PENGEMBANGAN METODE *DIFFICULTY ADJUSTMENT* SECARA PROSEDURAL PADA *GAME* "*LOST LABYRINTHS : ROGUE'S ODYSSEY*"

MUHAMMAD RAIHANDI JAMAL RITONGA

201401053

PROGRAM STUDI S1 ILMU KOMPUTER

FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI

UNIVERSITAS SUMATERA UTARA

MEDAN

2024

PENGEMBANGAN METODE *DIFFICULTY ADJUSTMENT* SECARA PROSEDURAL PADA *GAME* "*LOST LABYRINTHS : ROGUE'S ODYSSEY*"

SKRIPSI

Diajukan untuk melengkapi tugas dan memenuhi syarat memperoleh ijazah
Sarjana Ilmu Komputer

MUHAMMAD RAIHANDI JAMAL RITONGA
201401053

PROGRAM STUDI S1 ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA
MEDAN
2024

LEMBAR PERSETUJUAN

Judul	:	PENGEMBANGAN METODE DIFFICULTY ADJUSTMENT SECARA PROSEDURAL PADA GAME "LOST LABYRINTHS : ROGUE'S ODYSSEY"
Kategori	:	SKRIPSI
Nama	:	MUHAMMAD RAIHANDI JAMAL RITONGA
Nomor Induk Mahasiswa	:	201401053
Program Studi	:	SARJANA (S-1) ILMU KOMPUTER
Fakultas	:	ILMU KOMPUTER DAN TEKNOLOGI INFORMASI UNIVERSITAS SUMATERA UTARA
Komisi Pembimbing	:	

Pembimbing 2	Pembimbing 1
Dr. Jos Timanta Tarigan S.Kom., M.Sc.	Handrizal S.Si., M.Comp.Sc.
NIP. 198307232009122004	NIP. 197706132017061001

Diketahui/Disetujui
Program Studi S-1 Ilmu Komputer
Ketua

Dr. Amalia, S.T., M.T.
NIP. 197812212014042001

PERNYATAAN

PENGEMBANGAN METODE *DIFFICULTY ADJUSTMENT* SECARA PROSEDURAL PADA *GAME* "*LOST LABYRINTHS : ROGUE'S ODYSSEY*"

SKRIPSI

Saya mengakui bahwa skripsi ini adalah hasil karya saya sendiri, kecuali beberapa kutipan dan ringkasan yang masing-masing telah disebutkan sumbernya.

Medan, 17 Mei 2024

Muhammad Raihandi Jamal Ritonga

201401053

PENGHARGAAN

Puji syukur peneliti ucapkan kepada Allah SWT atas rahmat dan karunia-nya dan memberikan kesempatan peneliti untuk menyelesaikan skripsi yang berjudul *"Pengembangan metode difficulty adjustment secara procedural pada game "Lost Labyrinths : Rogue's Odyssey""*. Pada kesempatan kali ini peneliti ingin mengucapkan dan memberi rasa terima kasih kepada semua pihak yang ikut terlibat dalam proses pembuatan skripsi ini

Terima kasih kepada:

1. Bapak Dr. Muryanto Amin, S.Sos., M.Si, selaku Rektor Universitas Sumatera Utara.
2. Dr. Maya Silvi Lydia B.Sc., M.Sc Sebagai Dekan Fakultas Ilmu Komputer Universitas Sumatera Utara,
3. Dr. Amalia, ST., M.T., sebagai Ketua Prodi Program Studi Ilmu Komputer.
4. Bapak Handrizal S.Si., M.Comp.Sc sebagai Pembimbing I, yang yang memberikan arahan dan bimbingan yang sangat berharga dalam menyelesaikan skripsi ini.
5. Bapak Dr. Jos Timanta Tarigan S.Kom., M.Sc., selaku Ketua Laboratorium Computer Vision dan Multimedia Program Studi Ilmu Komputer, sekaligus Pembimbing II yang memberikan arahan dan bimbingan yang sangat berharga dalam menyelesaikan skripsi ini.
6. *Play-tester* yang dengan sukarela menyisihkan waktu untuk membantu peneliti dengan bermain game yang dibuat, memberikan tanggapan selama wawancara, serta memberikan kritik, masukan, dan saran..
7. Seluruh keluarga yang selalu mensupport dari segi mental ataupun materi selama peneliti menjalani kehidupan perkuliahan ini.
8. Teman teman terdekat peniliti, Nico dan Erick Yudha Pratama Sukku yang sangat berperan aktif dalam pengerjaan skripsi.
9. Teman-teman seperjuangan di program studi Ilmu Komputer yang telah memberikan dukungan dan bantuan yang berarti selama proses perkuliahan.

Peneliti juga ingin berterima kasih kepada semua pihak yang namanya tidak bisa disebutkan dalam kesempatan ini.

Dengan rendah hati, peneliti menyampaikan permohonan maaf atas segala kekurangan dan ketidaksempurnaan yang mungkin terdapat dalam penulisan ini. Semoga hasil dari penelitian ini memberikan efek yang positif terhadap perkembangan yang pengetahuan kedepannya Semoga hasil penelitian ini dapat memberikan kontribusi positif bagi perkembangan ilmu pengetahuan. Terima kasih atas kesempatan ini.

Medan, 17 Mei 2024

Peneliti,

Muhammad Raihandi Jamal Ritonga

NIM 201401053

ABSTRAK

Dalam industri permainan *video*, genre *2D platformer* memiliki basis penggemar yang besar karena menawarkan pengalaman bermain yang seru dan menghibur. Salah satu tantangan utama dalam pengembangan *game 2D platformer* adalah menjaga tingkat kesulitan yang menarik dan memuaskan bagi berbagai tingkat keterampilan pemain. Sistem *difficulty adjustment* otomatis adalah solusi yang menjanjikan, memungkinkan tingkat kesulitan menyesuaikan dengan kemampuan pemain secara dinamis. Studi ini berfokus pada pengembangan dan implementasi mekanika permainan berbasis penyesuaian kesulitan pada *game "Lost Labyrinths: Rogue's Odyssey"* sebuah *game 2D action platformer* yang menggunakan *prosedural content generation (PCG)* untuk desain *level*nya. Penelitian ini mengidentifikasi beberapa masalah utama, termasuk konsistensi tingkat kesulitan dan daya tarik jangka panjang permainan. Untuk mengatasi masalah ini, diterapkan sistem *difficulty adjustment* yang dinamis, dan penambahan mekanika permainan baru. Metode penelitian melibatkan pengujian respons pemain terhadap *mode* permainan *"Expert"* yang mengimplementasikan sistem penyesuaian kesulitan ini. Hasil penelitian menunjukkan bahwa pengimplementasian *difficulty adjustment* dalam *"Lost Labyrinths: Rogue's Odyssey"* memberikan hasil yang sangat memuaskan. Respons pemain terhadap *mode game "Expert"* sangat positif, dengan penilaian baik terhadap aspek-aspek terkait *difficulty adjustment* otomatis. Fitur *"Fog of War"* yang digunakan dalam *mode* ini dinilai efektif dalam menambahkan elemen tantangan dan ketegangan. Secara keseluruhan, tingkat kesulitan yang menantang dalam *mode "Expert"* memberikan pengalaman bermain yang lebih mendalam dan memuaskan, meningkatkan daya tarik dan keberlanjutan game. Hasil dari penelitian ini menyimpulkan bahwa penerapan mekanika permainan berbasis *difficulty adjustment* yang dinamis dapat meningkatkan kualitas dan daya tarik game 2D platformer, memberikan tantangan baru bagi pemain dan mempertahankan minat mereka dalam jangka panjang.

Kata kunci: *Difficulty Adjustment, Fog of War, Mode Expert, Game 2D Platformer, Lost Labyrinths : Rogue's Odessey*

DEVELOPMENT OF PROCEDURAL DIFFICULTY ADJUSTMENT METHOD IN THE GAME "LOST LABYRINTHS: ROGUE'S ODYSSEY"

ABSTRACT

In the video game industry, the 2D platformer genre has a large fan base as it offers a fun and entertaining gaming experience. One of the main challenges in 2D platformer game development is to keep the difficulty level interesting and satisfying for different skill levels of players. An automatic difficulty adjustment system is a promising solution, allowing the difficulty level to dynamically adjust to the player's ability. This study focuses on the development and implementation of difficulty adjustment-based game mechanics in the game "Lost Labyrinths: Rogue's Odyssey," a 2D action platformer game that uses procedural content generation (PCG) for its level design. This research identified several key issues, including difficulty consistency and the long-term appeal of the game. To address these issues, a dynamic difficulty adjustment system was implemented, and new game mechanics were added. The research method involved testing player responses to an "Expert" game mode that implemented this difficulty adjustment system. The results showed that the implementation of difficulty adjustment in "Lost Labyrinths: Rogue's Odyssey" yielded very satisfactory results. Player response to the "Expert" game mode was very positive, with favorable ratings on aspects related to automatic difficulty adjustment. The "Fog of War" feature used in this mode was effective in adding an element of challenge and suspense. Overall, the challenging difficulty level in the "Expert" mode provided a more immersive and satisfying gaming experience, increasing the game's appeal and sustainability. The results of this study conclude that the implementation of dynamic difficulty adjustment-based game mechanics can improve the quality and appeal of 2D platformer games, providing new challenges for players and sustaining their interest in the long run.

***Keywords**: Difficulty Adjustment, Fog of War, Expert Mode, Game 2D Platformer, Lost Labyrinths : Rogue's Odessey*

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR RUMUS

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1. Latar Belakang

Dalam dunia game, genre 2D *platformer* memiliki basis penggemar yang besar karena memberikan pengalaman permainan yang seru dan menghibur. Salah satu tantangan dalam mengembangkan *game* 2D *platformer* adalah menjaga agar tingkat kesulitan permainan tetap menarik dan memuaskan bagi para pemain. Dengan demikian, pengembangan *game mechanic* yang fokus pada peningkatan kesulitan dapat menjadi langkah penting dalam meningkatkan kualitas permainan. Ketika pemain memilih tingkat kesulitan yang mudah, *game* akan memiliki tingkat kesulitan yang rendah untuk keseluruhan *gameplay*. Masalahnya adalah, sebagai pemain, memiliki kemampuan untuk menguasai permainan setelah bermain beberapa kali. Jadi, setelah dapat menguasai permainan, kesulitan yang mudah sekarang menjadi terlalu membosankan karena tantangannya tidak cukup sulit (Sutoyo, et al., 2015).

Difficulty adjustments dalam *games* merupakan suatu konsep yang dirancang untuk mengatur tingkat kesulitan permainan secara otomatis. *Difficulty adjustment* menggunakan sistem otomatis untuk mengatur tingkat kesulitan, sehingga pemain tidak harus memilih tingkat kesulitan yang sesuai dengan kemampuan mereka tetapi tingkat kesulitan akan disesuaikan seiring dengan kenaikan *level*nya. Tujuannya agar permainan dapat dinikmati oleh berbagai tingkat pemain mulai dari pemula hingga pemain berpengalaman. Berdasarkan penelitian (Mitre-Hernandez, Carrillo, & Lara-Alvarez, 2021) menyatakan bahwa penyesuaian kesulitan harus dilakukan dan diimplementasikan sedemikian rupa sehingga pengguna tidak merasakan kesulitan perubahan permainan. Penyesuaian kesulitan *video game* bergantung pada data *game* seperti permainan yang berbeda

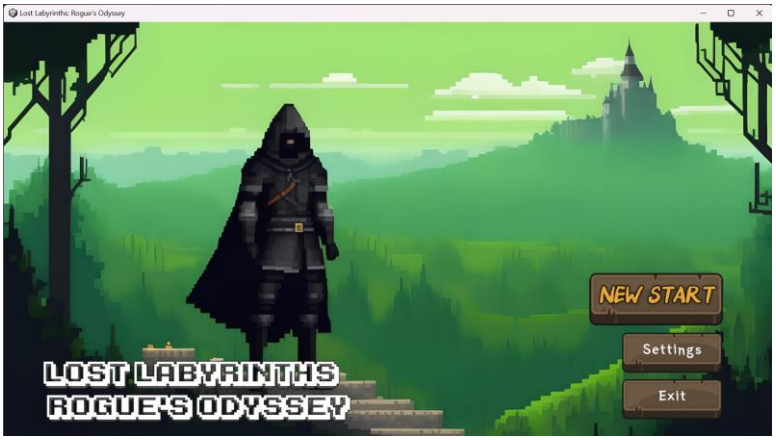
memerlukan fitur yang berbeda. Penyesuaian kesulitan dapat dirancang dengan menggabungkan fitur permainan umum (seperti skor, waktu, dan lainnya).

Seperti pada game lain tidak memiliki tingkat kesulitan atau setidaknya pemain tidak memiliki opsi untuk mengubah tingkat kesulitan permainan. *Game* seperti *Dark Souls 4* atau *Mario Odyssey 5* keduanya merupakan game yang berbeda genre yang berbeda tetapi tanpa pengaturan tingkat kesulitan permainan. Itu adalah cara permainan dimaksudkan untuk dialami, tetapi dengan kerumitan ini dapat memiliki banyak pemain lain yang tanpa tingkat kesulitan yang membuat mereka merasa nyaman, sebagian besar waktu akan kalah dan frustrasi atau bahkan bosan pada suatu saat jika permainan tidak cukup menantang (Muñoz, 2020). Faktor kesenangan dalam permainan tergantung pada tiga faktor, yaitu tantangan, fantasi, dan rasa ingin tahu. Menciptakan tingkat tantangan yang memadai tidaklah mudah ketika pemain dengan berbagai keterampilan diadu satu sama lain. Ketika lawan dikalahkan dengan mudah, permainan akan terlihat membosankan (Zohaib, 2018).

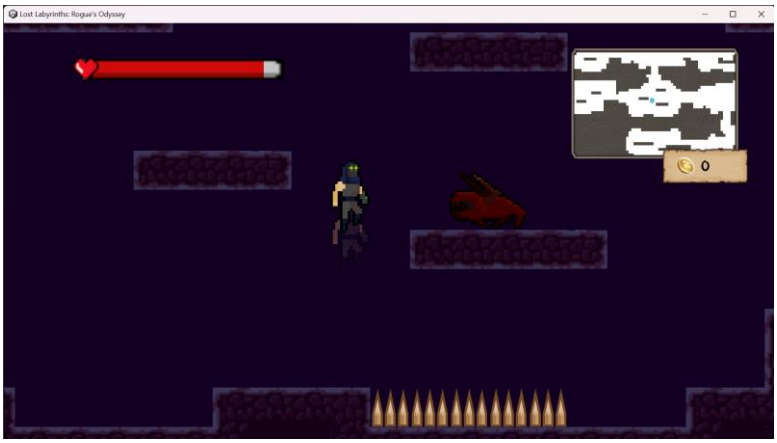
Dalam beberapa kasus, tingkat kesulitan *game* dikaitkan dengan konten atau aset yang digunakan. Misalnya, ketika melompat antar platform, ukuran platform dan jarak di antara mereka mempengaruhi kesulitan jika kecepatan pemain, tinggi lompatan, dan gravitasi tetap konstan. Sebuah metodologi untuk membuat adaptif untuk membuat *game* edukasi yang mencakup beberapa generasi dinamis cerita dan struktur yang dinamis. Namun dalam hal ini tujuannya adalah untuk mencapai menyeimbangkan kesulitan adaptif dengan cara yang lebih umum melalui penggunaan pembuatan konten prosedural (PCG). Sebuah teknik untuk menghasilkan konten dan aset baru yang memenuhi memenuhi persyaratan permainan secara real-time (Hendrix, et al., 2016).

Lost Labyrinth: Rogue's Odyssey, yang ditunjukkan pada gambar 1. 1 dan gambar 1. 2 di bawah, merupakan judul permainan yang dibuat oleh pengembang sebelumnya menggunakan pendekatan implementasi *prosedural content generation* secara hybrid untuk pembuatan levelnya. *Game* ini termasuk dalam genre 2D *action platformer*. Aset pada game ini tidak dibuat dari awal melainkan diambil dari tutorial yang ada pada youtube yang menyediakan *assets* untuk *game*

2D platformer pada deskripsi youtubenya. *Game* dimulai dengan *settingan level* yang acak tidak ada tingkatan *level*, di *game* ini *player* bisa menemukan berbagai musuh, *obstacle*, *treasure*, dan ada fungsi *shop* yang juga acak. *Game* ini akan berakhir apabila nyawa atau *bar hp* pemain mencapai 0 .



Gambar 1. 1 Tampilan *Main Menu* pada *"Lost Labyrinths : Rogue's Odyssey"*



Gambar 1. 2 Tampilan Contoh Ruangan pada *Lost Labyrinths : Rogue's Odyssey*

Maka dari itu dalam penelitian ini, *Game* "*Lost Labyrinths : Rogue's Odyssey*" sebagai *game* 2D *platformer* memiliki potensi untuk ditingkatkan dalam hal dinamika permainan, desain *level*, dan aspek-aspek lain yang memengaruhi tingkat kesulitan. Dalam konteks ini, pembaruan *game mechanic* dapat mencakup penyesuaian pada pola pergerakan karakter, penambahan elemen rintangan yang lebih menantang, dan pengaturan level yang dirancang untuk menguji

keterampilan pemain dengan cara yang berbeda. Lalu dalam pengembangan ini juga akan membuat *difficulty* baru yang dinilai berdasarkan item yang didapatkan, jumlah dan tipe musuh yang dikalahkan, dan damage yang diterima selama didalam *level*. Dengan memperhatikan aspek-aspek ini, pengembang dapat menciptakan pengalaman bermain yang lebih mendalam, memotivasi pemain untuk terus mencoba dan meningkatkan keterampilan mereka.

Seiring dengan meningkatnya persaingan di pasar *game* 2D *platformer*, pengembangan *game mechanic* yang fokus pada *difficulty adjustment* pada "*Lost Labyrinths : Rogue's Odyssey*" tidak hanya akan memberikan tantangan baru bagi para pemain, tetapi juga dapat menjaga daya tarik dan keberlanjutan *game* dalam jangka panjang.

1.2.Rumusan Masalah

Game "*Lost Labyrinths : Rogue's Odyssey*" belum menerapkan sistem pengembangan tingkat kesulitan dan hanya menyediakan satu *level* permainan. Oleh karena itu, diperlukan pengembangan lebih lanjut pada mekanika permainan dengan *Difficulty Adjustment* yang berfokus pada setiap *level*, memberikan tantangan yang lebih bervariasi kepada pemain dan menjadikan pengalaman bermain lebih menarik.

1.3.Tujuan Penelitian

Tujuan dari penelitian ini yaitu membuat *mode game* baru dengan mengimplementasikan *difficulty adjustment* kedalam game "*Lost Labyrinths : Rogue's Odyssey*". *Mode game* ini membuat tingkatan level baru yang disesuaikan setiap levelnya dengan penambahan seperti *fog of war*, dan musuh yang semakin sulit dan banyak.

1.4.Manfaat Penelitian

Manfaat dari penelitian ini adalah sebagai berikut.

1. Meningkatkan dinamika permainan dan aspek-aspek lain yang memengaruhi tingkat kesulitan pada game "*Lost Labyrinths: Rogue's Odyssey*". Dengan demikian, penelitian ini dapat memberikan pengalaman bermain yang lebih menarik.

2. Pengembangan *game mechanic* yang fokus pada peningkatan kesulitan dapat menjaga daya tarik *game* dalam jangka panjang. Dengan memanfaatkan metode *difficulty adjustment*, *game* ini dapat tetap menarik bagi berbagai tingkat pemain mulai dari pemula hingga pemain berpengalaman, sehingga meningkatkan keberlanjutan *game* di pasar yang semakin kompetitif Membantu memberikan kontribusi pada inovasi dalam pengembangan game, terutama dalam genre 2D *platformer*. Dengan fokus pada *difficulty adjustment* melalui penyesuaian mekanika permainan, penelitian ini dapat menjadi referensi bagi pengembang game lainnya untuk menghadapi tantangan yang serupa.
3. Membahas penggunaan metode *PCG* untuk mencapai keseimbangan kesulitan adaptif dalam game. Hal ini dapat memberikan inspirasi dan panduan bagi pengembang game untuk meningkatkan fleksibilitas dalam pengembangan permainan.

1.5.Batasan Masalah

Berikut batasan masalah dalam penelitian ini.

1. *Mode game* memiliki tingkatan *difficulty* yang terus meningkat.
2. Tingkatan *difficulty* memiliki pengaruh ke *obstacle*, *variant* dan *behavior* musuh, jumlah dan jenis *item treasure*, dan *item shop* disetiap *level* nya.
3. *Mode game* dalam penelitian ini terbatas hanya sampai 10 level.
4. Ruangan semakin besar seiring dengan peningkatan *level*.
5. Bahasa pemrograman C# digunakan untuk menulis algoritma pada *platform Unity*.
6. *Difficulty* baru dinilai berdasarkan item yang didapatkan, jumlah dan tipe musuh yang dikalahkan, dan damage yang diterima selama didalam level.

1.6.Penelitian Relevan

1) Penelitian oleh (Wheat, Masek, Lam, & Hingston, 2016) berjudul "*Modeling Perceived Difficulty in Game Levels*". Dalam penelitian ini membahas tentang minat baru dalam menciptakan konten prosedural untuk *game* menciptakan kebutuhan untuk menilai tingkat kesulitan. Penelitian ini menyelidiki faktor-faktor yang memengaruhi kesulitan dalam *level*

permainan yang dihasilkan prosedural. Dipenelitian ini mengidentifikasi faktor-faktor terkait pengalaman pemain dan mengembangkan model untuk memprediksi kesulitan. Penelitian ini fokus pada *platformer 2D*, dengan pengembangan *test-bed* untuk pengumpulan dan analisis data. Metrik permainan dan pemain yang diidentifikasi dievaluasi menggunakan klasifikasi *Multi-Layer Perceptron*, J48, dan *Random Forest* dari WEKA. Hasilnya memberikan wawasan awal tentang metrik untuk mengembangkan model klasifikasi kesulitan yang dirasakan.

2) Penelitian oleh (Wu, Lai, Lin, Huang, & Tai, 2018) berjudul "*Procedurally Generating Game Level with Specified Difficulty*". Dalam penelitian ini membahas perkembangan generasi konten prosedural, *game* menggunakan metode ini untuk menciptakan pengalaman berbeda setiap kali dimainkan. Namun, hasilnya tidak selalu menjamin variasi gameplay yang memadai. Penelitian ini memakai contoh game Pac-Man.Dalam penelitian ini, labirin *Pac-Man* dihasilkan dengan tingkat kesulitan berdasarkan jumlah persimpangan dan elemen tertentu. Labirin dihasilkan dalam sembilan tingkat kesulitan, dapat dibuat dalam waktu 40ms, dan dapat dimainkan. Pendekatan ini praktis dan cocok untuk aplikasi game waktu nyata.
3) Penelitian oleh (Sutoyo, Winata, Oliviani, & Supriyadi, 2015) berjudul "*Dynamic Difficulty Adjustment in Tower Defence*". Penelitian ini bertujuan untuk menciptakan *game* pertahanan menara yang dapat menyesuaikan tingkat kesulitannya dengan kemampuan pemain. Dalam game ini, tingkat kesulitan akan menjadi dinamis, disesuaikan berdasarkan nyawa pemain, kesehatan musuh, dan pilihan keterampilan pasif yang dipilih oleh pemain. Hal ini bertujuan untuk memberikan pengalaman bermain yang beragam, di mana tingkat kesulitan tinggi akan diatur jika pemain menggunakan strategi yang baik, sementara tingkat kesulitan rendah akan diatur untuk pemain dengan strategi yang kurang efektif. Hasil dari penelitian ini mencakup game pertahanan menara dengan tingkat kesulitan dinamis, dokumen penyesuaian tingkat kesulitan dinamis (DDA), serta hasil permainan untuk kasus strategi terbaik, rata-rata, dan terburuk.

BAB II
TINJAUAN PUSTAKA

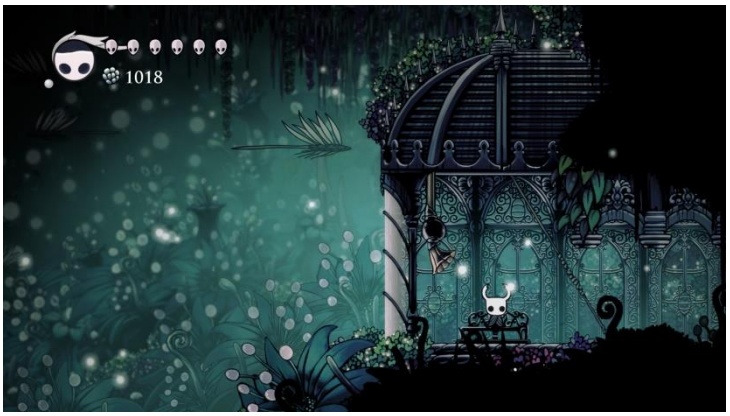
2.1. Game Platformer

Game Platformer juga dikenal sebagai *game* lari dan loncat menyajikan aksi melalui interaksi dengan avatar yang melompat dan turun mengelilingi rintangan. Gerakan dalam *game platformer* terjadi ke semua arah (misalnya ke atas, ke bawah, ke kiri, dan ke kanan) dibandingkan dengan genre lain yang membuat pemain tetap di tanah dan memungkinkan pergerakan sepanjang sumbu X dan Z (Oprean, Gould, Riedel, & Larsen, 2023). *Game platformer* menjadi salah satu genre *game* yang populer seperti *Hollow Knight*(2017), *Celeste*(2018) untuk yang 2D dan *Super Mario 3D World + Bowser's Fury* (2021), *Astro's Playroom* (2020) untuk yang 3D.

Game platformer bisa dibagi menjadi:

a. *2D platformer*

2D platformer adalah genre permainan video di mana pemain mengontrol karakter yang bergerak melintasi tingkatan horizontal, melompati rintangan, dan mengatasi tantangan untuk mencapai tujuan, sambil mengeksplorasi dunia yang dibuat dalam dua dimensi. Contohnya adalah *Hollow Knight (2017)* yang diilustrasikan pada gambar 2. 1 di bawah ini.



Gambar 2. 1 Contoh *Game 2D Platformer*, *Hollow Knight*(2017)

b. *3D platformer*

3D platformer adalah jenis *game* di mana pemain menjelajahi dunia tiga dimensi sambil menghadapi tantangan platform, melakukan lompatan, dan mengumpulkan objek untuk mencapai tujuan. Gambar 2. 2 menunjukkan *Super Mario 3D World + Bowser's Fury* (2021).

Gambar 2. 2 Contoh *Game 3D Platformer*, *Super Mario 3D World + Bowser's Fury*(2021)

2.2. Procedural Content Generation

Procedural Content Generation adalah konsep atau paradigma yang dimana semua bagian dari konten dapat dihasilkan dengan beberapa kode yang ditulis dengan baik. *PCG* dapat diterapkan pada hampir semua aspek dalam sebuah game melalui scripting. Yang menarik tentang *PCG* adalah bahwa komputer dapat mengambil beberapa tanggung jawab desainer dengan diberikannya beberapa instruksi dan dibiarkannya membuat bagian-bagian dari dunia game secara mandiri. *PCG* juga dapat hadir dalam beberapa bentuk yang praktis. Konten dapat dihasilkan dari awal, seperti tekstur, atau aset dapat dihasilkan dari serangkaian bagian yang sudah dibuat sebelumnya, seperti menghasilkan adegan tavern dari properti yang sudah dibuat sebelumnya seperti meja, kursi, tong, dan peti. Pilihan lainnya adalah memberikan alat kepada pemain untuk mengambil peran dalam menciptakan konten. Pemain yang membuat konten tidak selalu merupakan *PCG*, tetapi telah dibuat sistem *PCG* yang sekarang mengambil input pengguna sebagai parameter. Contohnya dapat dilihat pada gambar 2. 3 di bawah ini yang merupakan permainan populer *Minecraft* yang dikembangkan oleh Mojang (Watkins, 2016).

Gambar 2. 3 Contoh *Game PCG*, *Minecraft*(2009)

2.3. Difficulty Adjustment

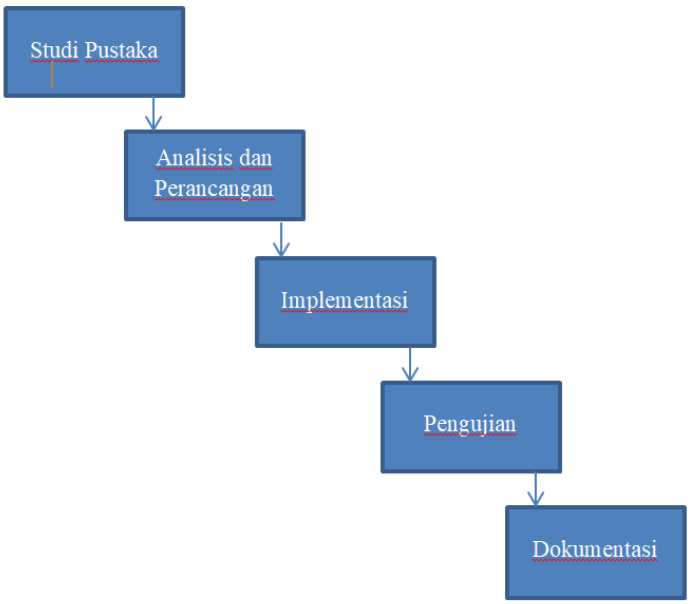
Difficulty adjustment didefinisikan sebagai suatu proses yang melibatkan pencarian kebijakan yang memadai untuk memilih properti *game*, dengan tujuan menyediakan keseimbangan yang optimal dalam pengalaman bermain (Sekhavat, 2017). Implementasi *difficulty adjustment* harus dilakukan dengan cermat sehingga perubahan dalam tingkat kesulitan tidak terlalu mencolok bagi pengguna. Penyesuaian tingkat kesulitan dalam video *game* didasarkan pada analisis data game yang mencakup berbagai fitur yang dibutuhkan oleh setiap permainan, mengingat bahwa setiap *game* memiliki kebutuhan yang unik (Mitre-Hernandez, Carrillo, & Lara-Alvarez, 2021).

Menyesuaikan variabel dalam game itu sendiri seperti tingkat kesulitan, atau memvariasikan jumlah musuh yang dihadapi. Dalam beberapa kasus, tingkat kesulitan game terkait dengan konten atau aset yang digunakan. Misalnya, ketika melompat antar platform, ukuran platform dan jarak di antara mereka mempengaruhi kesulitan jika kecepatan pemain, tinggi lompatan, dan gravitasi tetap konstan. Hal ini tujuannya adalah untuk mencapai menyeimbangkan kesulitan adaptif dengan cara yang lebih umum melalui penggunaan pembuatan konten prosedural (*PCG*), sebuah teknik untuk menghasilkan konten dan aset baru yang memenuhi memenuhi persyaratan permainan secara real-time (Hendrix, Bellamy-Wood, McKay, Bloom, & Dunwell, 2016).

BAB III
ANALISIS DAN PERANCANGAN SISTEM

3.1. Metodologi Penelitian

Metode pengembangan waterfall dipilih sebagai kerangka metodologi dalam penelitian ini. Pendekatan ini terkenal sebagai suatu proses linear yang terstruktur, di mana setiap tahapan pengembangan memiliki langkah-langkah yang jelas dan telah ditentukan sebelumnya. Tujuan dari menerapkan metodologi waterfall dalam penelitian ini adalah untuk menjamin bahwa proses pengembangan game dengan menggunakan metode difficulty adjustment pada pembuatan levelnya berjalan lancar. Metodologi ini diilustrasikan pada gambar 3. 1 di bawah ini.



Gambar 3. 1 Metode Pengembangan Waterfall

3.2. Analisis dan Perancangan

Pada tahap ini, dilakukan analisis *requirements* dan kebutuhan dalam penelitian. Selain itu dilakukan juga perancangan alur pengembangan game mode *"Expert"* dengan metode *difficulty adjustment.*

3.2.1 Analisis Masalah

Pertama-tama, diperlukan pemahaman mendalam terhadap tantangan dan masalah yang dihadapi dalam pengembangan game ”Lost Labyrinths : Rogue’s Odyssey” yang menerapkan *difficulty adjustment* dan *game 2D platformer*. Pada tahap ini, beberapa masalah utama yang muncul dari latar belakang ini adalah sebagai berikut:

1) **Tingkat Kesulitan yang Konsisten**

Salah satu masalah utama yang dihadapi dalam pengembangan *game 2D platformer* adalah mempertahankan tingkat kesulitan yang menarik bagi berbagai tingkat pemain. Ada risiko bahwa pemain yang lebih mahir akan merasa bosan dengan tingkat kesulitan yang terlalu mudah setelah mereka menguasai permainan. Ini menimbulkan masalah dalam mempertahankan minat pemain dalam jangka panjang.

2) **Penggunaan *Procedural Content Generation* (*PCG*)**

Penggunaan *PCG* untuk membuat level secara acak dalam game dapat menjadi solusi yang menarik, tetapi juga menimbulkan tantangan dalam memastikan bahwa tingkat kesulitan dan kualitas gameplay tetap konsisten. Ada risiko bahwa level yang dihasilkan secara acak mungkin tidak selalu memberikan pengalaman permainan yang optimal atau menantang.

3) **Mempertahankan Daya Tarik Jangka Panjang**

Dengan meningkatnya persaingan di pasar *game 2D platformer*, penting bagi pengembang untuk tidak hanya fokus pada menawarkan tantangan baru, tetapi juga menjaga daya tarik dan keberlanjutan game dalam jangka panjang. Ini memerlukan inovasi dalam desain *level*, mekanika permainan, dan penyesuaian kesulitan yang berkelanjutan.

3.2.2 Analisis Kebutuhan

Berdasarkan analisis masalah yang telah disajikan, berikut adalah beberapa kebutuhan yang dapat diidentifikasi untuk meningkatkan pengalaman bermain *"Lost Labyrinths: Rogue’s Odyssey"* yang menerapkan *difficulty adjustment* dan mengatasi tantangan yang dihadapi dalam genre *2D platformer* secara umum:

1) **Penyesuaian Kesulitan yang Dinamis**

Diperlukan sistem *difficulty adjustment* yang dinamis yang dapat mengukur kemampuan dan keterampilan pemain secara akurat sepanjang permainan. Ini memungkinkan tingkat kesulitan yang disesuaikan secara otomatis untuk mempertahankan tantangan yang memadai, tanpa membuat pemain merasa terlalu mudah atau terlalu sulit.

2) **Desain Level yang Dinamis**

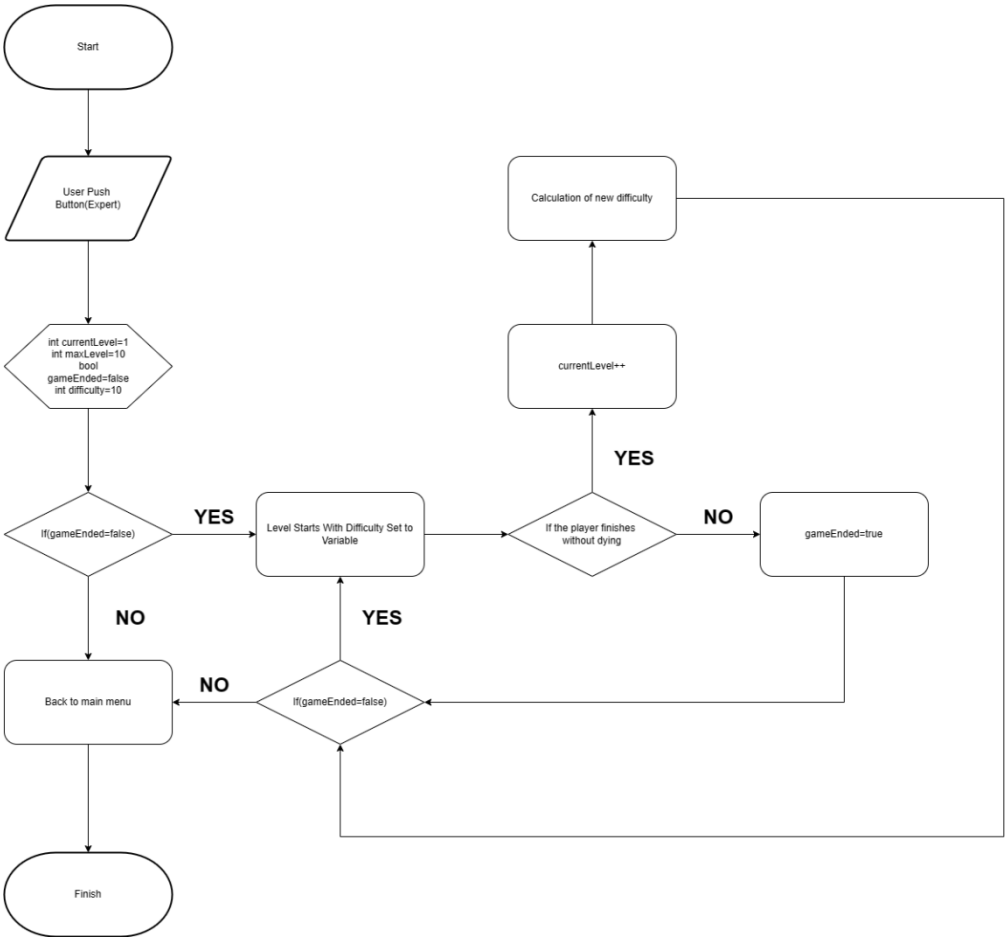
Diperlukan desain level yang dinamis dan bervariasi untuk menjaga keberagaman pengalaman permainan. Ini mencakup penggunaan PCG untuk membuat level secara acak, tetapi juga mempertimbangkan aspek-aspek seperti keseimbangan antara tantangan, distribusi aset, dan tingkat kesulitan yang berkembang seiring dengan kemajuan pemain.

3) **Penambahan Mekanika Permainan**

Mekanika permainan tambahan perlu ditambahkan untuk memberikan variasi dan kedalaman pada gameplay. Ini bisa berupa elemen rintangan baru, keterampilan khusus untuk karakter, atau fitur-fitur baru yang memengaruhi tingkat kesulitan. Penggunaan mekanika permainan yang inovatif dapat meningkatkan daya tarik permainan dan memberikan pengalaman bermain yang lebih menarik

3.3. Perancangan Arsitektur Sistem

Pada tahapan ini terdapat gambaran rencana pengerjaan kerja *system* pengembangan *game "Lost Labyrinths : Rogue's Odyssey"* dengan menggunakan *difficulty adjustment* sebagai pembuatan *mode game* yang baru. Berikut adalah *flowchart* yang dibuat dapat dlihat pada gambar 3. 2 di bawah ini.



Gambar 3. 2 *Flowchart* Arsitektur Sistem

3.3.1. Pengaruh Difficulty

Level dimulai dengan *difficulty* yang sudah di *set* divariabel yang bertipe data integer dengan range *difficulty* 0-100. *Difficulty* dapat mempengaruhi kemunculan jumlah ruangan, jumlah *obstacle*, jumlah dan *variant* dari musuh, jumlah ruangan *treasure*, dan *rarity* kemunculan *item* yang muncul dalam *shop*. Semakin tinggi *difficulty* maka semakin besar jumlah yang dipengaruhi.

3.3.2. Kalkulasi Difficulty Baru

Kalkulasi penilaian seberapa baik pemain bermain dalam *level* yang dilalui. Penilaian akan dibagi dalam 3 aspek:

1. *Item* yang didapatkan dari *shop* atau *treasure*
2. Jumlah dan tipe musuh yang dikalahkan

3. *Damage* yang diterima *player* selama didalam *level*

Dalam ketiga aspek tersebut terdapat kalkulasi yang harus dibuat:

a. Kalkulasi Penilaian *Item*

Setiap *item* yang memiliki *rarity* dan setiap *rarity* memiliki bobot penilaian yang berbeda

Tabel 3. 1 Tabel *Multiplier Rarity Item*

Rarity	***Multiplier***
Common	1
Rare	1.5
Legend	2.5

Pada tabel 3.1 di atas, penilaian dilakukan dengan membandingkan nilai *item* yang didapatkan dalam *level* dengan maksimum nilai *item* yang mungkin didapatkan dalam *level* tersebut. Terdapat 3 rumus dalam penilaian ini sebagai berikut:

1. $maxTreasureScore = treasureRoomCount * 2{,}5$

(1)

2. $maxShopScore = 3 * 2{,}5 = 7{,}5$

(2)

3 memiliki arti terdapat 3 *item* dalam 1 *shop*

2,5 didapat dari maksimal *rarity item shop*

3. $maxItemScore = maxTreasureScore + maxShopScore$

(3)

maxTreasureScore : Maksimal nilai *score* harta karun yang didapatkan

treasureRoomCount : Jumlah ruangan harta karun yang ada pada *level*

maxShopScore : Maksimal nilai *score shop* atau tempat membeli barang yang didapatkan

maxItemScore : Maksimal nilai *score item* yang didapatkan

Selanjutnya pengkalkulasian nilai *item* yang didapatkan oleh pemain dengan rumus sebagai berikut:

$$itemS = (commonIG * 1) + (rareIG * 1{,}5) + (legendIG * 2{,}5) \quad (4)$$

itemScore(itemS) : Nilai *score item* yang didapatkan
commonItemGet(commonIG) : Jumlah *item common* yang didapatkan
rareItemGe(rareIG) : Jumlah *item rare* yang didapatkan
legendItemGet(legendIG) : Jumlah *item legend* yang didapatkan

Untuk melakukan penilaian seberapa baik *player* mengumpulkan *item* dalam *level* dengan rumus sebagai berikut:

$$itemScorePercentage = \frac{itemScore}{maxItemScore} \quad (5)$$

itemScorePercentage(0-1) : Persentase *score* dari *item* yang hanya bernilai diantara 0 sampai 1

b. Kalkulasi Penilaian Musuh yang Dikalahkan

Setiap musuh memiliki *type* dan setiap *type* memiliki bobot penilaian yang berbeda

Tabel 3. 2 Tabel *Multiplier Type* Musuh

Type	***Multiplier***
Normal	1
Elite	1.5
Boss	2.5

Pada tabel 3. 2 di atas, penilaian dilakukan dengan membandingkan nilai musuh yang dikalahkan dalam *level* dengan jumlah

maksimal musuh yang ada dalam *level* tersebut. Untuk mengetahui nilai maksimum musuh yang dapat dibunuh didapatkan rumus sebagai berikut:

$$maxES = (enemyNC * 1) + (enemyEC * 1{,}5) + (enemyBC * 2{,}5) \quad \textbf{(6)}$$

maxEnemyScore(maxES) : Maksimal nilai *score* musuh yang bisa didapatkan
enemyNormalCount(enemyNC) : Jumlah musuh *normal* yang ada pada *level*
enemyEliteCount(enemyEC) : Jumlah musuh *elite* yang ada pada *level*
enemyBossCount(enemyBC) : Jumlah musuh *boss* yang ada pada *level*

Selanjutnya pengkalkulasian total nilai musuh yang dibunuh oleh pemain dengan rumus sebagai berikut:

$$enemyS = (enemyNK * 1) + (enemyEK * 1{,}5) + (enemyBK * 2{,}5) \quad \textbf{(7)}$$

enemyScore(enemyS) : Nilai *score* musuh yang dikalahkan
enemyNormalKilled(enemyNK) : Jumlah musuh *normal* yang dikalahkan
enemyEliteKilled(enemyEK) : Jumlah musuh *elite* yang dikalahkan
enemyBossKilled(enemyBK) : Jumlah musuh *boss* yang dikalahkan

Untuk menilai seberapa baik *player* membunuh musuh dalam *level* dengan rumus sebagai berikut:

$$enemyKP = \frac{enemyS}{maxES} \quad \textbf{(8)}$$

EnemyKilledPercentage(enemyKP) (0-1) : Persentase *score* dari mengalahkan musuh yang hanya bernilai diantara 0 sampai 1

c. Kalkulasi Penilaian *Damage* yang Diterima *Player*

Untuk melakukan penilaian seberapa baik *player* menghindari *damage* dari musuh dalam *level* digunakan rumus sebagai berikut:

$$damageTP = \frac{currentH}{maxH} \quad \textbf{(9)}$$

damageTakenPercentage(damageTP) (0-1) : Persentase *damage* yang diterima *player* dari musuh

currentHealth(currentH) : Sisa darah atau *HP player* setelah menyelesaikan level

maxHealth(maxH) : Maksimal darah atau *HP player*

Dari ketiga penilaian atau aspek tersebut dapat dikalkulasikan *difficulty* baru untuk *level* berikutnya. Terdapat variabel yang akan diset untuk menentukan *difficulty* baru sebagai berikut:

Variabel = *maxLevel* = 10

maxDifficulty = 100

maxLevel : Maksimal *level* yang ada pada mode

maxDifficulty : Maksimal nilai *difficulty* yang ada pada mode

Variabel ini akan digunakan untuk mengkalkulasi *difficulty* baru dengan menggunakan rumus sebagai berikut:

$$\Delta levelD = \left(\frac{enemyKP + itemSP + damageTP}{3} + 0.5\right) * \left(\frac{100 - currentLD}{maxL - currentL}\right) \quad \textbf{(10)}$$

ΔlevelDifficult(ΔlevelD): Jumlah nilai *difficulty* yang didapatkan pada *level* yang sudah diselesaikan yang akan dipakai untuk menentukan *difficulty* untuk *level* selanjutnya

currentLevelDifficulty(currentLD) : Nilai *difficulty* untuk *level* yang diselesaikan atau yang sedang dijalankan.

currentLevel(currentL) : *Level* yang sedang dijalankan

Selanjutnya pengkalkulasian *difficulty* baru akan dilanjutkan dengan rumus:

$$newLD = currentLD + \Delta levelD \quad (11)$$

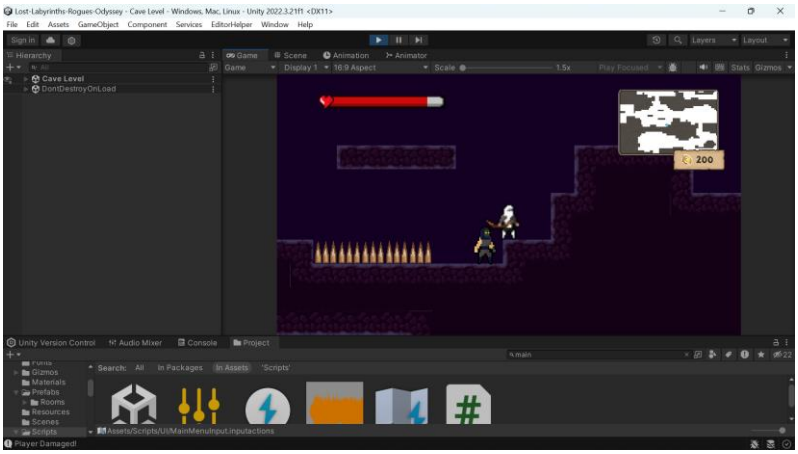
newLevelDifficulty(newLD) : Nilai *difficulty* untuk *level* yang selanjutnya

3.3.3. Tipe dan Varian Musuh

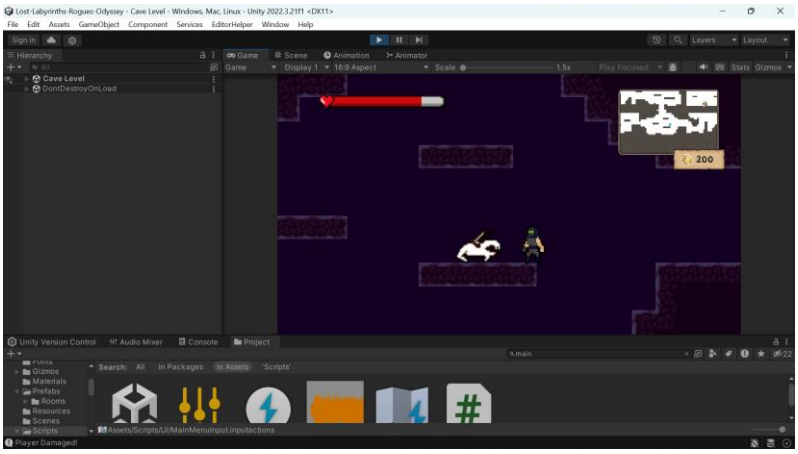
Dalam penelitian ini terdapat beberapa tipe musuh yang akan dijumpai dalam *level* seperti.

1. Musuh Normal

   Musuh tipe ini adalah musuh yang paling sering muncul dan paling mudah dikalahkan. Tipe ini juga tidak memiliki efek unik dalam serangannya. Didalam game ini terdapat 2 musuh normal yaitu *archer* yang bisa mengeluarkan serangan jarak jauh dan dekat, lalu terdapat anjing untuk serangan jarak dekat saja. Musuh tipe normal ini ditandai dengan warna putih. Kedua tipe ini ditampilkan pada gambar 3. 3 dan gambar 3. 4 di bawah ini.



Gambar 3. 3 Contoh Musuh *Normal Archer* di *Unity*



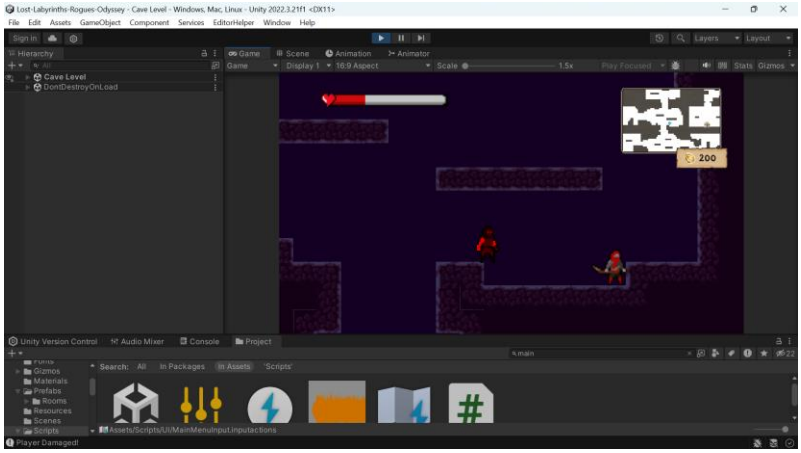
Gambar 3. 4 Contoh Musuh *Normal* Anjing di *Unity*

2. Musuh *Elite*

Musuh tipe ini adalah variasi dari tipe normal yang memiliki efek unik dalam serangannya. Untuk setiap varian dari *type* normal *archer* dan anjing terdapat 2 masing-masing sebagai berikut.

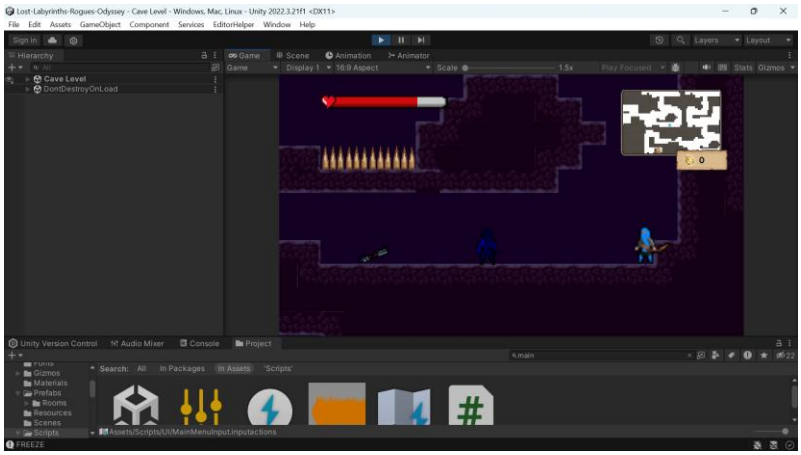
a. *Archer* :

- *archer* merah = warna merah pada *archer elite* memiliki efek *burn* terhadap musuh yang terkena serangannya dan merubah warna musuh tersebut menjadi merah. *Archer* merah dapat dilihat pada gambar 3. 5 dibawah ini.



Gambar 3. 5 Contoh Musuh *Elite Archer* Merah di *Unity*

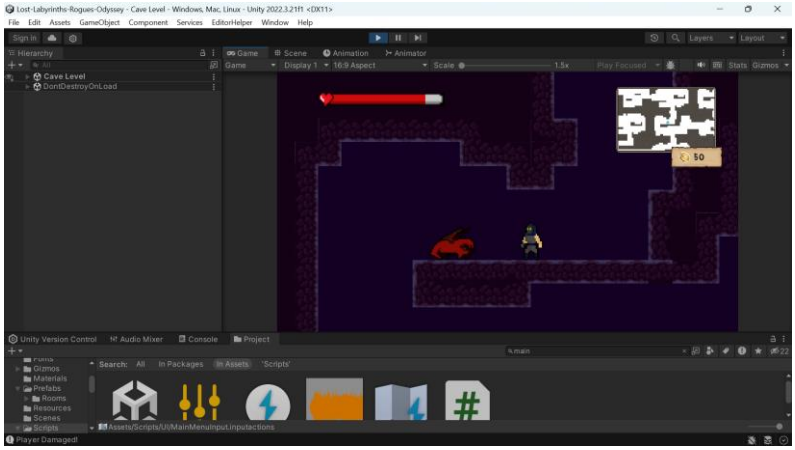
- *Archer* biru = warna biru pada *archer elite* memiliki efek *slow* terhadap musuh yang terkena serangannya dan merubah warna musuh tersebut menjadi warna biru. *Archer* biru dapat dilihat pada gambar 3. 6 dibawah ini.



Gambar 3. 6 Contoh Musuh *Elite Archer* Biru di *Unity*

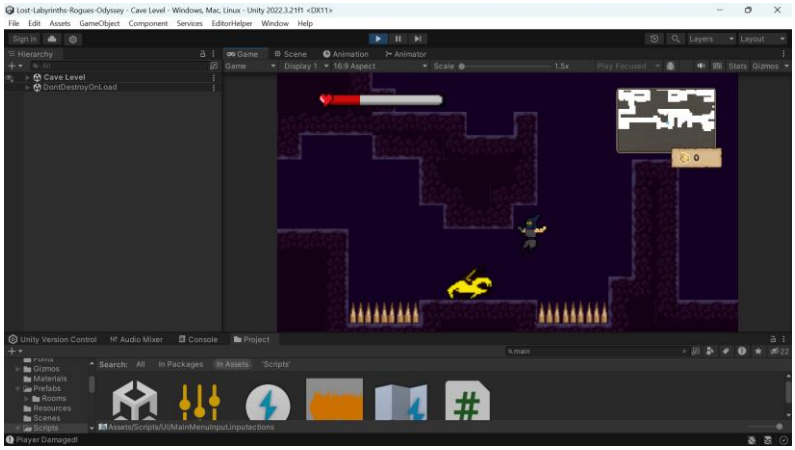
b. Anjing :

- Anjing merah = warna merah pada anjing *elite* memiliki efek *bleed* terhadap musuh yang terkena serangannya. Anjing merah dapat dilihat pada gambar 3. 7 dibawah ini.



Gambar 3. 7 Contoh Musuh *Elite* Anjing Merah di *Unity*

- Anjing kuning = warna kuning pada anjing *elite* memiliki efek *stun* pada musuh yang terkena serangannya. Anjing kuning dapat dilihat pada gambar 3. 8 dibawah ini.



Gambar 3. 8 Contoh Musuh *Elite* Anjing Kuning di *Unity*

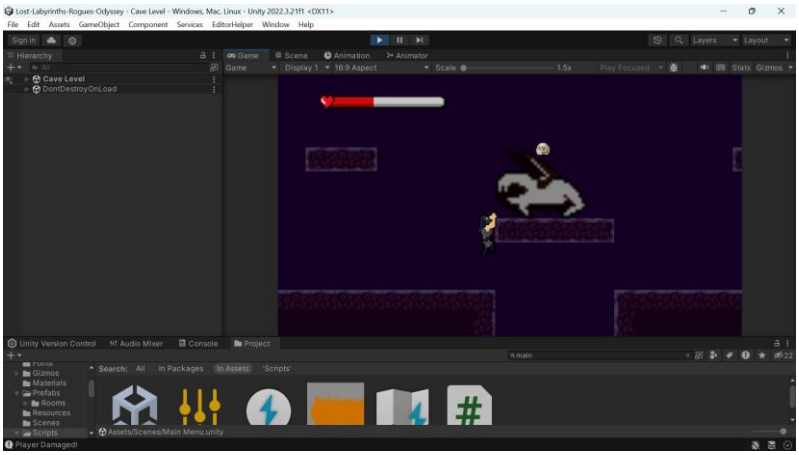
Tabel 3. 3 di bawah ini menampilkan efek musuh elite

Tabel 3. 3 Tabel Efek Musuh *Elite*

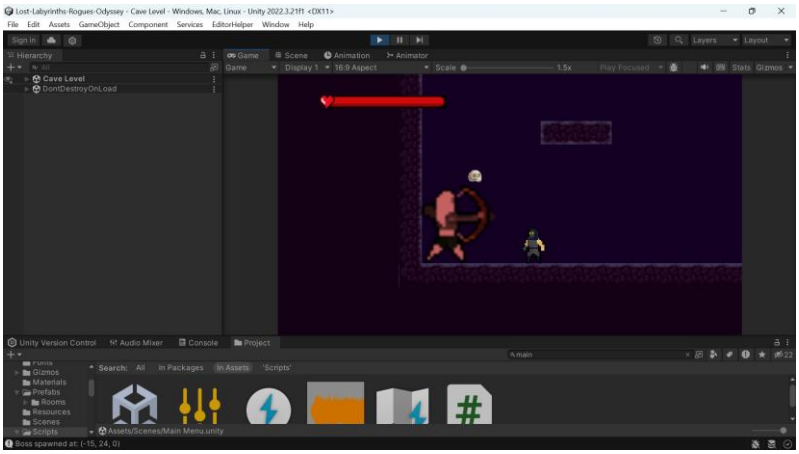
Effect	***Duration***	***Damage***
Burn	5 detik	3
Slow	4 detik	0
Bleed	6 detik	2
Stun	1 detik	0

3. Musuh *Boss*

Musuh tipe ini adalah variasi dari musuh normal juga tetapi dengan *hp* yang lebih banyak dan damage yang diberikan juga lebih besar. Kemudian ukuran dari musuh *boss* juga lebih besar dari musuh *normal*. Musuh *boss* ini hanya berjumlah 1 pada setiap *level*nya dan juga syarat untuk menuju ke *level* berikutnya diwajibkan harus membunuh musuh *boss* yang ada. Tipe musuh *boss* ini seperti musuh *normal* terdapat *archer* dan juga anjing yang akan diacak setiap *level*nya. Tipe ini digambarkan pada gambar 3. 9 dan gambar 3.10.



Gambar 3. 9 Contoh Musuh *Boss* Anjing di *Unity*

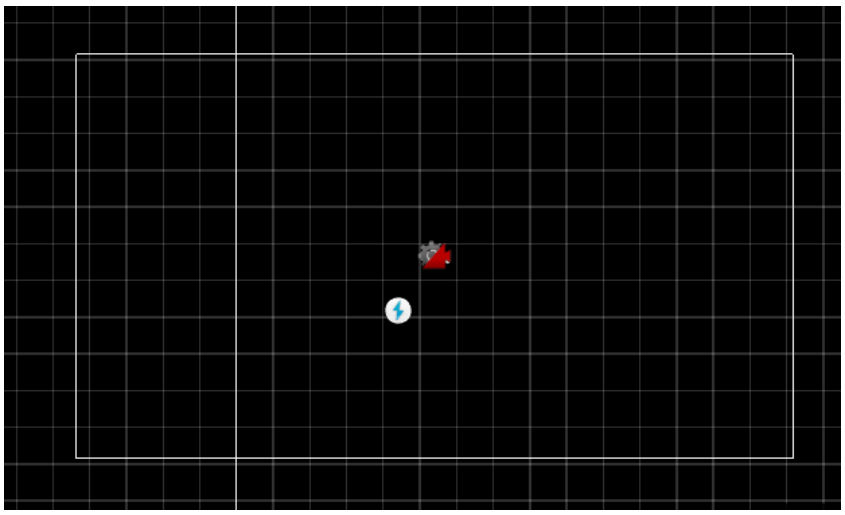


Gambar 3. 10 Contoh Musuh *Boss Archer* di *Unity*

3.3.4. *Fog of War*

Untuk implementasi *fog of war* akan dibantu oleh *grid* dan *tilemap*. *Grid* adalah sebuah alat bantu peletakan *tile* dalam sebuah koordinat. *Tilemap* adalah *layer*

peletakan *tile* pada *grid*. Dipenelitian ini dibuat sebuah *tilemap* yang diberinama *map tilemap*. *Tilemap* tersebut digunakan untuk meletakkan *tile* putih diruangan yang bukan tembok yang akan digunakan sebagai tampilan peta atau *map* pada game. *Grid* dapat dilihat pada gambar 3. 11 di bawah ini.

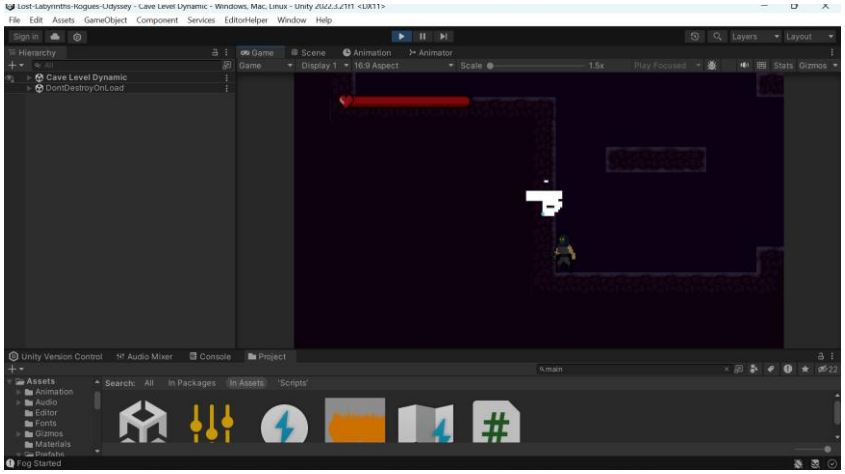


Gambar 3. 11 Contoh Gambar *Grid* di *Unity*

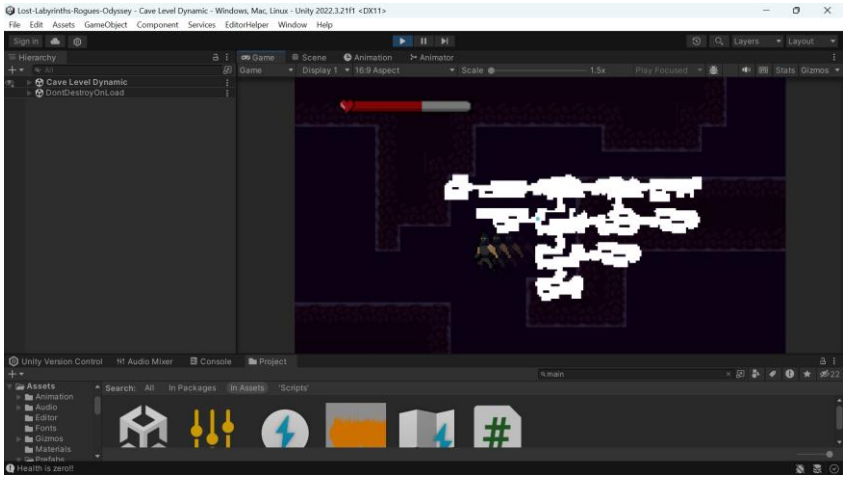
Pada gambar 3. 12 dan gambar 3. 13 di bawah ini menggambarkan implementasi *fog of war* dilakukan dengan beberapa langkah sebagai berikut:

1. Sembunyikan semua *tile* putih dalam *tilemap* diawal *level*
2. *Tracking* lokasi *player* setiap saat
3. Menentukan radius *fog of war*
4. Memunculkan *tile* dalam *map tilemap* dengan radius yang sudah ditentukan sebelumnya disekitar pemain.

Untuk implementasi dalam bentuk codingan programnya akan diletakkan dilampiran.



Gambar 3. 12 Contoh *Map* yang Masih Tertutup Fitur "*Fog of War*" di *Unity*



Gambar 3. 13 Contoh *Map* yang Sudah Terbuka setelah Menjelajahi *Level* di Unity\

3.4. Rencana Pengujian

Pengujian akan dilakukan dengan membuat kuisioner memakai *google form* dengan *target playtester* yang sering bermain atau berpengalaman dengan *video game* dan telah memainkan *mode game* yang dibuat oleh peneliti. Pertanyaan dalam kuisioner akan dibagi menjadi 3 kategori. Yang pertama, kategori yang akan membahas fitur atau aspek *game* yang berhubungan dengan *difficulty adjustment*. Yang kedua meliputi kategori tentang keefektifan penambahan *fog of war*. Dan yang

terakhir tentang kesan pengalaman pemain terhadap keseluruhan *gameplay* permainan.

BAB IV
IMPLEMENTASI DAN PENGUJIAN SISTEM

4.1. *Mode Game "Expert"* pada *Game "Lost Labyrinths : Rogue's Odyssey"*

Mode game "Expert" pada game ini adalah mode game yang dikembangkan oleh peneliti dengan menambahkan *difficulty adjustment* untuk memberikan pengalaman bermain lebih seimbang kepada pemain pemula dan pemain yang sudah berpengalaman. *Mode game* ini dibuat untuk meningkatkan mekanika *game* agar menjadi lebih menantang juga.

Gambar 4. 1 Tampilan *Main Menu* Mode *Game "Expert"*

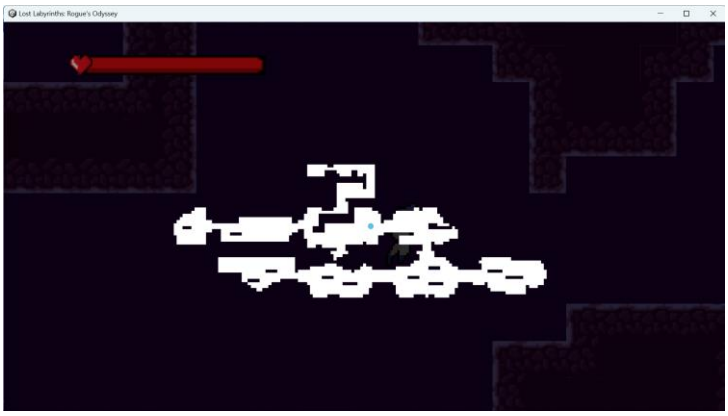
Gambar 4. 1 di atas menampilkan tampilan *menu* pada *game* dengan penambahan *mode game "Expert"*. Diawal *mode game "Expert"* ini dimulai, *level* akan dibuat secara acak dengan tingkat kesulitan yang rendah untuk membuat pemain mempelajari mekanika dalam memainkan *game* ini terlebih dahulu. Biasanya di*level* awal pemain hanya akan menghadapi musuh dengan tipe *normal* saja dan hanya ada beberapa *obstacle* yang tersedia didalam *level*. Berbeda dengan *mode* biasa, di*mode* ini terdapat fitur *fog of war* untuk menambahkan tantangan bermain.

Gambar 4. 2 Tampilan di Awal *Mode Game "Expert"* saat Map di Pojok Kanan Masih Tertutup Fitur *"Fog of War"*

Gambar 4. 3 Tampilan di Awal *Mode Game "Expert"* saat Map Diperbesar dan Masih Tertutup Fitur *"Fog of War"*

Seperti pada gambar 4. 2 dan 4. 3 diawal *level*, *map* akan tertutup dan *player* diharuskan menjelajahi *level* sendiri agar *map* terbuka berdasarkan arah *player*. Setelah *player* menjelajahi *map* maka *map* akan terbuka dengan sendirinya seperti pada gambar 4. 4 dan 4. 5.

Gambar 4. 4 Tampilan Mode *Game"Expert"* dengan *Map* Pojok Kanan yang Sudah Terbuka Karena Sudah Dijelajahi



Gambar 4. 5 Tampilan Mode *Game "Expert"* dengan *Map* yang Diperbesar Sudah Terbuka Karena Sudah Dijelajahi

Besarnya ruangan pada *level* akan bertambah seiring kenaikan *level* yang juga dipengaruhi oleh *difficulty* yang sudah disesuaikan terhadap performa pemain menggunakan *difficulty adjustment* . Contoh besar ruangan dengan *difficulty* yang berbeda dapat dilihat pada gambar 4. 6 dan 4. 7.

Gambar 4. 6 Tampilan *Map* yang Diperbesar dengan Contoh *Difficulty* 50

Gambar 4. 7 Tampilan *Map* yang Diperbesar dengan Contoh *Difficulty* 99

Dalam *mode game* ini terdapat beberapa tipe musuh dan varian yang dapat dijumpai. Dalam game ini terdapat musuh *archer* yang menyerang dari jarak jauh dan dekat dan juga ada musuh anjing yg menyerang dari jarak dekat saja .Dalam game ini terdapat 3 tipe musuh yaitu musuh *normal*, musuh *elite*, dan musuh *boss*, biasanya musuh *normal* akan dijumpai pada *level* awal game dimulai. Kemudian musuh *elite* akan dijumpai dipertengahan *level* seperti *level* 3 keatas. Tetapi itu semua tergantung kepada performa pemain dalam memainkan *level* yang dilewati semakin bagus performa pemain semakin sulit juga *level* yang akan dilewati selanjutnya. Musuh *elite* memiliki beberapa varian diantaranya, *archer* merah yang memiliki efek *burn*, *archer* biru yang memiliki efek *slow*, anjing merah yang memiliki efek *bleed*, dan anjing kuning yang memiliki efek *stun*. Kemudian disetiap level terdapat musuh boss yang menjadi tujuan atau syarat untuk menyelesaikan *level* tersebut. Contoh musuh unik

atau musuh yang memiliki efek spesial bisa dilihat pada gambar 4. 8, 4. 9, 4. 10, dan 4. 11

Gambar 4. 8 Tampilan Variasi Musuh *Elite Archer* yang Memiliki Efek *Burn*

Gambar 4. 9 Tampilan Variasi Musuh *Elite Archer* yang Memiliki Efek *Freeze*

Gambar 4. 10 Tampilan Variasi Musuh *Elite* Anjing dengan Efek *Bleed*

Gambar 4. 11 Tampilan Variasi Musuh *Elite* Anjing dengan Efek *Stun*

4.2. Hasil Pengujian oleh Sampel

Peneliti memberikan akses kepada 9 orang untuk memainkan permainan yang sudah dibuat oleh peneliti, Semua anggota sampel sudah terbiasa dan berpengalaman dalam bermain *video game*. Setelah sampel mencoba bermain game sampai melihat semua fitur yang telah dikembangkan oleh peneliti, sampel diberikan kuisoner tentang pengalaman bermain mereka dan juga fitur yang ada pada *mode game "Expert"* yang sudah diimplementasikan metode *difficulty adjustment* pada pembuatannya.

Kuisoner kategori pertama membahas tentang aspek *game* yang berhubungan dengan pengimplementasian *difficulty adjustment* pada *mode game "Expert"*

Tabel 4. 1 Tabel Hasil Kuisioner Aspek *Game* yang Berhubungan dengan *Difficulty Adjustment*

Aspek *Game*	**Hasil**					**Total Poin**	**Rata Rata**	**Kesimpulan**
	Sangat Baik (5)	**Baik (4)**	**Cukup (3)**	**Buruk (2)**	**Sangat Buruk (1)**			
Transisi kesulitan tiap *level*	7	2	0	0	0	43	4.778	Sangat Baik
Keseimbangan tingkat kemunculan *item* dari *shop* dan *treasure*	1	5	3	0	0	34	3.778	Baik
Keseimbangan tingkat kemunculan jumlah dan tipe musuh	5	4	0	0	0	41	4.556	Sangat Baik
Variasi musuh baru	6	3	0	0	0	42	4.667	Sangat Baik

Dari tabel 4. 1 di atas tentang aspek *game* yang melibatkan *difficulty adjustment* pada pembuatannya bisa dilihat kesimpulannya menunjukan hasil sangat baik (5), kecuali aspek dibagian kemunculan *item* dari *shop* yang mendapatkan hasil baik(4).

Lalu dipertanyaan kuisioner kategori kedua membahas tentang keefektifan pengimplementasian *"Fog of War"* dalam *mode game "Expert"*.

Tabel 4. 2 Tabel Hasil Kuisioner Penilaian Keefektifan implementasi *"Fog Of War"*

Penilaian	Hasil					Total Poin	Rata Rata	Kesimpulan
	Sangat Efektif (5)	Efektif (4)	Biasa saja (3)	Tidak Efektif (2)	Sangat Tidak Efektif (1)			
"Fog of War" menambah tingkat tantangan	8	1	0	0	0	44	4.889	Sangat Efektif
"Fog of War" memberikan ketegangan antisipasi menjelajahi area dimap	7	2	0	0	0	33	4.667	Sangat Efektif

Dari hasil kuisioner yang kedua yang ditampilkan pada tabel 4. 2 di atas tentang keefektifan *"Fog of War"* memberikan hasil sangat efektif(5) yang menunjukan kesimpulan bahwa *"Fog of War"* menambah tantangan dan ketegangan dalam bermain.

Selanjutnya dipertanyaan kuisioner kategori ketiga membahas tentang kesan pemain terhadap keseluruhan permainan dalam *mode game"Expert"* yang ditampilkan pada tabel 4. 3 di bawah ini.

Tabel 4. 3 Tabel Hasil Kuisioner Penilaian Seberapa Menantang Mode *Game "Expert"*

Penilaian	Hasil					Total Poin	Rata Rata	Kesimpulan
	Sangat Menantang (5)	Menantang (4)	Biasa saja (3)	Tidak Menantang (2)	Sangat Tidak Menantang (1)			
Seberapa menantang mode game "Expert"	8	1	0	0	0	44	4.889	Sangat Menantang

Tabel 4. 4 Tabel Hasil Kuisioner Penilaian Tingkat Kepuasan secara Keseluruhan Mode *Game "Expert"*

Penilaian	Hasil					Total Poin	Rata Rata	Kesimpulan
	Sangat Puas (5)	Puas (4)	Biasa saja (3)	Tidak Puas (2)	Sangat Tidak puas (1)			
Tingkat kepuasan keseluruhan	8	1	0	0	0	44	4.889	Sangat Puas

Tabel 4. 4 di atas menunjukan kesan pemain dan kepuasan pemain terhadap *mode game "Expert"* yang memiliki hasil sangat menantang(5) dan sangat puas(5) pada kesan bermainnya.

Berdasarkan dari semua hasil ketiga kategori kuisioner ini, bisa disimpulkan bahwa *mode game "Expert"* yang mengimplementasikan *difficulty adjustment* pada game *"Lost Labyrinths :Rogue's Odyssey"* memiliki hasil yang sangat memuaskan dengan tingkat survey yang sangat tinggi nilai hasil keseluruhannya.

BAB V
KESIMPULAN DAN SARAN

5.1. Kesimpulan

Berdasarkan penelitian pengembangan metode *difficulty* secara *procedural* pada *game "Lost Labyrinth : Rogue's Odessey"* dan juga respon sampel dapat disimpulkan:

1) Dengan adanya implementasi *difficulty adjustment* pada *game "Lost Labyrinths : Rogue's Odyssey"* memberikan tantangan yang lebih bervariasi kepada pemain berdasarkan performaanya yang dibuktikan dengan respons positif dari sampel yang berpengalaman dalam bermain *video game.*
2) Aspek-aspek terkait dengan *difficulty adjustment* pada mode *game "Expert"* dinilai baik secara keseluruhan oleh pemain, menunjukkan bahwa penyesuaian tingkat kesulitan secara otomatis telah berhasil diterapkan dengan efektif.
3) Penggunaan fitur *"Fog of War"* dalam *mode game "Expert"* dinilai sangat efektif karena menambahkan elemen tantangan dan ketegangan yang menyegarkan dalam pengalaman bermain.
4) Kesan dan kepuasan pemain terhadap keseluruhan permainan dalam *mode game "Expert"* sangat positif, dengan tingkat kesulitan yang menantang memberikan pengalaman bermain yang lebih memuaskan.

5.2. Saran

Saran yang dapat diberikan berdasarkan hasil penelitian ini adalah:

1) Mengembangkan *mode* permainan tambahan atau variasi lainnya yang dapat menarik minat pemain dengan tingkat keterampilan yang beragam, sehingga memperluas basis penggemar dan meningkatkan daya tarik *game* secara keseluruhan.

2) Terus melakukan pengujian dan pembaruan secara berkala berdasarkan umpan balik dari pemain, sehingga *game "Lost Labyrinths: Rogue's Odyssey"* tetap relevan dan kompetitif di pasar *game 2D platformer* yang semakin berkembang.
3) *Game* yang dibuat masih bisa ditambahkan lagi seperti membuat *mode* cerita tambahan, menambahkan konten baru seperti karakter, musuh, atau rintangan baru untuk memperkaya pengalaman bermain dan menjaga keberagaman permainan

DAFTAR PUSTAKA

Hendrix, M., Bellamy-Wood, T., McKay, S., Bloom, V., & Dunwell, I. (2016). *Implementing Adaptive . IEEE*.

Mitre-Hernandez, H., Carrillo, R. C., & Lara-Alvarez, C. (2021). *Pupillary Responses for Cognitive Load Measurement to ClassifyDifficulty Levels in an Educational Video Game: Empirical Study*. *JMIR SERIOUS GAMES, 9*.

Muñoz, E. M. (2020). *Project ADD: Adaptative Difficulty Dungeons.*

Oprean, D., Gould, H., Riedel, N., & Larsen, S. (2023). *Collect That Coin: Efficacy Testing of Platformer Game Mechanics . Proceedings of the 17th European Conference on Game-Based Learning: ECGBL 2023*, 459-466.

Sekhavat, Y. A. (2017). *MPRL: Multiple-Periodic Reinforcement Learning for Difficulty Adjustment in Rehabilitation Games. IEEE.*

Sutoyo, R., Winata, D., Oliviani, K., & Supriyadi, D. M. (2015). *Dynamic Difficulty Adjustment in Tower Defence. International Conference on Computer Science and Computational Intelligence, 436-444.*

Watkins, R. (2016). *Procedural Content Generation for Unity Game Development*.

Wheat, D., Masek, M., Lam, C. P., & Hingston, P. (2016). *Modeling Perceived Difficulty in Game Levels.*

Wu, Z.-H., Lai, K., Lin, L.-A., Huang, M.-H., & Tai, W.-K. (2018). *Procedurally Generating Game Level with Specified. IEEE,* 71-78.

Zohaib, M. (2018). *Dynamic Difficulty Adjustment (DDA) in Computer Games: A Review.*

LAMPIRAN

Lampiran 1.

CODINGAN PROGRAM UNTUK *"FOG OF WAR"*

```
public class FogOfWar : MonoBehaviour
{
    private Player player;
    private Tilemap tilemap;
    private Dictionary<Vector3Int, TileBase>
originalTiles = new Dictionary<Vector3Int, TileBase>();
    private List<SpriteRenderer> mapLayerSprites = new
List<SpriteRenderer>();
    public int tileRadius = 10;
    public int iconRadius = 20;
    private bool isStarted = false;

    public void StartFog()
    {
        player = SessionManager.player;
        tilemap = GetComponent<Tilemap>();
        HideAllTiles();
        HideAllMapLayerSprites();
        isStarted = true;
        Debug.Log("Fog Started");
    }
    {
        if (isStarted)
        {
            UpdateVisibleTiles();
            UpdateVisibleMapLayerSprites();
```

```
        }
    }

    void HideAllTiles()
    {
        BoundsInt bounds = tilemap.cellBounds;
        foreach (Vector3Int pos in
bounds.allPositionsWithin)
        {
            TileBase tile = tilemap.GetTile(pos);
            if (tile != null)
            {
                originalTiles[pos] = tile;
                tilemap.SetTile(pos, null);
            }
        }
    }

    void HideAllMapLayerSprites()
    {
        SpriteRenderer[] spriteRenderers =
FindObjectsOfType<SpriteRenderer>();
        foreach (SpriteRenderer renderer in
spriteRenderers)
        {
            if (renderer.gameObject.layer ==
LayerMask.NameToLayer("Map"))
            {
                mapLayerSprites.Add(renderer);
                renderer.enabled = false;
            }
        }
    }
```

```
    void UpdateVisibleTiles()
    {
        BoundsInt bounds = tilemap.cellBounds;
        foreach (Vector3Int pos in
bounds.allPositionsWithin)
        {
            if  (Vector3.Distance(tilemap.CellToWorld(pos),
player.transform.position) <= tileRadius)
            {
                if (originalTiles.ContainsKey(pos))
                {
                    tilemap.SetTile(pos,
originalTiles[pos]);
                }
            }
        }
    }

    void UpdateVisibleMapLayerSprites()
    {
        List<SpriteRenderer> destroyedSprites = new
List<SpriteRenderer>();
        foreach (SpriteRenderer spriteRenderer in
mapLayerSprites)
        {
            if (spriteRenderer == null ||
spriteRenderer.Equals(null))
            {
                destroyedSprites.Add(spriteRenderer);
                continue;
            }
```

```
            if
(Vector3.Distance(spriteRenderer.transform.position,
player.transform.position) <= iconRadius)
            {
                spriteRenderer.enabled = true;
            }
            else
            {
                spriteRenderer.enabled = false;
            }
        }
main list
        foreach (SpriteRenderer destroyedSprite in
destroyedSprites)
        {
            mapLayerSprites.Remove(destroyedSprite);
        }
    }
}
```

Lampiran 2.

KUISIONER ASPEK *GAME* YANG BERHUBUNGAN DENGAN *DIFFICULTY ADJUSTMENT* DALAM *MODE "EXPERT"* PADA *GAME "LOST LABYRINTHS : ROGUE'S ODESSEY"*

Sangat Buruk = 1, Buruk = 2, Cukup = 3, Baik = 4, Sangat Baik = 5

No.	Pertanyaan	1	2	3	4	5
1	Bagaimana penilaian anda terhadap transisi kesulitan tiap *level* dalam *mode game "Expert"* pada *game "Lost Labyrinths : Rogue's Odyssey"* ?					
2	Bagaimana penilaian anda terhadap keseimbangan tingkat kemunculan *item* yang didapatkan dari *shop* dan *treasure* dalam *mode game "Expert"* pada *game "Lost Labyrinths : Rogue's Odyssey"* ?					
3	Bagaimana penilaian anda terhadap keseimbangan tingkat kemunculan jumlah dan tipe musuh dalam *mode game "Expert*" pada *game "Lost Labyrinths : Rogue's Odyssey"* ?					
4	Bagaimana penilaian anda terhadap variasi musuh baru,seperti *archer* biru dan merah, dan anjing kuning dan merah dalam *mode game "Expert"* pada *game "Lost Labyrinths : Rogue's Odyssey"* ?					

Lampiran 3.

KUISIONER PENILAIAN KEEFEKTIFAN IMPLEMENTASI *"FOG OF WAR"*

Sangat Tidak Efektif = 1, Tidak Efektif = 2, Biasa saja = 3, Efektif = 4, Sangat Efektif = 5

No.	Pertanyaan	1	2	3	4	5
5	Seberapa efektif menurut anda *"Fog of War"* dalam menambah tingkat tantangan dalam *mode game "Expert"* pada *game "Lost Labyrinths : Rogue's Odyssey"*?					
6	Seberapa efektif menurut anda *"Fog of War"* dalam memberikan ketegangan antisipasi yang anda rasakan saat menjelajahi area yang belum terungkap dalam *mode game "Expert"* pada *game "Lost Labyrinths : Rogue's Odyssey"*?					

Lampiran 4.

KUISIONER PENILAIAN SEBERAPA MENANTANG MODE GAME"EXPERT" DAN PENILAIAN TINGKAT KEPUASAN SECARA KESELURUHAN MODE GAME "EXPERT"

Sangat Tidak Menantang/Puas = 1, Tidak Menantang/Puas = 2, Biasa saja = 3, Menantang/Puas = 4, Sangat Menantang/Puas = 5

No.	Pertanyaan	1	2	3	4	5
7	Seberapa menantang menurut anda *mode* permainan *"Expert"* dalam *game "Lost Labyrinths: Rogue's Odyssey"*?					
8	Bagaimana tingkat kepuasan anda terhadap keseluruhan pengalaman bermain dalam *mode "Expert"* pada *game "Lost Labyrinths : Rogue's Odyssey"*?					

Lampiran 5.

TABULASI JAWABAN RESPONDEN

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

Lampiran 6.

BIODATA

Nama	: Muhammad Raihandi Jamal Ritonga
Tempat, Tanggal Lahir	: Medan, 12 September 2002
Alamat	: Jl. Abd. Sani Muthalib Medan Marelan, Kota Medan
Telpon/HP	: 083140210881
Email	: raihanritonga170882@gmail.com
Agama	: Islam
Nama Orang Tua	
Ayah	: Khairul Amri Ritonga
Ibu	: Endah Widyastuti
Jumlah Saudara	: 2
Anak Ke	: 1
Riwayat Pendidikan	: SDIT Alfauzi (2007 - 2009)
	SD Asuhan Raya (2009 - 2013)
	SMP Laksamana Marthadinata (2013 - 2016)
	SMK Tritech Informatika (2016 - 2019)
	Universitas Sumatera Utara (2020 - sekarang)