SKRIPSI

PENERAPAN ALGORITMA CELLULAR AUTOMATA DALAM MENGHASILKAN MAP GAME RPG SECARA PROSEDURAL

DISUSUN OLEH:

I MADE ASTORA ANDISTYA

DBC 115 025

JURUSAN TEKNIK INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS PALANGKA RAYA

2021

PENERAPAN ALGORITMA CELLULAR AUTOMATA DALAM MENGHASILKAN MAP GAME RPG SECARA PROSEDURAL

SKRIPSI

Sebagai salah satu syarat untuk menyelesaikan Program Strata-1 pada Jurusan Teknik Informatika Fakultas Teknik Universitas Palangka Raya

Oleh

I MADE ASTORA ANDISTYA

DBC 115 025

Telah dipertahankan didepan tim penguji, pada:

Hari/Tanggal : Rabu, 10 Maret 2021

Waktu : 09.00 – 10.30

1. AGUS S. SARAGIH, ST., M.Eng
   NIP. 198508182012121003 : ..... (Ketua)
2. DEDDY RONALDO, ST., MT.
   NIP. 198012262008121002 : ..... (Anggota)
3. NOVA NOOR KAMALA SARI,ST.,M.Kom
   NIP. 198904072015042004 : ..... (Anggota)
4. NAHUMI NUGRAHANINGSIH, Ph.D
   NIP. 197910092008012016 : ..... (Anggota)
5. NOVERA KRISTIANTI, S.T., M.T.
   NIDN. 0016119301 : ..... (Anggota)

Mengetahui :

Fakultas Teknik
Universitas Palangka Raya
Dekan,

Ir. WALUYO NUSWANTORO, M.T.
NIP. 19681119 199302 1 001

Jurusan / Program Studi Teknik Informatika
Fakultas Teknik Universitas Palangka Raya
Ketua Jurusan,

ABERTUN SAGIT SAHAY, S.T., M.Eng
NIP. 19751212 200312 1 002

SKRIPSI

PENERAPAN ALGORITMA CELLULAR AUTOMATA DALAM MENGHASILKAN MAP GAME RPG SECARA PROSEDURAL

Sebagai salah satu syarat menyelesaikan Program Strata - 1
pada Jurusan Teknik Informatika Fakultas Teknik Universitas Palangka Raya

OLEH :

I MADE ASTORA ANDISTYA
NIM. DBC 115 025

Disetujui untuk diajukan dalam Seminar Akhir Skripsi,

Palangka Raya, 24 Februari 2021

Pembimbing I	Pembimbing II
DEDDY RONALDO, ST., MT.	**NOVA NOOR KAMALA SARI, ST.,M.Kom**
NIP. 19801226 200812 1 002	**NIP. 19890407 201504 2 004**

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS PALANGKA RAYA
2021

PERNYATAAN

Dengan ini saya menyatakan dengan sebenar-benarnya bahwa dalam Tugas Akhir ini tidak terdapat karya ilmiah yang pernah diajukan untuk memperoleh gelar kesarjanaan disuatu perguruan tinggi, serta tidak terdapat karya ilmiah atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis diacu dalam Tugas Akhir ini dan disebutkan dalam Daftar Pustaka.

Palangka Raya, 15 Maret 2021

I MADE ASTORA ANDISTYA

DBC 115 025

RIWAYAT PENYUSUN

Data Diri

Nama	:	I MADE ASTORA A.
NIM	:	DBC 115 025
Fakultas	:	Teknik
Jurusan/Program Studi	:	Teknik Informatika
Jenjang	:	Strata 1 (S-1)
Jenis Kelamin	:	Laki-Laki
Tempat, Tanggal Lahir	:	Sampit, 23 November 1997
Agama	:	Kristen Protestan
Status dalam Keluarga	:	Anak Kandung
Anak ke -	:	1
Alamat	:	Jl. Simpei Karuhei
No. Telpon/HP	:	+6281210198759

Data Orang Tua

Nama Ayah	:	Ir. I Made Dikantara
Pekerjaan Ayah	:	Pensiun PNS
Nama Ibu	:	Saruh., S.P.
Pekerjaan Ibu	:	PNS
Alamat Orang Tua	:	Sampit, Jl. R.A. Kartini No. 17
No. Telpon/HP	:	+628125094749

Riwayat Pendidikan

SD	:	SD Katholik Yos Soedarso	(Tahun Lulus 2009)
SMP	:	SMPN 1 Sampit	(Tahun Lulus 2012)
SMA	:	SMAN 3 Palangka Raya	(Tahun Lulus 2015)

Palangka Raya, 15 Maret 2021

I MADE ASTORA ANDISTYA

DBC 115 025

HALAMAN PERSEMBAHAN

Dengan hati yang penuh dengan sukacita,
saya panjatkan segala puji dan syukur dengan setulus hati saya kepada:

Tuhan Yesus Kristus

Kupersembahkan karya Skripsi ini untuk :

Ibu dan Bapak saya yang tercinta
Yang selalu mendoakan dan memberikan dukungan sepenuhnya
Yang tidak pernah habis-habisnya memberikan semangat dan motivasi
Yang selalu memberikan pengertian atas segala permasalahan saya
Dan tidak pernah gagal untuk memberikan motivasi dan nasihat untuk tetap maju

Sahabat-sahabat dan keluarga besar di Fakultas Teknik
Almamater tercinta Universitas Palangka Raya
Yang selalu membantu dan memberikan pengalaman yang berharga
Selama perjalanan menempuh studi saya.

KATA PENGANTAR

Puji dan Syukur kepada Tuhan Yang Maha Kuasa karena berkat karunia dan kasihnya yang berlimpah penulis dapat diberikan kemampuan, dan kekuatan untuk dapat menyelesaikan penulisan laporan skripsi ini dengan tepat waktu yang berjudul "*Penerapan Algoritma Cellular Automata dalam Menghasilkan Map Game RPG Secara Prosedural*".

Laporan Tugas Akhir ini disusun untuk memenuhi persyaratan Skripsi bagi mahasiswa Jurusan/Prodi Teknik Informatika, Fakultas Teknik, pada Universitas Palangka Raya. Dalam kesempatan ini penulis ingin mengucapkan terima kasih dan penghargaan setingginya kepada pihak bersangkutan:

1. Bapak Abertun Sagit Sahay, ST., M.Eng. Selaku Ketua Jurusan/Program Studi Teknik Informatika Fakultas Teknik Universitas Palangka Raya.
2. Bapak Deddy Ronaldo, S.T., M.T., dan Ibu Nova Noor Kamala Sari, S.T., M.Kom. selaku Dosen Pembimbing I dan Dosen Pembimbing II Skripsi ini.
3. Rekan – rekan mahasiswa angkatan 2015 yang selalu memberikan semangat, dukungan, saran, dan motivasi dalam proses menempuh ujian dan penulisan laporan Skripsi ini.

Penulis menyadari bahwa dalam penyelesaian skripsi ini dapat ditemukan berbagai kekurangan baik dalam penulisan maupun penyajiannya. Dengan demikian penulis dengan segala kerendahan hati mengharapkan segala jenis kritik dan saran yang bersifat membangun demi kesempurnaan skripsi ini.

Palangka Raya, 15 Maret 2021

I MADE ASTORA ANDISTYA

DBC 115 025

PENERAPAN ALGORITMA CELLULAR AUTOMATA DALAM MENGHASILKAN MAP GAME RPG SECARA PROSEDURAL

I Made Astora Andistya (DBC 115 025)
Jurusan Teknik Informatika, Fakultas Teknik, Universitas Palangka Raya
Universitas Palangka Raya Tunjang Nyaho, Jl. Yos Sudarso Palangka Raya 73113.
Email : twilightskies23@gmail.com

ABSTRAK

Videogame adalah permainan dimana pemain berinteraksi mengguanakan sebuah perangkat pengendali untuk menghasilkan umpan balik tampilan video. Dengan semakin meningkatnya tenaga komputasi yang juga berpengaruh dengan meningkatnya kompleksitas *videogame*, *developer* dapat mengimplementasikan beberapa teknik koding dalam mengembangkan suatu *videogame*. Salah satunya berupa *Procedural Content Generation*, dimana sebuah konten pada *videogame* digenerasikan menggunakan komputer berdasarkan parameter awal yang diberikan. Penerapan konsep ini telah dilihat di berbagai *genre* permainan, seperti *Role-Playing Game* (*RPG)* dimana pemain memerankan sebuah karakter dalam menjelajahi dunia yang dikembangkan dalam suatu naratif. Salah satu algoritma kandidat dalam penerapan *PCG* adalah Algoritma Cellular Automata, dikarenakan kemampuan algoritma untuk mengubah matriks 2D yang acak menjadi bentuk yang semi-teratur.

Subjek penelitian berupa implementasi algoritma cellular automata pada proses pembuatan arena permainan pada *game RPG*. Pembuatan arena dimulai dengan membuat sebuah matriks 2D yang diisi secara acak dengan nilai 0 atau 1, terhadap matriks tersebut kemudian akan dilakukan iterasi cellular automata untuk menghasilkan matriks akhir yang akan diubah menjadi bentuk 3D. Tahapan terakhir pembuatan arena berupa penempatan musuh pada wilayah yang sah dalam arena permainan.

Tahap perancangan game dilaksanakan dengan menggunakan metodologi *Waterfall* sebagai acuan, yang meliputi tahapan perancangan analisis dan desain, pengkodean, implementasi dan pengujian aplikasi. Analisis dan desain dilakukan dengan menjabarkan keperluan sistem dan pemodelan diagram *UML*. Pengkodean dan implementasi dilakukan menggunakan *software* Unity Editor, sementara pengujian dilakukan menggunakan metode *blackbox* beserta pengujian kecepatan generasi arena permainan.

Hasil pengujian menunjukan bahwa kecepatan generasi arena permainan sangat bergantung pada ukuran matriks 2D yang disediakan, dimana terdapat peningkatan waktu yang signifikan ketika ukuran arena yang dibuat semakin diperbesar, terutama pada waktu pembuatan objek mesh 3D dari matriks 2D yang didapat. Pengujian blackbox menunjukan bahwa aplikasi yang dibuat dapat dijalankan tanpa adanya kesalahan.

Kata-Kata Kunci : *Game RPG, Procedural Content Generation, Algoritma Cellular Automata, Metodologi Waterfall*

"IMPLEMENTATION OF CELLULAR AUTOMATA ALGORITHM IN GENERATING PROCEDURAL RPG GAME MAP"

I Made Astora Andistya (DBC 115 025)
Department of Informatics, Faculty of Engineering, University of Palangka Raya,
Campus of UPR Tunjung Nyaho Yos Sudarso Street Palangka Raya 73112
Email : twilightskies23@gmail.com

ABSTRACT

Videogames are a type of games where players use a control device to generate video display feedback. With the increase of computational power, the complexity of videogames are also affected, in turn developers can implement several coding techniques in developing a videogame. One of them is in the form of Procedural Content Generation, in which a content in a videogame is generated using a computer, based on the initial parameters given. The application of this concept has been seen from various game genres, such as Role-Playing Game (RPG) where the player plays a character in exploring a world that is developed in a narrative. One of the algorithms for implementing PCG is the Cellular Automata Algorithm, because of the algorithm's ability to convert randomized 2D matrices and turn it into semi-regular patterns.

The subject of researh is the implementation of cellular automata implementation in the process of making a game arena in RPG games. The arena creation process begins by creating a 2D matrix filled randomly with the value 0 or 1, the matrix will then iterated with the cellular automata algorithm to produce the final matrix, which will be converted into a 3D object. The final stage of making the arena is in the form of placing the enemy in a valid area in the game arena.

The game design stage is carried out using the Waterfall methodology as a reference, which includes the stages of design analysis and design, coding, implementation and application testing. Analysis and design are carried out with the system requirements and UML diagram modeling. Coding and implementation were carried out using the Unity Editor software, while testing was carried out using the blackbox method and the speed testing created a game arena. T

The test results show that the speed of the creation of the game arena is very dependent on the size of the 2D matrix provided, where the time increases significantly when the size of the arena is enlarged, especially in making 3D mesh objects from the 2D matrix. Blackbox testing shows that the application created can run without errors.

Keywords : *RPG Game, Procedural Content Generation, Cellular Automata Algorithm, Waterfall Methodology*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Video Game merupakan salah satu medium hiburan yang perkembangannya dapat dianggap berpegangan tangan dengan perkembangan teknologi, khususnya dalam teknologi komputasi. Dengan semakin meningkatnya tenaga komputasi dalam sebuah komputer, kompleksitas dari video game semakin juga meningkat, sehingga melahirkan beragam jenis game dengan yang sekarang dapat dikategorikan dalam beberapa genre berdasarkan karakteristik permainan tertentu.

Dengan tenaga komputasi yang semakin meningkat, *developer* game juga diberikan suatu kebebasan untuk mengimplementasikan beragam teknik koding. Salah satu teknik ini berupa implementasi *procedural content generation (PCG)*. PCG adalah sebuah penerapan dimana sebuah konten spesifik dalam game akan dihasilkan dengan otomatis secara algoritmik, berdasarkan parameter tertentu yang ditentukan oleh developer. Jika dilakukan dengan baik, hal ini dapat meningkatkan nilai *replayability* dari sebuah game.

Salah satu penerapan PCG yang umum dalam *video game* adalah dengan menggunakan sebuah algoritma untuk mengotomatisasikan proses pembuatan arena permainan dalam sebuah game. Cellular Automata berupa salah satu algoritma yang dipertimbangkan untuk penerapan PCG dikarenakan kemampuannya untuk mengubah input array 2D awal yang acak, kemudian mengubahnya menjadi pola yang lebih teratur. Salah satu bentuk yang dapat dihasilkannya berupa sebuah peta yang menyerupai sebuah terowongan gua, yang kemudian dapat diproses lebih lanjut untuk menghasilkan sebuah arena permainan yang fungsional.

Penggunaan PCG telah ditemui dalam berbagai genre game. Salah satunya berupa game *RPG(Role-Playing Game)*, dimana pemain mengambil kendali sebuah karakter, untuk menjelajahi dunia fiksi yang telah dibentuk oleh developer, biasanya sambil mengikuti sebuah alur cerita. Salah satu fitur khas dari game *RPG* adalah pemain dapat mengembangkan kekuatan karakter melalui proses *leveling*, dimana karakter yang dimainkan akan semakin kuat jika semaking tinggi *level*-nya. Untuk

meningkatkan *level* karakter, pemain biasanya harus memenuhi persyaratan tertentu dalam game, seperti telah mengalahkan sejumlah musuh, menyelesaikan suatu misi, dan sebagainya.

Berdasarkan paparan diatas, akan dibuat sebuah konsep game *RPG* dengan *procedural content generation* yang diimplementasikan menggunakan algoritma *cellular automata* dengan tujuan untuk menjelajahi kelayakan algoritma *cellular automata* dalam menghasilkan konten map atau arena permainan secara prosedural. Berdasarkan permasalahan tersebut maka diangkat sebuah materi Tugas Akhir dengan judul : "**Penerapan Algoritma Cellular Automata dalam Menghasilkan Map Game RPG Secara Prosedural**", dengan judul game berupa "**Ascent**".

1.2 Rumusan Masalah

Adapun rumusan masalah yang dapat ditarik berdasarkan latar belakang yang telah diuraikan adalah bagaimana cara menerapkan algoritma *Cellular Automata* sehingga dapat menghasilkan map sebuah game yang fungsional dan dapat dimainkan.

1.3 Batasan Masalah

Dalam pembuatan game "Ascent" akan memiliki batasan masalah sebagai berikut :

1. Game akan didesain untuk dimainkan secara single player dengan menggunakan platform PC (*Personal Computer*) dan sistem operasi Windows.
2. Konsep game yang akan dibuat dalam bentuk game 3D dengan tipe permainan *Action RPG* dengan gaya sudut pandangan *Top Down*.
3. Game akan dibuat menggunakan Unity3D Personal Edition dengan bahasa pemrograman C#, sementara aset-aset 3D yang digunakan akan dibuat menggunakan perangkat lunak modeling 3D Blender.
4. Area permainan akan terbagi menjadi dua, area aman dimana tidak terdapat musuh dan area labirin yang terdiri atas 4 tingkat.
5. Implementasi PCG akan dilakukan hanya pada proses pembuatan arena pada labirin, yang mencangkup pembuatan bentuk arena permainan dengan

mengunakan dasar algoritma selular automata, dan penempatan musuh dalam arena tersebut.

6. Labirin akan terdiri atas 3 area tingkat pertama yang di-generate berdasarkan parameter yang telah ditentukan, dan 1 area tingkat terakhir yang berupa arena bos.
7. Setiap area pada labirin terdiri atas beberapa musuh yang jumlahnya telah ditentukan, dimana area selanjutnya pada labirin akan memiliki lebih banyak musuh dibandingkan area sebelumnya.
8. Untuk dapat melanjutkan ke area selanjutnya pada labirin, pemain harus mengalahkan seluruh musuh yang ada pada area yang sedang ditempati.
9. Setelah menempuh 3 area pertama pada labirin, pemain akan memasuki arena bos, dimana pemain akan memenangkan permainan jika bos pada arena tersebut dapat dikalahkan.

1.4 Tujuan

Tujuan dari pembuatan game "Ascent" ini adalah untuk menjelajahi penggunaan algoritma Cellular Automata dalam menghasilkan sebuah area/arena permainan game RPG.

1.5 Manfaat

Adapun manfaat yang diberikan dalam pembuatan game " Ascent" ini adalah sebagai berikut:

1. Memberikan sebuah implementasi untuk pembuatan map game RPG menggunakan algoritma Cellular Automata.
2. Mengurangi beban pembuatan level dari tangan developer, dengan mengotomatisasi proses dengan bantuan algoritma.
3. Sebagai bahan pembelajaran awal untuk mendalami penerapan konsep algoritma *procedural content generation* dalam media video game.

1.6 Sistematika Penulisan

Penulisan laporan ini akan disusun dalam lima bab utama yang terdiri atas Pendahuluan, Landasan Teori, Perancangan, Implementasi, dan Penutup.

Berikut adalah susunan sistematika penulisan laporan:

1. **COVER**
2. **HALAMAN PERSETUJUAN**
3. **HALAMAN PENGESAHAN**
4. **HALAMAN PERNYATAAN**
5. **KATA PENGANTAR**
6. **DAFTAR ISI**
7. **DAFTAR TABEL**
8. **DAFTAR GAMBAR**
9. **ABSTRAKSI**
10. **BAB I PENDAHULUAN**

    Bab I Pendahuluan akan terdiri atas latar belakang masalah, rumusan masalah, batasan masalah, tujuan , sistematika penulisan, dan jadwal pelaksanaan kegiatan tugas akhir.
11. **BAB II LANDASAN TEORI**

    Bab II Landasan Teori akan membahas teori-teori dasar dan tinjauan pustaka yang akan digunakan sebagai basis penilitian tugas akhir ini.
12. **BAB III METODOLOGI PENELITIAN**

    Bab III Metodologi penelitian akan membahas metode penelitian yang digunakan beserta tahapan desain untuk game yang akan dibuat.
13. **BAB IV HASIL DAN PEMBAHASAN**

    Bab IV Hasil dan Pembahasan berisikan tahap implementasi desain sekaligus analisis dan evaluasi terhadap hasil penelitian tugas akhir dan aplikasi yang telah dihasilkan.
14. **BAB V PENUTUP**

    Bab V Kesimpulan dan Saran berisikan kesimpulan dan saran berdasarkan hasil dan pembahasan dari penelitian tugas akhir yang telah dilaksanakan.
15. **DAFTAR PUSTAKA**
16. **LAMPIRAN**

1.7 Jadwal Kegiatan

Tabel 1.1 Jadwal Kegiatan Pembuatan Tugas Akhir

Rencana Kegiatan	Bulan, Minggu, dan Tahun							
	Maret 2020	April 2020				Mei 2020 – November 2020	Desember 2020 – Januari 2021	Febuari 2021	
	IV	I	II	III	IV			I	II
Penyusunan dan Pengumpulan Proposal									
Pengumuman Dosen Pembimbing									
Perancangan									
Implementasi									
Pembuatan Laporan									
Seminar									

BAB II
LANDASAN TEORI

2.1. Tinjau Pustaka

2.1.1 An Analog History Of Procedural Content Generation (Gillian Smith, 2015)

Procedural Content Generation adalah dimana sebuah konten dihasilkan melalui sebuah perantara *agent* dengan mengikuti suatu prosedur atau langkah yang telah ditentukan. Dalam konteks video game, PCG adalah penggunaan sebuah algoritma formal untuk menghasilkan sebuah konten game yang biasanya dilakukan secara manual oleh manusia (Gillian Smith, 2015).

Menurut Gillian Smith, awal implementasi PCG dapat ditemukan pada permainan analog terdahulu, dimana implementasi PCG dilakukan dengan motivasi untuk menambahkan nilai pengulangan suatu permainan, membantu proses kreativitas pemain, sebagai sarana berekspresi, menjamin konten tanpa bias dan adil, dan menggantikan manusia dalam menghasilkan konten.

2.1.2 A Short Introduction To Procedural Content Generation Algorithm For Video Games (Nicolas A. Barriga, 2018)

Algoritma PCG dapat dikelompokan menjadi tiga kategori, yaitu metode tradisional yang mencangkup penggunaan pseudo random generator seperti *seed, fractal* dan *noise* dalam pembuatan konten, metode *search-based* dimana algoritma berperan sebagai penghasil konten sekaligus pengevaluasi kelayakan konten, dan metode *machine learning* dimana konten dihasilkan berdasarkan hasil prediksi dari dataset yang telah disediakan.

2.1.3 Procedural Generation : An Algorithmic Analysis Of Video Game Design And Level Creation (Logan Bond, 2017)

Fungsi dari procedural generation adalah untuk menghasilkan sebuah data secara algoritmik, yang biasanya perlu dilakukan secara manual. Pengimplementasian procedural generation dapat menggeserkan beban pekerjaan dari individu ke algoritma, sehingga dapat mengizinkan lebih banyak konten untuk

dapat dihasilkan. Procedural generation sering digunakan untuk menghasilkan berbagai komponen dalam video game, hingga seseorang dapat membuat game yang menggunakan algoritma untuk menghasilkan keseluruhan kontennya. Game yang menggunakan procedural generation sering kali digunakan dalam pembuatan peta, barang, karakter, grafis dan banyak hal lain.

2.1.4 Cellular automata for real-time generation of infinite cave levels (Lawrence Johnson dkk, 2010)

Jurnal yang bersangkutan menyajikan sebuah pendekatan untuk men-*generate* sebuah map game dengan memanfaatkan sifat algoritma Cellular Automata yang dapat mengatur diri sendiri. Algoritma berbasis CA yang dihasilkan mempunyai biaya komputasi yang sangat kecil. Sehingga dapat menghasilkan level tak terbatas secara *real-time*, sementara menyediakan fleksibilitas dalam perihal desain level.

2.2. Teori Pendukung

2.2.1 Cellular automata

Cellular Automata adalah sekumpulan 'sel' dalam sebuah grid yang berkembang seiring langkah waktu diskrit atau 'step' berdasarkan serangkaian aturan yang telah ditentukan berdasarkan kondisi sel yang ada disekitar suatu sel tertentu (Eric W. Weisstein, n.d.).

Salah satu contoh populer Cellular Automata adalah Conway's Game Of Life, yaitu sebuah permainan simulasi karya John Horton Conway yang dinamakannya berdasarkan analogi permainan terhadap kebangkitan, kejatuhan, dan perubahan suatu perkumpulan organisme (Martin Gardner, 1970). Permainan dapat dimainkan dengan menggunakan papan catur besar dan sejumlah potongan *counters* yang memiliki dua warna berbeda. Permainan dimulai dengan suatu konfigurasi penempatan *counter* yang sederhana, satu untuk setiap sel pada papan, kemudian mengamati perubahan yang terjadi ketika diterapkan 'peraturan genetik' yang telah ditentukan oleh Conway.

Peraturan genetik dari permainan Conway's Game of Life adalah sebagai berikut (Martin Gardner, 1970):

1. Jika suatu sel hidup memiliki 2 atau 3 tetangga sel yang hidup, maka sel tersebut akan tetap hidup
2. Jika suatu sel memiliki lebih dari 3 tetangga sel yang hidup, maka sel tersebut akan mati.
3. Suatu sel yang memiliki kurang dari 2 sel tetangga yang hidup akan mati.
4. Suatu sel yang mati akan hidup kembali jika memiliki tepat 3 sel tetangga yang hidup.

Perubahan pada tiap sel berlangsung secara bersamaan, sehingga perubahan pada suatu sel dalam satu generasi tidak akan mempengaruhi perubahan sel lain untuk generasi yang selanjutnya.

Gambar 2.1 Perubahan Cell Pada Conway's Game of Life

2.2.2 Video game

Video game merupakan sebuah bentuk permainan yang menggunakan sebuah perangkat elektronik untuk menciptakan sebuah sistem interaktif dimana pengguna dapat bermain. Di dalam sebuah video game, pemain dapat berinteraksi dengan game melalui perantara sebuah perangkat input seperti *keyboard*, *mouse* atau *joystick*, dimana hasil interaksi pemain tersebut akan dibentuk menjadi sebuah umpan balik visual dalam bentuk video yang dapat ditampilkan melalui perangkat yang kompatibel, seperti layar televisi atau layar komputer.

Menurut Mark J.P. Wolf (2008), video game dapat dibagikan menjadi beberapa genre dengan mempertimbangkan karakteristik dominan dari rasa interaksi pemain dengan game, tujuan dan objektif game, dan sifat kontrol karakter dalam game. Beberapa dari genre video game berdasarkan pembagian tersebut berupa *Adventure* (petualangan), *Racing* (balapan), *Role-Playing* (memainkan peran), dst.

2.2.3 Role-playing game (RPG)

Menurut Mark J.P. Wolf (2008), *Role-Playing Game* merupakan sebuah genre permainan dimana pemain dapat membuat atau mengendalikan sebuah karakter yang identitasnya telah ditentukan. Karakter juga dapat memilki beragam kemampuan atau kekuatan yang biasanya direpresentasikan tingkatannya dengan menggunakan angka.

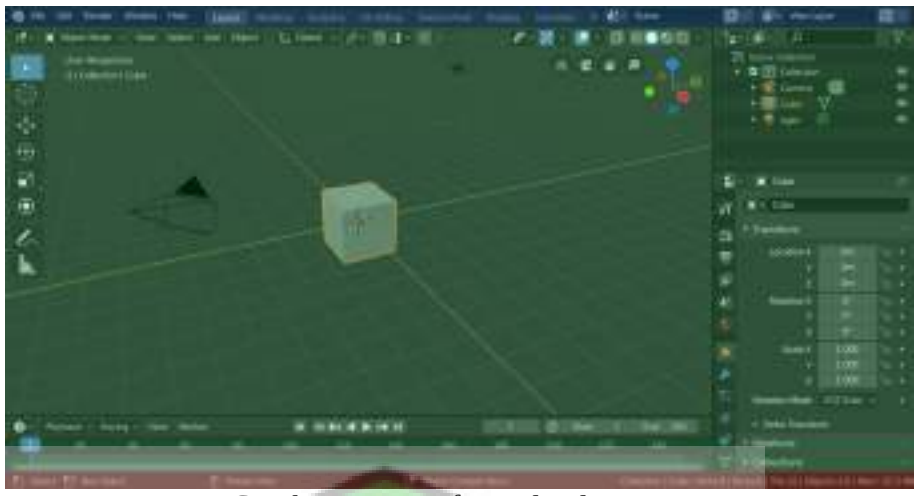
Role-Playing Game merupakan sebuah genre game dimana pemain memerankan suatu figur dalam suatu *setting* fiksi, yang aspek-aspek karakternya (nama, latar belakang, motivasi, dst) telah ditentukan. Pemain kemudian menjelajahi dan berinteraksi dalam dunia fiksi tersebut melaui perantara karakter yang dimainkan.

Salah satu fitur yang biasa dimiliki oleh *RPG* adalah sebuah sistem dimana kekuatan karakter yang dimainkan dapat dikembangkan sesuai dengan keinginan pemain. Sistem yang umum digunakan berupa sistem *Leveling*, dimana karakter memulai permainan dari *level* 0 dan dapat meningkatkan *level* ketika telah mencapai kriteria tertentu (menyelesaikan objektif, mengalahkan cukup musuh, dll.). Ketika karakter naik *level*, pemain akan diberikan suatu poin yang dapat digunakan untuk mengembangkan salah satu aspek kekuatan karakter yang dimainkan.

2.2.4 Blender

Blender adalah sebuah perangkat lunak *open source cross-platform* pemodelan 3D yang berada dibawah lisensi *GNU General Public License (GPL)*, memungkinkan pengguna untuk menggunakan Blender secara gratis untuk segala tujuan. Blender mendukung keseluruhan alur pemodelan 3D mulai dari proses *modeling*, *rigging*, *animation*, *simulation*, *rendering*, *compositing* dan *motion tracking*.

Interface Blender secara default terbagi menjadi tiga bagian utama, yaitu *topbar* (biru), *areas* (hijau) dan *status bar* (merah).

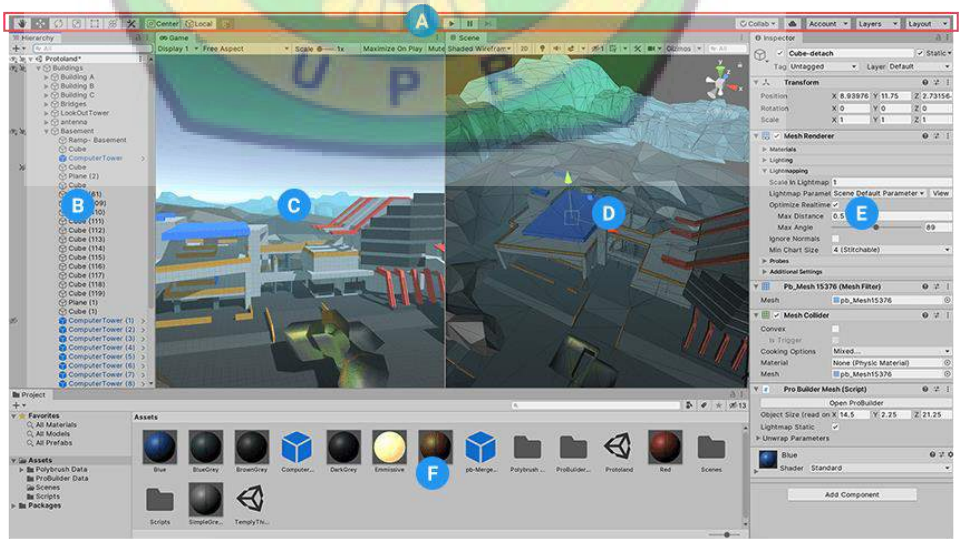


Gambar 2.2 Interface Blender

2.2.5 Unity development platform

Unity adalah sebuah Integrated Development Environment (IDE) yang digunakan untuk mengembangkan sebuah game 2D, 3D, *Virtual Reality (VR)* & *Augmented Reality (AR)* Game yang dikembangkan berjalan dengan menggunakan Unity Game Engine, sebuah *cross-platform game engine* yang dikembangkan oleh Unity Technologies.

Interface Unity Development Platform terdiri atas beberapa bagian yang memiliki fungsi tersendiri.



Gambar 2.3 Interface Unity

a. **Toolbar** memberikan akses terhadap peralatan/*tool* dasar yang digunakan untuk memanipulasi sebuah *Scene* atau *GameObject*
b. **Hierarchy Window** memberikan daftar hirarkis secara teks; seluruh *GameObject* yang ada dalam sebuah *Scene*
c. **Game View** men-simulasikan bentuk game akhir yang telah di-*render* melalui kamera yang ada pada scene.
d. **Scene View** digunakan untuk melihat dan mengubah scene yang sedang ditampilkan secara visual
e. **Inspector Window** digunakan untuk melihat dan mengubah seluruh properti yang dimiliki oleh sebuah *GameObject* yang sedang dipilih.
f. **Project Window** menampilkan seluruh aset game yang tersedia dan dapat digunakan dalam projek yang sedang dikerjakan.
g. **Inspector Window** digunakan untuk melihat dan mengubah seluruh properti yang dimiliki oleh sebuah *GameObject* yang sedang dipilih

2.2.6 Bahasa pemrograman C#

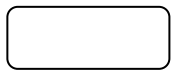
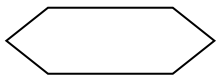
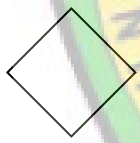
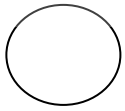
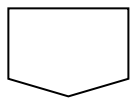
Bahasa pemrograman C# adalah sebuah bahasa pemrograman berorientasi objek yang dikembangkan oleh Microsoft yang ditujukan untuk menjadi sebuah bahasa pemrograman yang sederhana, modern dan *general-purpose* (ECMA-334, 2017). Distribusi secara luas pertama bahasa pemrograman C# dilakukan oleh Microsoft pada bulan Juli 2000 sebagai bagian dari inisiatif .NET Microsoft.

Bahasa pemrograman C# digunakan pada aplikasi Unity dalam pengembangan *script* sebuah game, yaitu sebuah file yang digunakan untuk mengontrol objek-objek dan logika game.

2.2.7 Flowchart

Flowchart atau diagram alir adalah sebuah jenis diagram yang merepresentasikan suatu algoritma, alur kerja atau proses, dengan cara menunjukan setiap langkah dalam berbagai bentuk kotak, dan urutan langkahnya menggunakan tanda panah. Flowchart digunakan untuk menganalisa, mendesain, mendokumentasikan, atau mengelola suatu proses atau program.

Tabel 2.1 Simbol-Simbol Flowchart

Simbol	Nama	Fungsi
	Terminator.	Permulaan atau akhir program.
	Garis Alir (Flow Line).	Arah aliran program.
	Preparation.	Proses inisialisasi atau pemberian harga awal.
	Proses.	Proses perhitungan atau proses pengolahan.
	Input/Output Data.	Proses input atau output data,parameter,informasi.
	Predefined Process (Sub Program).	Permulaan sub program atau proses menjalankan sub program.
	Decision.	Perbandingan pernyataan, penyeleksian data yang memberikan pilihan untuk langkah selanjutnya.
	On Page Connector.	Penghubung bagian-bagian flowchart yang berada pada satu halaman.
	Off Page Connector.	Penghubung bagian-bagian flowchart yang berada pada halaman berbeda.

2.2.8 Unified Modeling Language (UML)

UML yang merupakan singkatan dari Unified Modeling Language adalah sekumpulan alat yang digunakan untuk melakukan abstraksi terhadap sebuah sistem atau perangkat lunak berbasis objek. UML telah menjadi bahasa standar dalam industri untuk memvisualisasikan, merancang dan mendokumentasi sistem perangkat lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem (Dharwiyanti & Wahono, 2003).

a. Use-Case diagram

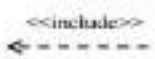
Sebuah use case menggambarkan sekumpulan aktivitas yang dilakukan untuk menghasilkan suatu hasil keluaran. Setiap use case menjelaskan bagaimana pengguna luar dapat memicu suatu 'kejadian' dimana sistem perlu menanggapinya (Dennis,Wixom & Roth, 2012). Diagram use case dapat digunakan untuk mendeskripsikan bagaimana suatu sistem akan terlihat di mata pengguna. .

Untuk menghasilkan diagram use case yang mudah dibaca dan dipahami, Ambler (2005) menyarankan beberapa panduan dalam pembuatan diagram use case, yaitu:

1. Mulai nama use case dengan kata kerja yang kuat.
2. Berikan nama use case sesuai dengan terminologi bidang kajian.
3. Susun use case sesuai dengan urutan perlakuan.
4. Letakan aktor utama pada kiri atas diagram.
5. Gambarkan aktor di luar tepi diagram.
6. Asosiasikan setiap aktor dengan satu atau lebih use case.
7. Namakan aktor sesuai perannya, bukan gelar pekerjaannya.
8. Gunakan <<system>> untuk menandakan aktor dari sistem.
9. Jangan perbolehkan interaksi antar aktor.
10. Tambahkan aktor "*Time"* untuk memulai kegiatan yang periodik.
11. Gunakan simbol asosiasi jika aktor muncul pada logika use case.
12. Hindari menggunakan tanda panah untuk relasi aktor-use case.

13. Gunakan <<include>> ketika diketahui pasti kapan use case dikerjakan.
14. Gunakan <<extend>> ketika pengerjaan use case memerlukan beberapa langkah use case lain.
15. Gunakan <<generalized>> untuk use case dimana satu kondisi dapat menghasilkan logika use case yang berbeda.
16. Hindari menggunakan relasi asosiasi use case pada dua atau lebih tingkat.
17. Letakan use case yang meng-*include* di sebelah kanan.
18. Letakan use case yang ter-*extend* dibawah use case induk.

Tabel 2.1 Simbol-Simbol Use Case Diagram

Simbol	Keterangan
	Aktor : Mewakili peran orang, sistem yang lain, atau alat ketika berkomunikasi dengan *use case*
	*Use case* : Abstraksi dan interaksi antara sistem dan aktor
	*Association* : Abstraksi dari penghubung antara aktor dengan use case
	*Generalisasi* : Menunjukkan spesialisasi aktor untuk dapat berpartisipasi dengan use case
	Menunjukkan bahwa suatu use case seluruhnya merupakan fungsionalitas dari use case lainnya
	Menunjukkan bahwa suatu use case merupakan tambahan fungsional dari use case lainnya jika suatu kondisi terpenuhi

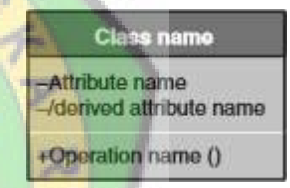
b. Class diagram

Class diagram pada UML adalah suatu diagram struktur statis yang menggambarkan suatu struktur dari sebuah sistem dengan cara memperlihatkan kelas-kelas yang dimiliki oleh sebuah sistem, beserta

atribut, operasi (atau method), dan relasi yang dimilikinya antara objek lainnya.

Kelas adalah balok pembangun utama dari sebuah diagram kelas. yang menyimpan dan mengelola informasi dalam sebuah sistem (Dennis,Wixom & Roth, 2012). Berikut adalah bentuk dasar kelas dalam sebuah diagram kelas.

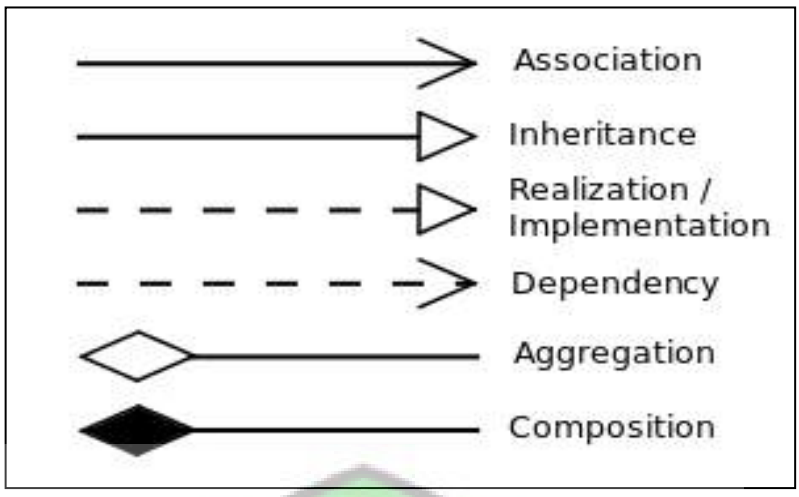
Tabel 2.2 Bagian Suatu Kelas (Dennis,Wixom & Roth, 2012)

Istilah dan Definisi	**Simbol**
Kelas • Merepresentasikan seseorang, tempat, atau sesuatu yang harus didata oleh sistem. • Memiliki nama yang ditulis tebal dan menengah pada bagian atas. • Memiliki daftar atribut pada bagian tengah. • Memiliki daftar operasi yang dapat dilakukan pada bagian bawah. • Tidak secara langsung menampilkan seluruh operasi yang tersedia bagi seluruh kelas.	
Atribut • Merepresentasikan properti yang menggambarkan kondisi suatu objek. • Dapat menurunkan dari atribut lain, ditunjukan dengan garis miring sebelum nama atribut.	Nama atribut /Nama atribut turunan

Method	
• Merepresentasikan aksi atau fungsi yang dapat dilakukan kelas. • Dapat diklasifikasikan sebagai konstruktor, *query*, atau operasi *update*. • Diikuti tanda kurung yang dapat berisi parameter atau informasi yang diperlukan untuk menjalani operasi.	Nama operasi ()

Pada class diagram juga digambarkan bagaimana interaksi hubungan antar kelas dalam sebuah konstruksi perangkat lunak seperti hubungan asosiasi, inheritance, generalisasi, agregasi, dan komposisi.

1. Asosiasi
   Menggambarkan kelas yang memiliki atribut berupa kelas lain, atau kelas yang harus mengetahui keberadaan kelas lain.
2. Inheritance
   Menggambarkan hubungan hirarkis antar kelas. Kelas dapat diturunkan dari kelas lain dan mewarisi semua atribut dan method kelas asalnya, dan menambahkan fungsionalitas baru, sehingga ia disebut anak dari kelas yang diwarisinya
3. Generalisasi
   Kebalikan dari inheritance, menggambarkan suatu hubungan dimana suatu kelas memiliki atribut yang sama dari dua atau lebih kelas lainnya.
4. Agregasi
   Menggambarkan hubungan antar kelas yang memiliki relasi “has-a”.
5. Komposisi
   Menggambarkan hubungan antar kelas yang memiliki relasi “part-of”.



Gambar 2.4 Simbol-Simbol Relasi Antar Kelas

c. Activity diagram

Activity diagram digunakan untuk menggambarkan rangkaian aliran dari aktivitas, dan berguna untuk mendeskripsikan aktifitas yang dibentuk dalam suatu operasi sehingga dapat juga digunakan untuk aktifitas lainnya seperti use case atau interaksi.

Tabel 2.3 Simbol-Simbol Activity Diagram

Keterangan	**Simbol**
Titik Awal atau permulaan.	
Titik Akhir atau akhir dari aktivitas.	
Aktivity, atau aktivitas yang dilakukan oleh aktor.	Login
Decision, atau pilihan untuk mengambil keputusan.	
Arah tanda panah alur proses.	

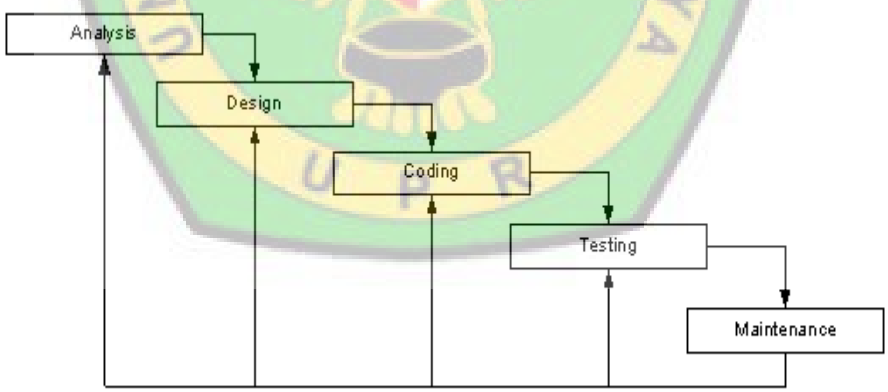
BAB III
METODOLOGI PENELITIAN

3.1 Metode Pengumpulan Data

Metode pengumpulan data yang dilakukan dalam proses desain dan pengembangan game yang dibuat adalah melalui studi pustaka yang relevan terhadap objek peneletian yang ditarik. Studi pustaka yang dilakukan mencangkup mempelajari literatur dalam bentuk buku dan jurnal, dan media online untuk mempelajari konsep dan teori yang akan digunakan dalam proses pengembangan game yang akan dibuat.

3.2 Metodologi Pengembangan Perangkat Lunak

Metodologi pengembangan perangkat lunak yang akan digunakan dalam pengembangan game "Ascent" ini adalah **Metodologi Waterfall**, yang terdiri atas fase analisis kebutuhan (*analysis*), desain sistem (*design*), pembuatan koding (*coding*), pengujian sistem (*testing*), dan pemeliharaan (*maintenance*).



Gambar 3.1 Alur Proses Metodologi Waterfall (Pressman, 1997)

1. Analysis

   Tahapan ini dilakukan analisis permasalahan, model aplikasi, dan perangkat yang diperlukan dalam pembuatan dan pengembangan game yang akan dibuat.

2. Design

Tahapan pelakukan perancangan bentuk game yang akan dibuat, tahapan perancangan akan dibagi menjadi perancangan cerita game, perancangan struktur antarmuka, perancangan aset game, perancangan arena permainan dan perancangan mekanisme game.

3. Coding

Pada tahapan ini akan dilakukan implementasi berdasarkan hasil perancangan yang telah didapat. Tahapan ini menyangkut pembuatan kode-kode game disertai dengan penggabungan aset-aset game yang telah dibuat menjadi sebuah produk akhir yang fungsional.

4. Testing

Pada tahapan ini akan dilakukan pengujian terhadap game yang telah dibuat dengan menggunakan metode *blackbox*, disertai dengan pembenahan terhadap kode game ketika terjadinya sebuah eror ketika game sedang dijalankan

5. Maintenance

Melakukan pemeliharaan terhadap aplikasi yang telah dibuat. Ini mencangkup penstrukturan ulang kode, perbaikan kode, dan penjagaan fungsionalitas aplikasi.

3.3 Analisis

Pada tahapan ini akan dilakukan analisis lebih dalam terhadap permasalahan yang telah didefinisikan sebelumnya dengan upaya untuk memecahkannya menjadi beberapa permasalahan kecil yang dapat diselesaikan secara terpisah. Berikut adalah rincian analisis yang didapat.

3.3.1 Analisis permasalahan

Analisis permasalahan dilakukan mendapatkan pemahaman terhadap permasalahan utama. Permasalahan utama yang menjadi fokus pengembangan adalah

bagaimana mengimplementasikan *Procedural Content Generation* dalam pembuatan arena permainan menggunakan dasar algoritma *Cellular Automata*.

3.3.2 Analisis model

Game yang dikembangkan akan didesain untuk dapat dimainkan pada platform komputer dengan sistem operasi Windows. Model interaksi yang dapat dilakukan oleh pemain dilakukan melalui perantara perangkat keras keyboard dan mouse.

3.3.3 Analisis pengguna

Target pengguna untuk game yang akan dibuat diperuntukan kepada mahasiswa sebagai bahan studi.

3.3.4 Analisis kebutuhan

Analisis kebutuhan mencangkup segala peralatan yang diperlukan dalam pengembangan aplikasi yang diinginkan. Peralatan yang digunakan akan dibagikan menjadi dua bagian berupa perangkat keras dan perangkat lunak.

a) Perangkat Keras.

Mencangkup segala perangkat keras yang digunakan dalam proses pengembangan game. Berikut adalah daftar perangkat keras yang digunakan:

1. Laptop

   Perangkat keras laptop digunakan untuk menjalankan perangkat-perangkat lunak yang diperlukan dalam proses pengembangan aplikasi. Semua kegiatan koding, pendesainan, dan dokumentasi dilakukan menggunakan perangkat keras laptop. Berikut adalah spesifikasi perangkat yang digunakan:

   a. Processor Intel Core i7 4720HQ
   b. VGA NVIDIA GeForce GTX 950M
   c. RAM 8 GB
   d. HDD 1 TB

b) Perangkat Lunak.

Mencangkup segala perangkat lunak yang digunakan dalam proses pengembangan game. Berikut adalah daftar perangkat lunak yang digunakan:

1. Unity Editor

Unity Editor adalah sebuah IDE (*Integrated Development Environment)* dari Unity Technologies didesain untuk mengembangkan game menggunakan game engine Unity. Versi Unity Editor yang digunakan adalah versi 2019.4.16f1 dengan lisensi Personal Edition.

2. Blender 3D

Blender 3D adalah aplikasi open source grafis 3D yang dikembangkan oleh Blender Foundation. Di dalam aplikasi Blender 3D tersedia peralatan-peralatan yang dapat digunakan untuk mebuat model 3D, animasi, dan efek visual. Versi Blender yang digunakan dalam pengembangan game adalah Blender 2.80.

3. Microsoft Paint

Microsoft Paint adalah sebuah editor grafis yang disediakan untuk setiap vesi sistem operasi Windows. Versi yang digunakan dalam pengembangan adalah versi bawaan sistem operasi Windows 7.

4. UMLet

UMLet adalah sebuah aplikasi pembuatan diagram UML yang open source. Seluruh diagram dibentuk dalam tahap pengembangan aplikasi akan dilakukan pada program UMLet. Versi program yang digunakan pada pengembangan aplikasi adalah versi 14.3.

3.3.5 Analisis Parameter Arena

Analisis Parameter Arena mencangkup analisis terhadap aspek-aspek dalam pembuatan arena yang dapat dimanipulasikan untuk menghasilkan bentuk arena yang berbeda.

Aspek-aspek yang dapat dikontrol dalam proses pembuatan arena dapat dibagi menjadi beberapa bagian berupa:

1. Dimensi

   Dimensi merupakan ukuran atau luas arena yang dibuat, hal ini mencangkup panjang, lebar, dan tinggi arena yang akan dibuat, sehingga untuk mengatu dimensi arena, akan dibuat parameter berupa ***Width*** untuk lebar, ***Height*** untuk panjang, ***Depth*** untuk ketinggian, ***Square Scaling*** untuk skala arena, dan tambahan ***Border Size*** untuk mengatur ketebalan perbatasan arena.

2. Nilai Awal Matriks

   Nilai awal yang diisikan kepada matriks yang akan diiterasikan dengan algoritma cellular automata. Dikarenakan proses pembuatan akan dilakukan sebagian besar oleh komputer, maka nilai yang diberikan pada matriks akan bersifat acak. Untuk dapat mengontrol nilai acak yang diberikan, maka akan digunakan parameter ***Fill Percent*** yang menentukan rasio jumlah sel hidup dan sel mati yang ada pada matriks awal.

3. Jumlah Iterasi

   Iterasi cellular automata akan dilakukan beberapa kali untuk mengubah matriks awal yang acak menjadi bentuk yang teratur. Jumlah maksimum iterasi cellular automata yang dilakukan akan dikontrol dengan menggunakan parameter ***Smooth Iteration***.

4. Penghapusan Wilayah

   Setelah proses iterasi telah dilakukan, perlu dilakukan pengecekan untuk mencari wilayah yang terlalu kecil untuk digunakan dalam arena permainan, kemudian menghapusnya. Untuk menentukan ukuran yang dianggap terlalu kecil untuk digunakan akan dibuat parameter ***Wall Culling Threshold*** *dan* ***Room Culling Threshold***.

5. Penempatan Musuh

Penempatan musuh dilakukan dengan membagikan arena menjadi beberapa segmen yang dicek apakah valid untuk ditempati oleh musuh. Parameter yang mengatur penempatan musuh berupa ***Segment Width***, yang mengatur luas segmen pembagi arena, ***Valid Segment Threshold***, digunakan untuk menentukan segmen yang dapat digunakan, ***Max Spawn Point*** menentukan jumlah maksimum segmen yang dapat ditempati musuh, dan terakhir ***Min/Max Enemy in Spawn*** digunakan untuk menentukan jumlah minimum dan maksimum musuh yang dapat muncul pada suatu segmen.

3.4 Perancangan Game

Tahapan perancangan dilakukan untuk memberikan gambaran terhadap game yang akan dibangun. Pada tahapan ini akan dilakukan perancangan berdasarkan hasil analisis yang telah didapat.

3.4.1 Cerita

Game yang dibuat akan menceritakan karakter utama yang berupa seorang teknisi sistem sebuah kapal luar angkasa. Ketika dalam sebuah perjalanan jauh menuju suatu planet di luar angkasa, kapal yang ditempati karakter utama diserang oleh sebuah virus komputer cerdas, yang menyebabkan perjalanan kapal berhenti, dan seluruh awak kapal yang sedang dalam hibernasi menjadi tidak dapat terbangun.

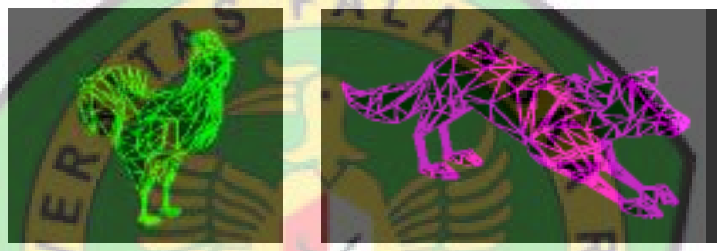
A.I. yang ditugaskan untuk mengamati kondisi kapal dan penumpangnya hanya sempat menyelamatkan karakter utama dengan cara memindahkan kesadaran karakter utama ke dalam dunia virtual sistem kapal, sebelum seluruh sistem telah dikunci oleh virus tersebut. Tetapi karena prosesnya yang sangat terburu-buru, A.I. tersebut tidak dapat menyalin bentuk tubuh karakter utama yang sebenarnya, sehingga dibuatlah sebuah badan virtual sederhana yang dapat cepat dibuat dalam bentuk seekor ayam yang dapat dikontrol oleh karakter utama sementara ia berada dalam dunia virtual.

Ketika menyadari bahwa karakter utama berhasil lolos dan telah memasuki dunia virtual kapal, virus tersebut mengubah bentuk dunia virtual kapal menjadi

labirin gua yang berliku-liku, dan menempatkan musuh-musuh yang berbentuk serigala dalam upaya untuk menghentikan karakter utama. Bersama dengan bantuan A.I. kapal, karakter utama berusaha untuk mengambil balik alih kapal dari virus tersebut, yang terletak pada level tertinggi di labirin.

3.4.2 Karakter

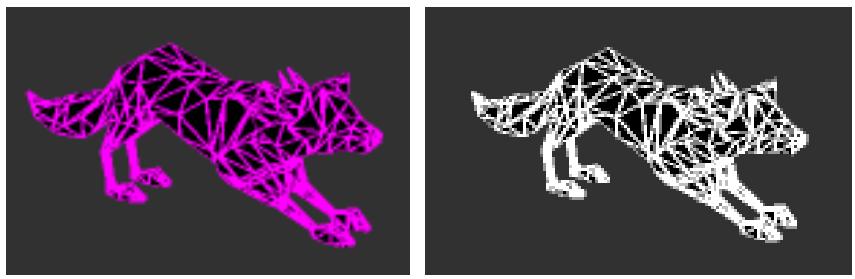
Terdapat dua model 3D yang akan digunakan sebagai karakter dalam game, yang pertama adalah **model 3D ayam** yang akan dimainkan oleh pemain. Yang kedua berupa **model 3D serigala** yang digunakan sebagai musuh yang akan dihadapi oleh pemain dalam arena labirin.



Gambar 3.2 Model 3D Ayam (Kiri) dan Model 3D Serigala (Kanan)

Selain dari karakter utama dan karakter musuh, terdapat juga karakter bos yang akan digunakan pada level terakhir dari permainan. Model 3D yang digunakan untuk karakter bos adalah sama dengan model yang digunakan pada musuh yang normal, tetapi selain dari varian normal yang memiliki warna **ungu**, terdapat satu varian tambahan yang mempunyai warna **putih**.

Kedua varian model musuh yang berwarna ungu dan berwarna putih akan digunakan pada level terakhir ketika pemain menghadapi bos.



Gambar 3.3 Bos Varian Normal (Kiri) dan Varian Putih (Kanan)

3.4.3 Gameplay

Model permainan yang akan digunakan dalam game adalah model *RPG*, dimana pemain dapat mengontrol karakter secara langsung dan menghadapi musuh secara *real-time*. Setiap musuh yang dikalahkan oleh pemain akan memberikan pemain sebuah poin bernama *exp*, dimana setelah mengakumulasi cukup poin *exp*, pemain dapat meningkatkan kekuatan karakater yang dimainkan.

Naratif permainan akan diceritakan melalui dialog antara karakter utama dengan karakter lainnya, melalui antarmuka dialog yang ada pada game. Dialog akan muncul ketika pemain memasuki bagian penting dalam permainan, seperti ketika pertama kali memasuki labirin.

Permainan akan mulai dari area aman, dimana tidak terdapat musuh. Dari area tersebut, pemain dapat mengakses labirin yang akan dibuat dengan algoritma Cellular Automata yang telah diimplementasi. Labirin terdiri atas 4 tingkat, 3 tingkat pertama berupa arena normal, sementara 1 terakhir berupa arena bos.

Untuk berpindah ke tingkat selanjutnya, pemain harus mengalahkan seluruh musuh pada tingkat yang ditempati. Untuk memenangkan permainan, pemain harus menyelesaikan seluruh tingkat labirin.

3.4.4 Antarmuka pengguna

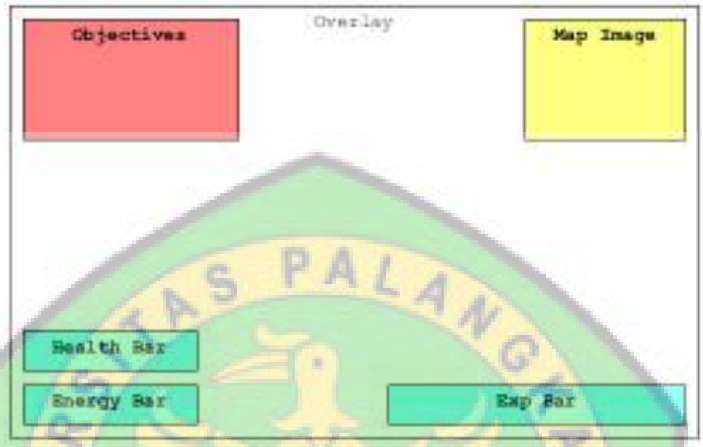
Antarmuka pengguna akan dibagikan menjadi dua, yang pertama adalah *overlay*, (tertempel diatas permainan). Interface ini akan selalu ditampilkan pada saat pemain berada dalam area permainan, dan interaksi pemain dengan antarmuka ini sangat minimal, dimana antarmuka hanya digunakan untuk menampilkan informasi penting yang relevan dengan permainan.

Interface kedua berupa dalam bentuk menu interaktif yang digunakan pemain untuk melakukan kegiatan seperti berpindah area dan meningkatkan kekuatan karakter. Antarmuka ini berisikan elemen-elemen interaktif seperti tombol dan bidang teks yang dapat diinteraksikan oleh pemain.

Berikut adalah daftar rancangan interface yang akan digunakan dalam game yang akan dibuat:

a) Desain Antarmuka Overlay

Antarmuka Overlay akan ditampilkan selama pemain berada dalam area permainan. Informasi-informasi yang ditampilkan berupa jumlah *Health* dan *Energy* yang dimiliki pemain, peta arena permainan, dan jumlah *EXP* yang dimiiliki oleh pemain.



Gambar 3.4 Rancangan Antarmuka Overlay

b) Desain Antarmuka Main Menu

Main Menu merupakan antarmuka pertama yang akan ditemui pemain ketika memasuki game. Main menu terdiri atas empat tombol, berupa *Test CA Map Generation* yang akan membawa pemain ke level pengujian algoritma CA, tombol *Continue* digunakan untuk memulai permainan dengan data karakter yang sebelumnya tersimpan, *New Game* digunakan untuk memulai permainan dengan data baru, dan tombol *Exit* untuk keluar dari game.

Gambar 3.5 Rancangan Antarmuka Main Menu

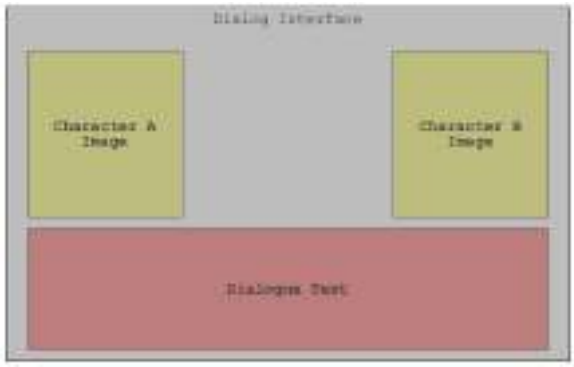
c) Desain Antarmuka Pause Menu

Pause menu dapat diakses oleh pemain ketika menekan tombol *Esc* pada keyboard ketika dalam area permainan. Menu terdiri atas tiga tombol, berupa tombol *Continue* menutup menu pause, *Return To Safe Zone* untuk mengembalikan pemain ke area aman, dan *Exit To Main Menu* untuk kembali ke Main Menu.

Gambar 3.6 Rancangan Antarmuka Pause Menu

d) Desain Antarmuka Dialog

Antarmuka Dialog digunakan untuk menampilkan percakapan antar karakter. Antarmuka terdiri atas dua gambar *Character A,* dan *Character B* yang merepresentasikan dua karakter yang sedang saling berbicara, dan sebuah lapang teks *Dialogue* yang menampilkan teks percakapan.



Gambar 3.7 Rancangan Antarmuka Dialog

e) Desain Antarmuka Status Menu

Antarmuka Status Menu dapat diakses oleh pemain dengan menekan tombol *Tab* pada keyboard ketika karakter berada pada area permainan. Pada menu ini, pemain dapat meningkatkan kekuatan karakter utama setelah karakter *level up*. Antarmuka terdiri atas *Player Level* yang menunjukan level karakter saat ini, *Skill Points* yang menampilkan poin yang dapat digunakan untuk menaikkan kekuatan karakter, tombol + untuk menambahkan kekuatan karakter dengan menggunakan poin, tombol – untuk mengembalikan poin yang digunakan, dan tombol *Confirm* untuk mengkonfirmasi penggunaan poin, dan menyimpan data karakter yang telah berubah.

Gambar 3.8 Rancangan Antarmuka Status Menu

f) Desain Antarmuka Map Generation Menu

Antarmuka Map Generation Menu dapat diakses dari Main Menu. Pada menu ini, pemain dapat menguji proses pembuatan arena permainan yang ada pada game.

Antarmuka Map Generation Menu terdiri atas *Controls* yang menampilkan tombol-tombol keyboard yang dapat digunakan, dan *Map Parameters* yang terdiri atas variabel-variabel yang dapat diubah sebelum men-*generate* arena permainan.

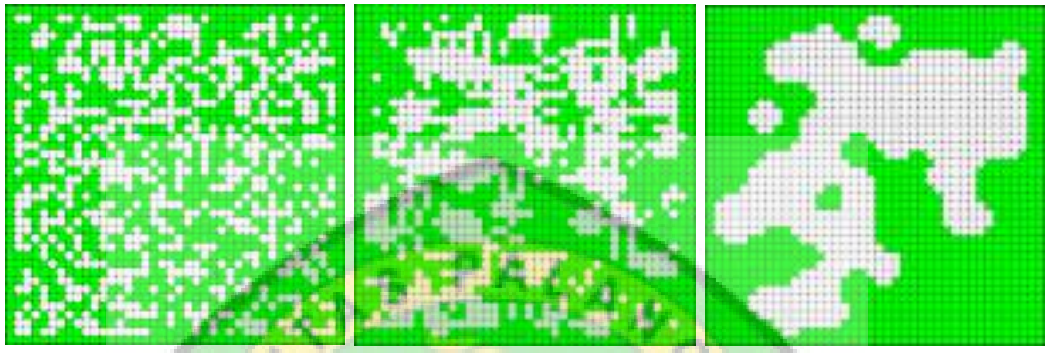
Gambar 3.9 Rancangan Antarmuka Map Generation Menu

3.4.5 Arena permainan

Arena permainan akan dibentuk dengan dasar algoritma Cellular Automata, pertama-tama akan dibentuk sebuah matriks dua dimensi dengan panjang dan lebar yang telah ditentukan. Pada setiap koordinat (x,y) atau sel dalam matriks akan memiliki nilai antara 0 (sel mati), dan 1 (sel hidup) yang akan diisikan secara acak.

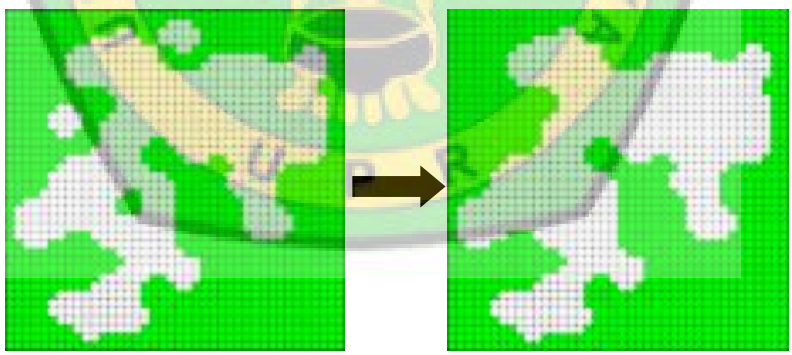
Gambar 3.10 Kondisi Awal Matriks 2D Setelah Diisi secara Acak

Terhadap matriks yang telah berisikan nilai, akan dilakukan iterasi dimana setiap sel akan dicek jumlah tetangga sekitarnya yang memiliki nilai 1. Jika jumlah tetangga tersebut melebihi 4, maka sel yang dicek akan diubah nilainya menjadi 1, dan sebaliknya jika jumlah tetangga kurang dari 4, nilai sel akan diubah menjadi 0. Iterasi akan dilakukan sebanyak yang telah ditentukan.



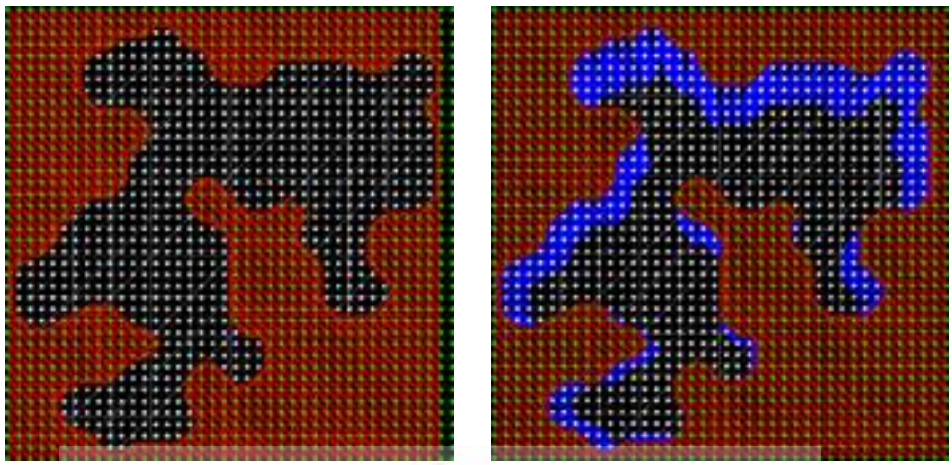
Gambar 3.11 Perubahan Mariks setelah Dilakukan Iterasi CA

Hasil terakhir iterasi akan diproses lebih lanjut untuk menghilangkan wilayah yang terlalu kecil, dan juga memastikan agar setiap wilayah yang terpisah dapat terhubung pada wilayah yang memiliki area terbesar.



Gambar 3.12 Penghapusan Wilayah Yang Terlalu Kecil

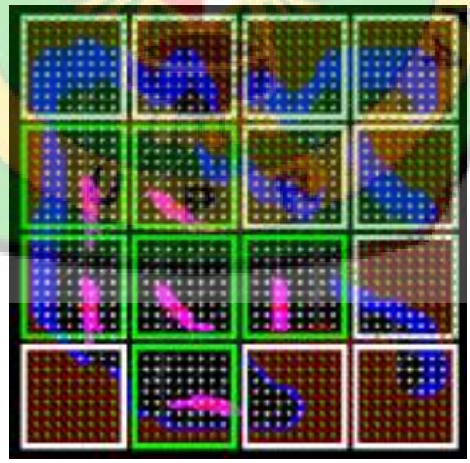
Matriks akhir yang telah selesai dibentuk akan digunakan untuk membuat bentuk objek 3D atau *mesh* yang akan digunakan pada game. Proses pembuatan *mesh* dilakukan secara algoritmik, dimulai dari pembuatan bagian atas dinding, diikuti dengan pembuatan tepi dinding.



Gambar 3.13 Pembuatan *Mesh* Dinding Arena Permainan

Untuk memastikan musuh dapat ditempatkan pada area yang sah, arena akan dibagikan menjadi beberapa persegi atau segmen yang akan dicek secara satu persatu untuk menentukan apakah area tersebut dapat digunakan untuk menempatkan musuh.

Di dalam setiap segmen akan dibandingkan jumlah antara sel hidup dan sel mati, jika jumlah sel mati melebihi nilai ambang yang telah ditentukan, maka segmen tersebut akan ditandakan sebagai area yang sah.



Gambar 3.14 Penempatan Musuh pada Segmen Valid (Kotak Hijau)

Terakhir, musuh akan ditempatkan pada area yang telah didapat. Dimana jumlah maksimum segmen yang akan digunakan dan jumlah total musuh yang akan ditempatkan dalam setiap segmen bergantung pada nilai parameter yang diberikan sebelum dilakukan generasi arena.

Agar pembuatan arena dapat dikontrol, akan diletakan beberapa parameter dalam script yang nilainya dapat diubah untuk mempengaruhi hasil akhir arena yang akan didapat. Berikut adalah parameter-parameter yang digunakan.

1. Width, Height, Depth

   Parameter height, width dan depth digunakan untuk mengatur masing-masing panjang, lebar, dan tinggi area permainan. Unit yang digunakan berupa unit koordinat bawaan dari game engine yang digunakan.

2. Square Scaling

   Digunakan untuk memperbesar ukuran area permainan berdasarkan kelipatan nilai yang diberikan. Hal ini dilakukan dengan memperlebar jarak antar sel pada matriks.

3. Border Size

   Digunakan untuk mengatur lebar dinding yang mengelilingi sekitar area permainan.

4. Fill Percent

   Fill percent digunakan untuk menentukan kondisi awal cell yang akan diproses, nilai yang diberikan berupa persentase antara 0 – 100, dimana angka persen menentukan jumlah persentase sel yang hidup pada awal pembuatan area permainan.

5. Seed/Random Seed

   Seed adalah sebuah runtutan karakter alfanumerik yang digunakan untuk meninisialisasikan kondisi awal sebuah pseudo random generator. Dengan menyediakan seed yang sama, maka hasil akhir yang diberikan oleh pseudo random generator akan serupa.

6. Smooth Iteration

Menentukan jumlah iterasi cellular automata yang akan dilakukan terhadap matriks 2D arena permainan. Iterasi dilakukan sebelum proses penghapusan wilayah.

7. Room/Wall Culling Threshold

Menentukan ambang batas jumlah sel yang akan dihapus pada hasil matriks yang telah di-iterasikan. Jika sebuah wilayah memiliki jumlah sel yang lebih kecil dibandingkan nilai ambang tersebut, maka wilayah tersebut akan dihapuskan.

8. Segment Width

Menentukan lebar segmen yang digunakan untuk membagikan arena menjadi beberapa bagian kecil. Segmen yang akan dibuat berbentuk persegi sama sisi, sehingga nilai Width digunakan untuk menentukan panjang dan juga lebar segmen.

9. Valid Segment Threshold

Nilai rasio antara sel hidup dan sel mati pada suatu segmen yang digunakan untuk menentukan apakah sebuah segmen dapat digunakan untuk memunculkan musuh

10. Max Enemy Spawn Points

Jumlah maksimum segmen yang akan digunakan untuk meletakan musuh pada arena. Segmen yang digunakan untuk penempatan musuh diambil secara acak dari daftar segmen yang valid.

11. Min/Max Enemy in Spawn

Jumlah musuh yang dapat berada sekaligus dalam sebuah segmen. Nilai *Min* menentukan jumlah minimum musuh, sementara nilai *Max* menentukan jumlah maksimum.

3.5 Perancangan Diagram

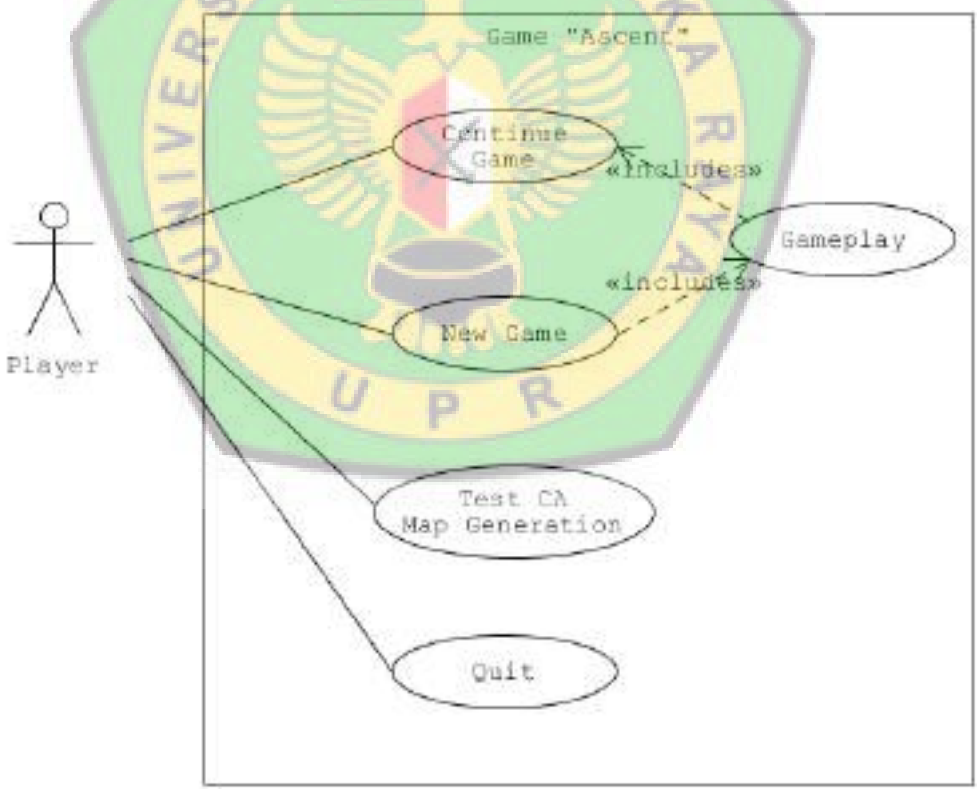
Pada tahapan ini dilakukan pemodelan diagram UML yang akan digunakan sebagai acuan dalam proses implementasi kode game. Diagram yang akan digunakan pada tahapan ini berupa diagram *use case*, diagram *activity*, dan diagram *class*. Berikut adalah diagram-diagram yang digunakan.

3.5.1 Use case

Use case diagram digunakan untuk menggambarkan urutan kegiatan yang dilakukan oleh seorang aktor dalam sistem untuk mencapai suatu tujuan tertentu.

a) Diagram Use Case

Diagram use case berikut memodelkan aktivitas yang ada dalam game yang akan dibuat.



Gambar 3.15 Diagram Use Case

b) Definisi Aktor

Aktor merupakan entitas eksternal yang terlibat dalam berbagai kegiatan yang dimiliki sistem. Berikut adalah deskripsi aktor yang

menjelaskan aktor-aktor yang akan terlibat, dan perannya dalam sebuah sistem yang akan dibangun.

Tabel 3.1 Definisi Aktor

Aktor	Deskripsi
Player	Aktor yang akan memainkan game yang dibuat beserta fitur-fitur game yang tersedia.

c) Definisi Use Case

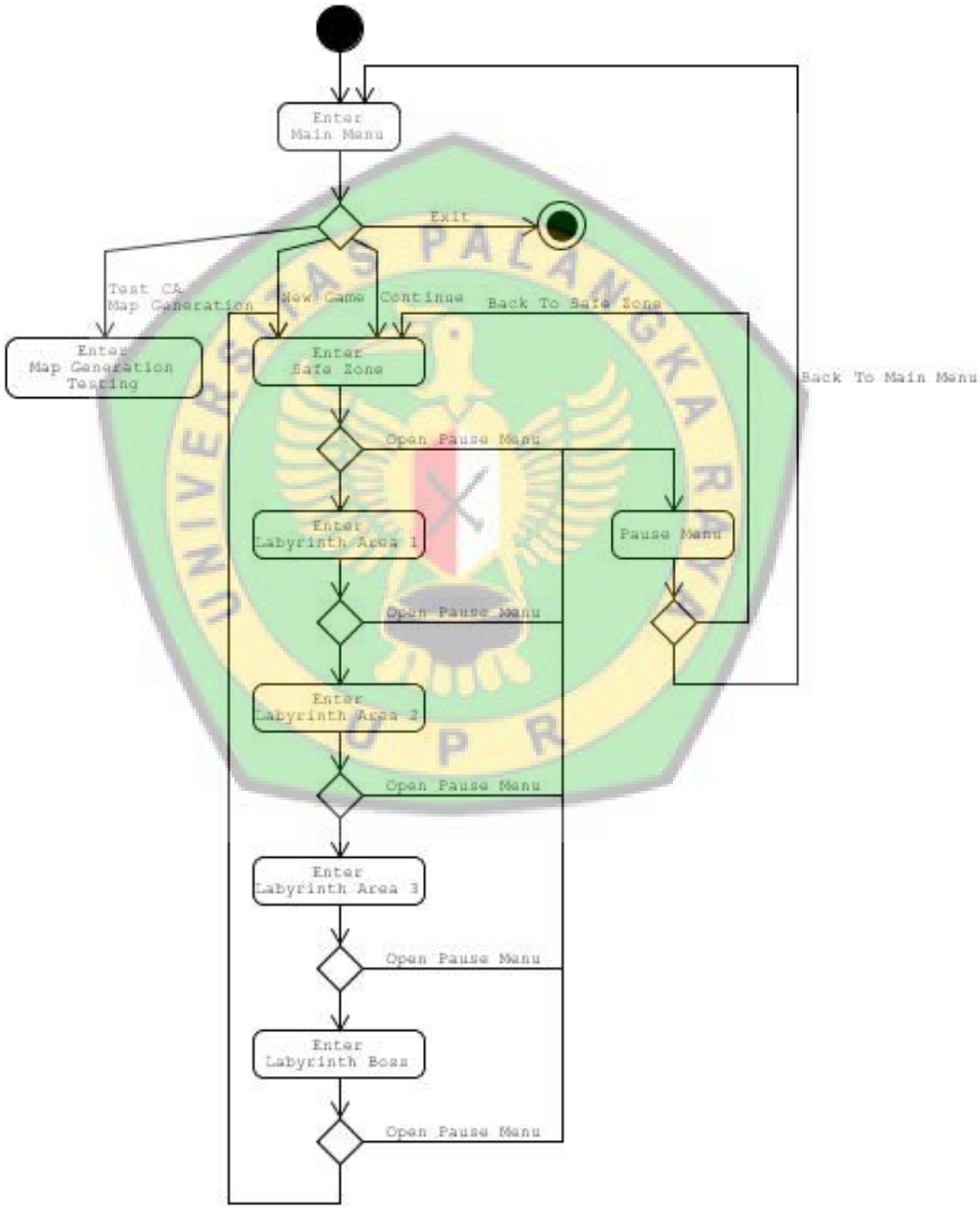
Deskripsi Use Case menggambarkan aktivitas-aktivitas yang dapat dilakukan oleh aktor dalam sebuah sistem, kondisi awal, proses, dan kondisi akhir. Berikut adalah deskripsi use case yang ada dalam sistem.

Tabel 3.2 Definisi Use Case

No.	Use Case	Deskripsi
1.	Continue Game	*Pre-condition:* Pemain pernah memainkan permainan sebelumnya *Post-condition:* Pemain melanjutkan permainan dengan data save yang sebelumnya
2.	New Game	*Pre-condition:* Pemain ingin memulai permainan dengan data save baru *Post-condition:* Pemain memulai permainan dengan data save baru
3.	Gameplay	*Pre-condition:* Pemain ingin memainkan permainan *Post-condition:* Pemain
4.	Test CA Map Generation	*Pre-condition:* Pemain ingin melihat proses pembuatan arena/map permainan *Post-condition:* Pemain memasuki menu testing
5.	Quit	*Pre-condition:* Pemain ingin menutup permainan *Post-condition:* Aplikasi ditutup

3.5.2 Activity diagram

Diagram Aktivitas digunakan untuk menggambarkan alur kerja yang dilalui ketika menjalani suatu proses. Alur kerja dalam aplikasi yang dibuat akan digambarkan dalam sebuah diagram aktivitas. Berikut adalah diagram aktivitas pada aplikasi.

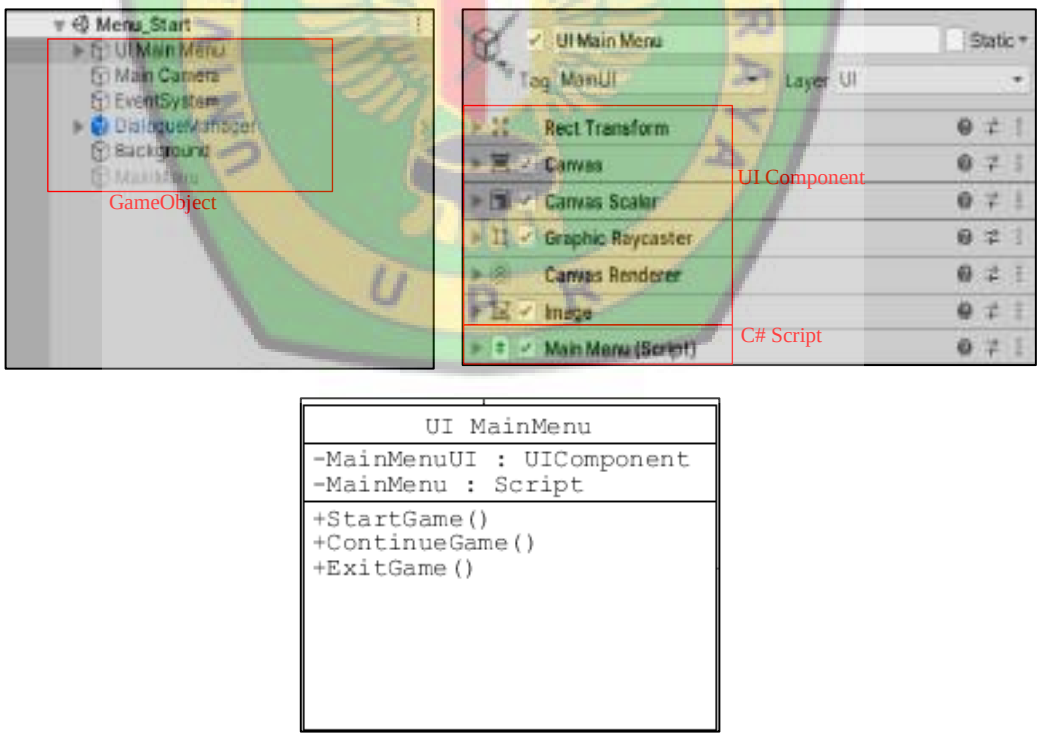


Gambar 3.16 Diagram Activity

3.5.3 Class diagram

Diagram kelas menunjukan struktur dari sistem yang akan dibuat dalam segi pendefinisian kelas-kelas yang membangun seluruh sistem. Dikarenakan struktur objek pada game engine *Unity* bersifat sedikit berbeda dari struktur kelas tradisional pada pemrograman berorientasi objek bahasa pemrograman C#, maka diagram kelas dalam konteks pembuatan game ini akan digunakan untuk menampilkan relasi antar *GameObject* yang ada pada game yang akan dibuat, dengan upaya untuk mempersingkat dan memperjelas .

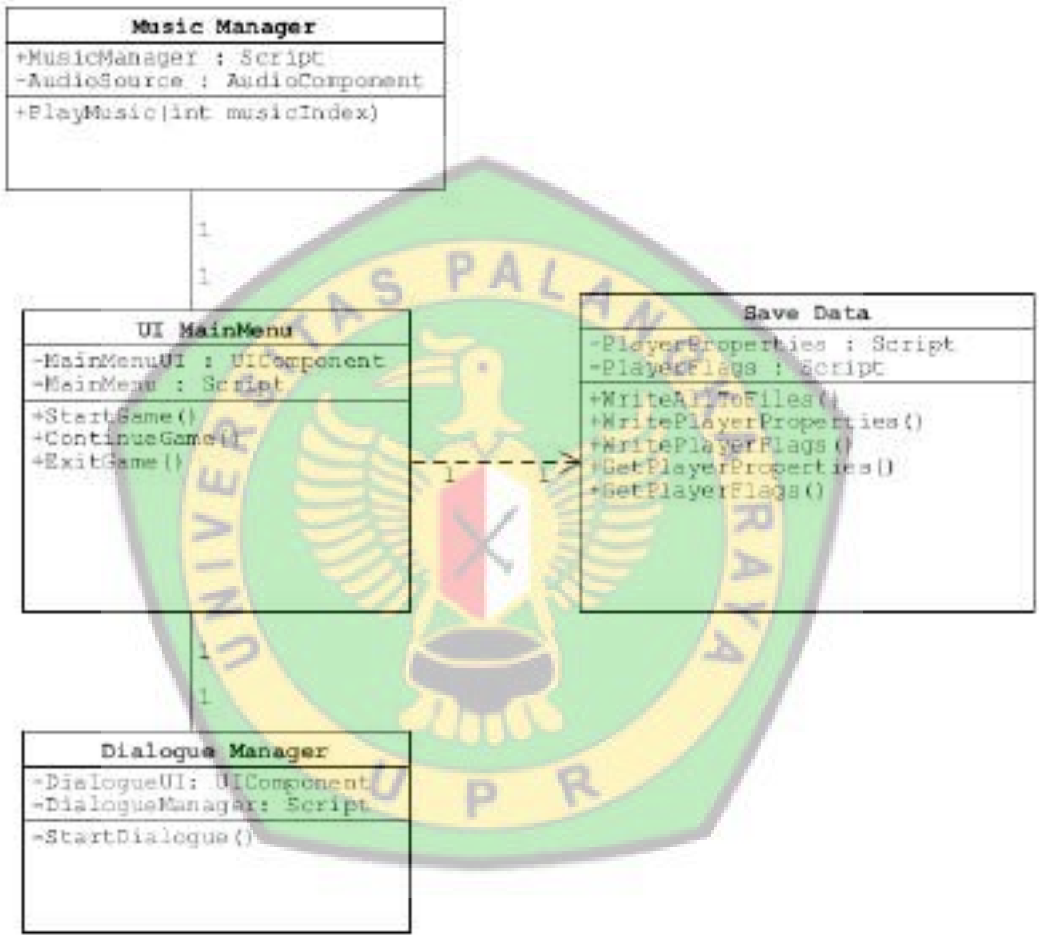
GameObject adalah sebuah kelas bawaan Unity yang menjadi basis dari seluruh entitas permainan. Serupa dengan struktur kelas tradisional, sebuah *GameObject* mempunyai peran dan karakteristik yang mempengaruhi jalannya aplikasi, dengan perbedaan dimana selain dapat memuat script atau koding, GameObject juga dapat memuat komponen-komponen tambahan Unity seperti *Audio*, *Mesh*, *UI*, *Navigation*, dll.



Gambar 3.17 Tampilan Daftar *GameObject* dan *Component* pada Unity

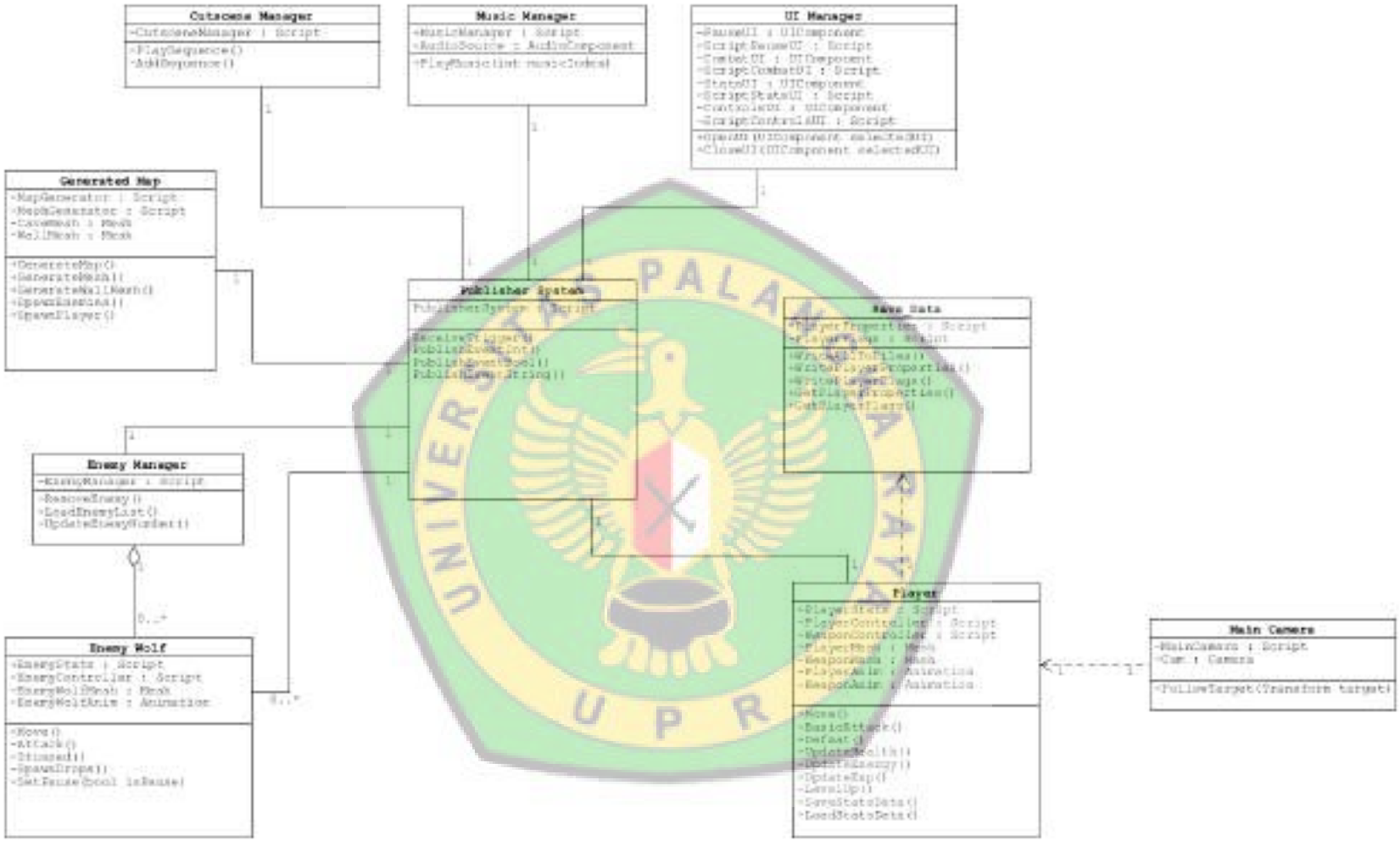
Untuk pembuatan diagram kelas akan dibagikan menjadi dua, yaitu struktur kelas diagram yang digunakan dalam *Scene Main Menu*, dan struktur kelas diagam yang digunakan pada *Scene SafeZone Cave1, Cave2, Cave3, CaveBoss*

Berikut adalah perancangan diagram kelas yang akan digunakan dalam proses implementasi pengkodean game.



Gambar 3.18 Diagram Kelas *Scene Main Menu*

Berikut adalah struktur diagram kelas yang digunakan pada scene arena permainan. Pada diagram tersebut terdapat satu kelas pusat bernama *Publisher System* yang berfungsi sebagai kelas perantara. Kelas ini bertugas untuk menerima sebuah *event* yang dapat dipicu oleh kelas lainnya. Tergantung pada *event* yang diterimanya, kelas *Publisher System* dapat memberitahukan kepada kelas lain yang bersangkut untuk melakukan aksi spesifik berdasarkan *event* yang telah terpicu.



Gambar 3.19 Diagram Kelas *Scene SafeZone Cave1, 2, 3*, dan *CaveBoss*

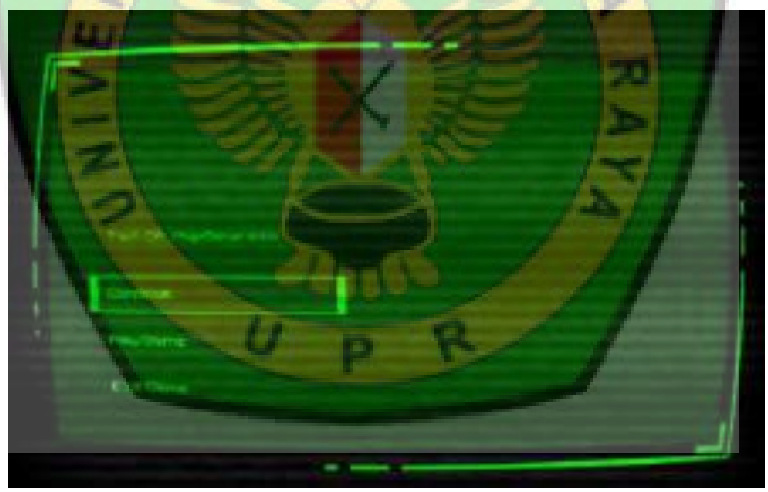
BAB IV
HASIL DAN PEMBAHASAN

4.1 Implementasi

Pada tahapan ini akan dilakukan penerapan hasil analisis dan desain yang telah didapat sebelumnya menjadi sebuah program yang dapat digunakan. Untuk mengimplementasikannya ke dalam game engine Unity, game akan dibagikan menjadi area/level terpisah yang disebut sebagai *scene*.

Sebuah *scene* terdiri atas sejumlah *game object* yang merepresentasikan sebuah entitas yang ada pada sebuah area game. Setiap *game object* dapat diberikan sejumlah *script* (file koding) yang dapat mempengaruhi tingkah laku dan peran objek tersebut dalam game.

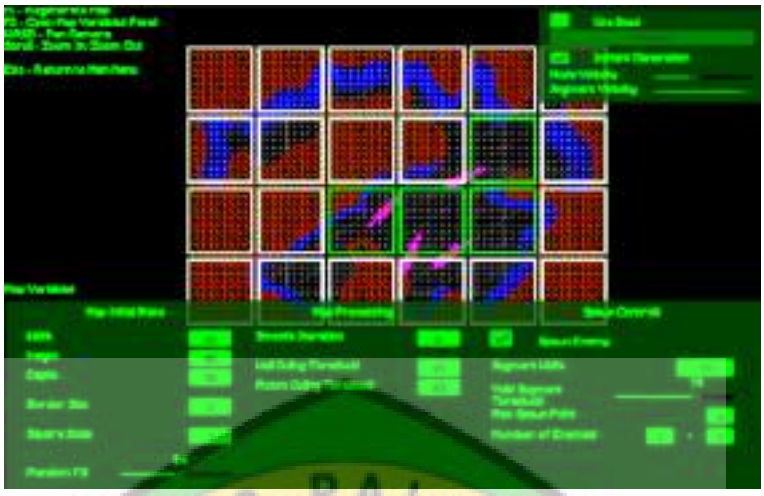
4.1.1 Scene Main Menu



Gambar 4.1 Scene Main Menu

Main menu merupakan scene yang paling pertama ditemui oleh pemain ketika membuka game. Scene terdiri atas 4 pilihan tombol yang akan memindahkan pemain ke scene lainnya pada game. Tombol **Test CA_MapGeneration** akan membawa pemain ke scene testing pembuatan arena labirin permainan; tombol **Continue** akan memulai permainan dengan data karakter yang tersimpan pada sesi permainan sebelumnya; tombol **New Game** akan memulai permainan dengan data karakter baru; dan tombol **Exit Game** akan menutup aplikasi game.

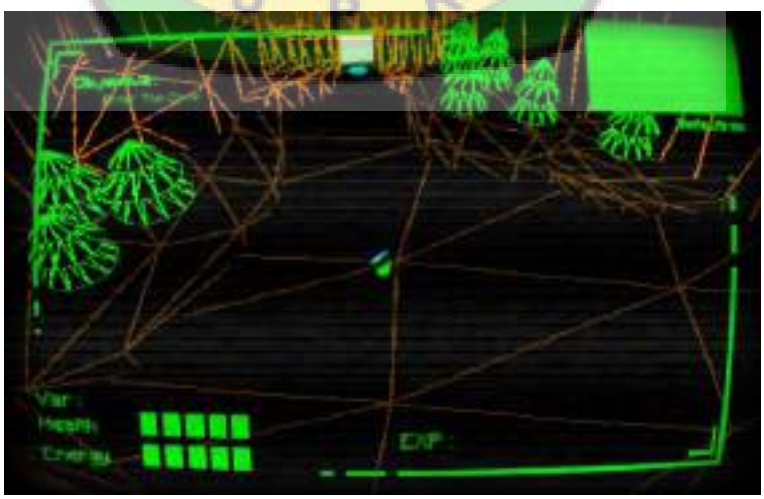
4.1.2 Scene Testing



Gambar 4.2 Scene Test CA_MapGeneration

Scene testing dapat diakses pemain melalui scene **Main Menu** dengan memilih tombol **Test CA_MapGeneration**. Pada scene ini pemain dapat menguji proses pembuatan arena permainan yang digunakan dalam game. Bentuk akhir arena yang dihasilkan dapat dipengaruhi dengan mengubah nilai parameter-parameter pembuatan arena pada antarmuka form yang berada di bagian bawah layar permainan. Pemain dapat mengulang pembuatan arena dengan menekan tombol **F1**.

4.1.3 Scene Safe Zone



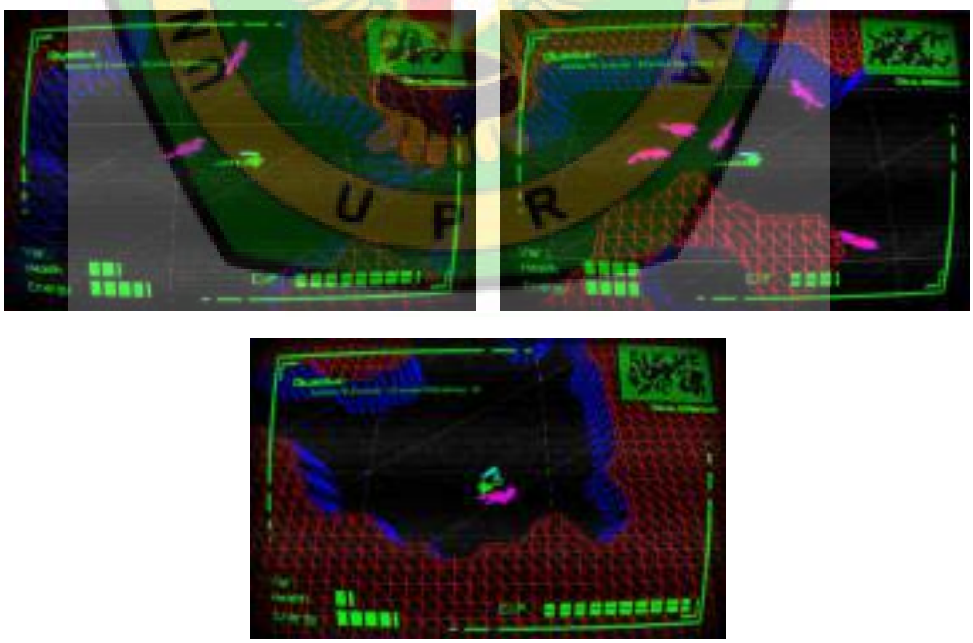
Gambar 4.3 Scene Safe Zone

Scene **Safe Zone** dapat diakses dengan memilih tombol **Continue** atau **New Game** yang ada pada scene **Main Menu**, dan pemain akan selalu memulai permainan dari scene ini. Karakter yang dimainkan pemain dapat dikontrol menggunakan tombol W,A,S,D pada keyboard, sementara untuk menyerang, pemain dapat menekan tombol kiri pada mouse.

Pada safe zone terdapat sebuah objek *gate* berupa lingkaran biru yang terletak pada mulut gua (gambar 4.3). Ketika karakter pemain menginjak *gate* tersebut, maka pemain akan dipindahkan ke tingkat pertama pada labirin.

4.1.4 Scene Cave 1,2, dan 3

Scene **Cave1**, **Cave2**, dan **Cave3** berupa 3 tingkat pertama labirin yang harus dilalui pemain sebelum dapat menghadapi bos yang berada pada tingkat terakhir labirin. **Cave1** dapat diakses dari scene **Safe Zone** dengan berjalan menuju mulut gua yang ada pada safe zone (gambar 4.3), sementara untuk berpindah ke scene selanjutnya, pemain harus mengalahkan semua musuh yang ada pada tingkat sebelumnya.



Gambar 4.4 Scene Cave1 (Kiri), Cave2 (Kanan), dan Cave3 (Bawah)

Ketika pemain telah berhasil mengalahkan seluruh musuh yang ada pada tingkat yang ditempati, maka akan diletakan sebuah *gate* dengan lokasi acak pada arena permainan yang ketika diinjak oleh karakter pemain, akan memindahkan pemain ke arena pada tingkat selanjutnya.

Gambar 4.5 Penempatan *Gate* (Lingkaran Biru) Pada Arena

Dalam pembuatannya, Cave1, Cave2 dan Cave3 menggunakan metode generasi yang sama. Yang membedakan diantaranya adalah nilai-nilai parameter yang ditentukan pada awal pembuatan masing-masing scene. Berikut adalah perbedaan antara **Cave1**,**Cave2**, dan **Cave3** :

Tabel 4.1 Perbedaan Parameter Cave1, 2, dan 3

Parameters	Cave1	Cave2	Cave3
Width	**80**	**120**	**160**
Height	**40**	**60**	**80**
Depth	10	10	10
Square Scaling	2	2	2
Border Size	10	10	10
Fill Percent	54	54	54
Seed	Random	Random	Random
Smooth Iteration	6	6	6
Room Culling	30	30	30
Wall Culling	50	50	50
Segment Width	10	10	10
Valid Segment	70	70	70
Max Spawn Point	**8**	**10**	**12**
Min Enemy	**1**	**1**	**2**
Max Enemy	**1**	**2**	**2**

Berdasarkan tabel diatas dapat dilihat semakin tinggi tingkat yang ditempuh oleh karakter pada labirin, maka tingkat kesulitannya juga akan meningkat dalam bentuk area yang lebih besar, dan bertambahnya jumlah musuh yang akan ditemui oleh pemain.

4.1.5 Scene Cave Boss

Gambar 4.6 Scene Cave Boss

Scene **Cave Boss** merupakan tingkat yang paling terakhir pada labirin. Scene ini berbeda dari scene sebelumnya, dimana arena berbentuk ruang terbuka dan pemain akan menghadapi bos yang hanya dapat dikalahkan dengan menggunakan suatu cara tertentu.

Berikut adalah konfigurasi nilai parameter yang digunakan untuk membentuk scene **Cave Boss**:

Tabel 4.2 Parameter Arena Cave Boss

Width	80	
Height	40	
Depth	10	
Square Scaling	2	
Border Size	10	
Fill Percent	0	
Seed	Random	
Smooth Iteration	6	
Room/Wall Culling	40	40
Segment Width	10	
Valid Segment	70	
Max Spawn Point	12	
Min/Max Enemy	1	1

Selain bentuk arena yang berbeda, terdapat beberapa peraturan tambahan yang diberikan pada scene ini yang berbeda dari scene sebelumnya. Peraturan tersebut berupa:

1. ‘Bos’ yang perlu dikalahkan berupa sekumpulan musuh dengan model 3D yang sama dengan musuh sebelumnya.
2. Musuh pada arena bos terbagi menjadi **dua varian**, satu musuh berupa varian berwarna **ungu** seperti normal, dan sisa musuh lainnya yang berwarna **putih**.
3. Pemain tidak dapat menyerang/mengalahkan musuh varian **putih**, pemain hanya bisa menyerang/mengalahkan musuh yang berwarna **ungu**.
4. Ketika pemain berhasil mengalahkan musuh yang berwarna **ungu**, maka jumlah total musuh berkurang satu, posisi seluruh musuh berubah, dan dari musuh yang tersisa akan dipilih salah satunya untuk diubah menjadi musuh varian **ungu**.
5. Pemain mengulang proses mengalahkan musuh berwarna **ungu** hingga jumlah total musuh pada arena mencapai 0.

Ketika seluruh musuh telah dikalahkan, maka akan muncul *gate* yang akan membawa pemain kembali ke scene **Safe Zone**, dimana akan ditampilkan pesan kepada pemain bahwa mereka telah memenangkan permainan.

4.1.6 Pause Menu

Gambar 4.7 Tampilan Pause Menu

Pause Menu dapat diakses dengan menekan tombol **Esc** pada keyboard ketika sedang dalam arena permainan. Selama menu ini terbuka, permainan akan dihentikan sementara. Didalam menu terdapat 3 pilihan tombol berupa tombol **Continue Game**, yang akan menutup menu pause dan melanjutkan alur permainan; tombol **Return To Safe Zone**, yang akan mengembalikan pemain ke scene **Safe Zone**; dan tombol **Exit To Main Menu**, untuk kembali ke scene **Main Menu**.

4.1.7 Stats Menu

Gambar 4.8 Tampilan Stats Menu

Stats Menu dapat diakses dengan menekan tombol **Tab** pada keyboard ketika sedang dalam arena permainan. Selama menu ini terbuka, permainan akan dijeda untuk sementara. Pada menu ini, pemain dapat meningkatkan kekuatan karakter yang sedang dimainkan.

Menu terdiri atas teks yang menunjukan ***Level*** karakter dan jumlah ***Skill Points*** yang dimiliki karakter beserta teks yang menunjukan nilai keempat atribut yang dimiliki karakter. Selain itu juga terdapat sejumlah tombol **"-"** dan **"+"** yang digunakan untuk mengubah nilai atribut, dan tombol ***Confirm*** yang digunakan untuk menyimpan perubahan atribut yang telah.

Dengan menggunakan skill point yang dimiliki, pemain dapat menaikkan salah satu atribut karakter. Atribut-atribut yang dapat dinaikkan adalah : ***Health*** berupa poin yang dikurangi ketika terkena serang musuh; ***Energy*** berupa poin yang digunakan ketika menyerang; ***Power*** berfungsi untuk menaikkan kekuatan serangan karakter; dan ***Defense*** berfungsi untuk mengurangi kekuatan serangan musuh ketika mengenai karakter.

4.2 Pengujian Aplikasi

Pada tahapan ini akan dilakukan pengujian terhadap berbagai fungsionalitas aplikasi untuk menentukan kelayakan aplikasi untuk digunakan oleh pengguna. Hasil dari pengujian yang dilakukan akan menghasilkan kesimpulan yang dapat digunakan untuk pengembangan aplikasi selanjutnya.

4.2.1 Pengujian *Blackbox*

Pengujian blackbox dilakukan untuk menentukan apakah segala input yang diberikan kepada sistem akan menghasilkan sebuah output yang sesuai dengan yang diprediksikan.

A. Pengujian *Blackbox* pada Scene Main Menu

Pengujian *Blackbox* scene Main Menu dilakukan untuk memastikan agar menu utama dapat memindahkan pemain ke scene yang diinginkan.

Tabel 4.3 Pengujian *Blackbox* Scene Main Menu

No.	Kondisi Awal	Hasil Yang Diinginkan	Hasil Keluaran	Hasil Uji
1.	Memasuki Main Menu	Menampilkan Main Menu beserta Pilihannya	Main Menu Telah Ditampilkan	✓
2.	Klik Tombol Test CA_MapGeneration	Menampilkan Scene Pengujian Pembuatan Arena	Halaman Pengujian Pembuatan Arena Ditampilkan	✓
3.	Klik Tombol Continue	Memulai permainan dengan data karakter yang tersimpan sebelumnya	Pemain memasuki permainan, data karakter berhasil dimuat	✓
4.	Klik Tombol New Game	Memulai permainan dengan data karakter baru	Pemain memasuki permainan, data karakter baru dibuat	✓

5.	Klik Tombol Exit	Keluar dari aplikasi	Jendela aplikasi ditutup	✓

B. Pengujian *Blackbox* pada Scene Test CA_MapGeneration

Pengujian *Blackbox* scene Test CA_MapGeneration dilakukan untuk memastikan agar pembuatan arena sesuai dengan parameter yang telah ditentukan oleh pemain.

Tabel 4.4 Pengujian *Blackbox* Scene Test CA_MapGeneration

No.	Kondisi Awal	Hasil Yang Diinginkan	Hasil Keluaran	Hasil Uji
1.	Memasuki Scene Pengujian	Menampilkan Scene Testing beserta antarmuka parameter	Scene Testing berserta antarmuka parameter berhasil ditampilkan	✓
2.	Pemain mengubah nilai-nilai parameter	Nilai dimasukan pemain ditampilkan pada antarmuka parameter	Nilai yang diinput ditampilkan pada antarmuka parameter	✓
3.	Pemain menekan tombol “F1” pada keyboard	Mengenerasikan arena labirin berdasarkan parameter yang ada	Arena berhasil digenerasikan berdasarkan nilai pada antarmuka parameter	✓
4.	Pemain menekan tombol W,A,S,D	Tampilan kamera bergeser sesuai dengan arah tombol yang ditekan	Kamera berhasil digerakan sesuai tombol yang ditekan	✓

5.	Pemain menggerakan *scroll wheel* pada mouse	Tampilan kamera mendekat atau menjauh sesuai arah geseran *scroll wheel*	Tampilan kamera berhasil mendekat/menjauh sesuai arah gerak *scroll wheel*	✓
6.	Pemain menekan tombol "F2" pada keyboard	Antarmuka parameter akan ditampilkan jika sedang tersembunyi, dan disembunyikan jika sedang ditampilkan	Berhasil menyembunyikan dan menampilkan antaruka parameter	✓
7.	Pemain menekan tombol "Esc" pada keyboard	Keluar dari scene testing, dan kembali ke scene main menu	Berhasil kembali ke Main Menu	✓

C. Pengujian *Blackbox* pada kontrol karakter

Pengujian *Blackbox* ini dilakukan untuk memastikan kontrol terhadap karakter pemain sesuai dengan input pemain.

Tabel 4.5 Pengujian *Blackbox* Kontrol Karakter

No.	Kondisi Awal	Hasil Yang Diinginkan	Hasil Keluaran	Hasil Uji
1.	Pemain menekan tombol W,A,S,D pada keyboard	Karakter pemain bergerak ke arah yang sesuai tombol yang ditekan	Karakter berhasil digerakan sesuai dengan tombol yang ditekan	✓
2.	Pemain menekan tombol "Space" dan menekan salah satu tombol W,A,S,D	Karakter pemain akan berlari ke arah yang sesuai tombol yang ditekan	Karakter pemain berhasil berlari ke arah yang sesuai	✓

3.	Pemain menekan tombol “Space” sekaligus menekan salah satu tombol W,A,S,D	Karakter pemain akan berlari ke arah yang sesuai tombol yang ditekan	Karakter pemain berhasil berlari ke arah yang sesuai	✓
4.	Pemain mengklik tombol kiri pada mouse	Karakter pemain menyerang	Karakter pemain berhasil menyerang	✓

D. Pengujian *Blackbox* pada Scene Safe Zone

Pengujian *Blackbox* scene Safe Zone dilakukan untuk memastikan pemain dapat memasuki arena permainan sesuai dengan opsi yang dipilih pada Main Menu.

Tabel 4.6 Pengujian *Blackbox* Scene Safe Zone

No.	Kondisi Awal	Hasil Yang Diinginkan	Hasil Keluaran	Hasil Uji
1.	Memasuki Scene Safe Zone	Menampilkan permainan beserta interface overlay	Permainan dan interface overlay berhasil ditampilkan	✓
2.	Karakter pemain menginjak *gate* yang berada pada mulut gua	Pemain berpindah ke scene Cave1	Pemain berpindah ke scene Cave1	✓

E. Pengujian *Blackbox* pada Scene Cave1, Cave2, dan Cave3

Pengujian *Blackbox* scene Cave1, Cave2, dan Cave3 dilakukan untuk memastikan agar pemain dapat melalui keseluruhan arena labirin tanpa menemui permasalahan.

Tabel 4.7 Pengujian *Blackbox* Scene Cave1, 2, dan 3

No.	Kondisi Awal	Hasil Yang Diinginkan	Hasil Keluaran	Hasil Uji
1.	Pemain memasuki scene arena labirin	Mengenerate arena beserta musuhnya, kemudian menempatkan karakter pemain pada arena tersebut	Pemain berhasil memasuki arena, dan musuh berhasil diletakan pada arena	✓
2.	Pemain menyerang musuh	*Health* musuh berkurang	*Health* musuh terlihat berkurang	✓
3.	*Health* musuh kosong	Musuh dihapus dari arena permainan	Musuh dikalahkan, dan dihilangkan dari arena	✓
4.	Pemain mengalahkan seluruh musuh pada arena	Muncul *gate* dengan lokasi acak pada labirin	*Gate* berhasil muncul	✓
5.	Karakter pemain menginjak gate yang muncul	Pemain dipindahkan ke arena labirin tingkat selanjutnya	Pemain berpindah ke arena labirin tingkan selanjutnya	✓

F. Pengujian *Blackbox* pada Scene CaveBoss

Pengujian *Blackbox* pada scene CaveBoss memastikan agar hasil yang didapatkan ketika pemain berhasil mengalahkan bos adalah sesuai dengan yang diharapkan.

Tabel 4.8 Pengujian *Blackbox* Scene CaveBoss

No.	Kondisi Awal	Hasil Yang Diinginkan	Hasil Keluaran	Hasil Uji
1.	Pemain memasuki scene arena labirin	Mengenerate arena beserta musuhnya, kemudian menempatkan karakter pemain pada arena tersebut	Pemain berhasil memasuki arena, dan musuh berhasil diletakan pada arena	✓
2.	Pemain menyerang musuh berwarna putih	*Health* musuh tidak berkurang	*Health* musuh terlihat tidak berkurang	✓
3.	Pemain menyerang musuh berwarna ungu	*Health* musuh berkurang	*Health* musuh terlihat berkurang	✓
4.	*Health* musuh kosong	Musuh dihapus dari arena permainan	Musuh dikalahkan, dan dihilangkan dari arena	✓
5.	Pemain mengalahkan seluruh musuh pada arena	Muncul *gate* dengan lokasi acak pada labirin	*Gate* berhasil muncul	✓
6.	Karakter pemain menginjak gate yang muncul	Pemain dipindahkan ke scene Safe Zone, menampilkan pesan bahwa pemain berhasil memenangkan game	Pemain berpindah ke arena labirin tingkan selanjutnya	✓

G. Pengujian *Blackbox* pada Menu Pause

Pengujian *Blackbox* pada bagian Menu Pause dilakukan untuk memastikan agar pilihan pada menu pause dapat memberikan hasil yang sesuai dengan yang diinginkan.

Tabel 4.9 Pengujian *Blackbox* Menu Pause

No.	Kondisi Awal	Hasil Yang Diinginkan	Hasil Keluaran	Hasil Uji
1.	Pemain menekan tombol "Esc" pada keyboard ketika sedang dalam arena permainan	Menampilkan menu pause dan menjeda permainan	Permainan berhasil dijeda dan menu pause ditampilkan	✓
2.	Pemain memilih tombol Continue Game	Menu pause ditutup dan permainan dilanjutkan	Menu pause berhasil ditutup, dan permainan berlanjut	✓
3.	Pemain memilih tombol Return To Safe Zone	Pemain dikembalikan ke scene Safe Zone	Pemain kembali ke Safe Zone	✓
4.	Pemain memilih tombol Exit To Main Menu	Pemain dikembalikan ke scene Main Menu	Pemain kembali ke Main Menu	✓

H. Pengujian *Blackbox* pada Menu Stats

Pengujian *Blackbox* Menu Stats dilakukan untuk memastikan agar atribut karakter dapat dinaikkan sesuai dengan yang diinginkan. Berikut adalah hasil yang didapat dari pengujian.

Tabel 4.10 Pengujian *Blackbox* Menu Stats

No.	Kondisi Awal	Hasil Yang Diinginkan	Hasil Keluaran	Hasil Uji
1.	Pemain menekan tombol “Tab” pada keyboard ketika sedang dalam arena permainan	Menampilkan Menu Stats dan menjeda permainan	Berhasil menjeda permainan dan menampilkan menu stats	✓
2.	Pemain menekan tombol “+“ disebelah salah satu atribut	Menambah nilai atribut yang dipilih, mengurangi nilai *skill point* yang tersedia	Nilai atribut meningkat , *skill point* yang tersedia berkurang	✓
3.	Pemain menekan tombol “-“ disebelah salah satu atribut	Mengurangi nilai atribut yang dipilih, menambah nilai skill point yang tersedia	Nilai atribut menurun , *skill point* yang tersedia bertambah	✓
4.	Pemain menekan tombol Confirm	Perubahan pada atribut diterapkan, menu stats ditutup	Perubahan atribut berhasil tersimpan dan menu stats ditutup	✓
5.	Pemain menutup menu stats tanpa menekan confirm	Perubahan pada atribut dibatalkan, menu stats ditutup	Perubahan atribut dibatalkan dan menu stats ditutup	✓

4.2.2 Pengujian Parameter Arena

Pada tahapan ini, akan dilakukan pengujian parameter dalam pembuatan arena permainan. Data pengujian yang didapat kemudian dapat digunakan dalam membentuk suatu kesimpulan

A. Pengujian Bentuk Arena

Pengujian bentuk arena dilakukan untuk menguji nilai parameter yang optimal untuk meraih bentuk arena yang menyerupai terowongan gua. Parameter yang akan digunakan dalam pengujian berupa ***Fill Percent*** dikarenakan parameter terkait telah diamati memiliki pengaruh yang signifikan dalam hasil akhir bentuk arena.

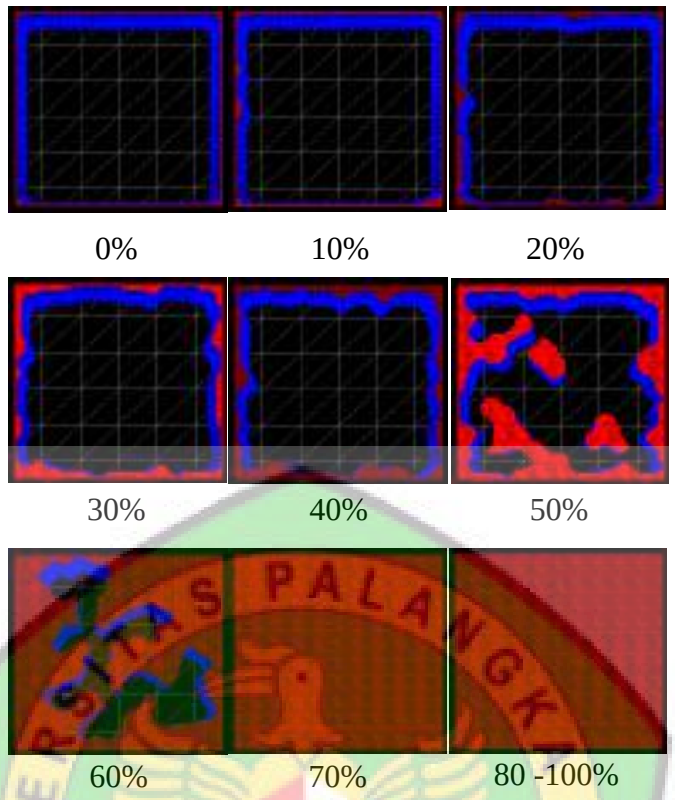
Parameter selain parameter ***Fill Percent*** akan bersifat sebagai variabel kontrol, sehingga akan diisikan dengan nilai default. Selain dari nilai parameter ***Fill Percent***, nilai parameter lainnya tidak akan berubah dalam bagian pengujian ini. Berikut adalah daftar parameter beserta nilai *default*-nya.

Tabel 4.11 Nilai Default Parameter Pengujian Bentuk Arena

Width	50
Height	50
Depth	10
Square Scaling	1
Border Size	0
Fill Percent	Variabel Uji
Smooth Iteration	6
Room Culling	40
Wall Culling	40
Segment Width	10
Valid Segment	70
Max Spawn Point	5
Min Enemy	1
Max Enemy	1

Pengujian dilakukan dengan memasukan nilai 0 – 100 kemudian mengamati bentuk arena yang didapat. Pengujian akan dilakukan sebanyak 9 kali dimana pada setiap pengujian nilai parameter ***Fill Percent*** akan dinaikan sebesar 10.

Setelah dilakukannya pengujian, berikut adalah beberapa hasil bentuk yang dipilih untuk setiap nilai ***Fill Percent*** yang berbeda.



Gambar 4.9 Bentuk Arena dengan Nilai Fill Percent yang Berbeda

Berdasarkan hasil yang didapat, terlihat bahwa nilai 0 – 40 hanya menghasilkan sebuah ruang terbuka dengan empat dinding yang bergerigi. Ini diakibatkan oleh nilai sel hidup (nilai 1) yang sangat sedikit, sehingga sebagian besar berubah menjadi sel mati (nilai 0) atau ruang terbuka.

Sebaliknya pada nilai 50 – 60 dapat terlihat bentuk arena lebih menyerupai terowongan gua, dengan nilai 50 bersifat lebih terbuka dan nilai 60 bersifat lebih sempit dan menjalar.

Untuk nilai 70 – 100 tidak terdapat sama sekali ruang terbuka. Ini dikarenakan oleh jumlah sel hidup yang terlalu banyak sehingga setelah dilakukannya iterasi, seluruh sel mati yang ada berubah menjadi sel hidup atau dinding.

Berdasarkan hasil pengujian diatas, dapat disimpulkan bahwa nilai optimal parameter ***Fill Percent*** untuk membuat arena yang menyerupai gua adalah diantara 50 dan 60, dimana semakin besar nilai parameter, maka terowongan yang ada pada arena semakin menyempit.

B. Pengujian Iterasi Cellular Automata

Pengujian Iterasi dilakukan untuk menentukan jumlah iterasi yang optimal dalam pembuatan arena. Pengujian dilakukan sebanyak 30 kali dimana data yang dikumpulkan berupa jumlah iterasi beserta dengan jumlah sel yang berubah dalam setiap iterasi. Berikut adalah nilai parameter yang digunakan.

Tabel 4.12 Nilai Parameter Pengujian Iterasi

Width	150
Height	150
Depth	10
Square Scaling	1
Border Size	0
Fill Percent	55
Smooth Iteration	99
Room Culling	40
Wall Culling	40
Segment Width	10
Valid Segment	70
Max Spawn Point	5
Min Enemy	1
Max Enemy	1

Setelah dilakukan pengujian, berikut adalah data jumlah iterasi beserta daftar jumlah sel berubah yang didapat.

Tabel 4.13 Hasil Pengujian Iterasi

No	Iterations	Number of Cell Changes Every Iteration
1	17	7859|4675|707|300|166|87|56|33|25|20|15|9|5|5|3|1|0|
2	13	7815|4445|686|336|170|77|43|26|8|3|1|1|0|
3	18	7947|4611|739|320|165|82|42|31|14|7|5|5|3|3|3|13|3|0|
4	23	7765|4492|710|384|225|103|57|29|18|15|13|7|2|1|2|3|2|1|2|3|3|3|0|
5	19	7837|4482|744|361|194|89|49|30|20|16|7|4|2|2|2|2|2|1|0|
6	16	7871|4722|707|367|169|88|57|28|15|10|8|4|4|5|2|0|
7	18	7842|4554|656|351|203|88|62|42|41|23|20|11|9|5|2|5|1|0|
8	16	7841|4639|777|324|186|109|43|29|11|12|6|3|2|1|1|0|
9	20	7774|4497|703|312|194|88|66|32|23|17|14|6|7|6|7|5|4|2|1|0

10	17	7901|4629|767|386|170|113|50|40|25|20|14|12|8|9|4|1|0|
11	32	7755|4533|717|308|149|90|42|29|19|13|5|4|4|3|3|1|2|1|1|1|2|1|1|2|1|1|2|1|2|1|1|0|
12	15	7868|4664|697|339|176|90|43|18|14|17|7|8|3|2|0|
13	14	7858|4728|747|375|181|95|44|19|19|9|5|3|2|0|
14	14	7869|4586|743|331|181|92|43|31|20|18|8|5|2|0|
15	15	7789|4438|667|368|184|116|73|44|21|14|8|6|1|1|0|
16	16	7861|4601|678|307|127|73|43|33|21|17|14|11|5|6|6|0|
17	15	7867|4572|697|357|205|125|58|47|24|11|6|7|6|2|0|
18	13	7908|4547|697|277|173|81|55|30|24|12|12|2|0|
19	15	7899|4609|717|357|147|67|27|19|12|8|6|4|2|1|0|
20	17	7773|4506|716|338|154|79|46|30|19|12|6|4|4|3|1|2|0|
21	15	7799|4615|756|365|165|78|23|12|7|7|9|4|2|1|0|
22	17	7858|4660|714|336|187|70|47|26|19|14|12|4|6|3|2|2|0|
23	23	7840|4627|751|364|167|81|50|30|14|8|5|7|6|8|4|2|4|2|2|2|1|1|0|
24	13	7836|4490|704|400|205|91|53|36|15|13|13|2|0|
25	15	7747|4438|729|389|212|119|75|48|22|13|13|5|3|2|0|
26	21	7871|4564|743|334|181|99|53|45|22|14|10|9|4|4|1|1|2|1|1|2|0|
27	17	7923|4578|635|294|160|65|39|20|14|13|6|4|5|5|3|3|0|
28	16	7824|4634|725|363|175|82|64|40|21|5|2|2|1|1|1|0|
29	18	7867|4557|699|358|154|81|51|42|17|17|11|9|6|5|3|1|1|0|
30	15	7806|4457|696|363|210|103|54|30|19|7|4|2|1|1|0|

Dapat dilihat pada kolom ***Iterations*** tercantum jumlah total iterasi yang dilakukan hingga matriks mencapai kondisi stabil, sementara pada kolom paling kanan tercantumkan jumlah sel yang berubah pada setiap iterasi. Semakin kecil jumlah sel yang berubah pada suatu iterasi, maka bentuk matriks yang dihasilkan semakin mendekati bentuk matriks yang sebenarnya, yaitu sebuah matriks dimana jumlah sel yang berubah berupa 0 (kondisi stabil).

Pada jumlah sel yang berubah juga dapat diamati bahwa pada setiap iterasi, jumlah sel yang berubah selalu menurun, sehingga semakin banyak iterasi yang dilakukan, maka perubahan yang terjadi pada keseluruhan arena semakin berkurang dan tidak signifikan. Hal ini menyebabkan meningkatnya waktu dalam pembuatan arena, dikarenakan untuk setiap iterasi dilakukan pengecekan setiap sel yang ada pada arena.

Untuk itu perlu diambil sebuah nilai ambang yang berfungsi untuk menentukan pada jumlah iterasi ke berapa bentuk arena sudah dianggap cukup menyerupai bentuk akhir yang sebenarnya.

Pada pengujian ini, akan diggunakan batas ambang berupa **5%** dari total sel yang ada, yang dianggap cukup kecil sehingga menghasilkan matriks akhir yang cukup mendekati hasil akhir matriks sebenarnya. Jika jumlah sel yang berubah dalam satu iterasi **< 5%** dari total jumlah sel yang ada pada matriks, maka pada pemrosesan matriks akan dihentikan pada iterasi tersebut.

Berdasarkan parameter ***Height*** dan ***Width*** yang telah ditentukan dalam pengujian, maka didapat total sel yang ada pada matriks :

150 x 150 = 22.500

Sementara batas ambang yang didapat berupa **5%** dari jumlah total sel yang dimiliki oleh arena yang dibuat:

5/100 * 22.500 = 1.125

Jika dilihat dari tabel 4.12, seluruh hasil uji mencapai jumlah sel berubah < 1.125 pada iterasi yang ketiga. Sehingga dapat direkomendasikan jumlah maksimum iterasi yang diberikan berupa 3, dimana nilai ini dapat dinaikkan untuk mendapatkan bentuk arena yang lebih akurat.

C. Pengujian *Seed*

Pengujian *seed* dilakukan untuk memastikan nilai *seed* yang sama ketika diinputkan ke dalam proses pembuatan arena akan menghasilkan bentuk arena yang sama. Pengujian dilakukan dengan cara meng-generasikan nilai *seed* secara acak sebanyak tiga kali dan mengamati bentuk arena yang dibuat, kemudian setelah aplikasi ditutup dan dibuka ulang, akan dimasukan nilai yang sama dalam pembuatan arena dan mengamati apakah bentuk arena yang dihasilkan sama dengan yang sebelumnya.

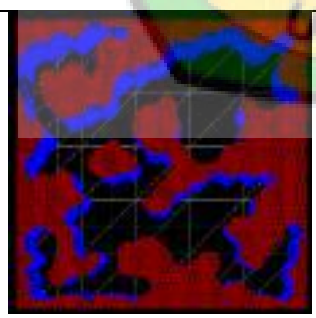
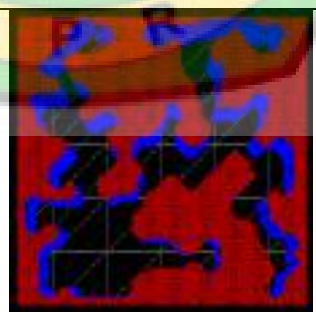
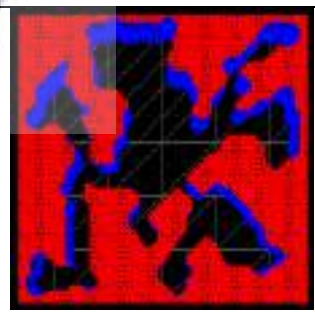
Berikut adalah nilai-nilai default parameter yang akan digunakan dalam pengujian.

Tabel 4.14 Nilai Parameter Pengujian Seed

Width	60
Height	60
Depth	10
Square Scaling	1
Border Size	0
Fill Percent	55
Smooth Iteration	6
Room Culling	40
Wall Culling	40
Segment Width	-
Valid Segment	-
Max Spawn Point	-
Min Enemy	-
Max Enemy	-

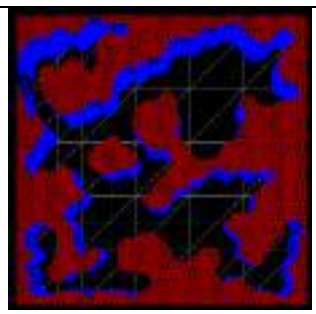
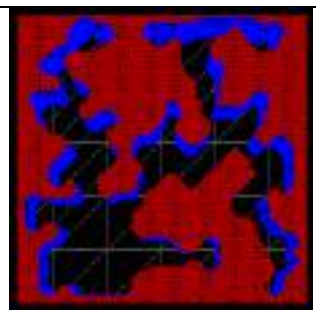
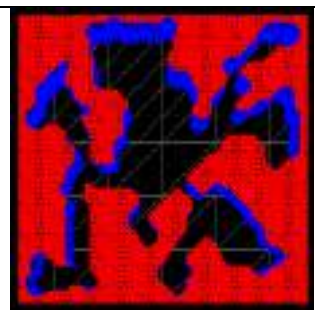
Setelah dilakukan pengujian, berikut adalah nilai ketiga seed beserta gambar bentuk arena yang didapat dari masing-masing seed.

Tabel 4.15 Nilai Seed Acak dan Arena yang Dihasilkan

Seed	ja4tdhj40t	ycg9kzh784	d4th68ntxe
Arena			

Setelah seed acak telah didapatkan, aplikasi akan ditutup kemudian dibuka ulang. Kemudian akan dibuat arena baru menggunakan seed yang telah didapat. Berikut adalah hasil dari pengujian yang dilakukan

Tabel 4.16 Nilai Seed yang Diinput dan Arena yang Dihasilkan

Seed	ja4tdhj40t	ycg9kzh784	d4th68ntxe
Arena			

Terlihat bahwa bentuk arena yang didapat berupa identik dengan bentuk yang sebelumnya. Sehingga dapat disimpulkan bahwa arena yang dibuat dengan menggunakan seed yang sama akan selalu menghasilkan bentuk yang identik.

D. Pengujian Waktu Eksekusi

Pengujian waktu eksekusi dilakukan untuk menentukan rata-rata waktu yang diperlukan untuk menghasilkan sebuah arena sekaligus menentukan pengaruh parameter terhadap waktu eksekusi pembuatan arena.

Pada pengujian waktu eksekusi, pengujian akan dibagikan menjadi beberapa kelompok parameter yang akan diuji secara satu persatu. Dalam pengujian suatu parameter, parameter yang diuji akan diberikan tiga variasi nilai yang berbeda, sementara parameter lainnya akan diberikan nilai *default*.

Berikut adalah nilai *default* untuk setiap parameter dalam pengujian kecepatan eksekusi ini. Kecuali secara eksplisit dikatakan berubah, maka nilai yang tercantumkan berupa nilai yang digunakan untuk keseluruhan bagian pengujian waktu eksekusi.

Tabel 4.17 Nilai Default Parameter Pengujian Waktu Eksekusi

Width	60
Height	60
Depth	10
Square Scaling	1
Border Size	0
Fill Percent	55
Seed	Random
Smooth Iteration	6
Room Culling	40
Wall Culling	40
Segment Width	10
Valid Segment	70
Max Spawn Point	5
Min Enemy	1
Max Enemy	1

Sampel pengujian akan dilakukan sebanyak 10 kali untuk setiap arena. Pada setiap iterasi, waktu yang diperlukan untuk membentuk arena secara keseluruhan akan dibagikan menjadi 3 waktu terpisah, yang berupa:

1. Initial Processing Time

   Waktu yang diperlukan untuk memproses matriks 2D acak menjadi matriks akhir yang akan digunakan untuk membentuk arena permainan.
2. Mesh Generation Time

   Waktu yang diperlukan untuk membentuk mesh atau objek 3D dari matriks 2D yang didapat.
3. Enemy Placement Time

   Waktu yang diperlukan untuk menentukan wilayah penempatan musuh pada arena permainan.

Bagian pertama yang akan diuji adalah bagaimana pengaruh luas ukuran arena permainan dalam kecepatan pembuatannya. Parameter yang akan diuji berupa ***Height***, ***Width*** dan ***Depth***. Berikut adalah perubahan nilai parameter pada ketiga arena yang diuji.

Tabel 4.18 Perbedaan Parameter Height,Width dan Depth Ketiga Arena

	Height	Width	Depth
Arena1	50	50	10
Arena2	100	100	20
Arena3	150	150	30

Dari tabel tersebut dapat dilihat perbedaan ukuran ketiga arena yang akan dibuat, dimana Arena2 berupa kelipatan dua kali dari ukuran Arena1, sementara Arena3 memiliki ukuran berupa kelipatan tiga kali dari ukuran Arena1.

Berikut adalah tabel hasil yang didapatkan setelah dilakukannya pengujian.

Tabel 4.19 Hasil Pengujian Arena1 (Height,Width,Depth)

No.	Initial Processing	Mesh Generation	Enemy Placement	Total Time
1	0,0041	0,25487	0,03406	0,29303
2	0,00576	0,34238	0,02029	0,36843
3	0,00517	0,36501	0,01442	0,3846
4	0,00605	0,34558	0,01771	0,36934
5	0,00463	0,31181	0,02037	0,33681
6	0,00585	0,34261	0,01543	0,36389
7	0,00532	0,34707	0,01612	0,36851
8	0,00474	0,29082	0,01888	0,31444
9	0,00373	0,29355	0,01542	0,3127
10	0,00475	0,283	0,01611	0,30386
AVG	**0,00501s**	**0,31767s**	**0,018881s**	**0,341561s**

Tabel 4.20 Hasil Pengujian Arena2 (Height,Width,Depth)

No.	Initial Processing	Mesh Generation	Enemy Placement	Total Time
1	0,04517	4,29428	0,0766	4,41605
2	0,03955	4,0936	0,06427	4,19742
3	0,04593	4,09403	0,07666	4,21662
4	0,01324	4,01953	0,06433	4,0971
5	0,03125	3,8458	0,06705	3,9441
6	0,04974	4,60907	0,08154	4,74035
7	0,0575	3,8558	0,06686	3,98016
8	0,02216	4,03922	0,05151	4,11289
9	0,02576	3,71597	0,05099	3,79272
10	0,03601	3,73489	0,05322	3,82412
AVG	**0,036631s**	**4,030219s**	**0,065303s**	**4,132153s**

Tabel 4.21 Hasil Pengujian Arena3 (Height,Width,Depth)

No.	Initial Processing	Mesh Generation	Enemy Placement	Total Time
1	0,20477	19,16846	0,20081	19,57404
2	0,22141	18,92346	0,1066	19,25147
3	0,0708	17,45813	0,11038	17,63931
4	0,04831	16,63312	0,09656	16,77799
5	0,03693	15,73334	0,11041	15,88068
6	0,11371	18,12024	0,11023	18,34418
7	0,0564	16,67743	0,11438	16,84821
8	0,26239	17,64673	0,12122	18,03034
9	0,17255	17,25708	0,10309	17,53272
10	0,10822	17,34863	0,10278	17,55963
AVG	**0,129549s**	**17,496662s**	**0,117646s**	**17,743857s**

Tabel 4.22 Rata-Rata Waktu Pembuatan Ketiga Arena (Height,Width,Depth)

	Initial Processing	Mesh Generation	Enemy Placement	Total Time
Arena1	0,005s	1,32s	0,053s	1,376s
Arena2	0,034s	11,43s	0,146s	11,613s
Arena3	0,092s	33,67s	0,229s	33,994s

Berdasarkan hasil pengujian dapat disimpulkan bahwa parameter ***Height***, ***Width***, dan ***Depth*** memiliki pengaruh yang signifikan dalam waktu pembuatan arena permainan. Pada dua kali lipat luas Arena1 terjadi peningkatan waktu sebesar **843%**, sementara pada tiga kali lipat luas Arena1 terjadi peningkatan waktu sebesar **2.470%.**

Pengujian selanjutnya dilakukan terhadap parameter ***Border Size***. Berikut adalah perubahan nilai parameter ***Border Size*** pada ketiga arena yang akan diuji.

Tabel 4.23 Perbedaan Parameter Ketiga Arena (Border Size)

	Border Size
Arena1	10
Arena2	20
Arena3	30

Setelah melakukan pengujian sebanyak 10 kali pada ketiga arena, berikut adalah hasil yang didapatkan.

Tabel 4.24 Hasil Pengujian Arena1 (Border Size)

	Initial Processing	Mesh Generation	Enemy Placement	Total Time
1	0,0041	0,25487	0,03406	0,29303
2	0,00576	0,34238	0,02029	0,36843
3	0,00517	0,36501	0,01442	0,3846
4	0,00605	0,34558	0,01771	0,36934
5	0,00463	0,31181	0,02037	0,33681
6	0,00585	0,34261	0,01543	0,36389
7	0,00532	0,34707	0,01612	0,36851
8	0,00474	0,29082	0,01888	0,31444
9	0,00373	0,29355	0,01542	0,3127
10	0,00475	0,283	0,01611	0,30386
AVG	**0,00501**	**0,31767**	**0,018881**	**0,341561**

Tabel 4.25 Hasil Pengujian Arena2 (Border Size)

	Initial Processing	Mesh Generation	Enemy Placement	Total Time
1	0,00513	1,40698	0,04907	1,46118
2	0,00659	1,4375	0,0415	1,48559
3	0,00635	1,39795	0,02783	1,43213
4	0,00586	1,3562	0,04102	1,40308
5	0,00586	1,39795	0,02612	1,42993
6	0,00488	1,34839	0,02393	1,3772
7	0,00415	1,50342	0,05713	1,5647
8	0,00562	1,69238	0,02881	1,72681
9	0,00586	1,86353	0,04395	1,91334
10	0,00659	1,59717	0,04761	1,65137
AVG	**0,005689**	**1,500147**	**0,038697**	**1,544533**

Tabel 4.26 Hasil Pengujian Arena3 (Border Size)

	Initial Processing	Mesh Generation	Enemy Placement	Total Time
1	0,01416	5,72998	0,11084	5,85498
2	0,00879	5,47217	0,08105	5,56201
3	0,02271	5,3772	0,08276	5,48267
4	0,00732	4,69824	0,06763	4,77319
5	0,00806	4,82446	0,06128	4,8938
6	0,00439	4,47192	0,07031	4,54662
7	0,00732	4,58887	0,05688	4,65307
8	0,00562	4,56836	0,12329	4,69727
9	0,00903	4,41016	0,05078	4,46997
10	0,0061	4,32031	0,04712	4,37353
AVG	**0,00935**	**4,846167**	**0,075194**	**4,930711**

Tabel 4.27 Rata-Rata Waktu Pembuatan Ketiga Arena (Border Size)

	Initial Processing	Mesh Generation	Enemy Placement	Total Time
Arena1	0,00501	0,31767	0,018881	**0,341561s**
Arena2	0,005689	1,500147	0,038697	**1,544533s**
Arena3	0,00935	4,846167	0,075194	**4,930711s**

Berdasarkan data yang didapat, dapat disimpulkan bahwa parameter mempunyai dampak yang signifikan juga dalam waktu pembuatan arena. Pada Arena2 dengan nilai parameter dua kali lipat dari Arena1 terdapat peningkatan sebesar **341%**, sementara pada Arena3 yang memiliki nilai parameter tiga kali lipat dari Arena1 memiliki peningkatan waktu sebesar **1.341%**.

Parameter selanjutnya yang akan diuji berupa ***Square Scaling***. Berikut adalah perubahan nilai parameter untuk ketiga arena yang akan dibuat.

Tabel 4.28 Perbedaan Parameter Ketiga Arena (Square Scaling)

	Square Scaling
Arena1	1
Arena2	2
Arena3	3

Berikut adalah hasil yang didapat dari pengujian arena dengan parameter ***Square Scaling*** yang berubah.

Tabel 4.29 Hasil Pengujian Arena1 (Square Scaling)

	Initial Processing	Mesh Generation	Enemy Placement	Total Time
1	0,00537	0,35059	0,03906	0,39502
2	0,01367	0,34277	0,01904	0,37548
3	0,00635	0,34131	0,01611	0,36377
4	0,00488	0,32471	0,01904	0,34863
5	0,00537	0,27588	0,01855	0,2998
6	0,00391	0,29492	0,02686	0,32569
7	0,00537	0,31543	0,01172	0,33252
8	0,00391	0,33105	0,01855	0,35351
9	0,00391	0,27637	0,021	0,30128
10	0,00342	0,34912	0,01367	0,36621
AVG	**0,005616**	**0,320215**	**0,02036**	**0,346191**

Tabel 4.30 Hasil Pengujian Arena2 (Square Scaling)

	Initial Processing	Mesh Generation	Enemy Placement	Total Time
1	0,00537	0,3042	0,06592	0,37549
2	0,00439	0,30859	0,06152	0,3745
3	0,0083	0,35938	0,04541	0,41309
4	0,00586	0,3335	0,05176	0,39112
5	0,00391	0,27002	0,07764	0,35157
6	0,00684	0,29883	0,05029	0,35596
7	0,00488	0,31836	0,04492	0,36816
8	0,00488	0,32324	0,05811	0,38623
9	0,00879	0,35107	0,05176	0,41162
10	0,00488	0,35938	0,05469	0,41895
AVG	**0,00581**	**0,322657**	**0,056202**	**0,384669**

Tabel 4.31 Hasil Pengujian Arena3 (Square Scaling)

	Initial Processing	Mesh Generation	Enemy Placement	Total Time
1	0,00488	0,35059	0,11719	0,47266
2	0,00732	0,34375	0,1001	0,45117
3	0,00488	0,34961	0,08496	0,43945
4	0,00732	0,33984	0,08447	0,43163
5	0,00781	0,25488	0,09912	0,36181
6	0,01807	0,30127	0,09619	0,41553
7	0,00684	0,29541	0,104	0,40625
8	0,01563	0,36914	0,08789	0,47266
9	0,00537	0,2959	0,10645	0,40772
10	0,0083	0,3291	0,09863	0,43603
AVG	**0,008642**	**0,322949**	**0,0979**	**0,429491**

Tabel 4.32 Rata-Rata Waktu Pembuatan Ketiga Arena (Border Size)

	Initial Processing	Mesh Generation	Enemy Placement
Arena1	0,005616	0,320215	0,02036
Arena2	0,00581	0,322657	0,056202
Arena3	0,008642	0,322949	0,0979

Berdasarkan data yang didapat, terlihat bahwa parameter Square Scaling tetap memiliki dampak kecil terhadap waktu pembuatan arena. Dari perbandingan Arena1 dan Arena2 terjadi peningkatan waktu sebanyak **11,7%**, sementara perbandingan Arena1 terhadap Arena3 terjadi peningkatan waktu sebesar **26,5%**.

Parameter selanjutnya berupa ***Room Culling Threshold*** dan ***Wall Culling Threshold***. Berikut adalah ketiga nilai variabel yang akan digunakan dalam pengujian.

Tabel 4.33 Perbedaan Parameter Ketiga Arena (Wall/Room Culling)

	Wall Culling Threshold	Room Culling Threshold
Arena1	30	30
Arena2	60	60
Arena3	90	90

Berikut adalah hasil yang didapatkan dari pengujian ketiga arena yang dibuat.

Tabel 4.34 Hasil Pengujian Arena1 (Wall/Room Culling)

	Initial Processing	Mesh Generation	Enemy Placement	Total Time
1	0,02368	0,41724	0,03345	0,47437
2	0,01733	0,47534	0,01611	0,50878
3	0,01611	0,54932	0,02344	0,58887
4	0,01147	0,55933	0,01733	0,58813
5	0,01001	0,49707	0,01831	0,52539
6	0,0105	0,50635	0,01343	0,53028
7	0,00806	0,4729	0,02441	0,50537
8	0,0083	0,41333	0,01855	0,44018
9	0,01123	0,58691	0,0249	0,62304
10	0,0105	0,55884	0,01758	0,58692
AVG	**0,012719**	**0,503663**	**0,020751**	**0,537133**

Tabel 4.35 Hasil Pengujian Arena2 (Wall/Room Culling)

	Initial Processing	Mesh Generation	Enemy Placement	Total Time
1	0,02539	0,51953	0,03662	0,58154
2	0,01074	0,56079	0,02734	0,59887
3	0,01318	0,60229	0,02002	0,63549
4	0,00977	0,5625	0,01855	0,59082
5	0,01709	0,51245	0,02905	0,55859
6	0,00928	0,47559	0,02637	0,51124
7	0,00928	0,53418	0,0271	0,57056
8	0,01001	0,46826	0,02075	0,49902
9	0,00879	0,57007	0,02588	0,60474
10	0,00977	0,40576	0,0271	0,44263
AVG	**0,01233**	**0,521142**	**0,025878**	**0,55935**

Tabel 4.36 Hasil Pengujian Arena3 (Wall/Room Culling)

	Initial Processing	Mesh Generation	Enemy Placement	Total Time
1	0,0144	0,4458	0,02808	0,48828
2	0,00854	0,50146	0,02661	0,53661
3	0,01025	0,61816	0,0249	0,65331
4	0,0083	0,53223	0,02246	0,56299
5	0,00903	0,46094	0,01904	0,48901
6	0,00854	0,45898	0,02515	0,49267
7	0,00879	0,55029	0,02295	0,58203
8	0,00757	0,52295	0,01807	0,54859
9	0,00659	0,51367	0,02588	0,54614
10	0,03027	0,56396	0,0249	0,61913
AVG	**0,011228**	**0,51684**	**0,0238**	**0,551876**

Tabel 4.37 Rata-Rata Waktu Pembuatan Ketiga Arena (Wall/Room Culling)

	Initial Processing	Mesh Generation	Enemy Placement	Total
Arena1	0,012719	0,503663	0,020751	0,537133
Arena2	0,01233	0,521142	0,025878	0,55935
Arena3	0,011228	0,516844	0,023804	0,551876

Parameter ***Wall Culling Threshold*** dan ***Room Culling Threshold*** memiliki pengaruh yang minimal terhadap waktu pebuatan arena. Perbandingan waktu Arena1 dan Arena2 terdapat kenaikan **4,1%**, sementara antara Arena1 dan Arena3 terdapat kenaikan **2,7%** waktu eksekusi.

Selanjutnya adalah pengujian parameter ***Segment Width***. Berikut adalah perubahan nilai parameter yang akan digunakan dalam pengujian.

Tabel 4.38 Perbedaan Parameter Ketiga Arena (Segment Width)

	Segment Width
Arena1	10
Arena2	20
Arena3	30

Setelah dilakukan pengujian, berikut adalah hasil yang telah didapatkan.

Tabel 4.39 Hasil Pengujian Arena1 (Segment Width)

	Initial Processing	Mesh Generation	Enemy Placement	Total Time
1	0,00766	0,60809	0,04288	0,65863
2	0,00552	0,50671	0,03046	0,54269
3	0,00876	0,64255	0,02838	0,67969
4	0,00781	0,59872	0,0274	0,63393
5	0,00827	0,58762	0,03024	0,62613
6	0,00952	0,55737	0,01904	0,58593
7	0,00522	0,6329	0,0257	0,66382
8	0,00491	0,4624	0,0318	0,49911
9	0,00571	0,52161	0,02832	0,55564
10	0,00653	0,60175	0,02319	0,63147
AVG	**0,006991**	**0,571972**	**0,028741**	**0,607704**

Tabel 4.40 Hasil Pengujian Arena2 (Segment Width)

	Initial Processing	Mesh Generation	Enemy Placement	Total Time
1	0,01135	0,68185	0,01208	0,70528
2	0,00577	0,58344	0,01733	0,60654
3	0,00735	0,58859	0,03784	0,63378
4	0,00488	0,62131	0,0144	0,64059
5	0,00696	0,61404	0,02707	0,64807
6	0,00638	0,58139	0,01694	0,60471
7	0,00766	0,60507	0,02252	0,63525
8	0,00699	0,57944	0,03387	0,6203
9	0,00604	0,6008	0,02399	0,63083
10	0,00784	0,56363	0,03696	0,60843
AVG	**0,007122**	**0,601956**	**0,0243**	**0,633378**

Tabel 4.41 Hasil Pengujian Arena3 (Segment Width)

	Initial Processing	Mesh Generation	Enemy Placement	Total Time
1	0,00787	0,62036	0,02197	0,6502
2	0,00775	0,54602	0,0191	0,57287
3	0,0047	0,5368	0,01959	0,56109
4	0,00983	0,54608	0,02496	0,58087
5	0,00818	0,57062	0,01337	0,59217
6	0,00775	0,54437	0,01349	0,56561
7	0,00537	0,53424	0,02142	0,56103
8	0,00427	0,53821	0,01282	0,5553
9	0,00745	0,5686	0,01147	0,58752
10	0,00555	0,64807	0,02148	0,6751
AVG	**0,006872**	**0,565337**	**0,017967**	**0,590176**

Tabel 4.42 Rata-Rata Waktu Pembuatan Ketiga Arena (Segment Width)

	Initial Processing	Mesh Generation	Enemy Placement	Total Time
Arena1	0,006991	0,571972	0,028741	0,607704
Arena2	0,007122	0,601956	0,0243	0,633378
Arena3	0,006872	0,565337	0,017967	0,590176

Sama seperti parameter sebelumnya, parameter ***Segment Width*** memiliki pengaruh yang minimal terhadap waktu pembuatan arena. Didapat peningkatan total waktu sebesar **4,2%** sementara terjadi penurunan sebesar **-2,8%** antara Arena1 dan Arena3.

Parameter selanjutnya yang akan diuji berupa ***Valid Segment Threshold***. Berikut adalah variasi nilai yang akan diberikan terhadap parameter dalam pengujian

Tabel 4.43 Perbedaan Parameter Ketiga Arena (Valid Segment)

	Valid Segment Threshold
Arena1	3
Arena2	6
Arena3	9

Setelah dilakukannya pengujian, berikut adalah hasil yang didapatkan.

Tabel 4.44 Hasil Pengujian Arena1 (Valid Segment)

	Initial Processing	Mesh Generation	Enemy Placement	Total Time
1	0,00781	0,27367	0,02359	0,30507
2	0,00307	0,22179	0,01257	0,23743
3	0,00346	0,2563	0,01895	0,27871
4	0,00435	0,27402	0,01082	0,28919
5	0,0034	0,29594	0,01495	0,31429
6	0,00385	0,22935	0,0148	0,248
7	0,00314	0,26343	0,0174	0,28397
8	0,00386	0,27379	0,01222	0,28987
9	0,00368	0,25868	0,01828	0,28064
10	0,00435	0,25101	0,01389	0,26925
AVG	**0,004097**	**0,259798**	**0,015747**	**0,279642**

Tabel 4.45 Hasil Pengujian Arena2 (Valid Segment)

	Initial Processing	Mesh Generation	Enemy Placement	Total Time
1	0,00366	0,23462	0,01392	0,2522
2	0,00391	0,25342	0,02026	0,27759
3	0,00366	0,271	0,01392	0,28858
4	0,00317	0,29565	0,01563	0,31445
5	0,00439	0,29517	0,03052	0,33008
6	0,00464	0,25903	0,01782	0,28149
7	0,00293	0,24902	0,01831	0,27026
8	0,00317	0,27612	0,02002	0,29931
9	0,00439	0,27051	0,01929	0,29419
10	0,00342	0,2627	0,01758	0,2837
AVG	**0,003734**	**0,266724**	**0,018727**	**0,289185**

Tabel 4.46 Hasil Pengujian Arena3 (Valid Segment)

	Initial Processing	Mesh Generation	Enemy Placement	Total Time
1	0,0058	0,27191	0,02856	0,30627
2	0,00439	0,26379	0,01166	0,27984
3	0,00354	0,26001	0,01923	0,28278
4	0,00366	0,22296	0,01294	0,23956
5	0,00305	0,26105	0,01874	0,28284
6	0,00421	0,24579	0,01538	0,26538
7	0,0033	0,25342	0,01819	0,27491
8	0,00305	0,25421	0,00995	0,26721
9	0,00403	0,28625	0,01093	0,30121
10	0,00385	0,28693	0,01208	0,30286
AVG	**0,003888**	**0,260632**	**0,015766**	**0,280286**

Tabel 4.47 Rata-Rata Waktu Pembuatan Ketiga Arena (Valid Segment)

	Initial Processing	Mesh Generation	Enemy Placement	Total Time
Arena1	0,004097	0,259798	0,015747	0,279642
Arena2	0,003734	0,266724	0,018727	0,289185
Arena3	0,003888	0,260632	0,015766	0,280286

Dapat dilihat dari hasil pengujian bahwa pengaruh parameter ***Valid Segment Threshold*** memiliki dampak yang minimal terhadap waktu pembuatan arena. Terlihat terjadi peningkatan waktu total sebesar **3,41%** antara Arena1 dan Arena2, sementara hanya terjadi peningkatan waktu total sebesar **0,2%** antara Arena1 dan Arena2.

Hal terakhir yang akan diujikan adalah pengaruh jumlah musuh yang dimunculkan pada arena terhadap waktu total pembuatan arena. Parameter-parameter yang terkait dengan jumlah penempatan musuh berupa ***Max Enemy Spawn Points*** dan ***Min/Max Enemy in Spawn***. Dikarenakan parameter yang diuji hanya terkait pada proses penempatan musuh, data yang diambil hanya berupa waktu penempatan musuh/***Enemy Placement***, selain itu akan diambil juga jumlah total musuh yang ada pada saat pembuatan arena.

Metode pengujian masih serupa dengan sebelumnya, terdapat tiga arena yang masing-masing memiliki nilai parameter yang berbeda yang akan diuji sebanyak 10 kali. Berikut adalah daftar parameter yang diuji beserta niliai-nilainya.

Tabel 4.48 Perbedaan Parameter Ketiga Arena (Max Spawn, Min/Max Enemy)

	Max Spawn Point	Min Enemy in Spawn	Max Enemy in Spawn
Arena1	3	1	3
Arena2	6	3	6
Arena3	9	6	9

Setelah melakukan pengujian pada setiap arena. Berikut adalah data yang telah didapatkan.

Tabel 4.49 Hasil Pengujian Arena1 (Max Spawn, Min/Max Enemy)

	Enemy Placement	Total Enemies
1	0,08497	2
2	0,08487	5
3	0,08339	5
4	0,06958	3
5	0,18206	4
6	0,07096	3
7	0,05026	3
8	0,07047	5
9	0,07369	4
10	0,07289	5
AVG	**0,084314s**	5

Tabel 4.50 Hasil Pengujian Arena2 (Max Spawn, Min/Max Enemy)

	Enemy Placement	Total Enemies
1	0,10211s	24
2	0,08191s	14
3	0,1004s	16
4	0,07983s	22
5	0,07794s	10
6	0,08813s	23
7	0,08112s	23
8	0,07666s	17
9	0,08551s	19
10	0,09039s	23
AVG	**0,0864s**	**19,1**

Tabel 4.51 Hasil Pengujian Arena3 (Max Spawn, Min/Max Enemy)

	Enemy Placement	Total Enemies
1	0,12122s	68
2	0,10754s	64
3	0,09821s	42
4	0,11041s	61
5	0,11432s	62
6	0,10406s	50
7	0,10303s	49
8	0,11047s	57
9	0,11298s	57
10	0,09528s	27
AVG	**0,107752s**	**53,7**

Tabel 4.52 Rata-Rata Total Musuh dan Waktu Penempatan Musuh

	Enemy Placement	Avg Enemy
Arena1	0,084314s	5
Arena2	0,0864s	19,1
Arena3	0,107752s	53,7

Dikarenakan parameter ***Max Enemy Spawn Points*** dan ***Min/Max Enemy in Spawn*** hanya mempengaruhi terhadap waktu penempatan musuh, dan tidak terhadap proses pembuatan matriks dan model 3d, maka yang akan menjadi fokus adalah waktu ***Enemy Placement***.

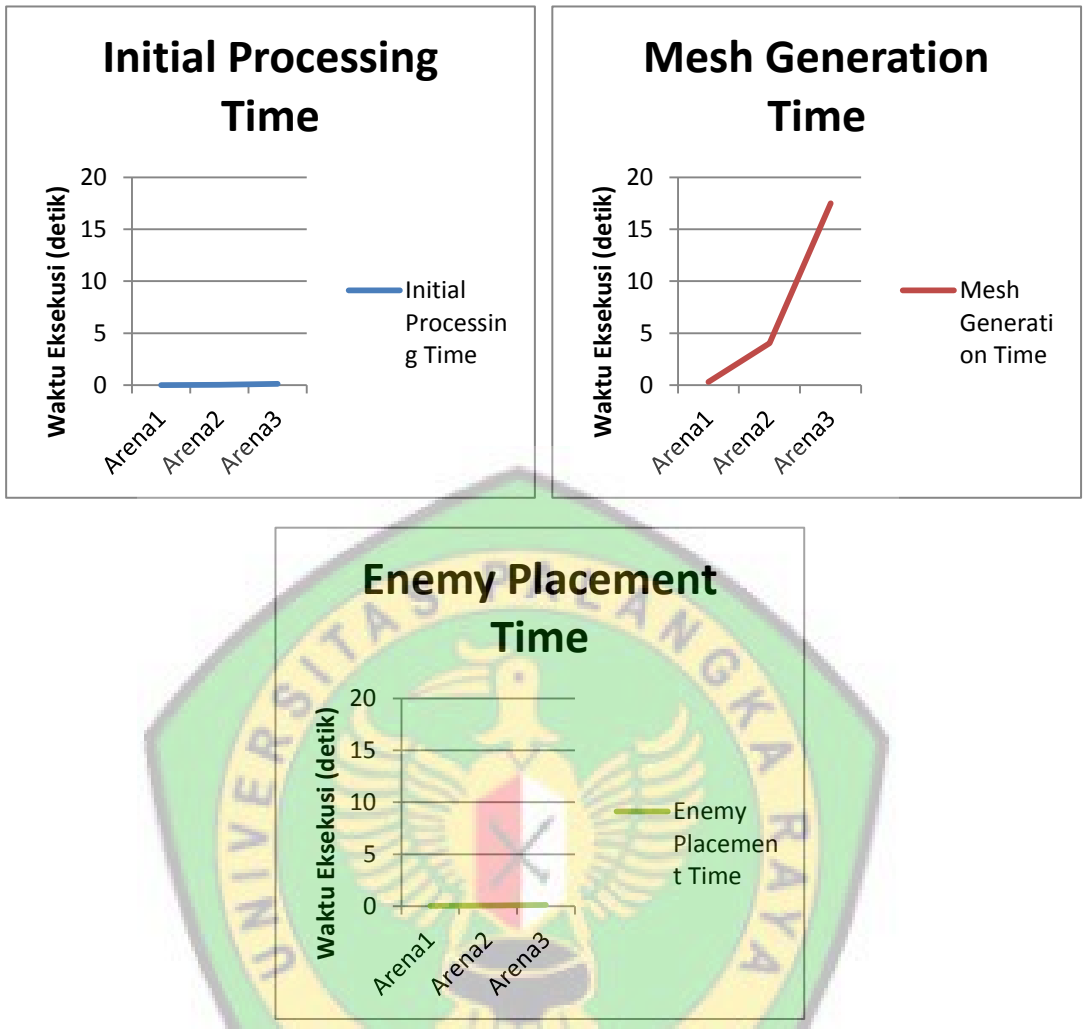
Ketika dilihat pada tabel, dapat diamati walaupun terjadi peningkatan waktu ketika semakin banyak musuh yang dimunculkan, peningkatan waktunya cukup kecil. Arena2 dengan rata-rata jumlah musuh total **19,1** hanya mengalami peningkatan waktu sebesar **2,4%** dibandingkan Arena1 dengan rata-rata jumlah total musuh sebanyak **5**. Sementara Arena3 yang memiliki rata-rata jumlah musuh **53,7**, mengalami peningkatan waktu sebesar **27,8%** dibandingkan Arena1

E. Pembahasan

Berdasarkan data yang telah dikumpulkan dari hasil pengujian seluruh parameter, dapat diamati bahwa parameter yang berhubungan dengan luas dimensi arena memiliki pengaruh yang sangat kuat terhadap waktu pembuatan arena dibandingkan parameter lainnya, dimana semakin luas arena yang dibuat, maka semakin signifikan waktu yang diperlukan untuk membuat arena tersebut.

Hal ini dapat jelas diamati pada hasil pengujian parameter ***Width***, ***Height***, dan ***Depth*** pada tabel 4.22. Hasil pada tabel menunjukan bahwa terdapat peningkatan sebesar **843%** ketika arena diperluas sebanyak dua kali lipat, sementara pada tiga kali lipat luas awal, terjadi peningkatan waktu sebesar **2.470%**. Hal yang sama juga dapat dilihat pada pengujian parameter ***Border Size*** pada tabel 4.27. Dimana peningkatan yang terjadi berupa **341%**, dan **1.341%** masing-masing untuk kelipatan dua dan tiga kali dari ukuran awal.

Ketika dilihat lebih teliti, sebagian besar peningkatan waktu yang terjadi berada pada proses ***Mesh Generation*** yaitu proses pembuatan objek 3D. Berikut adalah grafik yang dibuat beradasarkan hasil perbandingan waktu rata-rata pembuatan arena 1, 2, dan 3 dari pengujian parameter ***Height***, ***Width***, dan ***Depth***.



Gambar 4.10 Perbandingan Waktu Pembuatan Arena1,2,dan 3 (Height,Width,Depth)

Hal ini dikarenakan parameter-parameter yang mempengaruhi luas arena, juga mempengaruhi jumlah sel yang ada pada matriks yang akan dibuat menjadi objek 3D. Ketika dilakukannya pembuatan objek 3D, algoritma akan mengecek secara satu persatu setiap sel beserta tetangganya untuk menentukan bentuk yang akan digambarkan menjadi objek 3D. Sehingga jika matriks memilki jumlah sel yang banyak, maka proses pembuatan objek 3D akan semakin lama.

Sehingga dapat disimpulkan bahwa penerapan algoritma cellular automata yang dihasilkan lebih cocok digunakan untuk membuat arena permainan yang kecil.

Berdasarkan pengujian yang telah dilakukan berikut adalah beberapa poin rekomendasi dalam pengisian nilai parameter arena:

1. rekomendasi dimensi ***Height***, ***Width***, dan ***Depth*** secara berurutan berupa **100**, **100**, dan **10** atau yang memiliki luas/jumlah sel **<10.000** untuk memastikan bentuk terowongan gua arena yang cukup bervariasi, dan waktu yang digunakan untuk membuat arena tidak terlalu lama.

2. Untuk parameter Fill Percent direkomendasikan nilai **50 – 60**, dimana nilai 50 menghasilkan gua yang memiliki ruang yang bersifat lebih terbuka, sementara nilai 60 menghasilkan gua dengan terowongan yang sempit.

3. Rekomendasi untuk jumlah maksimum iterasi atau parameter ***Smooth Iteration*** diberikan nilai **3** untuk waktu pembuatan arena yang tercepat sementara masih menghasilkan bentuk yang cukup akurat dari bentuk yang sebenarnya.

4. Dikarenakan parameter lainnya memiliki dampak minimal terhadap waktu pembuatan arena, maka nilai yang diberikan dapat disesuaikan dengan keperluan.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan apa yang telah didapat dari bab-bab sebelumnya, maka kesimpulan yang didapat dari penyusunan skripsi "Penerapan Algoritma Cellular Automata dalam Menghasilkan Map Game RPG Secara Prosedural" adalah sebagai berikut:

1. Game dibuat menggunakan game engine Unity dengan bahasa pemrograman C# dengan target platform PC (*Personal Computer*) sistem operasi Windows, dan objek 3D yang digunakan dibuat menggunakan Blender.
2. Algoritma Cellular Automata diterapkan pada bagian pembuatan arena permainan pada game.
3. Pembuatan arena dimulai dengan sebuah matriks 2D yang diisi secara acak dengan nilai 0 dan 1, kemudian dengan melakukan iterasi algoritma cellular automata terhadap matriks tersebut, akan didapatkan sebuah matriks yang menyerupai bentuk peta gua yang dapat diproses untuk mendapatkan objek 3D yang bisa digunakan.
4. Algoritma yang diimplementasikan pada game lebih baik digunakan untuk membuat arena yang berukuran kecil (luas ±10.000 sel), dikarenakan dengan semakin meningkatnya bidang arena permainan, maka waktu yang diperlukan untuk membentuk arena tersebut meningkat secara eksponensial.
5. Ukuran rekomendasi untuk parameter pembuatan arena ketika diuji dengan komputer spesifikasi komputer processor Intel Core i7 4720HQ, VGA NVIDIA GeForce GTX 950M, dan RAM 8 GB adalah *Width* = ±100, *Height* = ±100, *Depth* = 10, *Smooth Iteration* = 3, dan *Fill Percent* = 50 – 60, sementara untuk nilai parameter lainnya dapat disesuaikan dengan kebutuhan.

5.2 Saran

Berdasarkan hasil yang diperoleh dari pengujian dan pembuatan aplikasi, disarankan dalam pengembangan selanjutnya untuk menggunakan metode yang berbeda dalam proses pembuatan objek 3D dari matriks 2D untuk mengurangi waktu dalam pembuatan arena permainan.

DAFTAR PUSTAKA

Ambler, Scott W. 2005. *The elements of UML 2.0 style*. Cambridge U.K. New York: Cambridge University Press.

Barriga, A. Nicolas. 2018. A Short Introduction to Procedural Content Generation Algorithm for Video Games. *International Journal on Artificial Intelligence Tools*. Retrieved from https://www.researchgate.net/publication/331717755_A_Short_Introduction_to_Procedural_Content_Generation_Algorithms_for_Videogames(diakses 13 Maret 2020)

Bond, Logan. 2017. *Procedural Generation: An Algorithmic Analysis of Video Game Design and Level Creation. Honors Theses*. 249. http://scholarlycommons.obu.edu/honors_theses/249. (diakses 13 Maret 2020)

Dennis, Alan, Barbara H. Wixom, and Roberta M. Roth. 2012. *System analysis and design*. Hoboken, NJ: John Wiley.

Dharwiyanti, Sri., Wahono, R. S. 2003. Pengantar Unified Modeling Language (UML). 1-13.

ECMA International. 2017. *C# Language Specification*. *ECMA-334*. https://www.ecma-international.org/publications/files/ECMA-ST/ECMA-334.pdf (diakses 15 Maret 2020)

Gardner, Martin., 1970. *Mathematical Games – The fantastic combinations of John Conway's new solitaire game "life"*. https://www.ibiblio.org/lifepatterns/october1970.html (diakses 13 Maret 2020)

Johnson, Lawrence, Georgios N. Yannakakis, and Julian Togelius. 2010. *Cellular automata for real-time generation of infinite cave levels*. https://www.researchgate.net/publication/228919622_Cellular_automata_for_real-time_generation_of. (diakses 10 Mei 2020)

Pressman, RS. 2010. *Software Engineering: A Practitioner's Approach*. New York, America: McGraw-Hill Higher Education.

Smith, Gillian. 2015. *An Analog History of Procedural Content Generation*. *FDG*. https://www.ecma-international.org/publications/files/ECMA-ST/ECMA-334.pdf (diakses 12 Maret 2020)

Weisstein, Eric W. *Cellular Automaton*. http://mathworld.wolfram.com/CellularAutomaton. html. (diakses 21 Maret 2020)

Wolf, Mark J.P. 2008. *The Video Game Explosion: a History from Pong to Playstation and Beyond*, Greenwood Press