement*

Yuan Akbarsyah Pandunegoro
NRP 5111100 182

Dosen Pembimbing
IMAM KUSWARDAYAN, S.Kom., M.T.
WIJAYANTI NURUL KHOTIMAH,S.Kom., M.Sc.

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2016

FINAL PROJECT – KI141502

Generating Dynamic World and Tileset Game On BATTLE Of THRONE By Genre Turn Based Strategy Game Using MIDPOINT DISPLACEMENT

Yuan Akbarsyah Pandunegoro
NRP 5111100 182

Advisor
IMAM KUSWARDAYAN, S.Kom., M.T.
WIJAYANTI NURUL KHOTIMAH,S.Kom., M.Sc.

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY
SEPULUH NOPEMBER INSTITUTE OF TECHNOLOGY
SURABAYA
2016

KATA PENGANTAR

Segala puji dan syukur penulis panjatkan kepada Allah SWT atas karunia dan kesempatan yang diberikanNya. Atas ijinNya, penulis akhirnya dapat menyelesaikan tugas akhir yang berjudul "PEMBANGKIT *WORLD* dan *TILESET* DINAMIS PADA *GAME* BATTLE of THRONE dengan *GENRE TURN BASED STRATEGY GAME* MENGGUNAKAN METODE *MIDPOINT DISPLACEMENT"*.

Proses pengerjaan tugas akhir ini juga tidak lepas dari bantuan berbagai pihak, baik secara langsung maupun tidak. Maka dari itu, pada kesempatan ini penulis ingin menyampaikan rasa terima kasih yang tulus dan sebesar-besarnya kepada semua pihak, khususnya:

1. Allah Subhanahu wa ta'ala yang telah melimpahkan rahmat, hidayah, dan, inayah-Nya sehingga penulis dapat menyelesaikan tugas akhir dengan baik.
2. Junjungan kita Nabi Muhammad Shallahu Alayhi Wasallam yang telah menjadi inspirasi dari penulis.
3. Ibu penulis, Evy Erviyanti, atas doa dan kasih sayangnya yang tulus serta motivasi, semangat dan dukungan yang tiada hentinya sehingga dapat mengantarkan penulis untuk menyelesaikan masa studinya.
4. Keluarga penulis, yang selalu memberikan doa, semangat, perhatian dan bantuan kepada penulis.
5. Calon Istri penulis, Indah Pudji Oktavia yang selalu memberikan doa dan support kepada penulis.
6. Partner penulis Rimby Kamesworo yang telah banyak menemani penulis selama berkuliah diinformatika its.
7. Bapak Victor Hariadi, selaku dosen wali dan juga dosen bimbingan PKM penulis, yang selalu memberikan bimbingan dalam bidang akademis maupun non akademis, dukungan serta motivasi selama masa perkuliahan di Teknik Informatika ITS.
8. Bapak Imam Kuswardayan selaku dosen pembimbing Tugas Akhir pertama dan yang telah memberikan arahan dalam pengerjaan Tugas Akhir ini.

9. Ibu Wijayanti Nurul Khotimah selaku dosen pembimbing Tugas Akhir kedua yang dengan sabar membimbing penulis dalam pengerjaan Tugas Akhir ini.
10. Segenap dosen dan karyawan Teknik Informatika ITS atas ilmu dan pengalaman yang telah diberikan selama penulis menjalani masa studi di Teknik Informatika ITS.
11. Wiby, Iswah, Habibi, Ilmi, Umak, Komo, Vendoc, Bagus, Ujek, Cokro, Alex, Fian, Royan, mas Hidari dan teman-teman seperjuangan TC 2011 yang selalu memberikan bantuan, semangat, canda tawa serta kebersamaan selama masa perkuliahan.
12. Serta pihak-pihak dan teman-teman yang tidak bisa di sebutkan satu persatu yang telah membantu penulis menyelesaikan studinya.

Akhirnya penulis menyadari masih banyak kekurangan dalam tugas akhir ini. Untuk itu penulis mengharapkan adanya kritik dan saran yang dapat menyempurnakan tugas akhir ini. Harapan penulis, semoga tugas akhir ini dapat bermanfaat bagi pembaca.

Surabaya, Januari 2016
Penulis

Yuan Akbarsyah Pandunegoro

LEMBAR PENGESAHAN

Pembangkit *World* dan *Tileset* Dinamis pada *Game* BATTLE Of THRONE dengan Genre *Turn Based Strategy Game* Menggunakan Metode *Midpoint Displacement*

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:
YUAN AKBARSYAH PANDUNEGORO
NRP 5111 100 182

Disetujui oleh Dosen Pembimbing Tugas Akhir

IMAM KUSWARDAYAN, S.Kom., M.T.
NIP 197612152003121001 ..... (pembimbing 1)

WIJAYANTI NURUL KHOTIMAH, S.Kom., M.Sc.
NIP 198603122012122004 ..... (pembimbing 2)

SURABAYA
JANUARI, 2016

Pembangkit *World* dan *Tileset* Dinamis pada *Game* BATTLE Of THRONE dengan Genre *Turn Based Strategy Game* Menggunakan Metode *Midpoint Displacement*

Nama Mahasiswa : Yuan Akbarsyah Pandunegoro
NRP : 5111 100 182
Jurusan : Teknik Informatika FTIf-ITS
Dosen Pembimbing I : IMAM KUSWARDAYAN, S.Kom., M.T.
Dosen Pembimbing II : WIJAYANTI NURUL KHOTIMAH,S.Kom., M.Sc.

ABSTRAK

Saat ini game memiliki perkembangan yang sangat pesat. Game saat ini juga tidak hanya pada perangkat komputer namun juga telah merabah ke perangkat mobile khususnya dalam sistem operasi android.

Pada perkembanganya selain bisa dimainkan secara bersama–sama, game juga dituntut untuk bisa memberikan tantangan bagi pengguna berupa level permainan. Untuk mendapatkan tantangan dari level permainan, digunakan pembangkit world dinamis.

Dalam tugas akhir ini akan dibangun suatu permainan bergenre turn-based strategy yang menggunakan pembangkit world menggunakan algoritma Midpoint Displacement yang merupakan salah satu algoritma pembangkit terrain. Algoritma ini dimulai dengan mengacak angka dari empat sudut persegi, kemudian mendapatkan nilai titik tengah dari persegi tersebut dengan menghitung rata-rata dari empat angka sudut tersebut kemudian ditambah angka acak. Kemudian mendapatkan empat titik tengah dari keempat sisi persegi. Kemudian diulangi langkah tersebut hingga semua titik terisi.

Dari hasil uji performa diketahui bahwa permainan yang telah dibangun berhasil membangkitkan world dengan waktu minimal 0,013 detik dan maksimal sampai 0,020 detik. Dan memiliki total rata-rata 0,0163 detik dari 20 kali percobaan. Dan juga

fungsionalitas dari permainan ini sudah di uji dan sesuai dengan apa yang diharapkan.

Kata kunci**: **Pembangkit World Dinamis, Turn-Based Strategy, Midpoint Displacement Algorithm, Perangkat Mobile, Sistem Operasi Android.

Generating Dynamic World and Tileset Game On BATTLE Of THRONE By Genre Turn Based Strategy Game Using MIDPOINT DISPLACEMENT

Student Name	: Yuan Akbarsyah Pandunegoro
NRP	: 5111 100 182
Major	: Teknik Informatika FTIf-ITS
Advisor I	: IMAM KUSWARDAYAN, S.Kom., M.T.
Advisor II	: WIJAYANTI NURUL KHOTIMAH,S.Kom., M.Sc.

ABSTRACT

. Currently the game has a very rapid development . Game today not only on computers but also have reached to a mobile device, especially in the android operating system.

On its development, not only be able to play together, the game also required to be able to provide a challenge for the user in the form of a game level . To get a challenge from the level of the game, used dynamic world generator

s.

In this final project nuilt an turn-based strategy game that uses dynamic world generators using Midpoint Displacement algorithm which is one of terrain generator algorithm. This algorithm begins to randomize the numbers from the four corners of the square , then get the value of the midpoint of the square by calculating the average of the four corner numbers are then added random numbers . Then get four midpoints of the four sides of a square . Then repeat these steps until all the points are filled.

From the results of the performance test is known that the game has been built successfully generated world with a time of 0.013 sec minimum and a maximum of up to 0.020 seconds. And have an average total 0.0163 seconds from 20 attempts. And also the functionality of this game is in conformity with what was expected.

***Keywords**: Dynamuc World Generator, Turn-Based Strategy, Midpoint Displacement Algorithm, Mobile Device, Android OS.*

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR KODE SUMBER

BAB I
PENDAHULUAN

Pada bagian ini akan dijelaskan beberapa hal dasar mengenai tugas akhir ini yang meliputi: latar belakang, tujuan, manfaat permasalahan, batasan permasalahan, metodologi serta sistematika penulisan tugas akhir. Penjelasan tentang hal-hal tersebut diharapkan dapat memberikan gambaran umum mengenai permasalahan sehingga penyelesaian masalah dapat dipahami dengan baik.

1.1. Latar Belakang

Perkembangan teknologi komputer pada saat ini berkembang sangat pesat, dimana yang telah kita ketahui dalam instansi pemerintahan maupun swasta, lebih mengutamakan menggunakan teknologi komputer dalam menyelesaikan pekerjaannya, dan juga bagi para pelajar atau mahasiswa/i serta dosen dalam mempermudah proses belajar mengajar.

Salah satu hasil dari perkembangan teknologi ini adalah *game*. *Game* merupakan sebuah alat rekreasi dan hiburan yang sangat popular dimasyarakat. Hampir seluruh lapisan masyarakat saat ini pernah memainkan sebuah *game*.

Pada perkembanganya selain bisa dimainkan secara bersama-sama, *game* juga dituntut untuk bisa memberikan tantangan bagi pengguna berupa *level* permainan. Untuk mengurangi tingkat kejenuhan pemain dan mendapatkan tantangan baru dari *level* permainan digunakan sistem *random map* atau *Battle field* yang dinamis.

Dengan berubah-ubahnya *map* yang di gunakan, diharapkan tantangan yang dihadapi pemain akan meningkat setiap permainannya dan tidak menyebabkan mudah bosan. Tugas akhir ini lebih difokuskan pada pembuatan *Battle field* Dinamis pada *Turn-Based Strategy Game*.

1.2. Rumusan Permasalahan

Rumusan masalah yang akan diangkat dalam Tugas Akhir ini dapat dipaparkan sebagai berikut:

1. Bagaimana aturan main, *level* dan skenario dalam permainan *Battle of Throne* yang bergenre *Turn-Based Strategy Game* ?
2. Merancang pembangkit *world* yang dinamis dengan metode *Midpoint Displacement*.
3. Merancang algoritma untuk menampilkan *tileset* yang sesuai dengan *world* yang akan di bangkitkan.
4. Bagaimana mengimplementasikan rancangan *world* generator pada unity yang ber-*platform mobile android* versi 4.2 keatas?

1.3. Batasan Permasalahan

Permasalahan yang dibahas dalam Tugas Akhir ini memiliki beberapa batasan, di antaranya sebagai berikut:

1. Aplikasi yang dibangun melalui komponen objek dua dimensi.
2. Dimensi yang akan dibangkitkan adalah 2n+1.
3. Dimensi yang diuji cobakan adalah 9x9 dan 17x17.

1.4. Tujuan

Tujuan dari pembuatan tugas akhir ini adalah membangun *Turn-Based Strategy Game* pada android yang mempunyai fitur pembangkit *Battle Field* yang dinamis menggunakan *Midpoint Displacement* serta mengaplikasikan *Tile-based game*.

1.5. Manfaat

Manfaat dari hasil pembuatan tugas akhir ini antara lain:

1. Pengguna mendapatkan tantangan baru disetiap permainan.
2. Memberikan media hiburan bagi pengguna.

3. Dapat digunakan sebagai eksplorasi untuk *development game* kedepan.

1.6. Metodologi

Pengerjaan tugas akhir ini menggunakan metodologi sebagai berikut:

A. Studi literatur

Pada studi literatur ini, akan dipelajari sejumlah referensi yang diperlukan dalam pembuatan aplikasi yaitu mengenai:

1. *Midpoint Displacement Algorithm.*
2. *Turn-Based Strategy*.
3. Instalasi pada *Platform* perangkat *Mobile* Android.
4. *Artificial Intelligence.*
5. *Tile-based Game.*

B. Perancangan perangkat lunak

Langkah-langkah yang akan digunakan pada tahap ini adalah sebagai berikut:

1. Pencarian dan pendataan materi yang akan digunakan dalam *game* Battle of Throne.
2. Perancangan sistem dan mekanisme *game* Battle of Throne.
3. Analisis kebutuhan non fungsional.
4. Analisis algoritma dan formula dalam membangun *game* Battle of Throne.
5. Perancangan pembangkit *world* dinamis pada *game* Battle of Throne.

C. Implementasi dan pembuatan sistem

Aplikasi ini akan dibangun dengan bahasa pemrograman C# menggunakan *game engine* unity. Aplikasi yang akan dibangun berbasis perangkat *mobile* android.

D. Pengujian dan evaluasi
   1. Pengujian kotak hitam
      Pengujian kotak hitam adalah pengujian yang berfokus pada spesifikasi fungsional dari perangkat lunak, tester dapat mendefinisikan kumpulan kondisi *input* dan melakukan pengujian pada spesifikasi fungsional program. Pengujian ini dilakukan untuk menguji apakah proses kinerja *game* ini sudah sesuai dengan kebutuhan pengguna atau tidak.

   2. Pengujian usabilitas
      Pengujian usabilitas dilakukan dengan cara melakukan survei ke pengguna yaitu beberapa pengguna yang suka bermain *game* di sekitar lingkungan ITS. Survei dilakukan untuk mengukur tingkat kegunaan dari aplikasi yang dibuat dalam membantu pengguna.

E. Penyusunan laporan tugas akhir
   Pada tahap ini dilakukan penyusunan laporan yang berisi dasar teori, dokumentasi dari perangkat lunak, dan hasil-hasil yang diperoleh selama pengerjaan tugas akhir.

1.7. Sistematika Penulisan

Buku tugas akhir ini terdiri dari beberapa bab, yang dijelaskan sebagai berikut.

BAB I PENDAHULUAN
Bab ini berisi latar belakang masalah, rumusan dan batasan permasalahan, tujuan dan manfaat pembuatan tugas akhir, metodologi yang digunakan, dan sistematika penyusunan tugas akhir.

BAB II TINJAUAN PUSTAKA
Bab ini membahas dasar pembuatan dan beberapa teori penunjang yang berhubungan dengan pokok pembahasan yang mendasari penyusunan tugas akhir.

BAB III ANALISIS DAN PERANCANGAN

Bab ini membahas analisis dari sistem yang dibuat meliputi analisis permasalahan, deskripsi umum perangkat lunak, spesifikasi kebutuhan, dan identifikasi pengguna. Kemudian membahas rancangan dari sistem yang dibuat meliputi rancangan skenario kasus penggunaan, arsitektur, data, dan antarmuka.

BAB IV IMPLEMENTASI

Bab ini membahas implementasi dari rancangan sistem yang dilakukan pada tahap perancangan. Penjelasan implementasi meliputi implementasi pembangkitan area permainan, dan antarmuka permainan.

BAB V PENGUJIAN DAN EVALUASI

Bab ini membahas pengujian dari aplikasi yang dibuat dengan melihat keluaran yang dihasilkan oleh aplikasi dan evaluasi untuk mengetahui kemampuan aplikasi.

BAB VI PENUTUP

Bab ini berisi kesimpulan dari hasil pengujian yang dilakukan serta saran untuk pengembangan aplikasi selanjutnya.

BAB II
TINJAUAN PUSTAKA

Pada bab ini akan dibahas mengenai teori-teori yang menjadi dasar dari penyusunan tugas akhir. Teori-teori tersebut adalah Unity *(Game Engine)*, *Turn-Based Strategy*, *C#,* Battle of Throne, *Midpoint Displacement Algorithm,* dan Android.

2.1. *Unity (Game Engine)*

Unity 3D adalah sebuah *game engine* yang berbasis *cross-platform.* Unity dapat digunakan untuk membuat sebuah *game* yang bisa digunakan pada perangkat komputer, ponsel pintar android, iPhone, PS3, dan bahkan XBOX. Unity juga dapat menjadi sebuah *tool* yang terintegrasi untuk membuat *game*, arsitektur bangunan dan simulasi. Unity bisa digunakan untuk membuat PC *game* dan *Online game*. Pada *Online game* diperlukan sebuah *plugin*, yaitu Unity *Web Player*, sama halnya dengan *Flash Player* pada *Browser*.

Fitur *scripting* yang disediakan, Unity telah mendukung 3 bahasa pemrograman, JavaScript, C#, dan Boo. Fleksibel dan *easy moving*, *rotating*, dan *scaling objects* hanya perlu sebaris kode. Demikian pula dengan *duplicating, removing*, dan *changing properties*. *Visual properties Variables* yang di definisikan dengan *scripts* ditampilkan pada editor. Bisa digeser, *drag and drop*, bisa memilih warna dengan *color picker*. Berbasis .NET, artinya penjalanan program dilakukan dengan *Open Source* .NET *platform.*

Pada setiap *project* unity terdapat sebuah *Assets folder*. Isi dari *Assets folder* ditampilkan dalam bentuk panel *project* dalam editor unity. *Assets folder* adalah tempat untuk menyimpan semua komponen dari *game* seperti tingkatan *game* (*level scenes*), *scripts, 3D models*, tekstur, dan *file audio*.

Agar dapat menambahkan *assets* ke dalam *project* cukup dengan menarik (*drag*) file yang ingin ditambahkan ke dalam panel *project* atau dengan memilih menu *Assets* lalu *Import New Asset*. Agar dapat membuat *scene* baru maka gunakan tombol *Control*+N (pada

keyboard). Selanjutnya untuk menyimpan *scene* yang sedang aktif, gunakan *Control*+S (pada *keyboard*).

Panel *Hierarchy* menampung semua *game object* yang terdapat di *scene* yang sedang aktif. Beberapa dari *game object* tersebut berhubungan langsung ke *asset* seperti objek 3D. Objek yang terdapat pada *hierarchy* dapat diseleksi dan dihapus. Jika objek dihapus atau ditambahkan pada *scene*, maka objek tersebut juga akan hilang atau muncul pada *hierarchy*.

Unity menggunakan sebuah konsep yang disebut *parenting*. Ini digunakan untuk membuat sebuah *game object* menjadi anak dari *game object* yang lain. Tarik sebuah *game object* dan pindahkan tepat di atas tulisan *game object* yang akan dijadikan *parent* dalam *hierarchy*. *Game object* yang terdapat dalam sebuah *game object* lainnya akan mengikuti perpindahan dan perputaran ketika *game object parent* mengalami perubahan posisi [1].

2.2. *Turn-Based Strategy*

Turn-Based Strategy (TBS) *game* adalah *game* strategi, pemain mempunyai giliran untuk melakukan suatu aksi, kemudian menyelesaikan gilirannya. Setelah giliran pemain tersebut selesai maka pemain tersebut harus menunggu giliran berikutnya untuk melakukan aksi selanjutnya [2].

Hal inilah yang menjadi daya tarik *game turn-based strategy*, sebab pemain harus mengatur strategi yang tepat sebelum bertindak untuk dapat memenangkan permainan. Salah satu *game turn based strategy* yang cukup dikenal adalah King's Bounty. King's Bounty adalah *game turn based strategy* yang didesain oleh Jon Van Canegham dari *New World Computing* pada tahun 1990. *Game* ini menggunakan tema fantasi, dimana *player* bermain sebagai ksatria dari Raja Maximus yang bertugas untuk mengambil kembali *Sceptre of Order* dari tangan *Forces of Chaos*.

2.3. *Bahasa Pemrograman C#*

Microsoft C# adalah sebuah bahasa pemrograman yang didesain untuk membangun jangkauan aplikasi *enterprise* yang berjalan di atas

framework .NET. Sebuah evolusi Microsoft C dan Microsoft C++, C# sederhana, moderen, aman dan *object oriented.* C# dikenal sebagai visual C# dalam visual studio .Net. Dukungan untuk visual C# termasuk proyek *template*, desainer, halaman poperti, kode, model objek, dan fitur lain dari lingkungan pengembangan. *Library* untuk pemrograman visual c# adalah .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. Menurut standar ECMA-334 *C# language specification*, nama C# terdiri atas sebuah huruf Latin C (U+0043) yang diikuti oleh tanda pagar yang menandakan angka # (U+0023). Tanda pagar # yang digunakan memang bukan tanda kres dalam seni musik (U+266F), dan tanda pagar # (U+0023) tersebut digunakan karena karakter kres dalam seni musik tidak terdapat di dalam *keyboard* standar.

Bahasa pemrograman C# digunakan dalam mengembangkan komponen perangkat lunak yang mampu mengambil keuntungan dari lingkungan terdistribusi. C# digunakan untuk menulis program aplikasi baik dalam sistem *client-server* (*hosted system*) maupun sistem *embedded* (*embedded system*), mulai dari perangkat lunak yang sangat besar yang menggunakan sistem operasi yang canggih hingga kepada perangkat lunak yang sangat kecil yang memiliki fungsi-fungsi terdedikasi. Meskipun aplikasi C# ditujukan agar bersifat 'ekonomis' dalam hal kebutuhan pemrosesan dan memori komputer, bahasa C# tidak ditujukan untuk bersaing secara langsung dengan kinerja dan ukuran perangkat lunak yang dibuat dengan menggunakan bahasa pemrograman C dan bahasa rakitan [3].

2.4. Final Fantasy Tactic

Game serupa yang bergenre *Turn-Based Strategy Game* dan memakai *world* generator adalah Final Fantasy Tactic yang diluncurkan oleh Square Enix untuk *game console* PS 1, Final Fantasy Tactic termasuk pada *strategy role-playing game*. Tidak seperti *game* Final Fantasy terdahulu, pertarungan di Final Fantasy Tactic di lakukan

di atas *map* yang di bagi berdasarkan *grid-grid* dan aksi yang dapat dilakukan pemain terbatas pada giliran. Pemain akan menjalankan beberapa karakter sekaligus layaknya permainan *strategy turn-based* pada umumnya.

Daya tarik utama dari permainan ini adalah kemampuan pemain untuk melakukan kostumisasi terhadap unit-unit yang dimiliki. Seiring dengan berjalannya permainan, *level* dari setiap karakter yang kita miliki akan bertambah. Dengan bertambahnya *level* karakter, beberapa pilihan kostumisasi akan dapat dipilih. Pilihan-pilihan ini antara lain adalah penambahan *skill point* pada atribut-atribut yang ada, pergantian *job*, dan penambahan *skill* karakter. Pemain juga dapat mengganti *equipment* dari karakter yang dijalankan. Hal ini menyebabkan pemain dapat membuat sendiri karakter mulai dari sebuah petarung garis depan yang bertarung dengan tinjunya sampai seorang penyihir yang dapat menghanguskan medan perang dengan satu *spell*.

Dalam Final Fantasy Tactic kamu akan berperan sebagai Ramza, seorang anak bangsawan yang terlibat dalam sebuah perang saudara yang terjadi di tempat dia tinggal. Sebagai seorang anak bangsawan, Ramza bersahabat karib dengan Delita yang merupakan rakyat biasa. Namun karena adanya kejenjangan sosial di Ivalice, persahabatan mereka sering kali terganggu hingga ke satu poin dan kejadian yang membuat Delita tidak tahan lagi untuk meninggalkan Ramza yang merupakan seorang bangsawan dan akhirnya mulai terjerumus melibatkan dirinya untuk memiliki peran besar di perang saudara yang terjadi [4].

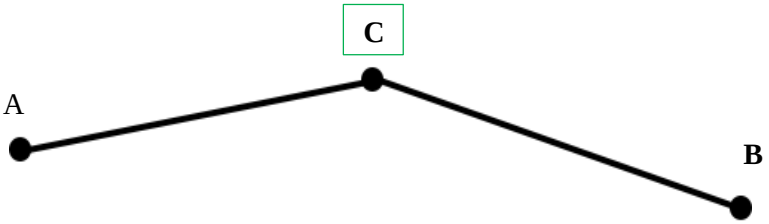
2.5. *Midpoint Displacement* Pada 1 Dimensi

Algoritma *Midpoint Displacement* merupakan algoritma yang digunakan untuk pembangkitan citra *terrain*. *Terrain* adalah bentuk permukaan bumi yang bisa dinaikkan atau direndahkan dengan memodifikasi elevasinya. Terdapat beberapa algoritma yang bisa digunakan untuk membangkitkan atau membentuk suatu *terrain*. Salah

satu dari algoritma pembangkit *terrain* adalah algoritma *Midpoint displacement*.

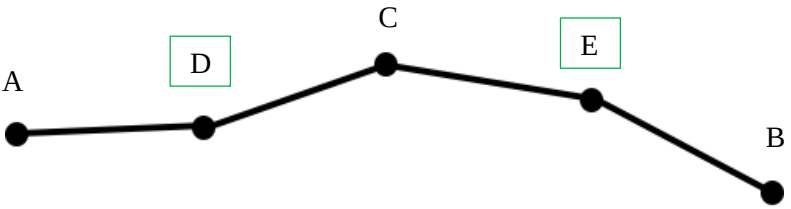
Midpoint displacement atau algoritma plasma, merupakan algoritma pembagian. *Terrain* dibentuk secara iteratif, pada setiap iterasi tingkat ketelitian bertambah. Algoritma ini dikonsep untuk membangkitkan *terrain* kotak dengan dimensi $(2^n + 1)$ x $(2^n + 1)$, dimana *n* merupakan jumlah iterasi. Dengan jumlah iterasi 8, berarti 257 x 257 titik *grid* akan dibangkitkan. Semakin besar jumlah iterasi, berarti jumlah memori yang digunakan akan semakin besar.

Pertama menentukan nilai acak/*random* pada ujung *terrain*. Rentang nilai *random* bebas, bisa 0-1 atau 50-100. Nilai tersebut dapat di ubah tergantung pada sistem yang akan digunakan. Lalu dari dua titik terluar yaitu titik A dan B diambil titik tengahnya dengan cara menghitung nilai rata-rata kedua poin dan menambahkan nilai acak yang menghasilkan titik C.



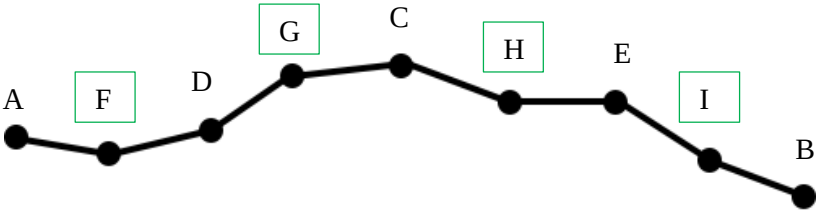
Gambar 2. 1 Langkah Awal Midpoint

Pada Gambar 2.1 diatas telah didapat 3 poin (A,B,C), lalu langkah berikutnya adalah mencari rata-rata tengah dari 2 titik tersebut dengan cara mencari nilai rata-rata dari dua poin yang berdekatan ditambah dengan nilai acak/*random*. Pada Gambar 2.2 titik D didapat dengan cara mengambil rata-rata dari titik A dan C dan ditambah dengan nilai acak, begitu juga dengan titik E didapat dari hasil rata-rata dari titik B dan C ditambah dengan nilai acak.



Gambar 2. 2 Langkah kedua Midpoint Displacement

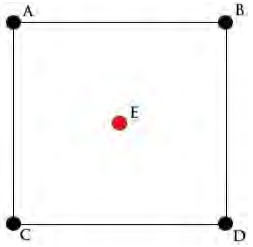
Pada Gambar 2.3, langkah selanjutnya adalah dengan mencari rata-rata dari setiap segmen yang ada dengan mencari rata-rata dari dua poin terdekat ditambah dengan nilai acak. Langkah ini berlangsung sampai pada jumlah iterasi yang diinginkan [3].



Gambar 2. 3 Langkah ketiga *Midpoint Displacement*

2.6. *Midpoint Displacement* Pada 2 Dimensi

Pada Gambar 2.4 langkah pertama adalah dengan menentukan titik-titik terluar ditambah dengan nilai acak, setelah itu mencari titik tengah dari kotak tersebut dengan cara mengambil rata-rata dari 4 titik terluar.

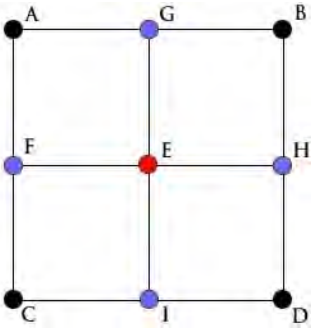


Gambar 2. 4 Midpoint Displacement 2 Dimensi

$$E = (A + B + C + D)/4 + rand$$

Persamaan 2. 4 Rumus yang didapat dari langkah 1

Lalu pada Gambar 2.5 menentukan titik-titik tengah diantara empat sudut segmen dengan mengambil rata-rata dari 2 sudut/titik yang berantara. Sehingga titik-titik tengah yang di dapat adalah titik F didapat dari titik A ditambah dengan titik C lalu di ambil rata-ratanya dan ditambah dengan nilai *random*, begitu juga dengan titik G, H dan I. Perhitungan *midpoint* dari empat buah titik tersebut adalah :

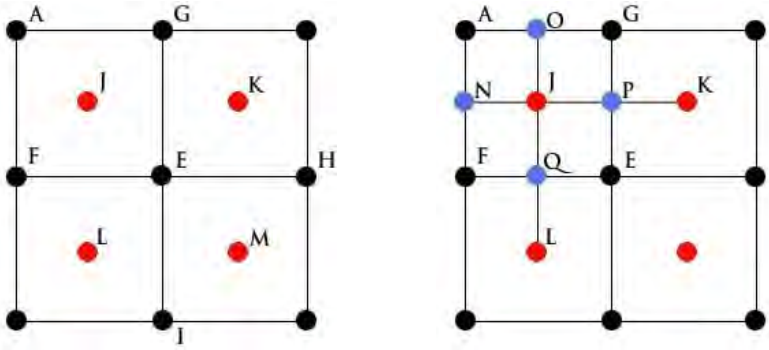


Gambar 2. 5 Midpoint Displacement 2 Dimensi (Langkah 2)

$$F = (A + C)/2 + rand$$
$$G = (A + B)/2 + rand$$
$$H = (B + D)/2 + rand$$
$$I = (C + D)/2 + rand$$

Persamaan 2. 1 Rumus yang didapat dari langkah 2

Selanjutnya pada Gambar 2.6 bagian kiri, melakukan hal yang sama dengan langkah satu pada Gambar 2.4 yaitu mencari titik tengah dari setiap kotak. kemudian pada Gambar 2.6 kanan melakukan langkah 2 (seperti pada Gambar 2.5) yaitu mencari titik tengah dari setiap sisi kotak, selanjutnya lakukan perulangan seperti langkah 1 dan 2 sebanyak jumlah iterasi yang di perlukan [5].



Gambar 2. 6 *Midpoint Displacement* 2 Dimensi (Langkah 3)

2.7. *Tileset*

Tileset atau terkadang di sebut juga sebagai *sprite sheet* adalah kumpulan gambar yang lebih kecil dan disebut sebagai *tile* yang di gabungkan dengan Gambar yang lebih besar. *Tile*set biasa di gunakan didalam *video games* 2D untuk membuat peta yang kompleks dari *tile-tile* yang dapat digunakan berulang kali didalam *set*. Ketika *tileset* digunakan pada *map* akan ditampilkan, *tile* tersebut disimpan bersama

map tersebut akan digunakan kembali untuk membuat *map* yang baru. Teknik ini telah digunakan di dalam permainan yang di kembangkan oleh Nitendo *Game Boy Advance system*. Keuntungan menggunakan *tileset* adalah mengurangi penggunaan sistem memori yang dibutuhkan untuk menampilkan *map* karena menggunakan *tileset* yang sama yang digunakan kembali. Keuntungan lainnya menggunakan *tileset* adalah mengurangi pembuatan *art work* untuk satu *map* karena dari satu *tileset* dapat menghasilkan *map* yang beragam [6].

2.8. Android

Android adalah sistem operasi berbasis Linux yang dirancang untuk perangkat seluler layar sentuh seperti telepon pintar dan komputer tablet. Android awalnya dikembangkan oleh android, Inc., dengan dukungan finansial dari Google, yang kemudian membelinya pada tahun 2005. Sistem operasi ini dirilis secara resmi pada tahun 2007, bersamaan dengan didirikannya Open Handset Alliance, konsorsium dari perusahaan-perusahaan perangkat keras, perangkat lunak, dan telekomunikasi yang bertujuan untuk memajukan standar terbuka perangkat seluler. Ponsel android pertama mulai dijual pada bulan Oktober 2008 [7].

BAB III
ANALISIS DAN PERANCANGAN

Pada bab ini akan dibahas mengenai analisis dan perancangan yang akan digunakan untuk menyelesaikan tugas akhir.

3.1. Analisis Sistem

Permainan video berkembang sangat pesat akibat tuntutan perkembangan zaman. Masyarakat berminat terhadap permainan video yang tentunya menyenangkan. Salah satu faktor yang mempengaruhi tingkat kesenangan pemain dari permainan video yaitu *world* atau *battlefield* dari permainan tersebut. *World* permainan yang beragam tentunya memberikan variasi tingkat kesulitan pada permainan. Pembuatan *world* permainan secara manual dengan *modeling* memakan waktu cukup lama. Dibutuhkan suatu sistem yang dapat membangkitkan *world* permainan setiap kali suatu *level* ingin diciptakan.

Aplikasi ini dibangun dengan tujuan membantu para pemain untuk dapat menikmati permainan yang variatif dan otomatis sekaligus dapat menambah pemikiran strategis. *World* pada sistem permainan dibangun secara dinamis. Aplikasi ini diharapkan dapat memberikan pembelajaran yang menyenangkan kepada para pengguna tanpa meninggalkan unsur *fun* pada suatu *game*.

Penulis menggunakan teknologi Unity *(Game Engine)* dengan Bahasa pemrograman C# untuk memfasilitasi pengembangan permainan. *World* pada permainan ini dibangun dengan menggunakan *midpoint displacement algorithm.*

3.2. Perancangan Perangkat Lunak

Tahap perancangan dalam subbab ini dibagi menjadi beberapa bagian yaitu deskripsi umum perangkat lunak, spesifikasi kebutuhan fungsional, spesifikasi kebutuhan nonfungsional, dan karakteristik pemain.

3.2.1. Deskripsi Umum Perangkat Lunak

Tugas akhir yang akan dikembangkan adalah sebuah permainan 2D dengan genre *turn-base stategy* dan *tile-based game. World* dari permainan ini dibangkitkan menggunakan *midpoint displacement algorithm. World* yang dibangkitkan mempunyai dua macam ukuran, yaitu 9x9 atau 17x17. Permainan ini akan dijalankan pada perangkat *smartphone* android agar dapat dimainkan secara fleksibel.

Pengguna utama dari permainan ini adalah semua orang yang ingin bermain. Pemain akan menjalankan pasukan dari sebuah kerajaan yang bertujuan untuk menguasai kerajaan lawan. Kerajaan lawan akan dijalankan oleh kecerdasan buatan (AI). Untuk memenangkan permainan ini, raja dari pasukan harus menguasai kastil kerajaan lawan. Agar dapat membantu raja menguasai kastil, pemain dapat membeli berbagai macam pasukan dengan uang yang didapatkan ketika giliran pemain.

Terdapat 3 macam tingkat kesulitan yang antara lain adalah *hard, medium* dan *easy.* Tingkat kesulitan yang dipilih dapat mempengaruhi *tile* yang akan ditampilkan, tolok ukur dari tingkat kesulitan dari setiap *map* yang dibangkitkan adalah banyaknya jumlah *tile* bergolongan *hard* sampai *easy* yang dibangkitkan. Urutan *tile* dari yang susah sampai mudah adalah *tile* air/*water*, pasir/*sand*, hutan/*forest*, padang rumput/*grass*. Setiap *tile* dapat mempengaruhi pergerakan dari pemain, sehingga semakin banyak *tile Hard* yang ditampilkan maka semakin berkurang jarak pergerakan dari setiap pemain.

3.2.2. Spesifikasi Kebutuhan Fungsional

Berdasarkan deskripsi umum sistem, maka disimpulkan bahwa kebutuhan fungsional dari aplikasi adalah:

1. **Memilih ukuran *map***
   Ukuran *map* yang dapat dipilih oleh pemain adalah 9x9 dan 17x17, ukuran ini menentukan seluas apa *map* yang akan dibangkitkan.

2. **Memilih tingkat kesulitan**
   Pemain dapat memilih tingkat kesulitan dari *map* yang akan dibangkitkan dengan cara memilih dari 3 tingkat kesulitan yang disediakan yaitu *hard*, *medium* dan *easy*.

3. **Menjalankan pasukan**
   Menjalankan pasukan dari posisi awal menuju posisi yang diinginkan.

4. **Membeli Pasukan**
   Menambah jumlah karakter pemain dengan cara membeli pasukan pada menu *Shop.*

5. **Menguasai Kastil**
   Setelah mengalahkan raja dari pihak lawan pemain harus berada pada posisi kastil lawan untuk menguasainya dan mengakhiri permainan.

6. **Menyerang Pasukan Lawan**
   Agar dapat memenangkan permainan pemain harus mengalahkan pemain lainnya dengan cara menyerang karakter pemain sampai hp musuh kurang dari sama dengan 0.

7. **Mengakhiri Giliran**
   Jika pemain sudah menjalankan semua karakternya maka pemain harus mengakhiri gilirannya dan menunggu giliran musuh berakhir.

3.2.3. Spesifikasi Kebutuhan Non-Fungsional

Terdapat beberapa kebutuhan non-fungsional yang apabila dipenuhi, dapat meningkatkan kualitas dari permainan ini. Berikut daftar kebutuhan non-fungsional:

1. ***Framerate***
   Permainan ini harus mampu dimainkan secara lancar, tidak ada *lag* dan nyaman di mata. Sebagian besar permainan 2D

berjalan optimal dengan *framerate* 24-30 fps (*Frame per Second*). Untuk pembangkitan *level* secara dinamis dapat mempengaruhi kelancaran *framerate* di awal pembentukan, serta kelancaran *framerate* dipengaruhi dan tergantung oleh spesifikasi *smartphone* yang akan digunakan.

2. **Kebutuhan Grafis**

   Kenyamanan bermain berbanding lurus dengan kualitas grafis yang disajikan dalam permainan. Efek seperti animasi merupakan salah satu daya tarik dalam suatu permainan. Efek-efek ini bisa membuat *drop rate fps* dan permainan melambat (*lag*), karena membutuhkan tambahan komputasi.

3.2.4. Karakteristik Pemain

Berdasarkan deskripsi umum diatas, maka dapat diketahui bahwa pengguna yang akan menggunakan aplikasi ini ada dua orang, yaitu pemain yang memainkan permainan, dan pengembang. Karakteristik pengguna tercantum dalam Tabel 3.1.

Tabel 3. 1 Karakteristik Pemain

Nama Aktor	Tugas	Hak Akses Aplikasi	Kemampuan yang harus dimiliki
Pemain	Menjalankan setiap fungsi yang berada dalam *game.*	Memainkan permainan.	Tidak ada.

3.2.5. Perancangan Pembangkit *World*/*Map*

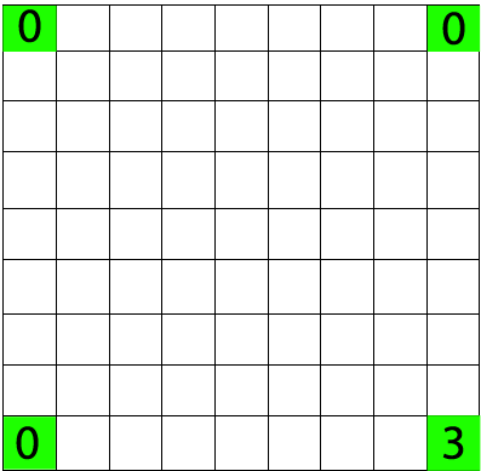
Dalam aplikasi tugas akhir ini, digunakan algoritma *Midpoint Displacement* untuk membangkitkan *world* secara dinamis. *Midpoint displacement* adalah salah satu algoritma yang digunakan untuk membangkitkan *map*. Algoritma ini membutuhkan *map* persegi yang berukuran 2^n+1.

Berikut ini adalah Gambaran langkah-langkah yang akan terjadi saat *world* dibangkitkan menggunakan algoritma *midpoint displacement.*

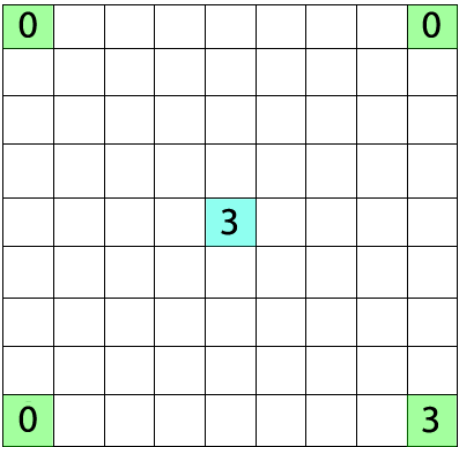
Pada Gambar 3.1 langkah pertama dari algoritma *Midpoint Displacement* adalah menentukan daerah terujung dari *terrain* dan memberikan nilai secara acak pada 4 titik sudut.

Pada Gambar 3.2, langkah kedua, menghitung titik tengah yang didapatkan dari nilai rata-rata dari keempat titik sudut yang kemudian ditambah dengan nilai *random.* Seperti yang terlihat pada Gambar 3.2, yaitu nilai pada warna biru didapatkan dari hasil rata-rata dari nilai pada warna hijau ditambah nilai *random.*

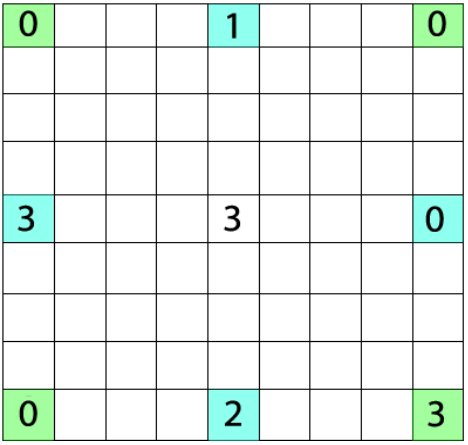
Selanjutnya pada Gambar 3.3 langkah ketiga pada algoritma *Midpoint Displacement* adalah menghitung titik tengah dari sisi terluar *world* dengan cara mencari nilai rata-rata dari 2 titik yang berdekatan, dari Gambar 3.3 terlihat bahwa nilai 1 yang berwarna biru berada diatas berasal dari rata-rata sudut kiri atas dan sudut kanan atas ditambah nilai *random* begitu juga dengan sisi kiri, kanan maupun bawah sehingga didapati 4 buah persegi.



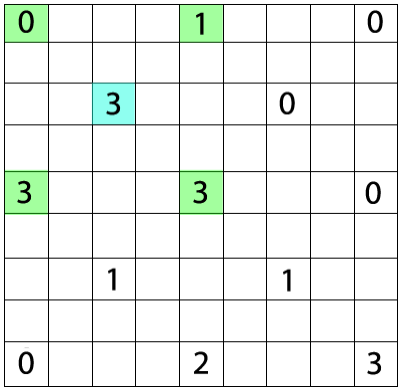
Gambar 3. 1 Implementasi *Midpoint* (Langkah 1)



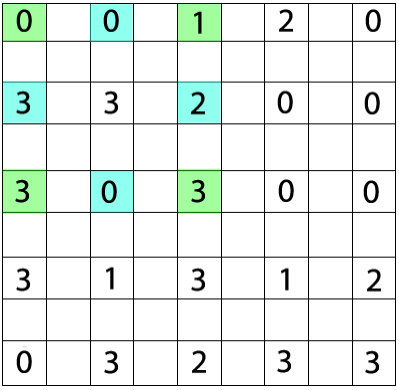
Gambar 3. 2 Implementasi *Midpoint* (Langkah 2)



Gambar 3. 3 Implementasi *Midpoint* (Langkah 3)

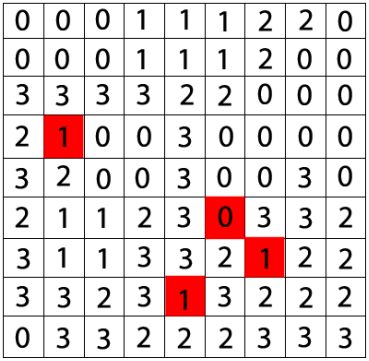


Gambar 3. 4 Implementasi *Midpoint* (Langkah 4)



Gambar 3. 5 Implementasi *Midpoint* (Langkah 5)

Pada Gambar 3.4 kemudian langkah keempat, memisah *world* tersebut menjadi empat bagian, lalu mengulangi langkah kedua sampai langkah kelima hingga nilai pada *world* tersebut terisi seluruhnya. Seperti yang terlihat pada Gambar 3.5 terlihat bahwa *map* setengah terisi, lalu nilai titik tengah yang berwarna biru diperoleh dari mendapatkan hasil rata-rata dari kotak-kotak yang berwarna hijau.



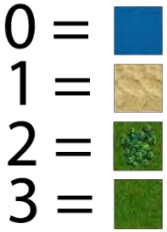
Gambar 3. 6 Implementasi *Midpoint* (Correction 1)

Selanjutnya untuk membangkitkan *tiles* yang direpresentasikan oleh nilai hasil dari perhitungan algoritma *Midpoint Displacement*, dilakukan pengecekan dari angka 0 dan 1 yang mempunyai sisi angka 0 dan 1 kurang dari 2, selanjutnya angka 0 dan 1 tersebut akan dirubah menjadi angka 2 atau 3 secara acak. Seperti yang terlihat pada Gambar 3.6 terdapat nilai 1 dan 0 berada sendiri.

Pada Gambar 3.7 merupakan hasil dari perbaikan dari nilai pada Gambar 3.6, dimana hasil nilai perbaikan adalah acak dari angka 2 sampai 3.

0	0	0	1	1	1	2	2	0
0	0	0	1	1	1	2	0	0
3	3	3	3	2	2	0	0	0
2	3	0	0	3	0	0	0	0
3	2	0	0	3	0	0	3	0
2	1	1	2	3	3	3	3	2
3	1	1	3	3	2	3	2	2
3	3	2	3	3	3	2	2	2
0	3	3	2	2	2	3	3	3

Gambar 3. 7 Implementasi *Midpoint* (Correction 2)

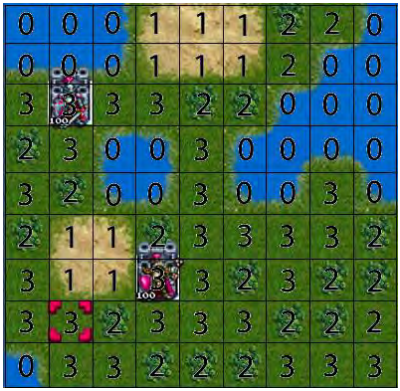


Gambar 3. 8 Macam-Macam *Tiles*

Pada Gambar 3.8 merupakan representasi dari nilai *random* yang didapatkan dari hasil *random* menggunakan *midpoint*

displcament, nilai 0 merepresentasikan *tile* air/*water,* nilai 1 merepresentasikan *tile* pasir/*sand,* nilai 2 merepresentasikan *tile* hutan/*forest,* nilai 3 merepresentasikan *tile* padang rumput/*grass* dan setiap *tile* yang dibangkitkan memiliki bobot yang dapat mempengaruhi jangkauan gerak dari *player*.

Pada Gambar 3.8 merupakan keterangan *tile* pada angka hasil pembangkitan *world* menggunakan algoritma *Midpoint Displacement*.

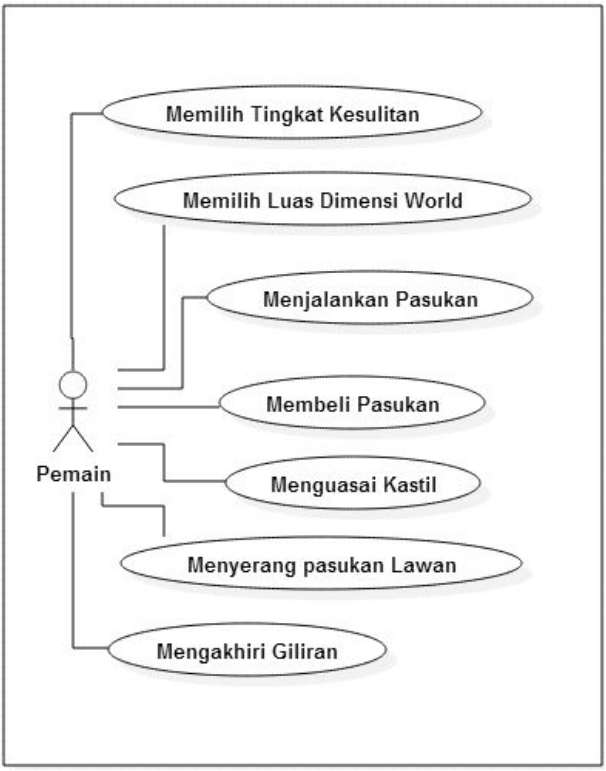


Gambar 3. 9 Contoh Implementasi *Map* dengan *Midpoint*

Pada Gambar 3.9 merupakan contoh hasil *world* implementasi *tiles* yang direpresentasikan oleh nilai yang diperoleh dari perhitungan menggunakan algoritma *Midpoint Displacement.* Dimana tiap nilai dalam kotak merepresentasikan *tile* tersendiri.

3.2.6. Perancangan Skenario Kasus Penggunaan

Pada subbab ini akan dijelaskan mengenai scenario kasus penggunaan yang ada pada tugas akhir ini. Terdapat 7 kasus penggunaan yang terdapat didalam sistem seperti yang tercantum pada Gambar 3.10 yaitu memilih tingkat kesulitan, memilih luas dimensi *world,* menjalankan pasukan, membeli pasukan, menguasai kastil, dan menyerang pasukan lawan.



Gambar 3. 10 Diagram Kasus Penggunaan

Penjelasan dari masing-masing kasus penggunaan dicantumkan pada Tabel 3.2. tabel tersebut berisi penjelasan skenario yang akan dilakukan ketika pengujian.

Tabel 3. 2 Skenario Kasus Penggunaan

No	Kode Kasus Penggunaan	Nama Kasus Penggunaan	Keterangan
1	UC-001	Memilih Tingkat kesulitan.	Pemain memilih tingkat kesulitan yang akan dimainkan.

2	UC-002	Memilih luas dimensi *world.*	Pemain memilih luas dimensi yang akan dimainkan.
3	UC-003	Menjalankan Pasukan.	Untuk menguji coba pemain dalam menjalankan pasukan.
4	UC-004	Membeli Pasukan.	Untuk menguji coba pemain dalam membeli pasukan.
5	UC-005	Menguasai Kastil.	Untuk menguji coba pemain dalam menguasai kastil.
6	UC-006	Menyerang Pasukan Lawan.	Untuk menguji coba pemain dalam menyerang pasukan lawan.
7	UC-007	Mengakhiri giliran.	Untuk menguji coba pemain dalam mengakhiri giliran.

3.2.6.1. Kasus Penggunaan Permainan

Penjelasan kasus penggunaan permainan untuk skenario UC-001 yakni memilih tingkat kesulitan dijelaskan pada Tabel 3.3.

Tabel 3. 3 Skenario Kasus Penggunaan Memilih tingkat kesulitan

Nama Kasus Penggunaan	Memilih tingkat kesulitan
Kode	UC-001
Deskripsi	Kasus penggunaan dimana aktor memilih tingkat kesulitan dari *map.*
Aktor	Pemain.
Kondisi Awal	Pemain sudah masuk ke aplikasi dan memilih menu *option*.
Alur Normal	1. Pemain memasuki halaman *option* dan memilih tingkat kesulitan yang tersedia. 2. Sistem menampilkan tombol yang sesuai dengan tingkat kesulitan yang dipilih. 3. Pemain kembali ke halaman awal lalu memilih *new game*. 4. Sistem menampilkan *map* yang sesuai dengan tingkat kesulitan yang di pilih.

Penjelasan kasus penggunaan permainan untuk skenario UC-002 yakni memilih luas dimensi *world* dijelaskan pada Tabel 3.4.

Tabel 3. 4 Skenario Kasus Penggunaan Memilih luas dimensi

Nama Kasus Penggunaan	Memilih luas dimensi *world*
Kode	UC-002
Deskripsi	Kasus penggunaan dimana aktor memilih luas dimensi dari *map.*
Aktor	Pemain.
Kondisi Awal	Pemain sudah masuk ke aplikasi dan memilih *option*.
Alur Normal	1. Pemain memasuki halaman *option* dan memilih tingkat kesulitan yang tersedia. 2. Sistem menampilkan tombol yang sesuai dengan tingkat kesulitan yang dipilih. 3. Pemain kembali ke halaman awal lalu memilih *new game*. 4. Sistem menampilkan *map* yang sesuai dengan tingkat kesulitan yang di pilih.

Penjelasan kasus penggunaan permainan untuk skenario UC-003 yakni menjalankan pasukan dijelaskan pada Tabel 3.5.

Tabel 3. 5 Skenario kasus penggunaan Menjalankan Pasukan

Nama Kasus Penggunaan	Menjalankan Pasukan
Kode	UC-003
Deskripsi	Kasus penggunaan dimana aktor menjalankan pasukan yang dipilih.
Aktor	Pemain.
Kondisi Awal	Pemain sudah masuk ke aplikasi dan memilih menu *new game*.
Alur Normal	1. Pemain memilih pasukan dengan menyentuh pasukan tersebut. 2. Sistem menampilkan menu aksi dari pasukan tersebut. 3. Pemain memilih menu jalan.

	4. Sistem menampilkan *tile* yang dapat didatangi oleh pasukan tersebut. 5. Pemain memilih salah satu dari *tile* tersebut. 6. Sistem menampilkan menu aksi dari pasukan tersebut. 7. Pemain memilih menu selesai**.**

Selanjutnya penjelasan kasus penggunaan permainan untuk skenario UC-004 yakni membeli pasukan dijelaskan pada Tabel 3.6.

Tabel 3. 6 Skenario penggunaan Membeli Pasukan

Nama Kasus Penggunaan	Membeli Pasukan
Kode	UC-004
Deskripsi	Kasus penggunaan dimana aktor membeli pasukan.
Aktor	Pemain.
Kondisi Awal	Pemain sudah masuk ke aplikasi dan memilih menu *new game*.
Alur Normal	1. Pemain memilih kastil dengan menyentuh kastil tersebut. 2. Sistem menampilkan menu aksi dari kastil tersebut. 3. Pemain memilih menu beli. 4. Sistem menampilkan toko pasukan. 5. Pemain memilih pasukan di toko tersebut dan menekan tombol beli. 6. Sistem membuat pasukan yang telah dibeli.

Kemudian penjelasan kasus penggunaan permainan untuk skenario UC-005 yakni Menyerang Pasukan Lawan dijelaskan pada Tabel 3.7.

Tabel 3. 7 Skenario penggunaan Menyerang Pasukan Lawan

Nama Kasus Penggunaan	Menyerang Pasukan Lawan
Kode	UC-005
Deskripsi	Kasus penggunaan dimana aktor menyerang pasukan lawan.

Aktor	Pemain.
Kondisi Awal	Pemain sudah masuk ke aplikasi dan memilih menu *new gam.*
Alur Normal	1. Pemain memilih pasukan yang berada di jarak serang pasukan lawan dengan menyentuh pasukan tersebut. 2. Sistem menampilkan menu aksi dari pasukan tersebut. 3. Pemain memilih menu serang. 4. Sistem menampilkan *tile* yang dapat diserang oleh pasukan tersebut. 5. Pemain memilih salah satu pasukan lawan yang berada di *tile* tersebut.

Selanjutnya penjelasan kasus penggunaan permainan untuk skenario UC-006 yakni Menguasai Kastil dijelaskan pada Tabel 3.8.

Tabel 3. 8 Skenario penggunaan Menguasai Kastil

Nama Kasus Penggunaan	Menguasai Kastil
Kode	UC-006
Deskripsi	Kasus penggunaan dimana aktor menguasai kastil.
Aktor	Pemain.
Kondisi Awal	Pemain sudah masuk ke aplikasi dan memilih menu *new game.*
Alur Normal	1. Pemain memilih raja yang berada di kastil yang belum dikuasai dengan menyentuh pasukan tersebut. 2. Sistem menampilkan menu aksi dari pasukan tersebut. 3. Pemain memilih menu kuasai.

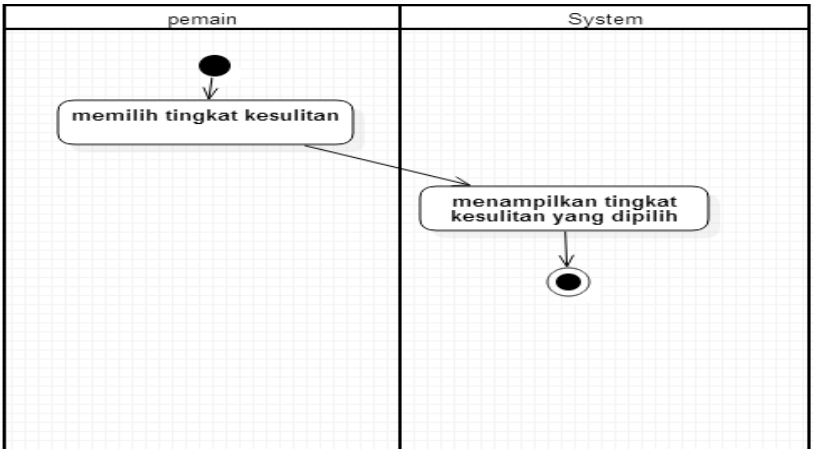
Kemudian penjelasan kasus penggunaan permainan untuk skenario UC-007 yakni Mengakhiri Giliran dijelaskan pada Tabel 3.9.

Tabel 3. 9 Skenario penggunaan Mengakhiri Giliran

Nama Kasus Penggunaan	Mengakhiri Giliran
Kode	UC-007
Deskripsi	Kasus penggunaan dimana aktor mengakhiri giliran.
Aktor	Pemain.
Kondisi Awal	Pemain sudah masuk ke aplikasi dan memilih menu *new game*.
Alur Normal	1. Pemain menyentuh tombol giliran selesai yang berada di pojok kanan bawah. 2. Sistem mengubah giliran musuh.

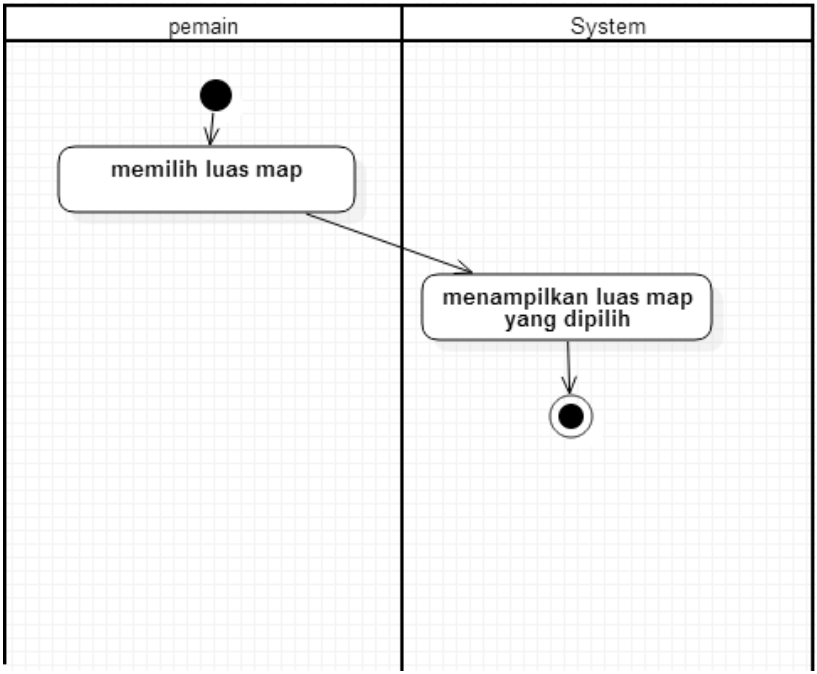
3.2.7. Diagram Aktifitas

Diagram aktivitas menampilkan langkah-langkah normal yang harus dilakukan pemain untuk menjalankan studi kasus permainan dimulai dari awal permainan hingga kondisi akhir. Berikut Gambar 3.11 diagram aktivitas untuk dari kasus penggunaan UC-001.



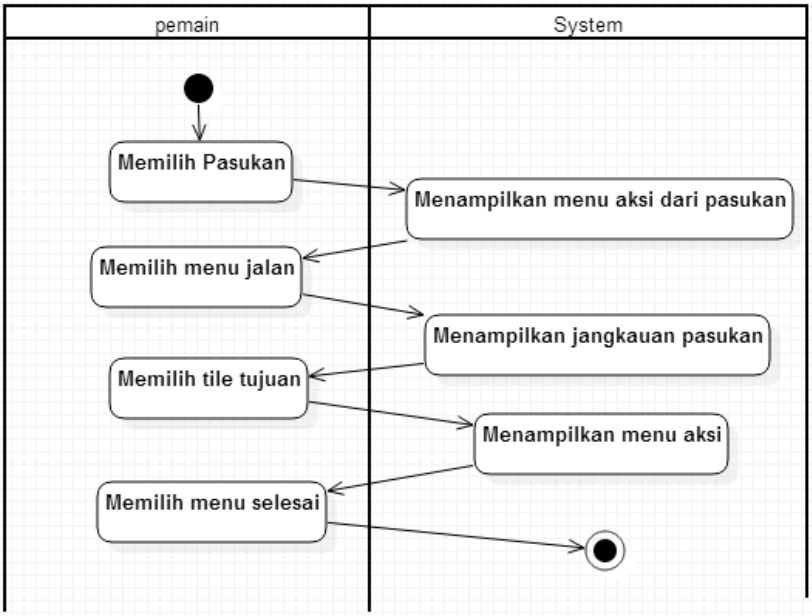
Gambar 3. 11 Diagram Aktifitas memilih tingkat kesulitan

Pada Gambar 3.11 merupakan diagram aktivitas untuk kasus penggunaan UC-001 yakni memilih tingkat kesulitan *world*/*map*. Diagram aktivitas menyatakan bahwa aktivitas dimulai dari memasuki menu *option* lalu memilih tingkat kesulitan yang akan digunakan dalam permainan.



Gambar 3. 12 Diagram Aktifitas memilih luas dimensi *world*

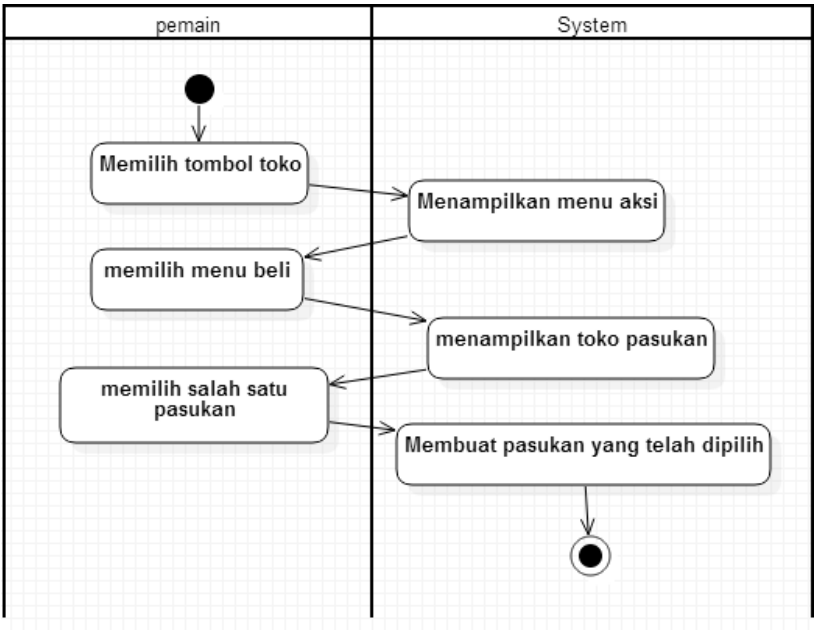
Kemudian pada Gambar 3.12 merupakan diagram aktivitas untuk kasus penggunaan UC-002 yakni memilih luas dimensi *world*. Diagram aktivitas menyatakan bahwa aktivitas dimulai dari memasuki menu *option* lalu memilih luas *map* yang akan digunakan. Dalam kasus penggunaan ini pemain dapat memilih luas dari *world* yang akan digunakan dalam permainan



Gambar 3. 13 Diagram Aktifitas Menjalankan Pasukan

Kemudian pada Gambar 3.13 merupakan diagram aktivitas untuk kasus penggunaan UC-003 yakni menjalankan pasukan. Diagram aktivitas menyatakan bahwa aktivitas dimulai dari memilih pasukan yang akan dijalankan, lalu sistem akan menampilkan jarak dari unit tersebut.

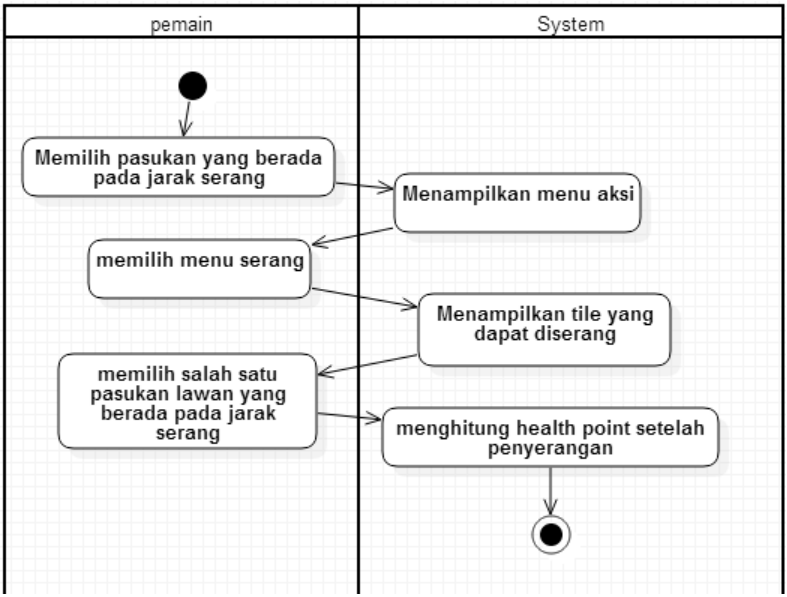
Kemudian pada Gambar 3.14 merupakan diagram aktivitas untuk kasus penggunaan UC-004 yakni Membeli Pasukan. Diagram aktivitas menyatakan bahwa aktivitas dimulai dari memilih kastil yang sudah dikuasai yang dilakukan oleh pengguna dan diakhiri dengan membuat pasukan yang telah dibeli yang dilakukan oleh sistem.



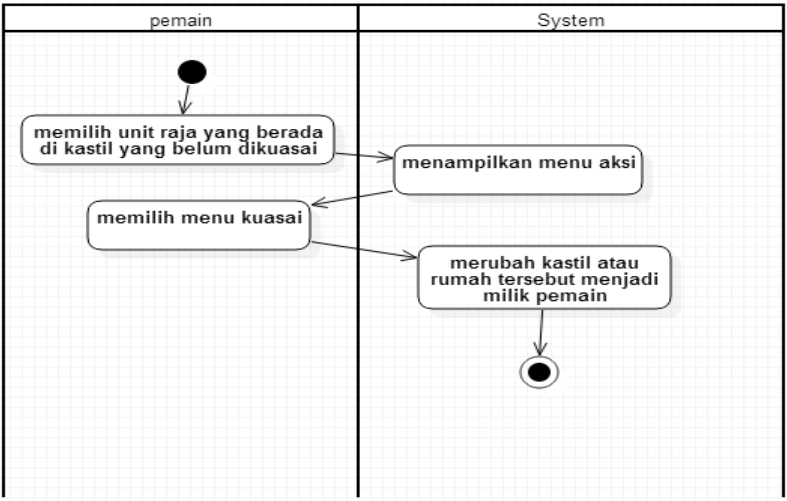
Gambar 3. 14 Diagram Aktifitas Membeli Pasukan

Kemudian Gambar 3.15 diagram aktivitas untuk kasus penggunaan UC-005 yakni Menyerang Pasukan Lawan. Diagram aktivitas menyatakan bahwa aktivitas dimulai dari pasukan yang berada di jarak serang pasukan lawan yang dilakukan oleh pengguna dan diakhiri dengan perhitungan *health point* yang dilakukan oleh sistem.

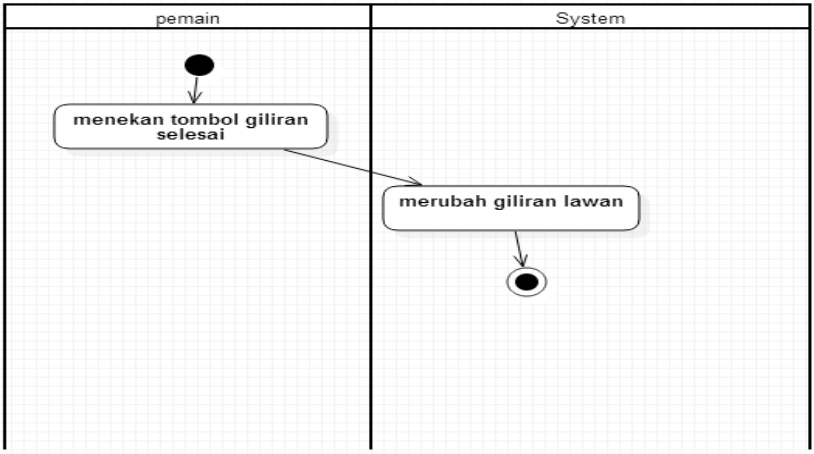
Kemudian pada Gambar 3.16 merupakan diagram aktivitas untuk dari kasus penggunaan UC-006 yakni Menguasai Kastil. Diagram aktivitas menyatakan bahwa aktivitas dimulai dari pasukan raja yang berada di kastil yang belum dikuasai yang dilakukan oleh pengguna dan diakhiri dengan mengubah kastil menjadi milik pemain yang dilakukan oleh sistem.



Gambar 3. 15 Diagram Aktifitas Menyerang Pasukan Lawan



Gambar 3. 16 Diagram Aktifitas Menguasai Kastil



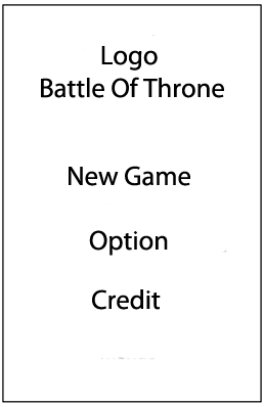
Gambar 3. 17 Diagram Aktifitas Mengakhiri Giliran

Kemudian pada Gambar 3.17 merupakan diagram aktivitas untuk dari kasus penggunaan UC-006 yakni Mengakhiri Giliran. Diagram aktivitas menyatakan bahwa aktivitas dimulai dari menekan tombol giliran selesai yang berada di bagian kanan bawah yang dilakukan oleh pengguna dan diakhiri dengan mengubah giliran lawan yang dilakukan oleh sistem.

3.2.8. Perancangan Antarmuka Pengguna

Pada subbab ini akan dibahas mengenai bagaimana rancangan antarmuka pengguna yang akan digunakan dalam tugas akhir ini. Rancangan antarmuka yang dibahas meliputi ketentuan masukan dan rancangan jendela tampilan. Adapun perancangan antaramuka yang akan dibahasa antara lain adalah tampilan *Main Menu,* tampilan *Option,* tampilan saat bermain, dan tampilan toko pasukan.

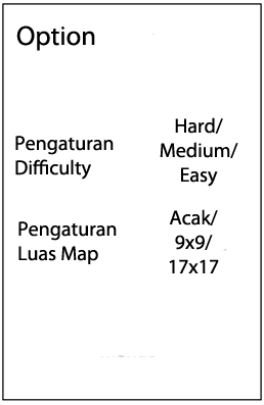
3.2.8.1. Tampilan *Main Menu*



Gambar 3. 18 Tampilan *Main Menu*

Tampilan *main menu* merupakan tampilan yang pertama kali muncul ketika aplikasi dijalankan dan pemain menyentuh layar, seperti pada Gambar 3.18 di atas.

3.2.8.2. Tampilan *Option*



Gambar 3. 19 Tampilan Option

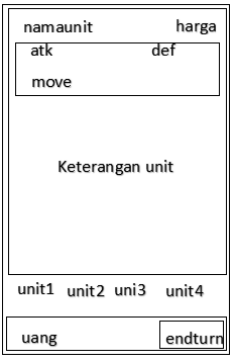
Tampilan *Option* merupakan tampilan yang muncul ketika pemain ingin mengubah setingan pada aplikasi, seperti pada Gambar 3.19 di atas.

3.2.8.3. Tampilan Saat Bemain

Gambar 3. 20 Tampilan Saat Bermain

Tampilan saat bermain, Gambar 3.20 merupakan tampilan yang muncul ketika pemain memilih *New Game* pada main menu. *World* pada tampilan saat bermain ini selalu berubah setiap permainan baru.

3.2.8.4. Tampilan Toko Pasukan



Gambar 3. 21 Tampilan Toko Pasukan

Tampilan toko pasukan, Gambar 3.21 merupakan tampilan yang muncul ketika pemain memilih menu beli pada menu aksi. Di dalam toko tersebut terdapat 4 macam pasukan yang mempunyai karakteristik dan harga yang berbeda-beda.

3.2.9. Perancangan Kontrol Permainan

Pada kontrol permainan ini menggunakan *tap* dan *swipe*. Semua interaksi dari *user* hanya menggunakan kontrol tersebut. *Tap* digunakan untuk menekan tombol dan memilih pasukan. Sedangkan *swipe* digunakan untuk menggeser *world* sesuai yang kita inginkan.

3.2.10. Perancangan Aturan Permainan

Aturan permainan pada permainan ini mengikuti aturan main dari *game* RPG *turn based* pada umumnya yaitu:

1. Memilih lebar *map* dari pilihan yang disediakan.
2. Memilih tingkat kesulitan *map* dari pilihan yang tersedia
3. *Tap* unit untuk menggerakkan pasukan yang diinginkan.
4. Kotak merah merupakan daya jelajah dari setiap unit.
5. Daya jelajah dari unit tersebut dipengaruhi oleh kemampuan jelajah dari unit tersebut dan medan yang dilewati.
6. Setelah unit dimainkan maka akan berubah menjadi abu-abu, yang menandakan bahwa unit tersebut sudah dijalankan pada giliran sekarang, dan tidak bisa dipakai lagi sampai giliran yang berikutnya.
7. Unit pasukan hanya dapat menyerang musuh apabila musuh tersebut memasuki jarak serang dari unit tersebut yang ditandai dengan kotak warna kuning.
8. Untuk menyerang pasukan musuh yang berada di jarak serang, gerakkan unit dengan memilih *icon* serang dan pilih serang.
9. Setelah selesai menggerakkan semua unit, tekan tombol *end turn* yang berada di pojok kanan bawah untuk mengakhiri giliran.

10. Hanya raja yang dapat menguasai kastil. Kastil yang sudah dikuasai bisa dipakai untuk membeli unit pasukan baru.
11. Untuk dapat memenangkan permainan ini, unit raja lawan harus dikalahkan dan juga unit raja pemain harus menguasai kastil dari kerajaan lawan.

3.2.11. Perancangan Skenario Permainan

Skenario dari permainan ini hampir menyerupai skenario dari *game* Ancient Empires 2. Pemain menjalankan unit biru yang dipimpin oleh raja Galamar yang tujuan utamanya adalah mengalahkan unit merah yang dipimpin oleh Demon Lord. Pada saat awal permainan pemain dan lawan masing-masing diberikan sebuah kastil yang berada di posisi acak dalam *map* yang berukuran acak antara 9x9 dan 17x17.

BAB IV
IMPLEMENTASI

Pada bab ini akan dibahas mengenai implementasi dari perancangan perangkat lunak. Di dalamnya mencakup proses penerapan dan pengimplementasian algoritma, dan antar muka yang mengacu pada rancangan yang telah dibahas sebelumnya.

4.1. Lingkungan Implementasi

Lingkungan implementasi dari tugas akhir dijelaskan pada Tabel 4.1.

Tabel 4. 1 Lingkungan Implementasi Perangkat Lunak

Perangkat Keras	Prosesor: Pentium(R) Core(TM) i3-32170 CPU @ 1.80GHz Memori: 4 GB
Perangkat Lunak	Sistem Operasi: Microsoft Windows 8.1 64-bit Perangkat Pengembang: Unity 4.3

4.2. Implementasi Permainan

Implementasi dari masing-masing fungsi utama dituliskan menggunakan *pseudocode* berdasarkan antarmuka utama yang ada pada permainan. Penjelasan implementasi hanya berupa antarmuka yang berhubungan dengan pembangkit *world* menggunakan algoritma *Midpoint Displacement*, menjalankan pasukan, dan menyerang lawan.

4.2.1. Implementasi Pembangkit *World*

Implementasi pembangkit *world* terdiri dari beberapa fungsi yang dituliskan pada Kode Sumber 4.1 dan Kode Sumber 4.2.

```
// fungsi Midpoint Displacement //
generatedMap[1,1]<-customRandom
generatedMap[9,1]<-customRandom
generatedMap[1,9]<-customRandom
generatedMap[9,9]<-customRandom
```

```
for(i=0 to 3)
   for(j=0 to 2^i)
      for(k=0 to 2^i)
         a=(k*y+1, j*y+1)      //pojok kiri bawah
         b=(k*y+y+1, j*y+1)    //pojok kanan bawah
         c=(k*y+1, j*y+y+1)    //pojok kiri atas
         d=(k*y+y+1, j*y+y+1) //pojok kanan atas
         FillMidpoint(a,b,c,d)
```

Kode Sumber 4. 1 Fungsi *Midpoint Displacement*

Pada fungsi midpoint() langkah pertama yang dilakukan adalah mengisi nilai dari keempat titik pojok dari *map* yang akan dibuat, yang direpresentasikan oleh variabel generatedMap[,]. Kemudian terdapat iterasi untuk membagi persegi tersebut menjadi 4 bagian dan seterusnya hingga samua nilai pada variabel generatedMap[,] terisi. Didalam iterasi tersebut memanggil fungsi FillMidpoint() untuk mengisi nilai-nilai pada variabel generatedMap[,].

```
// fungsi FillMidpoint //
Mid<-Average(generatedMap[a], generatedMap[b],
generatedMap[c], generatedMap[d]) + customRandom

Bawah<-Average(generatedMap[a], generatedMap[b],
generatedMap[Bawah.x,Bawah.y) +
customRandom

Kiri<-Average(generatedmap[a], generatedMap[c],
generatedmap[Kiri.x, Kiri.y]) +
customRandom

Kanan<-Average(generatedMap[b], generatedMap[d],
generatedMap[Kanan.x,Kanan.y]) +
customRandom

Atas<-Average(generatedMap[c], generatedMap[d],
generatedMap[Atas.x,Atas.y) +
customRandom
```

Kode Sumber 4. 2 Fungi FillMidpoint

Pada fungsi FillMidpoint() langkah pertama yang dilakukan adalah mengisi nilai titik tengah yang didapatkan dari nilai rata-rata dari keempat nilai di keempat sudut. Kemudian mengisi nilai titik tengah dari masing-masing sisi tepi persegi dari nilai rata-rata dari 2 poin yang berdekatan.

0	0	0	1	1	1	2	2	0
0	0	0	1	1	1	2	0	0
3	3	3	3	2	2	0	0	0
2	3	0	0	3	0	0	0	0
3	2	0	0	3	0	0	3	0
2	1	1	2	3	3	3	3	2
3	1	1	3	3	2	3	2	2
3	3	2	3	3	3	2	2	2
0	3	3	2	2	2	3	3	3

Gambar 4. 1 Isi Variable Setelah Fungsi Midpoint Dijalankan

Pada Gambar 4.1 merupakan tampilan isi dari variabel generatedMap[,] yang sudah terisi setelah pemanggilan fungsi Midpoint. Selanjutnya akan dilakukan pemanggilan fungsi generatedMap, yang berfungsi untuk mengubah nilai-nilai pada variabel generatedMap[,] menjadi *tile* sesuai dengan nilai yang dimiliki.

```
// fungsi GenerateMap //
for(i=1 to 9)
   for(j=1 to 9)
      position<- (i,j)
      gameObject<- tileType(generatedMap[i,j])
      Instantiate(gameObject, position)
```

Kode Sumber 4. 3 Fungi FillMidpoint

Fungsi generatedMap pada Kode Sumber 4.3 berfungsi untuk membangkitkan *tile* berdasarkan masing-masing nilai pada variabel generatedMap [,].

Gambar 4. 2 Tampilan Hasil Pembangkitan *Map* 9x9

Gambar 4.2 merupakan hasil pembangkitan *map* yang diperoleh dari fungsi yang terdapat dalam Kode Sumber 4.1, dan Kode Sumber 4.2. Map pada Gambar 4.2 merupakan hasil dari variabel generatedMap [,] yang terdiri dari beberapa *tile*. Misalnya *tile* hutan merupakan representasi dari nilai 2 dan *tile* air merupakan representasi dari nilai 0 yang terdapat dalam variabel generatedMap[,] tersebut. Sedangkan pada Gambar 4.3 merupakan hasil pembangkitan *map* dengan ukuran 17x17.

Gambar 4. 3 Tampilan Hasil Pembangkitan Map 17x17

4.2.2. Implementasi Menjalankan Pasukan

Implementasi Menjalankan Pasukan diawali dengan pemanggilan fungsi Cek() yang berada di *class* cekalpha.

```
// pemanggilan fungsi Cek //
_cekalpha <- gameObject.AddComponent<CekAlpha>
_cekalpha.Cek(transform.position)
```

Kode Sumber 4. 4 Pemanggilan Fungsi Cek

Fungsi Cek() bertujuan untuk menghitung jarak pergerakan dari suatu unit, yang nantinya akan direpresentasikan oleh kotak-kotak berwarna merah. Semakin kemampuan bergerak sebuah unit makan jangkauan atau kotak-kotak merah yang ditampilkan akan semakin banyak. Namun juga tidak dapat dilupakan faktor *terrain* yang memiliki bobot yang berbeda untuk setiap *terrain* yang ada.

```
// fungsi Cek //
for(i=1 to 9)
   for(j=1 to 9)
      if(tileMap[i,j]==grass || forest)
         groundMap<-1
      if(tileMap[i,j]==sand)
         groundMap<-2
      if(tileMap[i,j]==water)
         groundMap<-4

for(i=1 to 9)
   for(j=1 to 9)
      p<- unit.position
      jarak<- ShortestPath(groundMap[p],groundMap[i,j])
      if(jarak<=maxmoves)
         Instantiate(redTile, (i,j))
```

Kode Sumber 4. 5 Fungsi Cek

Terdapat variabel groundMap[,] yang digunakan untuk menyimpan bobot dari *tiles* yang akan dilewati oleh unit. Misalnya *tile sand* mempunyai bobot 3 yang maksudnya unit yang akan melewati *tile* tersebut akan menghabiskan 3 poin pergerakan, *tile water* mempunyai bobot 4 yang maksudnya unit yang melewati *tile* tersebut akan menghabiskan 3 point pergerakan dan *tile-tile* yang lain dengan bobotnya masing-masing.

Pada Kode Sumber 4.5 iterasi terakhir digunakan untuk perhitungan dalam mencari jalan terpendek dari sebuah unit menuju *tile* yang akan dituju. Jika *tile* tersebut mempunyai total jarak kurang dari jarak pergerakan suatu unit maka akan diberi tanda kotak merah pada *tile* tersebut.

```
// fungsi Moving //
If(isBlack==false)
   StartCoroutine(MoveToPosition(target))
```

Kode Sumber 4. 6 Fungsi *Moving*

Pada Kode Sumber 4.6 di atas berfungsi untuk memanggil fungsi movetoposition terus-menerus hingga unit yang dijalankan sampai di posisi target.

```
// fungsi IEnumerator MoveToPosition //
while(position!=target)
   unit.move(target)
   yield return 0;
if(position==target)
   isBlack=true
```

Kode Sumber 4. 7 Fungsi IEnumerator MoveToPosition

Pada Kode Sumber 4.7 di atas, unit akan berhenti bergerak jika sudah sampai target. Setelah itu sistem akan menampilkan menu aksi dari unit tersebut.

Gambar 4. 4 Tampilan Menjalankan Pasukan

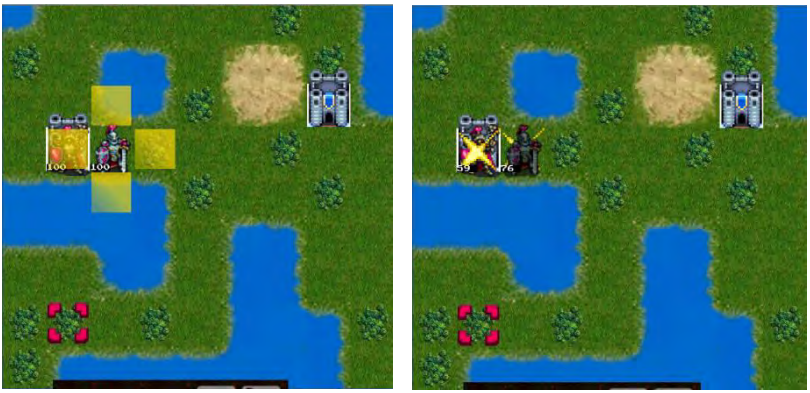
Gambar 4.4 di atas bagian kanan merupakan tampilan hasil dari fungsi Cek pada Kode Sumber 4.5. Untuk menjalankan unit, pemain harus memilih salah satu dari kotak merah tersebut.

Gambar 4.4 bagian kanan merupakan tampilan menu aksi dari unit setelah berjalan. Jika unit tersebut berada di jarak serang dari unit musuh maka akan keluar menu serang.

```
// fungsi Serang //
dmg<- Attacker.dmg * Attacker.hp/100
def<- Defender.armor + armorMap[Defender.position]
Defender.hp-=dmg-def
if(Defendeer.hp<=0)
   Destroy(Defender)

dmg<- Defender.dmg * Defender.hp/100
def<- Attacker.armor + armorMap[Attacker.position]
Attacker.hp-=dmg-def
if(Attacker.hp<=0)
   Destroy(Attacker)
```

Kode Sumber 4. 8 Fungsi Serang

Pada Kode Sumber 4.8 dijalankan ketika pemain memilih aksi serang. Dimana akan menampilkan kotak kuning yang menunjukkan pasukan lawan yang berada dalam jarak serang unit pemain. Kode sumber tersebut juga berisi perhitungan dari penyerangan. Dimana *damage* dari sebuah unit sama dengan *damage* unit tersebut dikalikan dengan prosentase *health point* dari unit tersebut. Sedangkan *armor* dari sebuah unit sama dengan *armor* dari unit tersebut ditambah pertahanan dari *tile* dimana unit tesebut berada.



Gambar 4. 5 Tampilan Menyerang Unit Lawan

Gambar 4.5 di atas bagian kanan merupakan implementasi dari perhitungan *damage* pada *pseudocode* yang berada pada Kode Sumber 4.8.

4.2.3. Implementasi Membeli Pasukan

Implementasi Membeli Pasukan diawali dengan menampilkan menu aksi beli.

```
// fungsi Beli //
If(tokopilih==1 && gold>=150)
   Instantiate(saber, castle.position)
If(tokopilih==2 && gold>=200)
   Instantiate(archer, castle.position)
If(tokopilih==3 && gold>=250)
   Instantiate(lancer, castle.position)
If(tokopilih==4 && gold>=250)
   Instantiate(berserker, castle.position)
```

Kode Sumber 4. 9 Fungsi Beli

Menu aksi beli akan muncul apabila pemain menekan tombol *Shop* pada layar. Pada Kode Sumber 4.9 di atas merupakan *pseudocode* untuk realisasi pembelian unit. Jika pemain membeli unit saber misalnya maka kondisi pertama pada *pseudocode* di atas yang akan di eksekusi. Pertama sistem membangkitkan unit yang telah dibeli di posisi kastil pemain. Kemudian sistem mengurangi uang dari pemain sebanyak harga dari unit yang dibeli tersebut.

Sistem memunculkan menu aksi beli ketika pemain menekan tombol *Shop* pada menu bar pada Gambar 4.9 bagian kanan. Gambar 4.9 di bawah merupakan implementasi dari *pseudocode* yang terdapat pada Kode Sumber 4.9. pada Gambar di bawah, pemain membeli unit saber yang berharga 150 gold. Kemudian pada Gambar 4.9 bagian kanan, uang dari pemain tersisa 350 gold setelah melakukan pembelian unit saber yang berharga 350 gold.

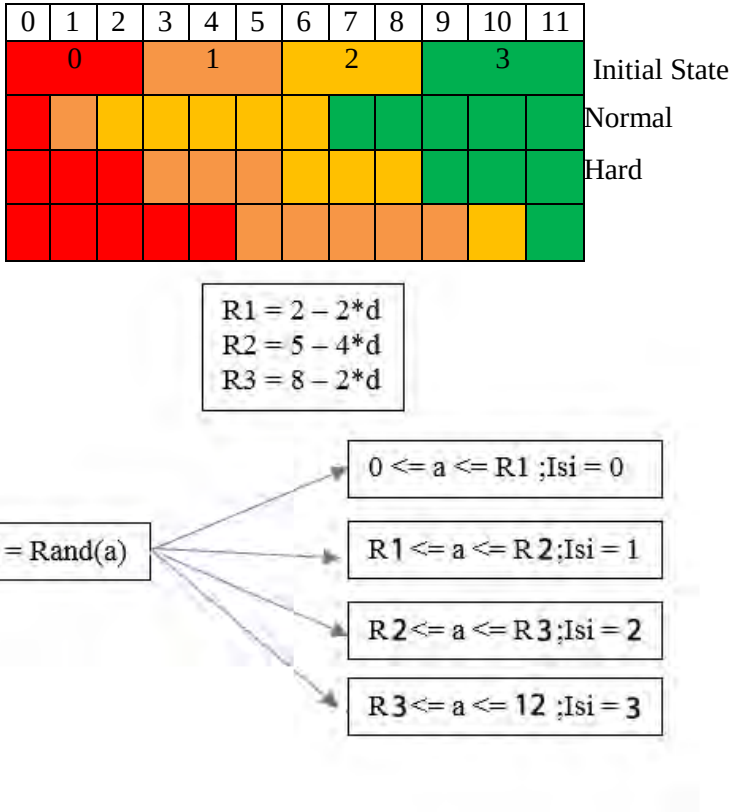
Gambar 4. 9 Tampilan Membeli Unit

4.2.4. Implementasi fungsi *random*

```
// fungsi Random //
public int customRandom (int difficulty)
       {
              int a = Random.Range (0, 12);
              int result = 0;
              int[] ii = {2,5,8};
              difficulty -= 1;

              ii[0]  -= 2*difficulty;
              ii [1] -= 4*difficulty;
              ii [2] -= 2*difficulty;

              if (a >= 0 && a <= ii[0] - (difficulty) )
              {result = 0;}
              else if(a > ii[0]   && a <= ii[1]   )
              {result = 1;}
              else if(a > ii[1]  && a <= ii[2] )
              {result = 2;}
              else if(a > ii[2] && a <= 11 )
              {result = 3;}

              return result;
      }
```

Kode Sumber 4. 10 Fungsi CustomRandom

Tabel 4. 2 Tabel Ilustrasi Fungsi CustomRandom



Gambar 4. 10 Persamaan CustomRandom

Fungsi CustomRandom adalah fungsi yang mengatur tingkat kesulitan pada *map* dengan cara memanipulasi jumlah *tile* tertentu sehingga dapat menyesuaikan dengan keinginan pemain akan tingkat kesulitan *tile* permainan. Tingkat kesulitan yang disediakan adalah *Hard, Medium, Easy.* Penjelasan dari Tabel 4.2 adalah jika pemain memilih tingkat kesulitan *easy* maka saat fungsi CustomRandom

berjalan maka fungsi CustomRandom akan mengubah *margin* dari setiap hasil yang keluar (0 adalah representasi dari *tile hard* yaitu air/*water*), dan hasil yang keluar akan digunakan sebagai nilai pada setiap *poin*.

BAB V
PENGUJIAN DAN EVALUASI

Pada bab ini akan dijelaskan mengenai rangkaian uji coba dan evaluasi yang dilakukan. Proses pengujian dilakukan menggunakan metode kotak hitam berdasarkan skenario yang telah ditentukan dan pengujian dilakukan dengan survei langsung kepada 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 dicantumkan pada Kode Sumber 5.1 dan Kode Sumber 5.2.

Tabel 5. 1 Lingkungan Uji Coba Perangkat Lunak (bagian 1)

Perangkat Keras	Prosesor: Pentium(R) Core(TM) i3-32170 CPU @ 1.80GHz Memori: 4 GB
Perangkat Lunak	Sistem Operasi: Microsoft Windows 8.1 64-bit Perangkat Pengembang: Unity 4.3

Tabel 5. 2 Lingkungan Uji Coba Perangkat Lunak (bagian 2)

Perangkat Keras	Alat : *Smartphone, 4.0 inch, 480×800 pixels Dual-core 1.2 GHz* Memori: 1 GB
Perangkat Lunak	Sistem Operasi: Android 4.4

5.2. Pengujian Performa Kecepatan Pembangkit *World*

Pengujian dilakukan untuk menguji seberapa cepat sistem dalam membangkitkan *world.* Pengujian dilakukan dengan

menambahkan *code* pada *script* untuk memudahan perhitungan waktu pembangkitan *world.*

5.2.1. Skenario Data Uji Coba Kecepatan

Skenario pengujian kecepatan performa digunakan untuk memberikan tahap-tahap dalam pengujian sistem dalam hal kecepatan. Skenario ini tertera pada Tabel 5.3.

Tabel 5. 3 Skenario Pengujian Performa

Tujuan	Bertujuan untuk mengukur *running time* dalam membangkitkan *world* area permainan
Langkah pengujian	1. Tambahkan *code* untuk menghitung waktu pembangkitan pada *script*. 2. Jalankan *scene game* pada *project* di unity. 3. Pada *console* unity akan menampilkan waktu pembangkitan *world*. 4. Ulangi langkah ke-1 sampai ke-4. 5. Cari rata-rata waktu kecepatan pembangkitan *world.*

```
// menghitung waktu GenerateMap//
waktuawal<- Time.realtimeSinceStartUp;
Midpoint()
GenerateMap()
waktuakhir<- Time.realtimeSinceStartUp;
Print(waktuakhir-waktuawal)
```

Kode Sumber 5. 1 Menghitung Waktu Fungsi GenerateMap

Pada Kode Sumber 5.1 ditambahkan pada *script* yang bertujuan untuk menghitung waktu pembangkitan *world* yang ada pada fungsi generatedMap().

5.2.2. Hasil Pengujian Performa

Hasil pengujian performa dikhususkan untuk menghitung waktu yang dibutuhkan dalam membangkitkan *world* dimana dijelaskan pada Tabel 5.4. semua nilai pada tabel dihitung dalam satuan detik.

Pada Tabel 5.4 merupakan hasil *log* dari unity *game engine* yang merupakan hasil *generate* dari tingkat kesulitan *hard.*

Tabel 5. 4 Hasil Uji Coba Performa Pembangkit *World Hard*

Uji Coba ke-	Waktu yang dibutuhkan (detik)
1	0.024
2	0.054
3	0.021
4	0.056
5	0.019
6	0.016
7	0.023
8	0.022
9	0.019
10	0.019
11	0.019
12	0.023
13	0.045
14	0.024
15	0.022
16	0.026
17	0.020
18	0.018
19	0.017
20	0.025
Rata-rata	**0.0256**

Berdasarkan Tabel 5.4, dapat disimpulkan kecepatan pembangkitan *world* dengan tingkat kesulitan *hard* membutuhkan waktu rata-rata **0.0256** detik. Waktu eksekusi yang singkat untuk membangun sebuah *world* dengan tingkat kesulitan *hard.*

Tabel 5.5 merupakan hasil *log* dari unity *game engine* yang merupakan hasil *generate* dari tingkat kesulitan *medium.*

Tabel 5. 5 Hasil Uji Coba Performa Pembangkit *World Medium*

Uji Coba ke-	Waktu yang dibutuhkan (detik)
1	0.016
2	0.016
3	0.019
4	0.016
5	0.015
6	0.018
7	0.013
8	0.015
9	0.015
10	0.016
11	0.020
12	0.018
13	0.015
14	0.017
15	0.015
16	0.019
17	0.016
18	0.015
19	0.015
20	0.017
Rata-rata	**0.0163**

Berdasarkan Tabel 5.5 dapat disimpulkan bahwa kecepatan pembangkitan *world* dengan tingkat kesulitan *medium* membutuhkan waktu rata-rata **0.0163** detik.

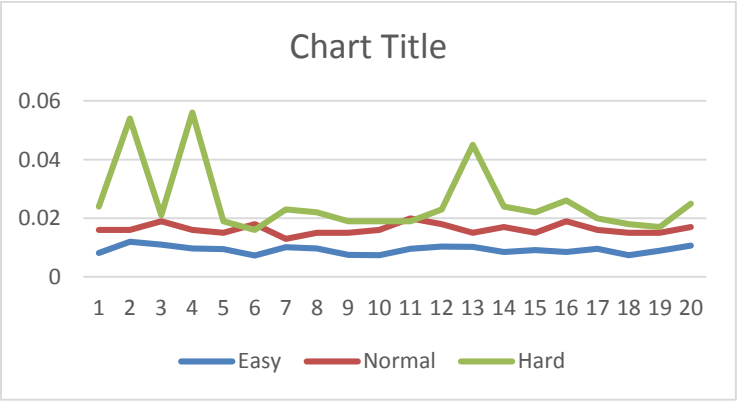
Tabel 5.6 merupakan hasil *log* dari unity *game engine* yang merupakan hasil *generate* dari tingkat kesulitan *easy*.

Tabel 5. 6 Hasil Uji Coba Performa Pembangkit *World Easy*

Uji Coba ke-	Waktu yang dibutuhkan (detik)
1	0.0082
2	0.0120
3	0.011

4	0.0097
5	0.0095
6	0.0073
7	0.0101
8	0.0097
9	0.0075
10	0.0074
11	0.0096
12	0.0103
13	0.0102
14	0.0085
15	0.0091
16	0.0085
17	0.0096
18	0.0074
19	0.0089
20	0.0107
Rata-rata	**0.011**

Berdasarkan table 5.6 dapat disimpulkan bahwa kecepatan pembangkitan *world* dengan tingkat kesulitan *easy* membutuhkan waktu rata-rata **0.011** detik.



Grafik 1. 1 Grafik Perbandingan

Berdasarkan Grafik 1.1, dapat disimpulkan bahwa kecepatan pembangkitan *world* dengan tingkat kesulitan *easy* dan *medium* membutuhkan waktu rata-rata **0.011** untuk tingkat kesulitan *easy* dan **0.0163** detik untuk tingkat kesulitan *hard* dibandingkan dengan waktu eksekusi dari *world* dengan tingkat kesulitan *hard*, waktu eksekusi yang dimiliki tingkat kesulitan *medium* dan *easy* lebih kecil dari pada tingkat kesulitan *hard* karena pada tingkat kesulitan *hard, tile* yang keluar lebih bervariasi. Pada *tile water* dan *sand* harus melakukan pengecekan kecocokan dengan *tile* yang bersebelahan sehingga semakin banyak *tile water* dan *sand* yang keluar maka sistem akan selalu melakukan pengecekan.

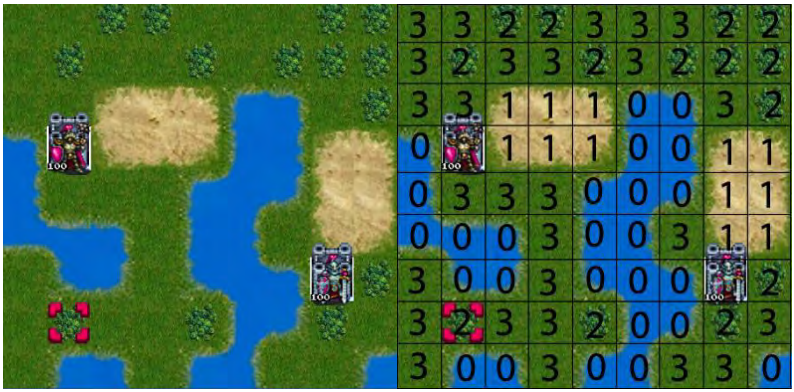
5.3. *Skenario Pengujian Kesesuaian Tingkat kesulitan World*

Skenario pengujian kesesuaian tingkat kesulitan *world* yang dibangkitkan digunakan untuk memberikan tahap-tahap dalam pengujian sistem dalam hal *world* yang dibangkitkan. Skenario ini tertera pada Tabel 5.7.

Tabel 5. 7 Skenario Pengujian Tingkat Kesulitan *World*

Tujuan	Bertujuan untuk mengetahui ketepatan dari *world* yang dibangkitkan
Langkah pengujian	1. Jalankan aplikasi sehingga membangkitkan *world*. 2. Simpan hasil *world* yang berukuran 9x9 dari *easy*, *medium* sampai *hard*. 3. Rubah *tile* pada *world* tersebut menjadi nilai-nilai yang merepresentasikan *tile* tersebut. 4. Hitung perbandingan nilai yang keluar pada *world*. 5. Ulangi langkah ke-1 sampai ke-4. 6. Lihat hasil dari keseluruhan data dari langkah percobaan di atas.

5.3.1. Skenario Data Uji Coba Kesesuaian Tingkat Kesulitan *Hard* Pada *World* yang Dibangkitkan



Gambar 5. 1 *World* Uji Coba *Hard*

Nilai pada *tile* merepresentasikan tingkat kesulitan tersendiri dikarenakan semakin banyak *tile* sulit maka akan semakin mempengaruhi pergerakan dari pemain, urutan *tile* dari yang tersulit adalah *tile* air/*water* yang memiliki bobot 4 selanjutnya adalah *tile* pasir/*sand* yang memiliki bobot 3 dan *tile* hutan/*forest*, padang rumput/*grass* yang memiliki bobot 2. Pada Gambar 5.1 didapatkan perbandingan nilai *random* antara nilai 0, 1, 2, dan 3 seperti pada Tabel 5.8.

Tabel 5. 8 Tabel Uji Coba Kesesuaian Tingkat Kesulitan *Hard*

Nilai	Jumlah
0	26
1	12
2	15
3	28

Dari Tabel 5.8 didapatkan data bahwa *tile* yang memrepresentasikan nilai 0 dan 1 berjumlah 38 *tile* dan *tile* yang merepresentasikan nilai 2 dan 3 berjumlah 43 *tile*.

5.3.2. Skenario Data Uji Coba Kesesuaian Tingkat Kesulitan *Medium* Pada *World* yang Dibangkitkan

Tabel 5. 9 Tabel Uji Coba Kesesuaian Tingkat Kesulitan *Medium*

Nilai	Jumlah
0	8
1	15
2	19
3	39

Gambar 5. 2 *World* Uji Coba *Medium*

Dari Tabel 5.9 didapatkan data bahwa *tile* yang memrepresentasikan nilai 0 dan 1 berjumlah 23 *tile* dan *tile* yang merepresentasikan nilai 2 dan 3 berjumlah 58 *tile*. *Tile-tile* tersebut kemudian di *map*-kan ke dalam *world* yang hasilnya dapat dilihat pada Gambar 5.2.

5.3.3. Skenario Data Uji Coba Kesesuaian Tingkat Kesulitan *Easy* Pada *World* yang Dibangkitkan

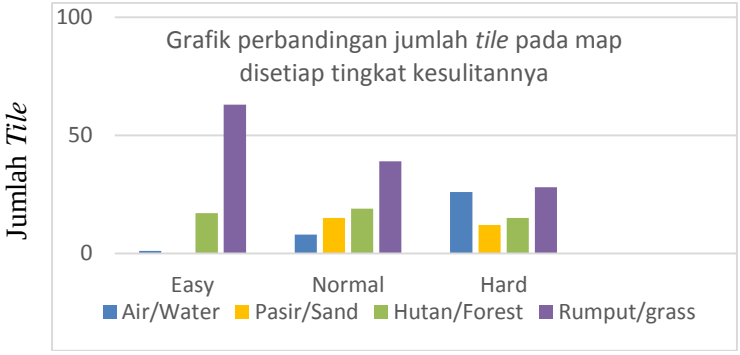
Dari Tabel 5.10 didapatkan data bahwa *tile* yang merepresentasikan nilai 0 dan 1 berjumlah 1 *tile* dan *tile* yang merepresentasikan nilai 2 dan 3 berjumlah 63 *tile*. *Tile-tile* tersebut kemudian di *map*-kan ke dalam *world* yang hasilnya dapat dilihat pada Gambar 5.3.

Tabel 5. 10 Tabel Uji Coba Kesesuaian Tingkat Kesulitan *Easy*

Nilai	Jumlah
0	1
1	0
2	17
3	63

Gambar 5. 3 *World* Uji Coba *Easy*

5.3.4. Hasil Uji Coba Kesesuaian Tingkat Kesulitan Pada *World* yang Dibangkitkan



Gambar 5. 4 Perbandingan Hasil Uji Coba Kesesuaian *World*

Pada Gambar 5.4 diatas dapat disimpulkan bahwa pada tingkat kesulitan *hard* memiliki lebih banyak *tile* yang tergolong susah yaitu

tile dengan nilai 0 (air/*water*) dan 1 (pasir/*sand*) dari tingkat kesulitan lainnya.

5.4. Pengujian Fungsionalitas dengan Metode Kotak Hitam

Pengujian ini dilakukan untuk menguji apakah fungsionalitas yang diidentifikasi pada tahap kebutuhan benar-benar diimplementasikan dan bekerja semestinya. Selain itu juga untuk mengetahui kesesuaian keluaran dari setiap tahapan atau langkah penggunaan fitur terhadap skenario yang dipersiapkan. Pengujian dilakukan dengan menggunakan metode kotak hitam.

5.4.1. Skenario Pengujian Fungsionalitas

Dalam subbab ini akan dijelaskan mengenai skenario pengujian yang akan dilakukan dalam menguji fungsionalitas aplikasi. Skenario pengujian fungsionalitas akan digunakan untuk mendiskripsikan tahap-tahap yang akan dilakukan dalam pengujian sistem. Detail dari skenario pengujian UFG01 dapat dilihat pada Tabel 5.11.

Tabel 5. 11 Pengujian Permainan

	UFG01
Kondisi Awal	Pengguna berada pada halaman awal permainan.
Prosedur Pengujian	Pengguna memainkan permainan hingga selesai dari beberapa fungsionalitas dan mencoba setiap tingkat kesulitan yang ingin diuji
Hasil yang diharapkan	Pengguna berhasil menyelesaikan permainan dan fungsionalitas permainan berjalan dengan lancar.
Hasil yang diperoleh	Pengguna berhasil menyelesaikan permainan dan fungsionalitas berjalan lancar. Akan tetapi masih terdapat *error* pada aplikasi.
Kesimpulan.	Pengujian berhasil.

5.4.1.1. Pengujian Skenario Permainan

Pada subbab ini akan dijelaskan mengenai pengujian Skenario Permainan. Pengujian yang dilakukan pada subbab ini bertujuan untuk melakukan pengecekan terhadap fungsionalitas dari layar permainan apakah sudah berjalan dengan baik dan sesuai dengan yang diinginkan atau tidak.

Skenario permainan pada Gambar 5.5 berjalan dengan baik seperti yang diinginkan, dimana pada awal permainan terdapat dua kerajaan, yaitu kerajaan biru (pemain) dan kerajaan merah (lawan). Dan juga tiap kerajaan diberikan kastil, raja dan 500 uang. Dan penempatan kastil sudah baik seperti yang diinginkan.

Gambar 5. 5 Tampilan Skenario Permainan

5.4.1.2. Pengujian Layar Main Menu

Pada subbab ini akan dijelaskan mengenai pengujian mengenai Layar Main Menu. Pengujian ini difokuskan pada saat pemain berada di layar main menu. dan bertujuan untuk mencoba fungsionalitas dari layar main menu pada permainan. Dapat dilihat pada Gambar 5.6, tampilan awal sudah berjalan sesuai yang diinginkan dengan menampilkan tombol *New Game*, *Option* dan *Credit*.

Pada Gambar 5.7 merupakan tampilan dari setelan menu permainan. Terdapat dua setelan yaitu tingkat kesulitan dan *map*. Untuk mengubah setelan, tap pada *status button*. Pada setelan *map*, kita dapat memilih ukuran *map* dari 9x9 atau 17x17 atau acak antara kedua ukuran tersebut dan untuk *level* kita dapat memilih tingkat kesulitan mulai dari *Easy*, *Medium* dan *Hard.*

Gambar 5. 6 Tampilan Awal Permainan

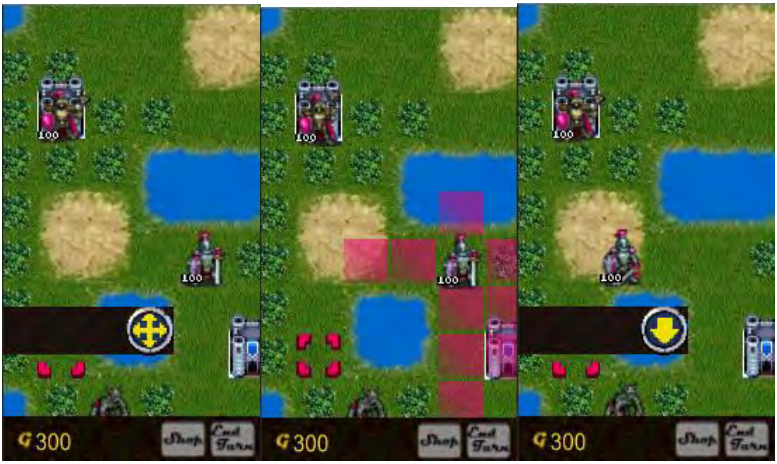
Gambar 5. 7 Tampilan Menu Setelan

5.4.1.3. Pengujian Layar Permainan

Pada subbab ini akan dijelaskan mengenai pengujian mengenai Layar Permainan. Pengujian ini difokuskan pada saat pemain berada di layar permainan dan bertujuan untuk mencoba fungsionalitas saat bermain. Pertama akan dilakukan pengujian saat membeli unit pasukan baru. Pengujian ini bertujuan untuk melakukan pengecekan terhadap fungsi pembelian unit yang ada didalam permainan, pergerakan unit, menyerang dan proses mengakhiri permainan.

Terlihat pada Gambar 5.8 adalah langkah-langkah yang dilakukan pemain saat membeli unit baru. Dalam pengujian ini pemain memilih tombol “Shop”. Permainan lalu menampilkan pilihan unit yang dapat dibeli dan detail dari unit tersebut. Setiap unit yang ada memiliki harga, kekuatan serangan, jumlah langkah dan pertahanan yang berbeda. Pemain lalu memilih salah satu unit yang tersedia. Unit yang dibeli lalu ditampilkan dalam layar permainan dan uang yang dimiliki oleh pemain berkurang sesuai dengan harga unit yang telah dibeli.

Gambar 5.8 Tampilan Membeli Pasukan



Gambar 5. 9 Tampilan Menjalankan Pasukan

Pada Gambar 5.9, menu aksi dari unit raja muncul ketika pemain memilih unit raja dengan menekan unit tersebut. Setelah memilih menu jalan pada menu unit maka akan ditampilkan kotak-kotak merah yang menandakan jarak pergerakan dari unit tersebut.

Permainan akan secara otomatis menghitung kotak-kotak yang dapat dijangkau oleh unit yang dipilih. Banyak kotak yang ditampilkan tergantung pada unit yang dipilih dan *terrain* yang ada disekitar unit. Semakin besar *cost* dari *terrain* yang dilewati makan semakin pendek jangkauan dari unit. Pada saat pemain memilih salah satu dari kotak merah maka unit tersebut akan bergerak menuju kotak merah yang dipilih tersebut. Pemain tidak dapat memilih kotak yang tidak berwarna merah karena kotak yang berwarna merah terletah diluar jangkauan gerakan unit yang dipilih.

Pada Gambar 5.9 bagian kanan, setelah unit tersebut sampai pada titik yang dituju, maka unit tersebut akan menampilkan menu aksi. Setelah mengakhiri giliran dari unit tersebut maka unit tersebut akan berubah warna menjadi abu-abu yang menandakan giliran unit tersebut sudah selesai untuk saat ini. Selanjutnya pengujian saat menyerang pasukan lawan.

Gambar 5. 10 Tampilan Menyerang Pasukan Lawan (bagian 1)

Pada Gambar 5.10 ditampilkan tetang proses yang dilakukan pemain saat melakukan penyerangan dengan unit. Dapat dilihat jika unit kita berada di jarak serang unit musuh maka akan keluar menu aksi serang. Kotak kuning menandakan jarak serang unit pemain.

Pemain harus memilih pasukan musuh yang berada dalam kotak kuning.

Pada Gambar 5.10 bagian kiri menampilkan animasi saat unit pemain menyerang unit lawan. Gambar 5.10 bagian kanan menampilkan hasil dari pertarungan berupa *health point* yang berkurang. Jika *health point* dari unit habis maka unit tersebut akan mati. Hasil dari pengujian menyerang pasukan lawan berjalan dengan baik seperti yang diinginkan. *Terrain* tempat unit yang diserang berada menjadi area pertarungan. Unit yang diserang mendapat bonus pertahanan sesuai dengan tipe *terrain.* Tetapi kemampuan masing-masing unit masih belum terimplementasikan secara keseluruhan. Misalnya terdapat unit yang memiliki jarak serangan lebih jauh dari unit lainnya. Contoh dari unit ini antara lain adalah archer, catapult, dragon, dan unit-unit yang memiliki jarak serangan jauh lainya. Selanjutnya pengujian dalam mengakhiri permainan.

Gambar 5. 11 Mengakhiri Permainan

Pada Gambar 5.11, untuk mengakhiri permainan, unit raja dari pasukan pemain harus berada pada kastil dari pasukan lawan. Kemudian akan muncul menu kuasai seperti pada Gambar 5.11 bagian kiri. Setelah berhasil menguasai kastil pasukan lawan, akan muncul keterangan tim biru menang, seperti pada Gambar 5.11 bagian kanan. Kemudian akan kembali ke main menu. Selanjutnya pengujian dalam menambah *health point* dari unit yang berada pada rumah atau kastil kerajaannya.

5.4.1.4. Hasil Pengujian Fungsionalitas

Hasil uji fungsionalitas yang sudah dilakukan berdasarkan pada empat pengujian pokok diatas. Empat uji coba yang telah dilakukan menunjukkan bahwa semua fungsionalitas permainan berjalan dengan baik dan sesuai dengan sebagaimana mestinya skenario yang telah dibuat pada perancangan. Rekap hasil pengujian fungsionalitas dicantumkan pada Tabel 5.12.

Tabel 5. 12 Hasil Pengujian Fungsionalitas

No	Nama Pengujian	Hasil Pengujian
1	Pengujian Hasil Pembangkitan World	Berhasil
2	Pengujian Skenario Permainan	Berhasil
3	Pengujian Layar Main Menu	Berhasil
4	Pengujian Layar Permainan	Berhasil

5.5. Pengujian Kepuasan Pengguna

Pengujian pada perangkat lunak yang dibangun tidak hanya dilakukan pada fungsionalitas yang dimiliki, tetapi juga pada pengguna untuk mencoba 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 perangkat lunak yang ada.

5.5.1. Skenario Uji Coba Pengguna

Dalam melakukan pengujian perangkat lunak, penguji diminta mencoba menggunakan perangkat lunak untuk mencoba semua fungsionalitas dan fitur yang ada. Pengujian aplikasi oleh pengguna dilakukan dengan sebelumnya memberikan informasi seputar aplikasi, kegunaan, dan fitur-fitur yang dimiliki. Setelah informasi tersampaikan, pengguna kemudian diarahkan untuk langsung mencoba aplikasi dengan spesifikasi lingkungan yang sama dengan yang telah diuraikan pada uji coba fungsionalitas.

Jumlah pengguna yang terlibat dalam pengujian perangkat lunak sebanyak sepuluh orang. Dalam melakukan pengujian, pengguna melakukan percobaan lebih dari satu kali penggunaan untuk masing-masing pengguna.

Dalam memberikan penilaian, penulis memberi lembar kuesioner kepada penguji. Setelah penguji selesai melakukan pengujian perangkat lunak, penguji mengisi kuesioner. Pertanyaan yang ditanyakan dalam pengujian perangkat lunak ini memiliki beberapa aspek penilaian.

5.5.2. Daftar Penguji Perangkat Lunak

Pada subbab ini ditunjukkan daftar pengguna yang bertindak sebagai penguji coba aplikasi yang dibangun. Daftar nama penguji aplikasi ini dapat dilihat pada Tabel 5.13.

Tabel 5. 13 Daftar Nama Penguji Coba Aplikasi

No	Nama	Pekerjaan
1	M Rizal Prihandoko	Mahasiswa
2	Bagus Prathista	Mahasiswa
3	Iswahyudi	Mahasiswa
4	Alfian Maulana Azhari	Mahasiswa
5	Rifi Febrio	Mahasiswa
6	Mulia Lovendo A	Mahasiswa
7	Irooyan Alvi Aziz	Mahasiswa
8	Lira Dewi Sufi	Pelajar
9	Bill Achmad Maliki	Pelajar
10	Dinar Aulia Rahman	Pelajar

5.5.3. Hasil Uji Coba Pengguna

Uji coba yang dilakukan terhadap beberapa pengguna memiliki beberapa aspek yang dipisahkan berdasarkan antarmuka dan fungsionalitas yang dimiliki. Sistem penilaian didasarkan pada skala penghitungan satu sampai empat di mana skala satu menunjukkan nilai terendah dan skala empat menunjukkan skala tertinggi. Penilaian akhir kemudian dilakukan dengan menghitung berapa banyak penguji yang memilih suata skala tertentu dan kemudian dicari nilai rata-ratanya. Hasil uji coba dipaparkan secara lengkap dengan disertai tabel yang dapat dilihat pada subbab. Keterangan nilai pada kuesioner dijelaskan pada Tabel 5.14.

Tabel 5. 14 Keterangan Nilai Pada Kuesioner

Nilai	Keterangan
1	Kurang
2	Cukup
3	Baik
4	Sangat Baik

5.5.3.1. Hasil Penilaian Antarmuka

Penilaian antarmuka difokuskan pada penilaian pengguna terhadap kemudahan penggunaan antarmuka dan sifat-sifat lain yang perlu dimiliki. Hasil penilaian pengguna terhadap antarmuka aplikasi dapat dilihat pada Tabel 5.15.

Tabel 5. 15 Penilaian Antarmuka

No.	Antarmuka	Penilaian				Rata-Rata
		1	2	3	4	
1	Kemudahan Penggunaan	0	0	8	2	3,2
2	Kelengkapan Menu	0	0	1	9	3,9
3	Keindahan Tampilan	0	0	5	5	3,5

4	Kecepatan Pemilihan Menu/Fitur	0	3	6	1	2,8
5	Kesesuaian tema	0	2	6	2	3
6	Ketertarikan Bermain	0	0	7	3	3,3
Nilai Akhir						**3,2**

5.5.3.2. Hasil Penilaian Performa Sistem

Penilaian performa sistem difokuskan pada penilaian pengguna terhadap kemampuan aplikasi dalam menghasilkan performa dari interaksi pengguna. Penilaian ini juga ditujukan untuk mendapatkan tingkat kecepatan dan kelancaran sistem atas interaksi yang dibuat oleh pengguna. Hasil penilaian performa sistem dapat dilihat pada Tabel 5.16.

Tabel 5. 16 Penilaian Performa Sistem

No.	Performa Sistem	Penilaian				Rata-Rata
		1	2	3	4	
1	Performa atau kinerja pada permainan	0	1	4	5	3,4
2	Nilai *world* yang dibangkitkan dan tingkat kesulitan *map*	0	0	5	5	3,5
3	Ketepatan dalam menampilkan data	0	0	3	7	3,7
4	Kelancaran animasi	0	0	7	3	3,3
5	Kesesuaian animasi dengan sistem permainan	0	0	7	3	3,3
6	Nilai penempatan kastil	0	0	2	8	3,8
Nilai Akhir						**3,5**

5.5.4. Hasil Pengujian Pengguna

Evaluasi pengujian pengguna dilakukan dengan menampilkan data rekapitulasi perangkat lunak yang telah dipaparkan. Dari data diketahui bahwa:

1. Aplikasi telah memenuhi unsur yang seharus seperti perancangan dimana mendapatkan nilai akhir 3,2 dan 3,5. Arti dari nilai 3,2 dan 3,5 ini merupakan penilaian yang baik.
2. Permainan yang dibuat telah mendapatkan apresiasi oleh pengguna sebenarnya. Sehingga dapat disimpulkan bahwa pengguna secara tidak langsung cukup menyetujui dan memberikan komentar baik kepada aplikasi permainan yang dibuat.
3. Mendapat nilai poin sedikit pada kecepatan pemilihan menu/fitur. Pada kecepatan pemilihan menu/fitur mungkin pada saat membangkitkan *world* dibutuhkan waktu dua sampai tiga detik jika menggunakan *smartphone* dengan spesifikasi seperti pada Tabel 5.1, sedangkan dengan menggunakan PC seperti pada Tabel 5.2 diperoleh rata-rata 0,0163 detik.

BAB VI
KESIMPULAN DAN SARAN

Bab ini membahas mengenai kesimpulan yang dapat diambil dari tujuan pembuatan perangkat lunak dan hasil uji coba yang telah dilakukan sebagai jawaban dari rumusan masalah yang dikemukakan. Selain kesimpulan, terdapat pula saran yang ditujukan untuk pengembangan perangkat lunak lebih lanjut.

6.1. Kesimpulan

Dalam proses pengerjaan tugas akhir mulai dari tahap analisis, desain, implementasi, hingga pengujian didapatkan kesimpulan sebagai berikut:

1. Aplikasi berhasil membangun permainan menggunakan pembangkit *world*. Kehandalan permainan diuji menggunakan kebutuhan fungsional dan kebutuhan pengguna yang meliputi menu, aturan main, *level* dan skenario berjalan seperti yang diharapkan. Berdasarkan uraian yang telah disampaikan dapat disimpulkan semua fungsionalitas permainan (menjalankan unit, menyerang unit lawan, membeli unit, menguasai kastil, serta mengakhiri giliran) baik itu aturan main ataupun skenario berfungsi seperti yang diharapkan.
2. Berdasarkan uji coba kesesuaian *world* yang dibangkitkan menggunakan algoritma *Midpoint Displacement*, aplikasi berhasil membangkitkan *world* seperti yang diharapkan.
3. Untuk tingkat kesulitan dan skenario dipengaruhi oleh posisi kastil dan *tile – tile* yang akan dibangkitkan yang di bangkitkan secara acak.

6.2. Saran

Berikut merupakan beberapa saran untuk pengembangan sistem di masa yang akan datang, berdasarkan pada hasil perancangan, implementasi dan uji coba yang telah dilakukan.

1. Penambahan kecerdasan buatan yang lebih bagus. Sehingga dapat memberikan tantangan kepada pemain.

2. Membuat permainan mode cerita yang mempunyai alur dan *map* yang sama dengan *game turn based strategy* pada umumnya.
3. Penambahan karakter yang dapat keistimewaan di salah satu *terrain.*
4. Pilihan pasukan kurang.

LAMPIRAN

Pertanyaan wawancara

No	Kuisioner	Penilaian			
		1	2	3	4
1	Kemudahan penggunaan				
2	Kelengkapan menu				
3	Keindahan tampilan				
4	Kecepatan pemilihan menu/tile				
5	Kesesuaian tema				
6	Ketertarikan bermain				
7	Performa atau kinerja pada permainan				
8	Nilai *world* yang dibangkitkan				
9	Kecepatan dalam menampilkan data				
10	Kelancaran animasi				
11	Kesesuaian animasi dengan sistem permainan				
12	Nilai penempatan *tile*				

DAFTAR PUSTAKA

[1] B. Plimmer and J. Jrundy, "Beautifying Sketching-based Design Tool Content: Issues and Experience," *AUIC '05 Proceedings of the Sixth Australasian conference on User interface - Volume 40,* pp. 31-38, 2005.

[2] B. Bersama, "Berbagi Bersama," Blogspot, November 2012. [Online]. Available: http://abhique.blogspot.com/2012/11/metode-prototyping-dalam-pengembangan.html. [Accessed 25 September 2014].

[3] Wikipedia, "Wikipedia," Wikipedia, 24 Maret 2014. [Online]. Available: http://id.wikipedia.org/wiki/Microsoft_Visual_Studio. [Accessed 26 September 2014].

[4] A. Nicolescu, "Black box techniques," 28 4 2014. [Online]. Available: https://www.qualitance.com/blog/black-box-techniques. [Accessed 19 12 2014].

[5] Unity Technologies, "Unity Overview," 19 December 2014. [Online]. Available: http://docs.unity3d.com/Manual/UnityOverview.html.

[6] J. Eldridge, "Best turn-based strategy games, definition and meaning," 13 3 2013. [Online]. Available: http://www.examiner.com/article/the-definition-of-turn-based-strategy-games-with-game-examples. [Accessed 19 12 2014].

[7] "C#.NET Programming," [Online]. Available: http://brainmatics.com/c-net-programming/. [Accessed 21 12 2014].

[8] "Explanation of the Diamond-Square algorithm," 27 9 2010. [Online]. Available: https://code.google.com/p/fractalterraingeneration/wiki/Diamond_Square. [Accessed 23 12 2014].

[9] C. Janssen, "Android Operating System," [Online]. Available: http://www.techopedia.com/definition/25106/android-operating-system. [Accessed 19 12 2014].

Biodata Penulis

Penulis, Yuan Akbarsyah Pandunegoro lahir pada tanggal 19 September 1993 di Surabaya, merupakan anak pertama dari Empat bersaudara. Penulis menempuh pendidikan formal pertamanya di SD Annur Pekanbaru-Riau (1999-2005). Kemudian penulis melanjutkan pendidikanya di SMP Negeri 3 Peterongan Jombang (2005-2008). Saat SMA penulis melanjutkan pendidikanya di SMA Ulul-Albab Sidoarjo (2008-2011). Tidak berhenti disitu masa pendidikan formal penulis. Saat kuliah penulis memilih untuk berkuliah di Jurusan Teknik Informatika Institut Teknologi Sepuluh Nopember Surabaya. Di Teknik Informatika ITS, penulis tidak hanya aktif di perkuliahan saja. Penulis juga sempat beberapa kali bergabung dengan beberapa organisasi mahasiswa, seperti HMTC ITS (Himpunan Mahasiswa Teknik Computer Informatika) dan BEM ITS(Badan Eksekutif Mahasiswa ITS). Untuk menghubungi penulis, dapat melewati surel Yuan.akbarsyah@gmail.com atau melewati akun *facebook* penulis https://www.facebook.com/yuan.pandu.