RANCANG BANGUN *GAME PLATFORMER* 2D *"FOXY TALES"* BERGENRE *ADVENTURE* MENGGUNAKAN *UNITY GAME ENGINE*

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat Kelulusan Program Strata I pada
Sekolah Tinggi Manajemen Informatika dan Komputer
(STMIK) Palangkaraya

OLEH

MUHAMMAD MAWANDI ANAJELI
C1655201045
PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TINGGI MANAJEMEN INFORMATIKA DAN KOMPUTER
(STMIK) PALANGKARAYA
2022

RANCANG BANGUN *GAME PLATFORMER* 2D *"FOXY TALES"* BERGENRE *ADVENTURE* MENGGUNAKAN *UNITY GAME ENGINE*

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat Kelulusan Program Strata I pada
Sekolah Tinggi Manajemen Informatika dan Komputer
(STMIK) Palangkaraya

OLEH

MUHAMMAD MAWANDI ANAJELI
C1655201045
PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TINGGI MANAJEMEN INFORMATIKA DAN KOMPUTER
(STMIK) PALANGKARAYA
2022

LEMBAR PERNYATAAN

Yang bertanda tangan di bawah ini:

Nama Mahasiswa	:	**MUHAMMAD MAWANDI ANAJELI**
N I M	:	**C1655201045**

menyatakan bahwa Tugas Akhir dengan judul :

RANCANG BANGUN *GAME PLATFORMER* 2D *"FOXY TALES"* BERGENRE *ADVENTURE* MENGGUNAKAN *UNITY GAME ENGINE*

adalah hasil karya saya dan bukan merupakan duplikasi sebagian atau seluruhnya dari karya orang lain, kecuali bagian yang sumber informasi dicantumkan.

Pernyataan ini dibuat dengan sebenar-benarnya secara sadar dan bertanggungjawab dan saya bersedia menerima sanksi pembatalan Tugas Akhir apabila terbukti melakukan duplikasi terhadap Tugas Akhir atau karya ilmiah lain yang sudah ada.

Palangka Raya, 25 Mei 2022

Yang Membuat Pernyataan,

MUHAMMAD MAWANDI ANAJELI

PERSETUJUAN

RANCANG BANGUN *GAME PLATFORMER* 2D *"FOXY TALES"* BERGENRE *ADVENTURE* MENGGUNAKAN *UNITY GAME ENGINE*

Tugas Akhir Ini Telah Disetujui Untuk Diujikan pada
Tanggal 23 Mei 2022

Pembimbing I	Pembimbing II
Lili Rusdiana, M.Kom	Catharina Elmayantie, M.Pd.
NIK 198707282011007	NIK 197610252015003

Mengetahui
Ketua STMIK Palangkaraya,

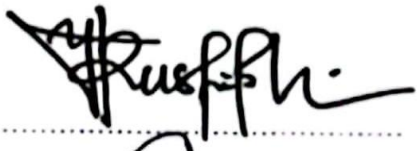
Suparno, M.Kom
NIK. 196901041995105

PENGESAHAN

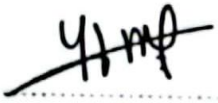
RANCANG BANGUN *GAME PLATFORMER* 2D *"FOXY TALES"* BERGENRE *ADVENTURE* MENGGUNAKAN *UNITY GAME ENGINE*

Tugas Akhir ini telah Diujikan Dinilai, dan Disahkan
Oleh Tim Penguji pada Tanggal 31 Mei 2022

Tim Penguji Tugas Akhir :

1. Sulistyowati, S.Kom., M.Cs.
   Ketua ..... 

2. Rommi Kaestria, M.Kom.
   Sekretaris ..... 

3. Veny Cahya Hardita, M.Kom.
   Anggota ..... 

4. Lili Rusdiana, M.Kom.
   Anggota ..... 

5. Catharina Elmayantie, M.Pd.
   Anggota ..... 

MOTTO DAN PERSEMBAHAN

Bahagia itu diciptakan bukan didapatkan

Ku persembahkan untuk :

- Skripsi ini adalah persembahan kecil saya untuk kedua orangtua saya. Ketika dunia menutup pintunya pada saya, ayah dan ibu membuka lengannya untuk saya. Ketika orang-orang menutup telinga mereka untuk saya, mereka berdua membuka hati untukku. Terima kasih karena selalu ada untukku.
- Skripsi ini saya persembahkan untuk teman dan sahabat yang selalu ada disisi saya. Saya bahkan tidak bisa menjelaskan betapa bersyukurnya saya memiliki kalian dalam hidup saya.
- Dengan penuh kesabaran, Para Dosen selalu membimbingku yang gemar melakukan kesalahan. Meski sering terdengar berang, tapi dirimu selalu rajin mengingatkanku untuk ikut bimbingan.

INTISARI

Muhammad Mawandi Anajeli, C1655201045, 2022. Rancang bangun *game platformer 2d foxy tales bergenre adventure* menggunakan *unity game engine*, Pembimbing I Lili Rusdiana, M.Kom., Pembimbing II Catharina Elmayantie, M.Pd.

Game platformer 2d foxy tales merupakan permainan yang bercerita tentang rubah bernama *Foxy*, seekor rubah yang hendak menyelamatkan pulau *Sunnyland* dari musuh – musuh yang menjajah pulau tersebut, tentu dalam aksi penyelamatannya *Foxy* harus melewati segala rintangan, memecahkan teka – teki , dan mengalahkan musuh – musuh yang menghadangnya hingga pulau *Sunnyland* pun aman dan tentram kembali.

Tujuan penelitian ini yaitu membuat rancang bangun *game platformer 2d Foxy Tales bergenre adventure* yang menarik dan juga cukup sulit untuk diselesaikan. Aplikasi ini dibuat menggunakan *Unity* dengan bahasa program *C#*

Metode yang digunakan pada tugas akhir ini adalah metode pengumpulan data studi pustaka, studi kuesioner, dan dokumentasi. Menggunakan metode penegembangan perangkat lunak model *MDLC*, serta menggukan metode *SWOT* untuk analisis sistem kelemahan.

Dari pengujian yang dilakukan, aplikasi berjalan pada *Windows* dan *Mac OS*, mudah digunakan oleh pengguna, dan tombol tampak jelas dan mudah dipahami di setiap *scenenya*. Semua fitur aplikasi bekerja dengan sempurna dan desain objek aplikasi menarik. Hasil dari pengguna yang disurvei adalah 94,44%

Kata Kunci: Aplikasi, *Adventure, Game, Foxy Tales, Platformer, Unity, 2D*

ABSTRACT

Muhammad Mawandi Anajeli, C1655201045, 2022. Design of a 2d foxy tales platformer game with adventure genre using the unity game engine, Advisor I Lili Rusdiana, M.Kom., Advisor II Catharina Elmayantie, M.Pd.

Platformer game 2d foxy tales is a game that tells the story of a fox named Foxy, a fox who wants to save the island of Sunnyland from enemies who colonize the island, of course in his rescue action Foxy must pass all obstacles, solve puzzles, and defeat enemies. who confronted him until the island of Sunnyland was safe and peaceful again.

The purpose of this research is to design a 2d Foxy Tales platformer game with an adventure genre that is interesting and also quite difficult to complete. This application is made using Unity with C# programming language.

The method used in this final project is the method of collecting data from literature study, questionnaire study, and documentation. Using the MDLC model software development method, and using the SWOT method for system weakness analysis.

From the tests carried out, the application runs on Windows and Mac OS, is easy to use by users, and the buttons are clear and easy to understand in every scene. All app features work perfectly and app object design is attractive. The results of the surveyed users are 94.44%

Keywords: Application, Adventure, Foxy Tales, Game, Platformer, Unity, 2D

KATA PENGANTAR

Puji syukur Alhamdulillah penulis panjatkan kehadirat Allah SWT atas berkat, rahmat, taufik, dan hidayah-Nya yang berjudul "Rancang Bangun Game Platformer 2D "Foxy Tales" Bergenre Adventure Menggunakan Unity Game Engine. Penyuntingan disertasi dapat diselesaikan dengan baik. Hingga akhir zaman selalu tercurah kepada Nabi besar Muhammad SAW beserta keluarga, sahabat, kerabat dan pengikutnya.

Tugas akhir ini disusun menjadi memenuhi kondisi pada memperoleh gelar Sarjana Komputer pada Sekolah Tinggi Managemen Informatika & Komputer (STMIK) Palangkaraya.

Selain itu, penulis mengucapkan terima kasih yang sebesar-besarnya kepada semua pihak yang telah membantu, membimbing dan memotivasi penulisan tugas akhir ini, yaitu kepada :

1. Suparno, M.Kom selaku Ketua Sekolah Tinggi Manajemen Informatika dan Komputer (STMIK) Palangkaraya.
2. Lili Rusdiana, M.Kom selaku Pembimbing I yang telah meluangkan waktunya untuk memberikan bimbingan dalam penyelesaian tugas akhir ini.
3. Catharina Elmayantie, M.Pd selaku dosen Pembimbing II dalam penulisan tugas akhir ini, yang juga banyak memberikan saran dan masukan dalam penulisan tugas akhir ini.
4. Terima kasih saya ucapkan kepada teman saya Rifa'I Al Amin yang berkenan membantu dalam pembuatan audio dari game yang saya rancang.

5. Seluruh responden dan penulis yang berpartisipasi dalam survey yang dilakukan mengharapkan kontribusi, saran, dan kritik yang membangun untuk penyempurnaan tugas akhir ini. Kami berharap tugas akhir ini dapat membantu penulis dan pembaca khususnya untuk memperdalam ilmunya.

Palangkaraya, 30 Mei 2022

Penulis

DAFTAR ISI

LEMBAR PERNYATAAN ii
PERSETUJUAN iii
PENGESAHAN iv
MOTTO DAN PERSEMBAHAN v
INTISARI vi
ABSTRACT vii
KATA PENGANTAR viii
DAFTAR ISI x
DAFTAR TABEL xii
DAFTAR GAMBAR xiii
DAFTAR LAMPIRAN xiv
BAB I 1
PENDAHULUAN 1
1.1 Latar Belakang Masalah 1
1.2 Rumusan Masalah 3
1.3 Batasan Masalah 3
1.4 Tujuan dan Manfaat 4
1.5 Sistematika Penulisan 5
BAB II 6
LANDASAN TEORI 6
2.1 Tinjauan Pustaka 6
2.1.1 Kajian Penelitian yang Relevan 6
2.2 Kajian Teori 9
2.2.1 *Game* 9
a. Jenis – jenis *game* 9
b. *Platformer* 12
c. Elemen – elemen *game* 13
2.2.2 *Game Unity Engine* 15
2.2.3 *C# (C Sharp)* 17
2.2.4 Teknik Pengumpulan Data 18
2.2.5 *Multimedia Development Life Cycle (MDLC)* 19
2.2.6 *Storyboard* 21

2.2.7 Skala *Likers* 21
BAB III 23
METODE PENELITIAN 23
3.1 Jenis Penelitian 23
3.2 Pengumpulan Data 23
3.2.1 Studi Pustaka 23
3.2.2 Studi Kuesioner 24
3.2.3 Studi Dokumentasi 24
3.3 Analisis Kebutuhan 24
3.3.1 Kebutuhan Perangkat Keras 24
3.3.2 Kebutuhan Perangkat Lunak 25
3.3.3 Analisis Proses 26
3.3.4 Analisis Kelemahan 27
3.4 Desain 29
3.4.1 *Storyline* 29
3.4.2 *Storyboard* 29
3.4.3 Desain Proses 31
3.4.4 Desain Perangkat Lunak 34
BAB IV 38
HASIL DAN PEMBAHASAN 38
4.1 Hasil 38
4.1.1 Implementasi Program 38
4.1.2 Manual Program 39
4.1.3 Pengujian 43
4.1.4 Manual Instalasi 47
4.2 Pembahasan 48
4.2.1 Pembahasaan Hasil Respon Pengguna 48
BAB V 52
KESIMPULAN DAN SARAN 52
5.2 Kesimpulan 52
5.2 Saran 53
DAFTAR PUSTAKA 54

DAFTAR TABEL

Tabel 1. Perbandingan Penelitian 6
Tabel 2. Kebutuhan Perangkat Keras 25
Tabel 3. Kebutuhan Perangkat Lunak 25
Tabel 4. *Storyboard Game* 29
Tabel 6. Pengujian Buka Aplikasi 43
Tabel 7. Pengujian Halaman Menu Utama 44
Tabel 8. Pengujian Halaman *New Game* 44
Tabel 9. Pengujian Halaman *About* 45
Tabel 10. Pengujian *Scene Victory* 46
Tabel 11. Skor Pilihan Jawaban 48
Tabel 12. Jawaban Responden 49

DAFTAR GAMBAR

Gambar 1. Konsep *The Magic Circles* 14
Gambar 2. Tahapan *Metode MDLC* 20
Gambar 3. *Use Case Diagram* 31
Gambar 4. *Activity Diagram Play* 32
Gambar 5. *Activuty Diagram Continue* 33
Gambar 6. *Activity Diagram About* 33
Gambar 7. *Activity Diagram Exit* 34
Gambar 8. Tampilan *Splash Screen* 35
Gambar 9. Tampilan *Menu Awal* 35
Gambar 10. Tampilan *Gamplay Game* 36
Gambar 11. Tampilan *Menu About* 36
Gambar 12. Tampilan *Scene Quiz Time* 37
Gambar 13. Sumber Assets Game. 39
Gambar 14. Tampilan *Splash Screen*. 39
Gambar 15. Tampilan Menu Utama 40
Gambar 16. Tampilan *Gameplay Foxy Tales* 40
Gambar 17. Tampilan *Quiz Time* 41
Gambar 18. Tampilan Halaman *About* 41
Gambar 19. Tampilan Halaman *Select Menu* 42
Gambar 20. Tampilan *Scene Victory* 42
Gambar 21. *Page Itch.io Foxy Tales* 47
Gambar 22. Hasil Kuesioner 51

DAFTAR LAMPIRAN

Lampiran 1. Lembar Konsultasi Pembimbing Tugas Akhir
Lampiran 2. Surat Tugas Akhir Penguji Sidang Tugas Akhir
Lampiran 3. Berita Acara Penilaian Sidang Tugas Akhir
Lampiran 4. Lembar Kuisioner
Lampiran 5. Listing Program

BAB I

PENDAHULUAN

1.1 Latar Belakang Masalah

Pada awal kemunculan *game* pertama kalinya, *game* masih disajikan secara sederhana proyek yang bernama *Computer Games* pada tahun 1962 dengan produk andalannya bernama *Star Wars*. Beberapa puluh tahun kemudian, banyak game bermunculan dengan 2 dimensi dan game 3 dimensi. Serta yang bersifat sebagai hiburan maupun bersifat sebagai media pembelajaran atau edukatif (Pratama, 2014). *Game* pada dasarnya bersifat huburan karena jika pengguna memainkan game maka akan terasa senang. Dalam era saat ini, game disajikan dengan kualitas visualisasi yang cukup canggih karena didukung oleh teknologi sehingga pemain lebih interaktif sesuai kemaunnya sendiri dan pemain terasa hidup dengan *game* tersebut. Maka bisa disebutkan bahwa *game* berkembang beriringan dengan teknologi.

Berdasarkan riset dari perusahaan *Quantic Foundry* yang berfokus dibidang pasar *game*. Motivasi seorang gamer dibagi menjadi 6 kelompok utama yang diantara masing – masing kelompok memiliki dua submotivasi serupa. Enam kelompok utama tersebut adalah *Action*, *Achievement*, *Creativity*, *Immersion*, dan *Mastery*. Dari keenam kategori ini mewakili motivasi paling umum Ketika seseorang bermain *game* (Yee, 2015).

Berbagai penelitian telah membuktikan efektifitas *game* dalam mempengaruhi pemainnya. Muatan atau konten yang berada pada *game*,

dengan mudah dipahami oleh pemain hal ini dikarenakan interaktifitas yang ada dalam *game*, sekaligus imersi yang diberikan oleh game menjadikan pemain kondisi yang paling rileks dan terbuka dalam menerima materi. Hal inilah yang dimanfaatkan oleh beberapa pengembangan *game* untuk mempengaruhi pemain, dalam artian positif maupun negatif. Kecepatan *game* dalam mempengaruhi pemain ini sangat terkait dengan perkembangan siswa didik di era teknologi seperti ini (Wibawanto, 2020).

Sebelumnya sudah pernah ada *game platformer* 2D yang bertemakan *adventure* yang popular yaitu *Mario Bros.* Mengacu pada *game* tersebut maka penulis terinspirasi membuat konsep yaitu dengan menambah fitur dalam *game*, membuat tampilan *game* lebih menarik lagi, dan menambahkan sisi edukasi dalam *game* berupa kuis. Konsep dari *game* ini sendiri lebih menonjolkan pada rintangan demi rintangan yang akan dilalui dan memecahkan teka – teki berupa kuis. Meskipun *game* ini pada dasarnya untuk hiburan semata, tetapi game ini juga memasukan unsur edukasi. Sehingga orang yang memainkan *game* ini tidak hanya bersenang – senang saja, tetapi game ini juga melatih ketangkasan dalam berpikir untuk menyelesaikan rintangan demi rintangan dan menyelesaikan kuis yang disajikan dalam game ini.

Berdasarkan pemaparan diatas, penulis memilih untuk mengembangan *game platformer* 2 dimensi yang berjudul **“ RANCANG BANGUN *GAME PLATFORMER* 2D *“FOXY TALES”* BERGENRE *ADVENTURE* MENGGUNAKAN *UNITY GAME ENGINE* ”**.

1.2 Rumusan Masalah

Masalah yang dapat dirumuskan berdasarkan latar belakang tersebut, adalah bagaimana cara merancang dan membangun *game platformer* 2d *"Foxy Tales"* Bergenre *Adventure* Menggunakan *Unity Game Engine* ?

1.3 Batasan Masalah

Pembatasan suatu masalah digunakan untuk menghindari adanya penyimpangan maupun pelebaran pokok masalah agar penelitian tersebut lebih terarah dan memudahkan dalam pembahasan sehingga tujuan penelitian akan tercapai. Adapun beberapa batasan masalah dalam penelitian ini adalah sebagai berikut :

1. *Game* hanya dapat dijalankan pada sistem komputer atau berbasis *desktop* dan tidak dapat dijalankan pada *smartphone*.
2. *Engine* yang digunakan adalah *Unity* dan menggunakan bahasa pemrograman *C#.*
3. Game tidak memerlukan koneksi internet atau menggunakan sistem *offline*.
4. Survei *game* yang dilakukan menggunakan kuesioner online.
5. Gaya *visual game* yang digunakan yaitu *pixel art*.
6. *Game* dirancang dengan menggunakan grafis 2D.
7. *Game* bersifat *single-player*.
8. Minimal *requirement* dari *game* ini adalah *OS Windows* 7, *prosessor* 2 *GHz*, memori 2 *GB RAM*, dan *graphics* 256 MB memori sedangkan

dengan *macOS Mac OSX* 10.10+, *prosessor* 2 *GHz*, memori 2 *GB RAM*, dan *graphics* 256 *MB* memori.

1.4 Tujuan dan Manfaat

a. Tujuan

Tujuan yang hendak dicapai oleh penulis dalam penelitian ini adalah untuk membangun game *platformer* 2d *"Foxy Tales"* Bergenre *Adventure* Menggunakan *Unity Game Engine*.

b. Manfaat

1) Bagi Penulis

Manfaat yang didapat penulis adalah bertambahnya wawasan dan pengalaman langsung tentang pembuatan game dan bisa mengimplementasikan ilmu yang telah dipelajari selama kuliah, serta sebagai syarat kelulusan Strata 1 (S1) Program Studi Teknik Informatika di STMIK Palangkaraya.

2) Manfaat bagi STMIK Palangkaraya

Sebagai penambah literatur pustaka pada perpustakaan STMIK Palangkaraya serta dapat digunakan sebagai salah satu referensi atau kajian untuk mahasiswa STMIK Palangkaraya yang ingin mengembangkan game platformer 2 dimensi bergenre adventure ataupun genre lain.

3) Manfaat bagi pengguna

Bertambahnya aplikasi game pilihan untuk dimainkan pada sistem computer, serta memberikan manfaat pengembangan diri dengan melatih resource management dengan bermain game dan sebagai sarana hiburan.

1.5 Sistematika Penulisan

Adapun sistematika penulisan dalam penulisan tugas akhir ini terdiri dari beberapa bab dan masing – masing bab membahas dan menguraikan pokok permasalahan yang berbeda, sebagai gambaran disini penulis menyertakan garis – garis besarnya yaitu :

BAB I PENDAHULUAN

Bab ini berisikan latar belakang masalah, rumusan masalah, batasan masalah, tujuan dan manfaat, serta sistematika penulisan.

BAB II LANDASAN TEORI

Bab ini berisikan tentang tinjauan pustaka yang diambil dari penelitian yang relevan beserta susunan kajian teori yang disesuaikan dengan tema tugas akhir.

BAB III METODE PENELITIAN

Bab ini berisikan tentang tahapan yang dilakukan peneliti dalam mengumpulkan informasi atau data yang dibutuhkan.

BAB IV HASIL DAN PEMBAHASAN

Pada bab ini dijelaskan pembahasan hasil dari penelitian yang dilakukan peneliti tentang aplikasi ini. Sebagai laporan observasi tentang penelitian terhadap sesuatu. Hasil pembahasan sebagai pertimbangan atau acuan, untuk dijadikan sebagai sebuah teori.

BAB V KESIMPULAN DAN SARAN

Pada bab ini berisi kesimpulan dari hasil penelitian yang telah dilakukan secara keseluruhan dan juga diberikan saran-saran untuk pengembangan selanjutnya.

BAB II

LANDASAN TEORI

2.1 Tinjauan Pustaka

Dalam suatu penelitian diperlukan dukungan hasil – hasil penelitian yang telah ada sebelumnya yang berkaitan dengan penelitian serupa. Berikut adalah hasil – hasil penelitian yang relevan dan perbandingan yang sedang dilakukan yang disajikan dalam bentuk tabel.

2.1.1 Kajian Penelitian yang Relevan

Tabel 1. Perbandingan Penelitian

No	Penulis / Tahun	Topik Penelitian	Metode	Hasil	Keterangan
1	Moh. Viqi Zulfiant / 2021	Rancang Bangun *Game Survival* 3D "*FUN SCIENCES ADVENTURE*"	*Waterfall*	Menghasilkan *game survival* 3D edukatif yang mengambil konsep ilmu pengetahuan alam juga mengimplementasikan *Artificial Intellegence* pada *NPC Zombie.*	Penelitian ini menggunakan algoritma *A** yang diimplementasikan pada *AI.*

No	Penulis / Tahun	Topik Penelitian	Metode	Hasil	Keterangan
2	Edy Santoso, dkk. / 2016	Rancang Bangun *Game Adventure Gyro* Berbasis *Android* Menggunakan *Model Rational Unifed Process (RUP)*	*Model Rational Unifed Process (RUP)*	Menghasilkan *game* yang diberi nama *"Adventure Gyro"* dan memiliki grafis 2D yang mengandung pesan selamatkan hewan yang ada disekitar kita.	Perbedaan terletak pada fitur dan platform yaitu pada penelitian ini memiliki stage yang beragam visualnya dan platform yang digunakan adalah *mobile device* yang ber OS *Android*.
3	Nahdia Asri Umami, dkk. / 2018	Rancang Bangun *Game Android Finding Diamond* dengan Unity 3D Menggunakan Metode *Dynamic Weighting A**	*Waterfall*	Hasil dari penelitian ini adalah sebuah game memiliki *perspective third – person* dan menggunakan objek 3D.	Pada penelitian ini menggunakan metode *Dynamic Weighting A** dalam penyebaran titik untuk menempatkan diamond disetiap level.

No	Penulis / Tahun	Topik Penelitian	Metode	Hasil	Keterangan
4	Rio Andriyati Krisdiawan & Ramdoni, Aji Permana / 2020	Rancang Bangun *Game Treasure of Labyrinth* dengan Algoritma *Backtracking* berbasis Android	*Game Development Life Cycle (GDLC)*	Hasil dari penelitian adalah game berhasil dibuat dengan *perspective third – person* dan metode *Backtracking* dapat membangun sistem pembangkit labirin *dynamis*.	Perbedaan terletak pada perspective *Top-Down* dan menggunakan objek 3D dan platform yang digunakan adalah *Mobile Device* terutama Android
5	Lourent Stefano Mongi dkk. / 2018	Rancang Bangun *Game Adventure of Unsrat* menggunakan *Game Engine Unity*	XP (*Extreme Programming*)	Menghasilkan *game* yang berkonsep simulasi dengan menyampaikan informasi lokasi gedung – gedung yang berada di Universitas dengan menggunakan 3D sebagai objeknya.	Penelitian ini berkonsep *RPG* atau memainkan peran dalam game dan menjelajah dunia yang ada di game ini menggunakan objek 2D

2.2 Kajian Teori

2.2.1 *Game*

Game berasal dari kata bahasa inggris yang memiliki arti dasar "Permainan". Permainan dalam ini merujuk pada pengertian "kelincahan intelektual" (*intellectual playability*). *Game* juga bisa diartikan sebagai arena keputusan dan aksi pemainnya. Ada target – target dan misi untuk dapat dicapai pemainnya. Kelincahan intelektual, pada tingkat tertentu merupakan ukuran sejauh mana *game* itu menarik untuk dimainkan secara maksimal (Siti Asmiatun, 2017).

a. Jenis – jenis *game*

Ada banyak jenis atau genre video *game*, dan biasanya mereka dikategorikan berdasarkan karakteristik atau tujuan yang mendasarinya. terdapat beberapa genre - genre game yaitu sebagai berikut :

1) *Action game*

Jenis *game* aksi adalah permainan dimana pemain mengendalikan dan menjadi pusat aksi, yang sebagian besar terdiri dari tantangan fisik yang harus diatasi pemain. Terdapat subgenre dari *game action* itu tersebut adalah *Platformer, Shooter, Fighting, Beat-em up, Stealth, Survival, dan Rhythm*.

2) *Action – Adventure Game*

Game aksi peteluangan menggabungkan dua mekanisme *game*, quest dan rintangan sepanjang *game* yang

harus ditaklukan menggunakan alat atau *item* yang dikumpulkan, serta elemen aksi tempat *item* digunakan. Adapun subgenre dari *action – adventure games adalah Survival Horror, Metroidvania*.

3) *Adventure Game*

Dalam *adventure game*, pemain biasanya berinteraksi dengan lingkungan mereka dan karakter lain untuk memecahkan teka – teki dengan petunjuk untuk memajukan cerita dan *gameplay*. Subgenre dari adventure games ini adalah *Text adventure, Graphic adventure, Visual novels, Interactive movie, Real – time 3D.*

4) *Role – Playing Games (RPG)*

Role – playing games adalah jenis *genre game* dimana pemain mengontrol aksi dari karakter atau grup dalam sebuah dunia *fantasy* dalam video *game*. Tiap pemain kemudian harus memerankan karakter sesuai dengan perannya untuk bisa mengembangkan permainan. *Role – playing games* juga membuat tiap pemain bisa memilih kostum sesuai dengan yang mereka inginkan. Dalam genre ini juga ada *non – controllable player* atau *NPC*, yakni karakter yang tidak dapat dimainkan, tetapi bisa berinteraksi dengannya. *RPG* umumnya memiliki misi – misi yang harus diselesaikan dengan interaksi dengan *NPC*, menjelajahi dunia *game*, atau mengumpulkan

item – item tertentu. Subgenrenya adalah *MMORPG, Rougelike, Tactical RPG, Sandbox RPG, First – person – party*.

5) *Simulation Games*

Game dalam genre simulasi semuanya dirancang untuk meniru realitas nyata atau fiksi, untuk mensimulasikan situasi atau peristiwa nyata. Subgenre dari *simulation game* adalah *life simulation, vehicle simulation*.

6) *Strategy Games*

Permainan ini mengharuskan pemain untuk menggunakan strategi dan taktik yang dikembangkan dengan hati – hati untuk mengatasi tantangan. Subgenrenya adalah *4X, Artillery, Real – time strategy (RTS), Multiplayer online battle arena (MOBA), Tower defense, Turn – based strategy (TBS), Turn – based tactics (TBT), Wargame, Grand strategy wargame*.

7) *Sport Games*

Permainan olahraga mensimulasikan olahraga seperti golf, sepak bola, bola basket, baseball, ski, bahkan sampai olahraga pub seperti panah dan biliar. Adapun subgenre dari *sport games* yaitu *Racing, Team sports, Team sports, Competitive, Sport – based fighting*.

8) *Puzzle Games*

Permainan teka – teki atau logika biasanya berlangsung di satu lapangan bermain dan mengharuskan pemain untuk memecahkan masalah. Subgenre dari genre ini adalah *Logic game, Trivia game*.

9) *Idle Games*

Idle games adalah permainan yang berkembang tanpa interaksi dari pemain. Sebuah *game* dengan genre yang terbilang baru. Subgenrenya adalah *Idle gaming, Casual game, Party game, Programming game, Board game, Massive multiplayer online (MMO), Advergame, Art game, Education game, Exergame*.

b. *Platformer*

Game platformer adalah permainan dimana pemainnya yang mengontrol karakternya untuk lari dan loncat ke berbagai arena, tangga, lantai, atau barang lain dalam tampilan *game scrolling/single*. Meski mirip, tapi beda dengan *game auto runner*. *Platformer* biasanya dikategorikan sebagai salah satu subgenre dari *game action*. *Game platformer* pertama dikembangkan pada awal 1980an sehingga *platformer* menjadi salah satu genre yang paling popular diawal kemunculannya. Tapi istilah *platformer* baru digunakan beberapa tahun kemudian. Selain menjadi salah satu genre *game* yang paling pertama dan populer, *platformer* juga

merupakan genre yang menggabungkan elemen – elemen dari genre lain seperti *leveling* dan kekuatan karakter yang dapat dijumpai di *game rpg*. Ada banyak contoh di mana *game platformer* mempunyai elemen dari genre – genre yang lain juga.

Banyak sejarahwan dan *fans* menganggap bahwa *space panic* yang dirilis pada tahun 1980 adalah *game platformer* pertama, sedangkan orang lain menganggap bahwa *Nintendo Donkey Kong* yang rilis pada tahun 1981 adalah *game platformer* pertama, namun sudah jelas bahwa *game – game* klasik seperti *donkey kong, space panic,* dan *mario bros* sangat berpengaruh terhadap pembentukan genre *platformer* (Klappenbach, 2019).

c. Elemen – elemen *game*

Elemen – elemen penting dari sebuah *game* terdiri dari *play, pretending, a goal, dan the rules*. Menurut (Adams, 2014) Adapun definisi dari elemen – elemen tersebut adalah :

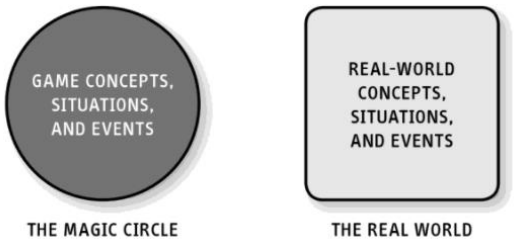
1) *Play*

Bermain adalah salah satu bentuk hiburan yang membutuhkan keikutsertaan (komunikasi dua arah), yang dimana buku, film, dan teater adalah bentuk hiburan yang berupa pertunjukkan (komunikasi dua arah). Seperti penulis yang menghibur pembacanya, tetapi ketika bermain, pemainlah yang menghibur dirinya sendiri. Sebuah buku tidak

dapat berubah meskipun sudah sering dibaca, namun ketika bermain, pemain dapat membuat keputusan yang mempengaruhi kejadian dalam permainan tersebut.

2) *Pretending*

Pretending adalah kegiatan untuk menciptakan sebuah realitas khayalan di dalam pikiran. Nama lain dari realitas yang diciptakan dari *pretending* adalah *magic circle*. Istilah ini pertama kali digunakan oleh sejarahwan belanda bernama Johan Huizinga dalam bukunya yang berjudul "*Homo Ludens*" di tahun 1971 yang berhubungan dengan dunia imajinasi dalam sebuah drama atau novel fiksi. Dalam *game*, *magic circle* mengacu pada batas yang memisahkan ide dan aktifitas yang berarti berada dalam *game* dari ide dan aktifitas yang berada dalam dunia nyata. Dengan kata lain batas antara dunia nyata dan khayalan (*make believe*).



Gambar 1. Konsep The Magic Circles *(Adams, 2014)*

3) *A Goal*

Sebuah permainan harus memiliki satu atau lebih tujuan. Tujuan dari permainan tersebut adalah peraturan dan bentuk dari permainan itu sendiri karena pembuat *game* dapat menentukan tujuan dari permainan tersebut sesuai dengan keinginannya. Dalam mencapai tujuan dalam permainan, ada tantangan yang harus diselesaikan. Meskipun kesulitan dari tantangan itu tergantung dari persepsi masing – masing pemain.

4) *The Rules*

Peraturan adalah definisi dan instruksi dimana pemain harus mengikutinya selama permainan berlangsung. Peraturan memiliki beberapa fungsi yaitu membuat objek permainan dan arti dari setiap aktifitas dan kejadian yang berbeda yang berlangsung di dalam *circles magic*. Peraturan juga membuat pemain mengetahui aktifitas apa yang diperbolehkan dan juga mengevaluasi setiap tindakan yang akan membuat pemain mencapai tujuan dari permainan tersebut.

2.2.2 *Game Unity Engine*

Unity merupakan suatu aplikasi yang digunakan untuk mengembangkan *game multi platform* yang didesain untuk mudah digunakan. Menurut (Wahyu, 2018) *Unity* itu bagus dan penuh perpaduan dengan palikasi yang *professional*. *Editor* pada *unity* dibuat

dengan *user interface* yang sedehana. *Editor* ini dibuat setelah ribuan jam yang mana telah dihabiskan untuk membuatnya menjadi nomor satu dalam urutan ranking teratas untuk *editor game*. Grafis pada *unity* dibuat dengan format umum seperti semua format *art applications*. *Unity* cocok dengan versi *64-bit* dan dapat beroperasi pada *Mac OS* dan *Windows* dan dapat menghasilkan *game* untuk *Mac, Windows, Wii, Iphone, Ipad, dan Android*.

Unity ini adalah sebuah aplikasi berbasis *Multi Platform, Multi Platform* merupakan aplikasi yang dapat beroperasi di banyak sistem operasi dan sanggup mempublish ke banyak format tipe *file*, misalnya : *exe, apk*, dan lainya.

Unity Technologies dibangun pada tahun 2004 oleh David Helgason (CEO), Nicholas Francis (CCO), dan Joachim Ante (CTO) di Copenhagen, Denmark sesudah *game* pertama mereka *GooBall*, gagal lagi dalam meraih sukses. Ketiganya menyadari nilai sebuah *engine* dan *tool* dalam sebuah pengembangan game dan berencana untuk menciptakan sebuah *engine* yang sanggup dipakai oleh semua dengan harga terjangkau. *Unity Technologies* menerima proteksi dana dari *Sequoia Capital*, *WestSummit Capital*, and *iGlobe Partners*.

Pada tahun 2008, *Unity* melihat kebangkitan *iPhone* dan menjadi *game engine* pertama yang melaksanakan dukungan penuh pada platform tersebut. *Unity* kini di gunakan oleh 53.1% *developers* (termasuk mobile game developer) dengan ratusan *game* yang dirilis

baik untuk *iOS* maupun *Android*. Pada tahun 2009, *Unity* mulai meluncurkan produk mereka secara gratis. Jumlah *developer* yang mendaftar melonjak drastis semenjak pengumuman tersebut. Pada April 2012, *Unity* mencapai popularitas yang sangat tinggi dengan lebih dari 1 juta *developer*.

2.2.3 *C# (C Sharp)*

C# (dibuat : *C Sharp*) merupakan sebuah bahasa pemrograman yang berorientasi objek yang dikembangkan oleh *Microsoft* sebagai bagian dari inisiatif kerangka *NET Framework*. Bahasa pemrograman ini dibuat berbasiskan bahasa *C++* yang telah dipengaruhi oleh aspek – aspek ataupun fitur bahasa yang terdapat pada bahasa – bahasa pemrograman lainnya seperti *Java*, *Delphi*, *Visual Basic*, dan lain – lain dengan beberapa penyederhanaan. *C#* juga dapat di jalankan ke dalam komputer dan dapat diproses dalam *mode offline*. *C#* merupakan bahasa pemrograman untuk pengembangan *game* dan juga bisa dapat dipakai dalam unity untuk pembuatan *game model 2D* dan *3D* oleh karena itu *C#* dapat terintegrasi dengan *unity* untuk membuat *game arsitektur* bangunan dan simulasi yang dirancang untuk *modeling* dan *rendering* dalam aplikasi *unity*. Dalam *unity C#* adalah fitur untuk *scripting* dan mudah digunakan untuk *rotating* dan *scaling object* hanya perlu sebaris kode. Begitu pula dengan *dublicating*, *removing*, dan *changing properties*. *C#* juga mudah digunakan untuk *visual properties variables* yang didefinisikan dengan *scripts* ditampilkan pada *editor*, yang dapat

dijalankan dalam aplikasi *unity*, berbasis *NET* artinya untuk *run program* dilakukan dengan *open source*.

2.2.4 Teknik Pengumpulan Data

Menurut (Sugiyono, 2019), teknik pengumpulan data merupakan langkah yang paling strategis dalam penelitian, karena tujuan utama dari penelitian adalah mendapatkan data. Prosedur pengumpulan data dapat juga diartikan sebagai suatu usaha untuk mengumpulkan data. Teknik yang digunakan peneliti dalam penelitian ini adalah :

1) Observasi (Pengamatan)

Observasi merupakan pengamatan, perhatian atau pengawasan. Moh. Nazir mendefinisikan observasi sebagai pengambilan data dengan menggunakan mata tanpa pertolongan alat standar lain untuk keperluan tersebut.

2) Wawancara

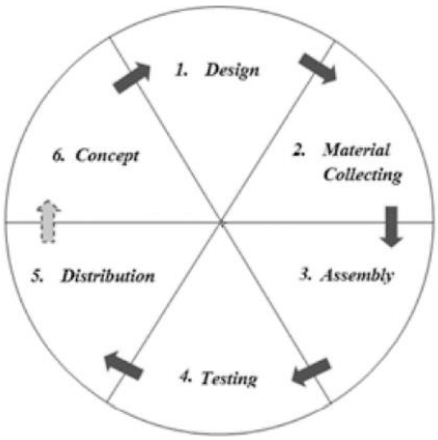
Wawancara merupakan metode pengumpulan data dengan jalan tanya jawab sepihak yang dilakukan secara sistematis dan berlandaskan kepada tujuan penelitian. Peneliti menerapkan jenis pembicaraan informal, pertanyaan yang diajukan muncul secara spontanitas. Pembicaraan dimulai dari segi umum menuju yang khusus. Peneliti mengajukan pertanyaan yang bebas kepada subyek menuju fokus penelitian.

3) Dokumentasi

Dokumentasi yaitu proses untuk memperoleh keterangan untuk tujuan penelitian yang berasal dari data yang berbentuk arsip (dokumen), karena dokumen merupakan sumber data yang berupa bahasa tertulis, foto atau dokumen elektronik. Metode dokumentasi bermanfaat dalam melengkapi hasil pengumpulan data melalui observasi dan wawancara.

2.2.5 *Multimedia Development Life Cycle (MDLC)*

Pengembangan metode multimedia ini dilakukan berdasarkan enam tahap, yaitu *concept* (pengonsepan), design (perancangan), *material collecting* (pengumpulan bahan), *assembly* (pembuatan), testing (pengujian), dan distribution (pendistribusian). Menurut Luther dalam (Mustika, 2017) keenam tahap ini tidak harus berurutan dalam praktiknya, tahap – tahap tersebut dapat saling bertukar posisi. Meskipun begitu, tahap concept memang harus menjadi hal yang pertama kali dikerjakan.



Gambar 2. Tahapan metode *MDLC (Mustika, 2017)*

Adapun penjelasan tentang tahapan – tahapan diatas sebagai berikut :

1) *Concept* (Konsep)

Concept adalah tahap untuk menentukan tujuan dan siapa pengguna program (identifikasi audien). Selain itu menentukan macam aplikasi (presentasi, interaktif, dll) dan tujuan aplikasi (hiburan, pelatihan, pembelajaran, dll).

2) *Design* (Desain)

Design adalah tahap membuat spesifikasi mengenai arsitektur program, gaya, tampilan, dan kebutuhan material/bahan untuk program.

3) *Material Collecting* (Pengumpulan Bahan)

Material collecting adalah tahap pengumpulan bahan yang sesuai dengan kebutuhan yang dikerjakan. Tahap ini dapat dikerjakan secara parallel dengan tahap *assembly*. Namun, pada beberapa kasus, tahap *material collecting* dan tahap *assembly* akan dikerjakan secara linear dan tidak *parallel*.

4) *Assembly* (Pembuatan)

Assembly adalah tahap pembuatan semua objek atau bahan *multimedia*. Pembuatan aplikasi berdasarkan pada tahap desain.

5) *Testing* (Pengujian)

Testing dilakukan setelah menyelesaikan tahap pembuatan (*assembly*) dengan menjalankan aplikasi/program dan dilihat apakah ada kesalahan atau tidak. Tahap ini disebut juga sebagai tahap pengujian *alpha* (*alpha test*) yang pengujiannya dilakukan oleh pembuat atau lingkungan pembuatnya sendiri.

6) *Distribution* (Pendistribusian)

Tahap ini aplikasi akan disimpan dalam suatu media penyimpanan. Pada tahapan ini jika media penyimpanan tidak cukup untuk menampung aplikasinya, maka dilakukan kompresi terhadap aplikasi tersebut.

2.2.6 *Storyboard*

Storyboard merupakan gambaran mengenai sistem yang dibuat dan menunjukkan tampilan dari sistem tersebut. Dalam *storyboard* akan ditunjukkan juga elemen *multimedia* yang digunakan pada setiap scenenya (Umi Khulsum, 2018).

2.2.7 Skala *Likers*

Angket dengan *Skala Likert* biasanya menyajikan pernyataan yang disertai dengan pilihan. Pilihan pada skala *Likert* berupa frekuensi (selalu, sering, jarang,tidak pernah) atau persetujuan (sangat setuju, setuju, netral, tidak setuju, sangat tidak setuju). Pilihan jawaban dengan skalaini diskor secara berjenjang (*ordinal*).

Instrumen model *Likert* ini relatif mudah membuatnya, dan responden juga mudah meresponnya. Namun kelemahan dari instrumen

ini adalah adanya kecenderungan responden untuk mengisi instrumen sesuai dengan harapan masyarakat (*desireability bias*). Instrumen dengan skala ini merupakan bentuk yang sering digunakan peneliti untuk melakukan pengukuran.

Model lain dari skala *Likert* adalah angket dengan pilihan ganda. Instrumen ini disarankan oleh ahli untuk mengurangi desireability bias. Pada instrumen ini, butir - butirnya berupa pernyataan yang disertai dengan pilihan-pilihan. Pilihan-pilihan yang disajikan sesuai dengan kondisi yang mungkin dialami oleh responden. Pilihan - pilihan ini kemudian diskor berjenjang. Karena pilihannya menyajikan kondisi kondisi yang mungkin dialami oleh responden, menyebabkan peneliti yang menyusun instrumen ini relatif sulit dan memerlukan waktu untuk membuatnya. Selain itu, pilihan-pilihan yang cukup panjang akan menyulitkan responden ketika membaca dan meresponnya.

BAB III

METODE PENELITIAN

3.1 Jenis Penelitian

Penelitian ini termasuk penelitian *Research and Development* (*R&D*). Menurut (Sugiyono, 2019) *Research and Development* (*R&D*) merupakan metode penelitian yang digunakan untuk menghasilkan produk tertentu dan menguji keefektifan produk tersebut.

Berdasarkan pendapat diatas, maka *Research and Development* (*R&D*) merupakan metode penelitian yang digunakan secara sengaja dan sistematis untuk menyempurnakan produk yang telah ada maupun mengembangkan suatu produk baru melalui pengujian, sehingga produk tersebut dapat dipertanggungjawabkan.

3.2 Pengumpulan Data

Penulis menggunakan beberapa tahapan atau metode dalam melakukan penelitian untuk menyusun tugas akhir ini, yaitu :

3.2.1 Studi Pustaka

Teknik pengumpulan data dengan cara mengumpulkan sumber referensi buku – buku yang berkaitan dengan penelitian, ada pula mengumpulkan dari teknologi yang sudah ada saat ini, literatur, jurnal, internet dan bacaan – bacaan sebagai sumber referensi yang berhubungan dengan *game* yang akan dibuat, serta teori yang berhubungan dengan perancangan *game* dengan *unity engine*.

3.2.2 Studi Kuesioner

Menyebarkan kuisioner *online* yang berisi pertanyaan – pertanyaan kepada pengguna dan meneliti data yang didapatkan dari reponden kuisioner *online* untuk membantu perancangan dan juga evaluasi *game platformer* yang dibuat penulis untuk mengukur tingkat ketertarikan pengguna pada *game* ini.

3.2.3 Studi Dokumentasi

Dokumentasi bertujuan untuk memperoleh dan mengumpulkan data yang membantu penulis dalam pembuatan *game* misalkan mengumpulkan assets gambar dan musik yang relevan dengan perancangan *game*.

3.3 Analisis Kebutuhan

Kebutuhan sistem haruslah sesuai dengan kondisi dan kemampuan pengguna, maka dari itu penulis yang juga adalah sebagai pembuat atau membangun program ikut serta melibatkan pengguna dalam mencari dan menganalisis kebutuhan – kebutuhan sistem yang menunjang dalam proses perancangan serta membangun aplikasi game platformer adventure "*Foxy Tales*". Adapun kebutuhan sistem yang diperlukan itu sebagai berikut :

3.3.1 Kebutuhan Perangkat Keras

Dalam kebutuhan perangkat keras (*hardware*) yang penulis gunakan dalam pembuatan aplikasi seperti Tabel 2 dibawah ini.

Tabel 2. Kebutuhan Perangkat Keras

No	Perangkat Keras	Spesifikasi
1	Type	Asus ROG Strix GL503GE-EN129T Hero Edition (15.6-inch Full HD)
2	Processor	Intel Core i7-8750H
3	Memory	8 GB DDR4 2666 MHz
4	Disk	128 GB SSD + 1 TB SSHD
5	Graphic	Nvidia GeForce GTX 1050 Ti 4 GB

3.3.2 Kebutuhan Perangkat Lunak

Dalam kebutuhan Perangkat Lunak (*software*) dalam aplikasi ini yang digunakan seperti Tabel 3 dibawah ini.

Tabel 3. Kebutuhan Perangkat Lunak

No	Perangkat Lunak	Spesifikasi
1	Unity	Sebagai perangkat lunak dalam pembuatan game.
2	Adobe Photoshop	Sebagai perangkat lunak untuk pembuatan sprite.
3	FL Studio	Sebagai perangkat lunak untuk pembuatan music dan sound fx.
4	Windows 11 Pro	Digunakan sebagai OS dalam menjalankan aplikasi perangkat lunak.
5	Canva	Digunakan sebagai pembuatan Splash Screen.

3.3.3 Analisis Proses

1) *Multimedia Development Life Cycle (MDLC)*

1. *Concept* (Pengonsepan)

   Tujuan dari pembuatan aplikasi *game* ini adalah sebagai media bermain dan belajar didalam ruang lingkup *game* dengan target pengguna berumur dari 16 – 25 tahun.

2. *Design* (Perancangan)

   Tahapan ini menggunakan perancangan perangkat lunak untuk menggambarkan rancangan tampilan aplikasi *game* ini nantinya.

3. *Material Collecting* (Pengumpulan bahan)

   Bahan – bahan yang digunakan pada *game* ini diperoleh dengan membuat sendiri dan juga pemesanan kepada pihak lain sesuai dengan rancangan yang telah dibuat.

4. *Assembly* (Pembuatan)

   Pada tahapan ini semua bahan yang sudah dikumpulkan akan dibuat sesuai dengan rancangan. Pembuatan *sprite character* dari *game* ini penulis buat dengan menggunakan *Adobe Photoshop* dan untuk musik penulis buat mengguanakan *Fl Studio* sedangkan untuk pembuatan *splash screen* penulis menggunakan *Canva* dan untuk pemrograman penulis lakukan pada *Unity Game Engine*.

5. *Testing* (Pengujian)

   Pengujian pertama kali dilakukan adalah *Alpha Test* dimana pengujian ini dilakukan oleh penulis sendiri atau orang – orang dilingkungan penulis. *Alpha* testing ini terdiri dari pengujian pada fitur aplikasi, dan pengujian terhadap fitur *game mechanic* dan juga tampilan, apakah sudah berjalan sesuai dengan yang penulis harapkan.

   Setelah *alpha* testing dilakukan dan mendapat hasil yang sesuai dengan skenario pengujian, maka tahap selanjutnya adalah *Beta testing*, pada tahapan ini penulis menggunakan kuesioner *online* untuk mengukur tingkat ketertarikan pengguna dalam game yang penulis buat dan fitur – fitur yang telah di uji di *alpha testing*.

6. *Distribution* (Pendistribusian)

   Aplikasi *game* ini didistribusikan melalui *itch.io* dengan tujuan *platform Windows* dan *Mac*.

3.3.4 Analisis Kelemahan

Berikut adalah hal-hal yang dapat di analisis menggunakan *SWOT* :

1) *Strength* (kekuatan)

Strength adalah kekuatan yang dijadikan dasar dalam membangun *game genre platformer*. Adapun kekuatan yang dimaksud yakni berbicara tentang kualitas, terlebih mekanisme

game yang mudah di mainkan, musik yang dibuat dengan nuansa *chiptune* dan memiliki visual yang menarik yaitu *pixel art* sehingga membuat pengguna bernostalgia kembali ke zaman *retro*.

2) *Weakness* (kelemahan)

Kelemahan sistem *game* yang dibuat penulis adalah *game* ini hanya berjalan pada sistem berbasis desktop saja seperti Windows dengan minimum requirement *Intel Pentium Dual-Core G4400 3.30 GHz* atau bisa lebih tinggi, dengan *OS Windows 7 64bit, 4 GB RAM, Graphics Card Nvidia GeForce GTX 750* sedangkan minimum requirement untuk sistem operasi Mac *Processor 2 GHz*, memiliki *2 GB RAM*, dengan *Graphics 256 MB video memory*.

3) *Opportunity* (peluang)

Kesempatan atau peluang yang dapat diperolah dalam rancang bangun *game* bergenre *platformer* adventure "*Foxy Tales*" yaitu menarik minat orang-orang yang suka memainkan game dengan nuasa *pixel art* yang bergenre *platformer* dan melatih ketangkasan dalam menyelesaikan rintangan dan kuis yang ada pada *game*.

4) *Threat* (ancaman)

Ancaman yang memungkinkan dalam *game* ini telah banyak *game* bergenre *platformer* yang dikembangkan oleh *developer* besar

dan lebih menarik baik dari segi tampilan, fitur, *gameplay*, dan hal - hal lainnya yang dapat lebih menarik pemain untuk memainkannya. Hal ini memungkinkan terjadi persaingan yang ketat dalam pembuatan game bagi para pembuat *game*.

3.4 Desain

3.4.1 *Storyline*

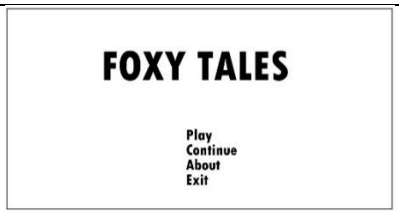
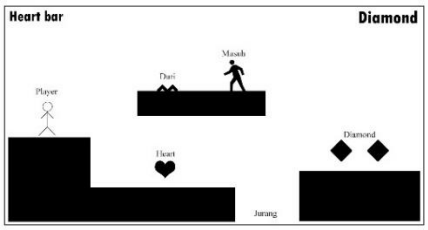
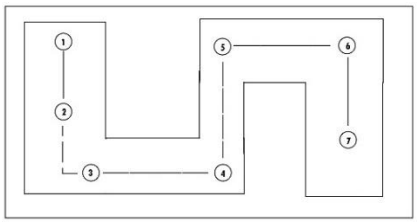
Bercerita tentang rubah bernama *Foxy,* seekor rubah yang hendak menyelamatkan pulau Sunnyland dari musuh – musuh yang menjajah pulau tersebut, tentu dalam aksi penyelamatannya *Foxy* harus melewati segala rintangan, memecahkan teka – teki , dan mengalahkan musuh – musuh yang menghadangnya hingga pulau *Sunnyland* pun aman dan tentram kembali.

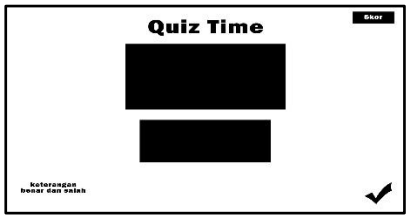
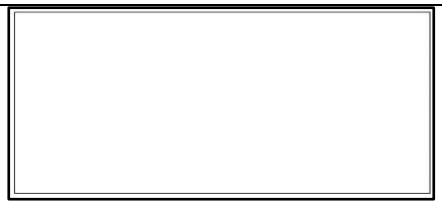
3.4.2 *Storyboard*

Storyboard ini sistem terdiri dari *scene splash scene*, *scene* menu utama, *scene* menu *play*, *scene* menu c*ontinue*, *scene about*, dan *scene exit*.

Tabel 4. *Storyboard Game*

No	Nama	Desain	Keterangan
1	*Splash Screen*		– Menampilan nama pembuat g*ame*. – Lanjut kehalaman menu utama.

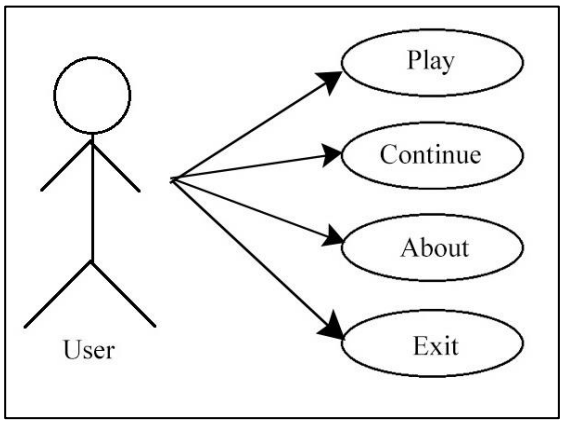
No	Nama	Desain	Keterangan
2	Menu Utama		– Judul *Game* – Menu *Start* – Menu *Continue* – Menu *About* – Menu *Exit*
3	Menu *Play*		– Menampilkan *Gameplay* – Melewati rintangan dalam *game* – Menyelesaikannya agar lanjut ke *stage* berikutnya.
4	Menu *Continue*		– Menampilkan Stage yang ingin dimainkan – Memilih *Stage* yang tersedia di peta – Lanjut ke *Gameplay* setelah memilih *stage*
5	Menu *About*		– Keterangan tentang *game* – Panduan bermain game – Tombol *Back* untuk kembali ke Menu Utama

No	Nama	Desain	Keterangan
6	Menu Exit		– Menjawab pertanyaan kuis – Tekan tombol ceklis jika sudah menjawab – Jika pertanyaan selesai dijawab scene akan beralih ke *select level* atau beralih ke *victory* scene apabila sudah menyelesaikan *boss level*
7	Quiz		– Keluar dari game

3.4.3 Desain Proses

1) *Use Case Diagram*

Use Case Diagram merupakan gambaran *user* yang menggunakan sistem dan perilaku *user* terhadap sistem.



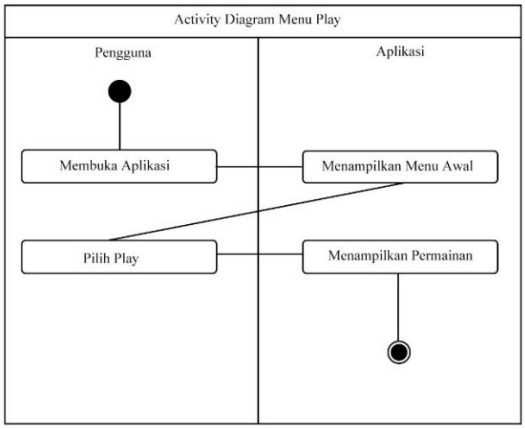
Gambar 3. *Use Case Diagram*

Dijelaskan bahwa dalam *Use Case Diagram* aplikasi ini terdapat tampilan menu – menu yang ada pada aplikasi *Game Foxy Tales*.

2) *Activity Diagram*

Activity Diagram merupakan gambaran alur proses atau cara kerja sistem. Pada diagram ini digambarkan aktivitas – aktivitas apa saja yang dikerjakan oleh sebuah sistem.

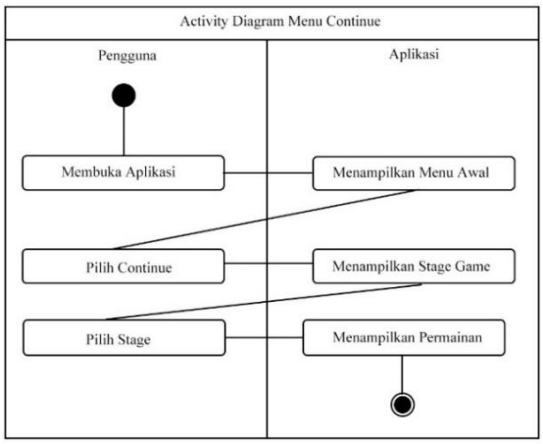
1. Activity Diagram Play



Gambar 4. *Activity Diagram Play*

Pada gambar 4 dijelaskan bahwa pengguna membuka aplikasi, kemudian akan muncul tampilan menu awal. Pada menu awal ini pengguna memilih tombol "*Play*" yang berfungsi untuk menampilkan permainan yang ada di dalam tombol tersebut.

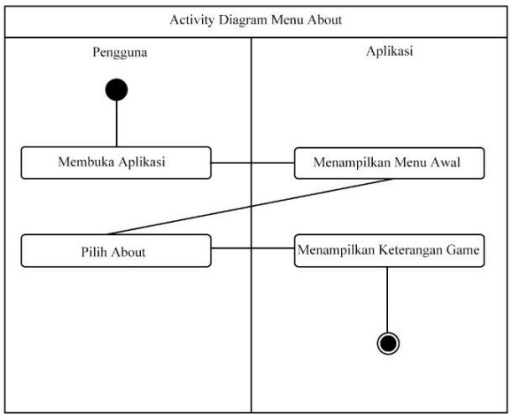
2. *Activity Diagram Continue*



Gambar 5. Activuty Diagram Continue

Pada gambar 5 dijelaskan bahwa pada saat pengguna memilih menu *Continue* maka aplikasi langsung menampilkan *Stage game*, pengguna pun memilih *stage* yang ingin dilanjutkan, setelah memilih *stage*, aplikasi pun menampilkan permainan.

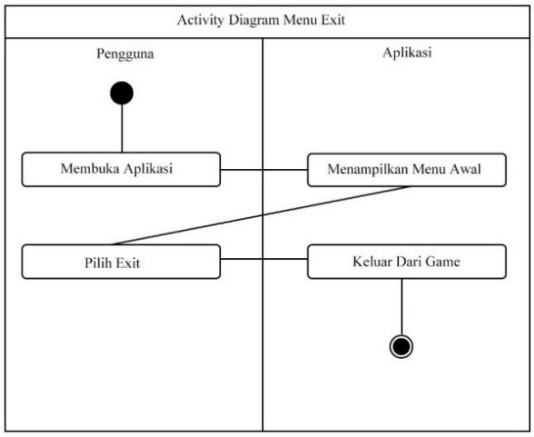
3. *Activity Diagram About*



Gambar 6. *Activity Diagram About*

Pada gambar 6 dijelaskan bahwa aplikasi menampilkan informasi kepada pengguna tentang *game* yang dimainkan dan cara bermain.

4. Activity Diagram Exit



Gambar 7. *Activity Diagram Exit*

Pada gambar 7 diatas dijelaskan bahwa aplikasi menampilkan informasi kepada pengguna jika tombol "*Exit*" dipilih makan akan keluar dari permainan.

3.4.4 Desain Perangkat Lunak

1) *Splash Screen*

Tampilan awal dari *game* adalah splash screen sambutan dengan nickname penulis yang bertransisi sebelum masuk ke permainan.

Gambar 8. Tampilan *Splash Screen*

2) Menu Utama

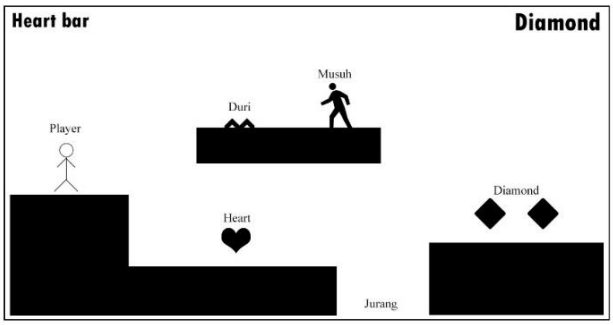
Tampilan menu utama terdapat judul dari game, dan tombol "*Play*" untuk memulai permainan, "*Continue*" untuk melanjutkan permainan, "*About*" untuk mengetahui tata cara bermain, dan "Exit" keluar dari permainan.

Gambar 9. Tampilan Menu Awal

3) *Gameplay*

Menu ini berisi tentang gambaran *gameplay* permainan, dimana sebuah area *gameplay* yang menjadi tempat dari pengguna untuk melewati rintangan, memiliki perspektif *2D*, dan memiliki *Heads up Display* (HUD) seperti tampilan sisa

heart, *diamond* yang terkumpul, musuh, dan rintangan yang perlu dilewati.



Gambar 10. Tampilan *Gamplay Game*

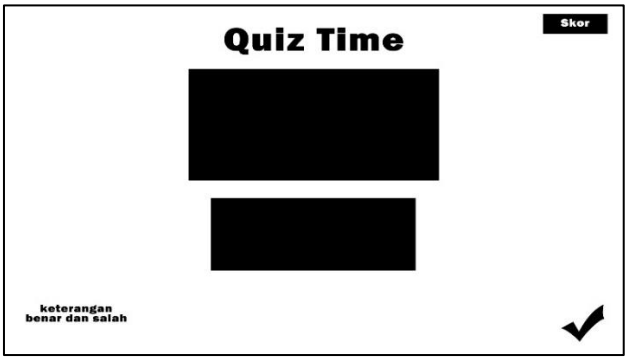
4) Menu *About*

Menu ini berisikan keterangan panduan bermain game.

Gambar 11. Tampilan *Menu About*

5) Quiz Time

Menu ini berisikan kuis yang harus diselesaikan agar dapat menyelesaikan level.



Gambar 12. Tampilan *Scene Quiz Time*

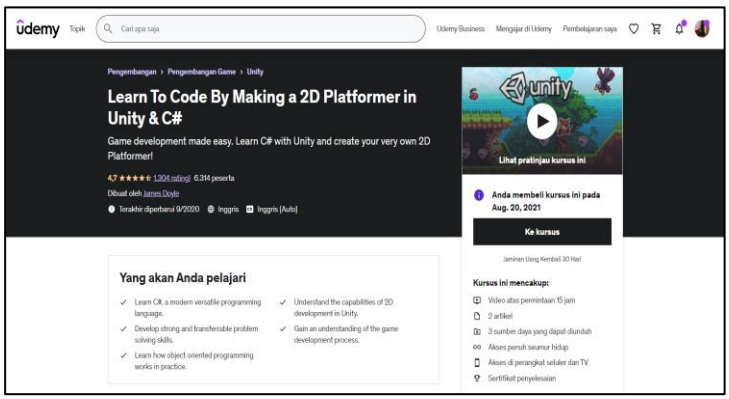
BAB IV

HASIL DAN PEMBAHASAN

4.1 Hasil

4.1.1 Implementasi Program

Dalam implementasi program *software* yang digunakan penulis untuk membuat *game Foxy Tales* adalah *Unity* dengan bahasa program C#. Untuk perancangan desain dalam game penulis membeli *assets game* dari sebuah website yang bernama *Udemy* dan *developer* pembuat *assets* bernama James Doyle.



Gambar 13. Sumber *assets game*

Kemudian penulis menggunakan aplikasi *Adobe Photoshop* sebagai aplikasi mengedit dan memodifikasi *sprites*, *object* dan *background* pada *game Foxy Tales* yang telah diperoleh dari sumber tersebut.

4.1.2 Manual Program

Game Foxy Tales adalah *game* yang dijalankan di sistem berbasis desktop. Penulis akan menjelaskan cara menggunakannya, yaitu sebagai berikut :

1. Buka aplikasi *game Foxy Tales* di *Personal Computer* atau Laptop pengguna.
2. Selesainya membuka *game*, *game* akan menunjukan tampilan *Splash Screen* seperti gambar dibawah.

Gambar 14. Tampilan *Splash Screen*.

3. Setelah tampilan *Splash Screen* selesai, maka *scene* selanjutnya adalah tampilan *scene* menu utama seperti gambar dibawah.

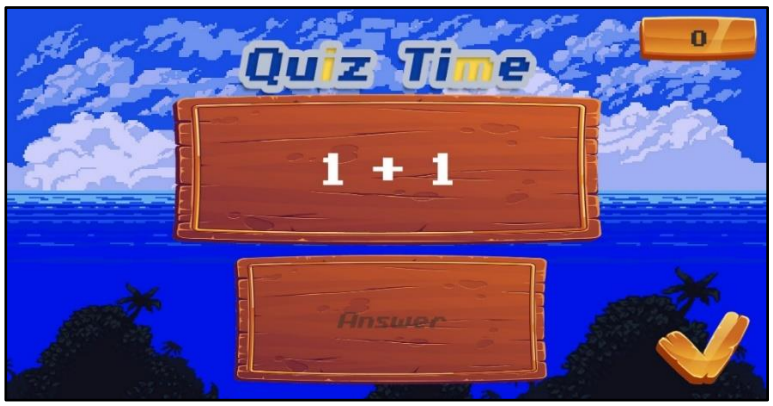
Gambar 15. Tampilan menu utama

Pada menu utama ini terdapat tombol *new game* yang apabila ditekan akan menampilkan *scene gameplay*, kemudian tombol *continue* apabila ditekan akan menampilkan menu *select level* yang sudah kita selesaikan sebelumnya, tombol *about* untuk menampikan keterangan pada *game* dan cara bermain, kemudian tombol *quit* untuk menutup *game*.

4. Ketika tombol *new game* ditekan maka *scene gameplay* akan tampil dan permainan pun dimulai.

Gambar 16. Tampilan *gameplay Foxy Tales*

Pada scene ini pengguna harus melewati rintangan yang dan setelah rintangan tersebut berhasil dilewati, maka *scene* akan beralih ke *scene quiz time*.



Gambar 17. Tampilan *quiz time*

Scene quiz time ini akan muncul apabila pengguna berhasil menyelesaikan rintangan – rintangan disetiap levelnya.

5. Ketika tombol *about* ditekan maka *scene* about akan tampil, menu about berisi tentang *storyline* dari *game Foxy Tales* dan *tutorial* cara bermainnya. Jika tombol *back* ditekan maka akan kembali ke menu utama.

Gambar 18. Tampilan halaman about

6. Ketika sudah kembali ke menu utama dan tombol *continue* ditekan maka *scene* akan menampikan *level select* dan pengguna bisa memilih *level* yang ingin dilanjutkan.

Gambar 19. Tampilan halaman *select menu*

7. Ketika pengguna telah berhasil melewati semua *level* dan berhasil menjawab semua *quiz* maka *scene victory* akan tampil dan jika tombol *main menu* ditekan maka *scene* akan beralih ke menu utama.

Gambar 20. Tampilan *scene victory*

8. Ketika tombol *exit* ditekan maka aplikasi akan tertutup.

4.1.3 Pengujian

Pengujian adalah bagian penting dari pengembangan perangkat lunak. Pengujian dirancang untuk menemukan bug dalam sistem dan memastikan bahwa sistem dibangun sesuai rencana. Sistem yang digunakan oleh penulis adalah menggunakan pengujian *black box*. Fokus pengujian sistem ini adalah kebutuhan fungsional dari sistem aplikasi. Berikut ini adalah hasil pengujian dari *game* Rancang Bangun *Game Platformers 2D "Foxy Tales"* Bergenre *Adventure* Menggunakan *Unity Game Engine*.

1. Pengujian Buka Aplikasi

Tabel 5. Pengujian Buka Aplikasi

Pengujian	**Skenario Uji**	**Hasil yang Diharapkan**	**Hasil Pengujian**
Membuka dan memulai aplikasi	Buka aplikasi Rancang Bangun *Game Platformers 2D "Foxy Tales"* Bergenre *Adventure* Menggunakan *Unity Game Engine*.	Aplikasi terbuka dan menampilkan *splash screen* kemudian masuk ke menu utama	Sesuai

2. Pengujian Halaman Menu Utama

Tabel 6. Pengujian Halaman Menu Utama

Pengujian	**Skenario Uji**	**Hasil yang Diharapkan**	**Hasil Pengujian**
Tombol *New Game*	Tekan tombol *new game* pada menu utama	Aplikasi akan beralih ke *scene gameplay*	Sesuai
Tombol *Continue*	Tekan tombol *continue* pada menu utama	Aplikasi akan berpindah ke tampilan *select level*	Sesuai
Tombol *About*	Tekan tombol *about* pada menu utama	Aplikasi akan membuka *scene about*	Sesuai
Tombol *Quit*	Tekan tombol *quit* pada menu utama	Aplikasi akan tertutup	Sesuai

3) Pengujian Halaman *New Game*

Tabel 7. Pengujian Halaman New Game

Pengujian	**Skenario Uji**	**Hasil yang Diharapkan**	**Hasil Pengujian**
Pergerakan *Player*	Gerakan *player* dengan A,D,←,→	*Player* bergerak ke kanan dan kiri	Sesuai
Player Jump	Menekan tombol *space* pada *keyboard*	*Player* akan melompat	Sesuai

Player Double Jump	Menekan tombol *space* dua kali pada *keyboard*	*Player* akan melompat dua kali	Sesuai
Pickup Items	Ambil item yang ada di *stage*	*Player* mengumpulkan item atau heart terisi setengah hati	Sesuai
Jump Attack	Menakan tombol *space* pada *keyboard*	*Player* melompat dan menginjak musuh	Sesuai

4) Pengujian Halaman About

Tabel 8. Pengujian Halaman About

Pengujian	**Skenario Uji**	**Hasil yang diharapkan**	**Hasil Pengujian**
Tombol *Back*	Menekan tombol *back*	Aplikasi akan kembali ke halaman menu utama	Sesuai

5) Pengujian Halaman Scene Victory

Tabel 9. Pengujian scene victory

Pengujian	**Skenario Uji**	**Hasil yang diharapkan**	**Hasil Pengujian**
Tombol *Main Menu*	Menekan tombol *main menu*	Aplikasi akan kembali ke halaman menu utama	Sesuai

Seperti yang terlihat dari pengujian yang telah dilakukan, aplikasi bekerja dengan baik dan memberikan *output* atau hasil yang benar secara fungsional dan sistem berjalan seperti yang diharapkan. Setelah uji *black box*, langkah selanjutnya adalah merancang kuesioner untuk melihat umpan balik pengguna. Kuesioner berisi beberapa pernyataan yang akan diberikan kepada pengguna dan jawabannya akan dievaluasi.

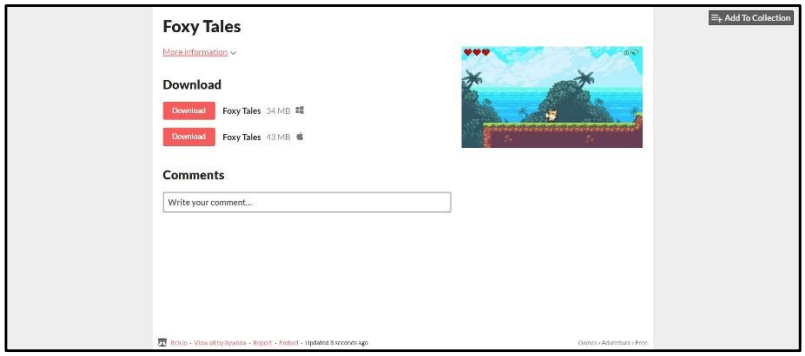
Aspek yang ditanyakan dalam kuesioner adalah :

1. Tampilan antar muka dalam aplikasi *game* menarik.
2. Cara bermain atau petunjuk dalam *game* mudah dipahami.
3. Tampilan objek seperti *player*, musuh, dan *pixel art* dari *game* cukup menarik.
4. Musik dan efek suara dari *game* menarik.
5. Konfigurasi dalam *game* mudah dipahami.
6. Game ini menyenangkan dan *friendly*.

4.1.4 Manual Instalasi

Pada bagian ini penulis akan menjelaskan bagaimana langkah – langkah dalam melakukan download sampai dengan instalasi *game Foxy Tales* dengan benar, Adapun langkah – langkahnya sebagai berikut :

1. *Game Foxy Tales* dapat di download pada *link* https://ilywnda.itch.io/foxy-tales, kemudian pilih *OS* komputer atau laptop yang sesuai dengan pengguna.



Gambar 21. Page Itch.io Foxy Tales

2. Jika *file* sudah selesai di *download* maka selanjutnya *extract zip file* yang sudah pengguna *download*.
3. Kemudian game siap langsung dimainkan, pengguna dapat membuka *folder game* yang sudah di *extract* dan tekan *file* berekstensi *.exe* untuk *Windows* dan *.app* untuk *macOS*.

4.2 Pembahasan

4.2.1 Pembahasaan Hasil Respon Pengguna

Suatu program dikatakan berhasil apabila *input*, proses, dan *output* dari aplikasi tersebut berhasil sesuai dengan tujuan yang ingin dicapai. Oleh karena itu, adalah ujian untuk mengetahui kekuatan dan kelemahan program. Untuk dapat menganalisis data, diperlukan tes melalui kuesioner dengan menggunakan skala *likert*.

Tabel 10. Skor Pilihan Jawaban

Jawaban	**Tidak Setuju**	**Biasa saja**	**Setuju**
Skor	1	2	3

Hasil survei terhadap Rancang Bangun *Game Platformers* 2D "*Foxy Tales*" Bergenre *Adventure* Menggunakan *Unity Game Engine* mengungkapkan sebagai berikut :

Tabel 11. Jawaban Responden

No	Pertanyaan	Jawaban Responden								Jumlah
		1	2	3	4	5	6	7	8	
1	Tampilan antar muka dalam aplikasi *game* menarik.	3	3	3	3	3	3	3	3	24
2	Cara bermain atau petunjuk dalam *game* mudah dipahami.	2	3	2	3	3	3	3	3	19
3	Tampilan objek seperti *player*, musuh, dan *pixel art* dari *game* cukup menarik.	3	3	2	3	2	3	3	3	22
4	Musik dan efek suara dari *game* menarik	3	3	3	3	2	3	3	3	23
5	Konfigurasi dalam *game* mudah dipahami	3	3	3	3	3	3	3	3	24
6	Game ini menyenangkan dan *friendly*.	3	3	3	3	3	3	3	3	24
Jumlah		136								

Untuk mendapatkan hasil, Anda perlu mengetahui interpretasi *interval* (jarak) dan persentase, dan mengetahui skor dengan metode pencarian persentase *interval* skor (I):

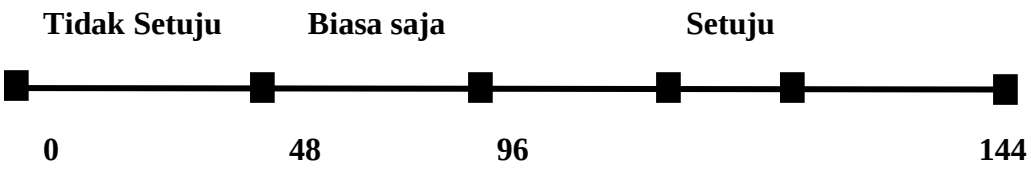
Rumus *Interval*

I = Jumlah Responden x Jumlah Kuesioner x Skor (*likert*)

Maka = 8 x 6 x 3 = 144

= 8 x 6 x 2 = 96

= 8 x 6 x 1 = 48



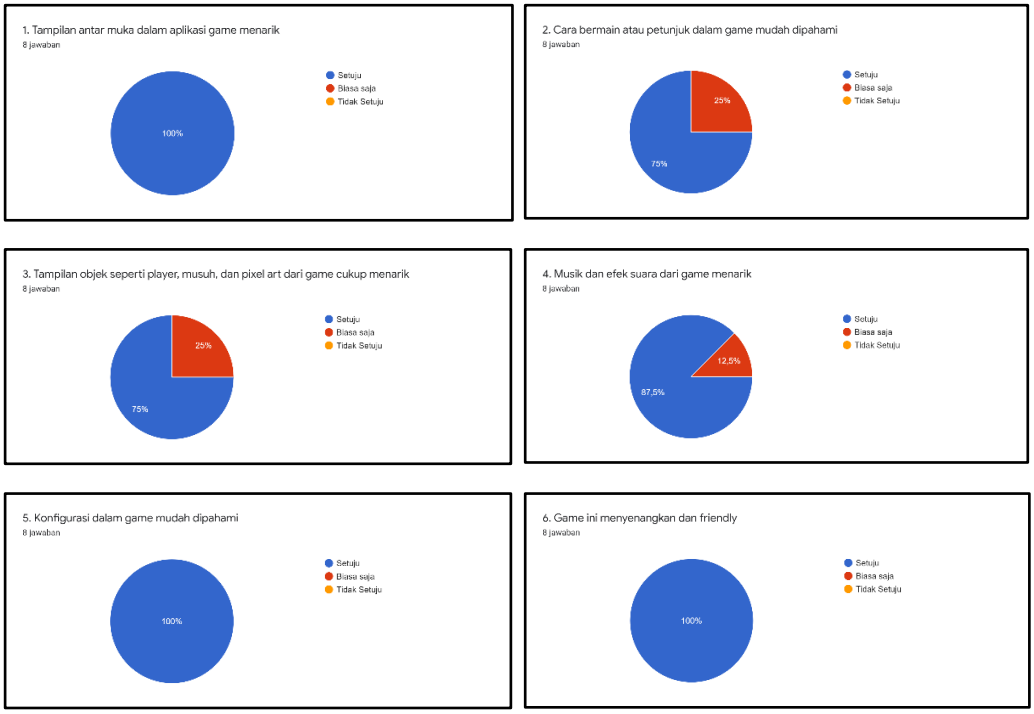
Maka penyelesaian dari kasus diatas adalah

= (Total Skor/Y) x 100%

= (136/144) x 100%

= 94,44%

Berdasarkan hasil enam pertanyaan responden, aplikasi Rancang Bangun *Game Platformers* 2D "*Foxy Tales*" Bergenre *Adventure* Menggunakan *Unity Game Engine* berada pada kategori sangat baik dengan rata-rata *share* sebesar 94,44%. Survei diperoleh dari tanggapan pengguna.



Gambar 22. Hasil Kuesioner

BAB V

KESIMPULAN DAN SARAN

5.2 Kesimpulan

Berdasarkan pemaparan pada bab - bab sebelumnya, seperti analisis, hasil penelitian, pembahasan, dan implementasi, penulis dapat menarik kesimpulan sebagai berikut :

1. Game dimainkan satu orang *player* atau *single player*.
2. Telah berhasil dibuat Rancang Bangun *Game Platformers 2D "Foxy Tales"* Bergenre *Adventure* Menggunakan *Unity Game Engine*.
3. Berdasarkan kuesioner rata – rata share 94,44% dari 8 responden menyatakan game Foxy Tales memiliki visual, grafik, dan audio yang sangat baik. Foxy Tales juga cukup baik sebagai hiburan dan pembelajaran, secara keseluruhan Foxy Tales memiliki tampilan yang baik dan cukup mudah dimengerti oleh pengguna serta baik digunakan sebagai media hiburan dan media pembelajaran.
4. Dari hasil pengujian, dapat diketahui aplikasi dapat dijalankan pada *macOS* dengan spesifikasi *Processor 2 GHz*, memiliki *2 GB RAM*, dengan *Graphics 256 MB video memory*, aplikasi dinyatakan optimal dan tidak ada masalah pada saat dijalankan.
5. Pengujian aplikasi dapat dijalankan pada *Windows* dengan spesifikasi *Intel Pentium Dual-Core G4400 3.30 GHz* atau bisa lebih tinggi, dengan *OS Windows 7 64bit, 4 GB RAM, Graphics Card Nvidia GeForce GTX*

750 atau bisa lebih tinggi, aplikasi berjalan dengan optimal dan tidak ada masalah saat dijalankan.

5.2 Saran

Adapun saran dan harapan yang penulis berikan untuk penelitian ini selanjutnya adalah sebagai berikut :

1. Menambahkan fitur *setting* dalam *game*, menambahkan *level* yang lebih menarik dan rintangan – rintangan yang lebih sulit lagi, menambahkan berbagai macam kuis lagi seperti essay dan pilihan ganda, menambahkan objek *sprite* lainnya seperti musuh dan *collectible item* lainnya, dan menambahkan algoritma *pathfinding* untuk musuh seperti *A** atau *Djikstra*.
2. Diharapkan pengembangan *game* mengarah ke *game mobile* berbasis *android* atau *ios* dan memiliki ciri khas seperti unsur budaya – budaya daerah kalimantan tengah.

DAFTAR PUSTAKA

Abdul Kadir, A. S. (2013). *Teori dan Aplikasi, Pengolahan Citra.* Semarang: ANDI.

Adams, E. (2014). *Fundamental of Game Design.* United States: New Riders.

Agus Perdana Windarto, D. N. (2020). *Jaringan Syaraf Tiruan : Algoritma Prediksi dan Implementasi.* Yayasan Kita Menulis .

Ahmad, A. (2017). Mengenal Artificial Intelligence, Machine Learning, Neural Network, dan Deep Learning. *ACADEMIA*, 2.

Dahria, M. (2008). Kecerdasan Buatan (Artificial Intelligence). *Saintikom*, 185.

Dr. Marina Silalahi, M. (2016). *Morfologi Tumbuhan.* Jakarta.

Ekawati Yulsilviana, H. E. (2019). PENERAPAN METODE FINITE STATE MACHINE (FSM) PADA GAME. *SEBATIK*, 117.

Ferdinand F, A. M. (2009). *Praktis Belajar Biologi* (I ed.). Jakarta: Pusat Perbukuan Departemen Pendidikan Nasional.

Harlanto, R. A. (2020, Juni 16). *Berkenalan dengan Fitur-Fitur Unity 3D*. Diambil kembali dari GAMELAB Indonesia: https://www.gamelab.id/news/211-berkenalan-dengan-fitur-fitur-unity-3d

Hendini, A. (2016). PEMODELAN UML SISTEM INFORMASI MONITORING PENJUALAN DAN STOK . *JURNAL KHATULISTIWA INFORMATIKA, IV*, 108-110.

Klappenbach, M. (2019, 11 27). *What is a Platform Game?* Diambil kembali dari LifeWife: https://www.lifewire.com/what-is-a-platform-game-812371

Mustika, E. P. (2017). Pengembangan Media Pembelajaran Interaktif dengan Menggunakan Metode Multimedia Development Life Cycle. *JOIN (Jurnal Online Informatika)*, 122-123.

Pratama, W. (2014). GAME ADVENTURE MISTERI KOTAK PANDORA. 13.

Pulung Nurtantio Andono, T. M. (2018). *Pengolahan Citra Digital* (I ed.). Yogyakarta: ANDI.

Putra, D. (2010). *Pengolahan Citra Digital.* Yogyakarta: C.V ANDI.

Retnawati, H. (2015). PERBANDINGAN AKURASI PENGGUNAAN SKALA LIKERT DAN PILIHAN GANDA UNTUK MENGUKUR SELF-REGULATED LEARNING. *JURNAL KEPENDIDIKAN*, 158 - 159.

Rikky Firmansyah, A. M. (2012). *Mudah dan Aktif Belajar Biologi* (I ed.). Jakarta: Pusat Perbukuan Departemen Pendidikan Nasional.

Rosa Andrie Asmara, S. M. (2018). *Pengolahan Citra Digital* (I ed.). Malang: POLINEMA PRESS.

Siti Asmiatun, A. N. (2017). *Belajar Membuat Game 2D dan 3D Menggunakan Unity*. Yogyakarta: Deepublish.

Solikhun Solikhun, M. W. (2020). *JARINGAN SARAF TIRUAN Backpropagation Pengenalan Pola Calon Debitur.* Yayasan Kita Menulis.

Sugiyono, P. D. (2019). *Metode Penelitian dan Pengembangan* (4th ed.). Yogyakarta: Abata press.

Sulistyorini, A. (2012). *Biologi 1* (I ed.). Jakarta: PT. Balai Pustaka.

Umi Khulsum, Y. H. (2018). PENGEMBANGAN BAHAN AJAR MENULIS CERPEN DENGAN MEDIA STORYBOARDPADA SISWA KELAS X SMA. *DIGLOSIA*, 6.

Vince. (2018, April 12). *The Many Different Types of Video Games & Their Subgenres*. Diambil kembali dari iD Tech: https://www.idtech.com/blog/different-types-of-video-game-genres

Wahyu. (2018, Juli 12). *Apa Itu Unity 3D*. Diambil kembali dari eventkampus: https://eventkampus.com/blog/detail/1474/apa-itu-unity-3d

Wibawanto, W. (2020). *Game Edukasi RPG (Role Play Game).* Semarang: LPPM UNNES.

Yee, N. (2015, Desember 15). *The Gamer Motivation Model in Handy Reference Chart and Slides*. Diambil kembali dari Quanty Foundry: https://quanticfoundry.com/2015/12/15/handy-reference/

Zonyfar, C. (2020). *Pengolahan Citra Digital : Sebuah Pengantar*. BANTEN: DESANTA MULIAVISITAMA.

SEKOLAH TINGGI MANAJEMEN INFORMATIKA DAN KOMPUTER
(STMIK) PALANGKARAYA
Jl. G. Obos No.114 Telp.0536-3224593, 3225515 Fax.0536-3225515 Palangka Raya
email : humas@stmikplk.ac.id - website : www.stmikplk.ac.id

SURAT TUGAS PENGUJI TUGAS AKHIR

No. 142/STMIK-3.C.2/KP/V/0

Ketua Program Studi Teknik Informatika Sekolah Tinggi Manajemen Informatika dan Komputer (STMIK) Palangkaraya menugaskan kepada nama- nama berikut :

1. Nama : Sulistyowati, S.Kom., M.Cs.
   NIK : 198212162007002
   Sebagai Ketua
2. Nama : Rommi Kaestria, M.Kom.
   NIK : 198605242011103
   Sebagai Sekretaris
3. Nama : Veny Cahya Hardita, M.Kom
   NIK : 199504302020002
   Sebagai Anggota
4. Nama : Lili Rusdiana, M.Kom.
   NIK : 198707282011007
   Sebagai Anggota
5. Nama : Catharina Elmayantie, M.Pd.
   NIK : 197610252015003
   Sebagai Anggota

Tim Penguji Tugas Akhir Mahasiswa :

Nama	:	Muhammad Mawandi Anajeli
NIM	:	C1655201045
Hari/ Tanggal Ujian	:	Selasa, 31 Mei 2022
Waktu	:	09.00 WIB
Judul Tugas Akhir	:	Rancang Bangun Game Platformer 2D "Foxy Tales" Bergenre Adventure Menggunakan Unity Game Engine

Demikian surat ini dibuat agar dapat dipergunakan sebagaimana mestinya dan dilaksanakan dengan penuh tanggung jawab.

Palangka Raya, 30 Mei 2022
Ketua Program Studi Teknik Informatika,

Lili Rusdiana, M.Kom.
NIK. 198707282011007

Tembusan :
1. Dosen Penguji
2. Mahasiswa yang Bersangkutan
3. Arsip

SEKOLAH TINGGI MANAJEMEN INFORMATIKA DAN KOMPUTER
(STMIK) PALANGKARAYA
Jl G Obos No 114 Telp.0536-3224593, 3225515 Fax.0536-3225515 Palangkaraya
email humas@stmikplk.ac.id - website : www.stmikplk.ac.id

BERITA ACARA
UJIAN TUGAS AKHIR

Periode (Bulan) : Mei Tahun 2022

1. Hari/Tanggal Ujian : Selasa / 31 mei 2022
2. Waktu (Jam) : 08.00 WIB sampai dengan ..... WIB
3. Nama Mahasiswa : Muhammad Mawardi Anjuu
4. Nomor Induk Mahasiswa : C1B5T201045
5. Program Studi : Teknik Informatika
6. Tahun Angkatan : 2016
7. Judul Tugas Akhir : Rancang Bangun Game Platformer 2D "Foxy Tales" Bergenre Adventure Menggunakan Unity Game Engine.
8. Dosen Penguji

	Nama	Nilai	Tanda Tangan
1.	Shirlyowati, S.Kom., M.Cs		( )
2.	Rommi Kuswria, M.Kom		( )
3.	Very Cahya Hardila, M.Kom		( )
4.	Lili Kurdiana, M.Kom		( )
5.	Catharina Cimayanthi, M.Pd		( )

9. Hasil Ujian : LULUS / ~~TIDAK LULUS~~ *) NILAI = 82,5

Dengan Perbaikan/ ~~Tanpa Perbaikan~~ *)

10. Catatan Penting :
    1. Lama Perbaikan : 14 hari
    2. Jika lebih dari 1 (satu) bulan dikenakan sanksi berupa denda sebesar Rp. 600.000,- (Enam ratus ribu rupiah) per bulan dari tanggal ujian
    3. Jika lebih dari 3 (tiga) bulan dari tanggal ujian maka hasil ujian dibatalkan dan wajib mengajukan judul dan pembimbing baru

Palangka Raya, 31 Mei 2022

Mengetahui :
Ketua Program Studi Teknik Informatika,

Ika Rahmania, M.Kom.
NIK. 1987072820110007

Ketua Penguji,

NIK.

Tembusan:
1. Arsip Prodi Teknik Informatika
2. Mahasiswa yang bersangkutan

Dibawa saat konsultasi perbaikan dengan dosen penguji

**) Coret yang tidak perlu*

SEKOLAH TINGGI MANAJEMEN INFORMATIKA DAN KOMPUTER
(STMIK) PALANGKARAYA
Jl. G. Obos No.114 Telp.0536-3224593, 3225515 Fax.0536-3225515 Palangkaraya
email : humas@stmikplk.ac.id - website : www.stmikplk.ac.id

KARTU KEGIATAN KONSULTASI
TUGAS AKHIR

Nama Mahasiswa : MUHAMMAD MAWANDI ANAJELI
NIM : C1655201045
Tanggal Persetujuan Judul : 25 November 2020
Judul Tugas Akhir : Rancang Bangun Game PlatFormer 2D "Foxy Tale" Bergenre Adventure Menggunakan Unity Game Engine.

No	Tanggal Konsultasi		Uraian	Tanda Tangan
	Terima	Kembali		
1	20/nov/2020	20/nov/2020	Pengajuan judul tugas akhir	
2	25/nov/2020	25/nov/2020	Pergantian judul di acc, Pengambilan Surat tugas.	
3	20/Feb/2021	1/mar/2021	Revisi Bab 1 dan Bab 2.	
4	22/Agus/2021	22/Agus/2021	Pergantian judul tugas akhir	
5	9/Okto/2021	9/okto/2021	Revisi judul baru dan Bab 1, 2, dan 3	
6	22/nov/2021	22/nov/2021	Perbaikan latar belakang dan perancangan Storyboard	
7	15/Jan/2022	15/Jan/2022	Perencanaan tanggal seminar Proposal Acc	
8	18/Jan/2022	18/Jan/2022	Revisi Bab 2	
9	19/Jan/2022	19/Jan/2022	Revisi proposal selesai diperbaiki.	
10	26/4 2022		lanjutkan program dan penulisan	
	9/5/2022		Acc program dan lanjutkan pelaporan	
	11/5/2022		Penyelesaian akhir	
	17/5/2022		Penyelesaian antara Program & penulisan laporan	
	21/5		lengkapi untuk paporlaporan	
	23/5		Acc Sidang	

No	Tanggal Konsultasi		Uraian	Tanda Tangan
	Terima	Kembali		
	21/5 2022	23/5 2022	- Perbaiki halaman persetujuan	
			- Perbaiki halaman pengesahan	
			- Lengkapi naskah dengan : abstrak, kata pengantar, Lampiran - lampiran	
			- Hilangkan Jadwal Penelitian pada naskah	
			- Perbaiki responden, terutama tabel halaman 54	
			- Perbaiki cara penulisan Daftar Pustaka	

Menyetujui :

Dosen Pembimbing I,

Lili Rusdiana, M.kom.

Dosen Pembimbing II,

Catharina Elmayantie, M.Pd.

Kuesioner Respon Pengguna Game Foxy Tales

Bercerita tentang rubah bernama Foxy, seekor rubah yang hendak menyelamatkan pulau Sunnyland dari musuh – musuh yang menjajah pulau tersebut, tentu dalam aksi penyelamatannya Foxy harus melewati segala rintangan, memecahkan teka – teki , dan mengalahkan musuh – musuh yang menghadangnya hingga pulau Sunnyland pun aman dan tentram kembali

Nama *

Aditya Bakti

Umur *

23

1. Tampilan antar muka dalam aplikasi game menarik *

- (●) Setuju Biasa
- ( ) saja
- ( )

2. Cara bermain atau petunjuk dalam game mudah dipahami *

- ( ) Setuju
- (●) Biasa saja
- ( )

3. Tampilan objek seperti player, musuh, dan pixel art dari game cukup menarik *

- (●) Setuju Biasa
- ( ) saja
- ( )

4. Musik dan efek suara dari game menarik *

- (●) Setuju Biasa
- ( ) saja
- ( )

5. Konfigurasi dalam game mudah dipahami *

- (●) Setuju Biasa
- ( ) saja
- ( )

6. Game ini menyenangkan dan friendly *

- (●) Setuju Biasa
- ( ) saja
- ( )

Kuesioner Respon Pengguna Game Foxy Tales

Bercerita tentang rubah bernama Foxy, seekor rubah yang hendak menyelamatkan pulau Sunnyland dari musuh – musuh yang menjajah pulau tersebut, tentu dalam aksi penyelamatannya Foxy harus melewati segala rintangan, memecahkan teka – teki , dan mengalahkan musuh – musuh yang menghadangnya hingga pulau Sunnyland pun aman dan tentram kembali

Nama *

Amanda

Umur *

22

1. Tampilan antar muka dalam aplikasi game menarik *

- (●) Setuju Biasa
- ( ) saja
- ( )

2. Cara bermain atau petunjuk dalam game mudah dipahami *

- (●) Setuju Biasa
- ( ) saja
- ( )

3. Tampilan objek seperti player, musuh, dan pixel art dari game cukup menarik *

- (●) Setuju
- ( ) Biasa saja
- ( ) [illegible]

4. Musik dan efek suara dari game menarik *

- (●) Setuju Biasa
- ( ) saja
- ( )

5. Konfigurasi dalam game mudah dipahami *

- (●) Setuju Biasa
- ( ) saja
- ( )

6. Game ini menyenangkan dan friendly *

- (●) Setuju Biasa
- ( ) saja
- ( )

Kuesioner Respon Pengguna Game Foxy Tales

Bercerita tentang rubah bernama Foxy, seekor rubah yang hendak menyelamatkan pulau Sunnyland dari musuh – musuh yang menjajah pulau tersebut, tentu dalam aksi penyelamatannya Foxy harus melewati segala rintangan, memecahkan teka – teki , dan mengalahkan musuh – musuh yang menghadangnya hingga pulau Sunnyland pun aman dan tentram kembali

Nama *

Elizabeth Cipta

Umur *

20

1. Tampilan antar muka dalam aplikasi game menarik *

- (●) Setuju Biasa
- ( ) saja
- ( )

2. Cara bermain atau petunjuk dalam game mudah dipahami *

- ( ) Setuju Biasa
- (●) saja
- ( )

3. Tampilan objek seperti player, musuh, dan pixel art dari game cukup menarik *

- [ ] Setuju
- [x] Biasa saja
- [ ]

4. Musik dan efek suara dari game menarik *

- [x] Setuju Biasa
- [ ] saja
- [ ]

5. Konfigurasi dalam game mudah dipahami *

- [x] Setuju Biasa
- [ ] saja
- [ ]

6. Game ini menyenangkan dan friendly *

- [x] Setuju Biasa
- [ ] saja
- [ ]

Kuesioner Respon Pengguna Game Foxy Tales

Bercerita tentang rubah bernama Foxy, seekor rubah yang hendak menyelamatkan pulau Sunnyland dari musuh – musuh yang menjajah pulau tersebut, tentu dalam aksi penyelamatannya Foxy harus melewati segala rintangan, memecahkan teka – teki , dan mengalahkan musuh – musuh yang menghadangnya hingga pulau Sunnyland pun aman dan tentram kembali

Nama *

Yoga Kristiano

Umur *

23

1. Tampilan antar muka dalam aplikasi game menarik *

- (●) Setuju Biasa
- ( ) saja
- ( )

2. Cara bermain atau petunjuk dalam game mudah dipahami *

- (●) Setuju Biasa
- ( ) saja
- ( )

3. Tampilan objek seperti player, musuh, dan pixel art dari game cukup menarik *

- (●) Setuju
- ( ) Biasa saja
- ( ) [illegible]

4. Musik dan efek suara dari game menarik *

- (●) Setuju Biasa
- ( ) saja
- ( )

5. Konfigurasi dalam game mudah dipahami *

- (●) Setuju Biasa
- ( ) saja
- ( )

6. Game ini menyenangkan dan friendly *

- (●) Setuju Biasa
- ( ) saja
- ( )

Kuesioner Respon Pengguna Game Foxy Tales

Bercerita tentang rubah bernama Foxy, seekor rubah yang hendak menyelamatkan pulau Sunnyland dari musuh – musuh yang menjajah pulau tersebut, tentu dalam aksi penyelamatannya Foxy harus melewati segala rintangan, memecahkan teka – teki , dan mengalahkan musuh – musuh yang menghadangnya hingga pulau Sunnyland pun aman dan tentram kembali

Nama *

Wanda

Umur *

21

1. Tampilan antar muka dalam aplikasi game menarik *

- (●) Setuju Biasa
- ( ) saja
- ( )

2. Cara bermain atau petunjuk dalam game mudah dipahami *

- (●) Setuju Biasa
- ( ) saja
- ( )

3. Tampilan objek seperti player, musuh, dan pixel art dari game cukup menarik *

- ○ Setuju
- ◉ Biasa saja
- ○ [illegible]

4. Musik dan efek suara dari game menarik *

- ○ Setuju Biasa
- ◉ saja
- ○

5. Konfigurasi dalam game mudah dipahami *

- ◉ Setuju Biasa
- ○ saja
- ○

6. Game ini menyenangkan dan friendly *

- ◉ Setuju Biasa
- ○ saja
- ○

Kuesioner Respon Pengguna Game Foxy Tales

Bercerita tentang rubah bernama Foxy, seekor rubah yang hendak menyelamatkan pulau Sunnyland dari musuh – musuh yang menjajah pulau tersebut, tentu dalam aksi penyelamatannya Foxy harus melewati segala rintangan, memecahkan teka – teki , dan mengalahkan musuh – musuh yang menghadangnya hingga pulau Sunnyland pun aman dan tentram kembali

Nama *

Marlo

Umur *

20

1. Tampilan antar muka dalam aplikasi game menarik *

- (●) Setuju Biasa
- ( ) saja
- ( )

2. Cara bermain atau petunjuk dalam game mudah dipahami *

- (●) Setuju Biasa
- ( ) saja
- ( )

3. Tampilan objek seperti player, musuh, dan pixel art dari game cukup menarik *

- (●) Setuju
- ( ) Biasa saja
- ( ) [illegible]

4. Musik dan efek suara dari game menarik *

- (●) Setuju Biasa
- ( ) saja
- ( )

5. Konfigurasi dalam game mudah dipahami *

- (●) Setuju Biasa
- ( ) saja
- ( )

6. Game ini menyenangkan dan friendly *

- (●) Setuju Biasa
- ( ) saja
- ( )

Kuesioner Respon Pengguna Game Foxy Tales

Bercerita tentang rubah bernama Foxy, seekor rubah yang hendak menyelamatkan pulau Sunnyland dari musuh – musuh yang menjajah pulau tersebut, tentu dalam aksi penyelamatannya Foxy harus melewati segala rintangan, memecahkan teka – teki , dan mengalahkan musuh – musuh yang menghadangnya hingga pulau Sunnyland pun aman dan tentram kembali

Nama *

Yuan

Umur *

20

1. Tampilan antar muka dalam aplikasi game menarik *

- (●) Setuju Biasa
- ( ) saja
- ( )

2. Cara bermain atau petunjuk dalam game mudah dipahami *

- (●) Setuju Biasa
- ( ) saja
- ( )

3. Tampilan objek seperti player, musuh, dan pixel art dari game cukup menarik *

- (●) Setuju
- ( ) Biasa saja
- ( ) [illegible]

4. Musik dan efek suara dari game menarik *

- (●) Setuju Biasa
- ( ) saja
- ( )

5. Konfigurasi dalam game mudah dipahami *

- (●) Setuju Biasa
- ( ) saja
- ( )

6. Game ini menyenangkan dan friendly *

- (●) Setuju Biasa
- ( ) saja
- ( )

Kuesioner Respon Pengguna Game Foxy Tales

Bercerita tentang rubah bernama Foxy, seekor rubah yang hendak menyelamatkan pulau Sunnyland dari musuh – musuh yang menjajah pulau tersebut, tentu dalam aksi penyelamatannya Foxy harus melewati segala rintangan, memecahkan teka – teki , dan mengalahkan musuh – musuh yang menghadangnya hingga pulau Sunnyland pun aman dan tentram kembali

Nama *

Reki

Umur *

21

1. Tampilan antar muka dalam aplikasi game menarik *

- (●) Setuju Biasa
- ( ) saja
- ( )

2. Cara bermain atau petunjuk dalam game mudah dipahami *

- (●) Setuju Biasa
- ( ) saja
- ( )

3. Tampilan objek seperti player, musuh, dan pixel art dari game cukup menarik *

- (●) Setuju
- ( ) Biasa saja
- ( ) [illegible]

4. Musik dan efek suara dari game menarik *

- (●) Setuju Biasa
- ( ) saja
- ( )

5. Konfigurasi dalam game mudah dipahami *

- (●) Setuju Biasa
- ( ) saja
- ( )

6. Game ini menyenangkan dan friendly *

- (●) Setuju Biasa
- ( ) saja
- ( )

LISTING PROGRAM

```
public class AudioManager :
MonoBehaviour
{
    public static AudioManager
instance;

    public AudioSource[]
soundEffects;

    public AudioSource bgm,
levelEndMusic, bossMusic;

    private void Awake()
    {
        instance = this;
    }

    // Start is called before
the first frame update
    void Start()
    {

    }

    // Update is called once
per frame
    void Update()
    {

    }

    public void PlaySFX(int
soundToPlay)
    {

soundEffects[soundToPlay].Stop
();


soundEffects[soundToPlay].pitc
h = Random.Range(.9f, 1.1f);


soundEffects[soundToPlay].Play
();
    }

    public void
PlayLevelVictory()
    {
        bgm.Stop();
        levelEndMusic.Play();
    }

    public void
PlayBossMusic()
    {
        bgm.Stop();
        bossMusic.Play();
    }

    public void
StopBossMusic()
    {
        bossMusic.Stop();
        bgm.Play();
    }
}

public class BossActivator :
MonoBehaviour
{
    public GameObject
theBossBattle;

    // Start is called before
the first frame update
    void Start()
    {

    }

    // Update is called once
per frame
    void Update()
    {

    }
    private void
OnTriggerEnter2D(Collider2D
other)
    {
        if(other.tag ==
"Player")
        {

theBossBattle.SetActive(true);


gameObject.SetActive(false);


AudioManager.instance.PlayBoss
Music();
        }
    }
}
```

```
public class BossBullet :
MonoBehaviour
{
    public float speed;

    // Start is called before
the first frame update
    void Start()
    {

AudioManager.instance.PlaySFX(
2);
    }
    // Update is called once
per frame
    void Update()
    {
        transform.position +=
new Vector3(-speed *
transform.localScale.x *
Time.deltaTime, 0f, 0f);
    }

    private void
OnTriggerEnter2D(Collider2D
other)
    {
        if(other.tag ==
"Player")
        {

PlayerHealthController.instanc
e.DealDamage();
        }


AudioManager.instance.PlaySFX(
1);

        Destroy(gameObject);
    }
}
public class
BossTankController :
MonoBehaviour
{
    public enum bossStates {
shooting, hurt, moving, ended
};
    public bossStates
currentState;

    public Transform theBoss;
    public Animator anim;

    [Header("Movement")]
    public float moveSpeed;
    public Transform
leftPoint, rightPoint;
    private bool moveRight;
    public GameObject mine;
    public Transform
minePoint;
    public float
timeBetweenMines;
    private float mineCounter;

    [Header("Shooting")]
    public GameObject bullet;
    public Transform
firePoint;
    public float
timeBetweenShots;
    private float shotCounter;

    [Header("Hurt")]
    public float hurtTime;
    private float hurtCounter;
    public GameObject hitBox;

    [Header("Health")]
    public int health = 5;
    public GameObject
explosion, winPlatform;
    private bool isDefeated;
    public float shotSpeedUp,
mineSpeedUp;



    // Start is called before
the first frame update
    void Start()
    {
        currentState =
bossStates.shooting;
    }

    // Update is called once
per frame
    void Update()
    {
        switch(currentState)
        {
            case
bossStates.shooting:

                shotCounter -=
Time.deltaTime;

                if(shotCounter
<= 0)
```

```
                {

shotCounter =
timeBetweenShots;

                    var
newBullet =
Instantiate(bullet,
firePoint.position,
firePoint.rotation);

newBullet.transform.localScale
= theBoss.localScale;
                }

                break;

            case
bossStates.hurt:

                if(hurtCounter
> 0)
                {

hurtCounter -= Time.deltaTime;


if(hurtCounter <= 0)
                    {

currentState =
bossStates.moving;


mineCounter = 0;


if(isDefeated)
                        {

theBoss.gameObject.SetActive(f
alse);

Instantiate(explosion,
theBoss.position,
theBoss.rotation);


winPlatform.SetActive(true);


AudioManager.instance.StopBoss
Music();


currentState =
bossStates.ended;
                        }
                    }
                }

                break;

            case
bossStates.moving:

                if(moveRight)
                {

theBoss.position += new
Vector3(moveSpeed *
Time.deltaTime, 0f, 0f);


if(theBoss.position.x >
rightPoint.position.x)
                    {

theBoss.localScale =
Vector3.one;


moveRight = false;


EndMovement();
                    }
                } else
                {

theBoss.position -= new
Vector3(moveSpeed *
Time.deltaTime, 0f, 0f);

                    if
(theBoss.position.x <
leftPoint.position.x)
                    {

theBoss.localScale = new
Vector3(-1f, 1f, 1f);


moveRight = true;


EndMovement();
                    }
                }
```

```
                mineCounter -=
Time.deltaTime;

                if(mineCounter
<= 0)
                {

mineCounter =
timeBetweenMines;


Instantiate(mine,
minePoint.position,
minePoint.rotation);
                }

                break;
        }

#if UNITY_EDITOR


if(Input.GetKeyDown(KeyCode.H)
)
        {
            TakeHit();
        }

#endif
    }

    public void TakeHit()
    {
        currentState =
bossStates.hurt;
        hurtCounter =
hurtTime;


anim.SetTrigger("Hit");


AudioManager.instance.PlaySFX(
0);

        BossTankMine[] mines =
FindObjectsOfType<BossTankMine
>();
        if(mines.Length > 0)
        {

foreach(BossTankMine foundMine
in mines)
            {

foundMine.Explode();
            }
        }

        health--;

        if(health <= 0)
        {
            isDefeated = true;
        } else
        {
            timeBetweenShots
/= shotSpeedUp;
            timeBetweenMines
/= mineSpeedUp;

        }
    }

    private void EndMovement()
    {
        currentState =
bossStates.shooting;

        shotCounter = 0f;


anim.SetTrigger("StopMoving");


hitBox.SetActive(true);
    }
}
public class BossTankHitBox :
MonoBehaviour
{
    public BossTankController
bossCont;

    // Start is called before
the first frame update
    void Start()
    {

    }

    // Update is called once
per frame
    void Update()
    {

    }

    private void
OnTriggerEnter2D(Collider2D
other)
    {
```

```
        if (other.tag ==
"Player" &&
PlayerController.instance.tran
sform.position.y >
transform.position.y)
        {

bossCont.TakeHit();


PlayerController.instance.Boun
ce();


gameObject.SetActive(false);
        }
    }
}

public class BossTankMine :
MonoBehaviour
{
    public GameObject
explosion;

    // Start is called before
the first frame update
    void Start()
    {

    }
    // Update is called once
per frame
    void Update()
    {

    }

    private void
OnTriggerEnter2D(Collider2D
other)
    {
        if(other.tag ==
"Player")
        {

Destroy(gameObject);


Instantiate(explosion,
transform.position,
transform.rotation);


PlayerHealthController.instanc
e.DealDamage();


AudioManager.instance.PlaySFX(
3);
        }
    }

    public void Explode()
    {
        Destroy(gameObject);


AudioManager.instance.PlaySFX(
3);

        Instantiate(explosion,
transform.position,
transform.rotation);
    }
}
public class CameraController
: MonoBehaviour
{
    public static
CameraController instance;

    public Transform target;

    public Transform
farBackground,
middleBackground;

    public float minHeight,
maxHeight;

    public bool stopFollow;

    //private float lastXPos;
    private Vector2 lastPos;

    private void Awake()
    {
        instance = this;
    }

    // Start is called before
the first frame update
    void Start()
    {
        //lastXPos =
transform.position.x;
        lastPos =
transform.position;
    }
```

```
    // Update is called once
per frame
    void Update()
    {
        /* transform.position
= new
Vector3(target.position.x,
target.position.y,
transform.position.z);

        float clampedY =
Mathf.Clamp(transform.position
.y, minHeight, maxHeight);
        transform.position =
new
Vector3(transform.position.x,
clampedY,
transform.position.z); */

        if (!stopFollow)
        {
            transform.position
= new
Vector3(target.position.x,
Mathf.Clamp(target.position.y,
minHeight, maxHeight),
transform.position.z);

            //float
amountToMoveX =
transform.position.x -
lastXPos;
            Vector2
amountToMove = new
Vector2(transform.position.x -
lastPos.x,
transform.position.y -
lastPos.y);


farBackground.position =
farBackground.position + new
Vector3(amountToMove.x,
amountToMove.y, 0f);

middleBackground.position +=
new Vector3(amountToMove.x,
amountToMove.y, 0f) * .5f;

            //lastXPos =
transform.position.x;
            lastPos =
transform.position;
        }
    }
}
```

```
public class Checkpoint :
MonoBehaviour
{
    public SpriteRenderer
theSR;

    public Sprite cpOn, cpOff;


    // Start is called before
the first frame update
    void Start()
    {

    }

    // Update is called once
per frame
    void Update()
    {

    }

    private void
OnTriggerEnter2D(Collider2D
other)
    {

if(other.CompareTag("Player"))
        {

CheckpointController.instance.
DeactivateCheckpoints();

            theSR.sprite =
cpOn;


CheckpointController.instance.
SetSpawnPoint(transform.positi
on);
        }
    }

    public void
ResetCheckpoint()
    {
        theSR.sprite = cpOff;
    }
}

public class
CheckpointController :
MonoBehaviour
{
```

```
    public static
CheckpointController instance;

    private Checkpoint[]
checkpoints;

    public Vector3 spawnPoint;

    private void Awake()
    {
        instance = this;
    }

    // Start is called before
the first frame update
    void Start()
    {
        checkpoints =
FindObjectsOfType<Checkpoint>(
);

        spawnPoint =
PlayerController.instance.tran
sform.position;
    }

    // Update is called once
per frame
    void Update()
    {

    }

    public void
DeactivateCheckpoints()
    {
        for(int i = 0; i <
checkpoints.Length; i++)
        {

checkpoints[i].ResetCheckpoint
();
        }


    }

    public void
SetSpawnPoint(Vector3
newSpawnPoint)
    {
        spawnPoint =
newSpawnPoint;
    }
}
```

```
public class DamagePlayer :
MonoBehaviour
{
    // Start is called before
the first frame update
    void Start()
    {

    }

    // Update is called once
per frame
    void Update()
    {

    }

    private void
OnTriggerEnter2D(Collider2D
other)
    {
        if (other.tag ==
"Player")
        {


//Debug.Log("Hit");


//FindObjectOfType<PlayerHealt
hController>().DealDamage();


PlayerHealthController.instanc
e.DealDamage();

        }
    }
}
public class DestroyOverTime :
MonoBehaviour
{
    public float lifeTime;

    // Start is called before
the first frame update
    void Start()
    {

    }

    // Update is called once
per frame
    void Update()
    {
```

```
        Destroy(gameObject,
lifeTime);
    }
}

public class EnemyController :
MonoBehaviour
{
    public float moveSpeed;

    public Transform
leftPoint, rightPoint;

    private bool movingRight;

    private Rigidbody2D theRB;
    public SpriteRenderer
theSR;
    private Animator anim;

    public float moveTime,
waitTime;
    private float moveCount,
waitCount;

    // Start is called before
the first frame update
    void Start()
    {
        theRB =
GetComponent<Rigidbody2D>();
        anim =
GetComponent<Animator>();

        leftPoint.parent =
null;
        rightPoint.parent =
null;

        movingRight = true;

        moveCount = moveTime;
    }

    // Update is called once
per frame
    void Update()
    {
        if (moveCount > 0)
        {
            moveCount -=
Time.deltaTime;

            if (movingRight)
            {
                theRB.velocity
= new Vector2(moveSpeed,
theRB.velocity.y);

                theSR.flipX =
true;

                if
(transform.position.x >
rightPoint.position.x)
                {

movingRight = false;
                }
            }
            else
            {
                theRB.velocity
= new Vector2(-moveSpeed,
theRB.velocity.y);

                theSR.flipX =
false;

                if
(transform.position.x <
leftPoint.position.x)
                {

movingRight = true;
                }
            }

            if(moveCount <= 0)
            {
                waitCount =
Random.Range(waitTime * .75f,
waitTime * 1.25f);
            }


anim.SetBool("isMoving",
true);
        } else if(waitCount >
0)
        {
            waitCount -=
Time.deltaTime;
            theRB.velocity =
new Vector2(0f,
theRB.velocity.y);

            if(waitCount <= 0)
            {
```

```
                moveCount =
Random.Range(moveTime * .75f,
waitTime * .75f);
            }

anim.SetBool("isMoving",
false);
        }
    }
}
public class KillPlayer :
MonoBehaviour
{
    // Start is called before
the first frame update
    void Start()
    {

    }

    // Update is called once
per frame
    void Update()
    {

    }

    private void
OnTriggerEnter2D(Collider2D
other)
    {
        if(other.tag ==
"Player")
        {

LevelManager.instance.RespawnP
layer();
        }
    }
}

public class LevelExit :
MonoBehaviour
{
    // Start is called before
the first frame update
    void Start()
    {

    }

    // Update is called once
per frame
    void Update()
    {

    }

    private void
OnTriggerEnter2D(Collider2D
other)
    {
        if(other.tag ==
"Player")
        {

LevelManager.instance.EndLevel
();
        }
    }
}

public class LevelManager :
MonoBehaviour
{
    public static LevelManager
instance;

    public float
waitToRespawn;

    public int gemsCollected;

    public string levelToLoad;

    public float timeInLevel;

    private void Awake()
    {
        instance = this;
    }

    // Start is called before
the first frame update
    void Start()
    {
        timeInLevel = 0f;
    }

    // Update is called once
per frame
    void Update()
    {
        timeInLevel +=
Time.deltaTime;
    }

    public void
RespawnPlayer()
    {

StartCoroutine(RespawnCo());
```

```
    }

    private IEnumerator
RespawnCo()
    {

PlayerController.instance.game
Object.SetActive(false);

AudioManager.instance.PlaySFX(
8);

        yield return new
WaitForSeconds(waitToRespawn -
(1f /
UIController.instance.fadeSpee
d));


UIController.instance.FadeToBl
ack();

        yield return new
WaitForSeconds((1f /
UIController.instance.fadeSpee
d) + .2f);


UIController.instance.FadeFrom
Black();


PlayerController.instance.game
Object.SetActive(true);


PlayerController.instance.tran
sform.position =
CheckpointController.instance.
spawnPoint;


PlayerHealthController.instanc
e.currentHealth =
PlayerHealthController.instanc
e.maxHealth;

UIController.instance.UpdateHe
althDisplay();
    }

    public void EndLevel()
    {

StartCoroutine(EndLevelCo());
    }

    public IEnumerator
EndLevelCo()
    {

AudioManager.instance.PlayLeve
lVictory();


PlayerController.instance.stop
Input = true;


CameraController.instance.stop
Follow = true;


UIController.instance.levelCom
pleteText.SetActive(true);

        yield return new
WaitForSeconds(1.5f);


UIController.instance.FadeToBl
ack();

        yield return new
WaitForSeconds((1f /
UIController.instance.fadeSpee
d) + 3f);


PlayerPrefs.SetInt(SceneManage
r.GetActiveScene().name +
"_unlocked", 1);

PlayerPrefs.SetString("Current
Level",
SceneManager.GetActiveScene().
name);

        if
(PlayerPrefs.HasKey(SceneManag
er.GetActiveScene().name +
"_gems"))
        {
            if(gemsCollected >
PlayerPrefs.GetInt(SceneManage
r.GetActiveScene().name +
"_gems"))
            {

PlayerPrefs.SetInt(SceneManage
r.GetActiveScene().name +
"_gems", gemsCollected);
```

```
            }
        }
        else
        {

PlayerPrefs.SetInt(SceneManage
r.GetActiveScene().name +
"_gems", gemsCollected);
        }

        if
(PlayerPrefs.HasKey(SceneManag
er.GetActiveScene().name +
"_time"))
        {
            if(timeInLevel <
PlayerPrefs.GetFloat(SceneMana
ger.GetActiveScene().name +
"_time"))
            {

PlayerPrefs.SetFloat(SceneMana
ger.GetActiveScene().name +
"_time", timeInLevel);
            }
        }
        else
        {
PlayerPrefs.SetFloat(SceneMana
ger.GetActiveScene().name +
"_time", timeInLevel);
        }


SceneManager.LoadScene(levelTo
Load);
    }
}

public class
LSCameraController :
MonoBehaviour
{
    public Vector2 minPos,
maxPos;

    public Transform target;

    // Start is called before
the first frame update
    void Start()
    {

    }

    // Update is called once
per frame
    void LateUpdate()
    {
        float xPos =
Mathf.Clamp(target.position.x,
minPos.x, maxPos.x);
        float yPos =
Mathf.Clamp(target.position.y,
minPos.y, maxPos.y);

        transform.position =
new Vector3(xPos, yPos,
transform.position.z);
    }
}
public class LSManager :
MonoBehaviour
{
    public LSPlayer thePlayer;

    private MapPoint[]
allPoints;

    // Start is called before
the first frame update
    void Start()
    {
        allPoints =
FindObjectsOfType<MapPoint>();


if(PlayerPrefs.HasKey("Current
Level"))
        {
            foreach(MapPoint
point in allPoints)
            {

if(point.levelToLoad ==
PlayerPrefs.GetString("Current
Level"))
                {

thePlayer.transform.position =
point.transform.position;

thePlayer.currentPoint =
point;
                }
            }
        }
    }

    // Update is called once
per frame
```

```
    void Update()
    {

    }

    public void LoadLevel()
    {

StartCoroutine(LoadLevelCo());
    }

    public IEnumerator
LoadLevelCo()
    {

AudioManager.instance.PlaySFX(
4);


LSUIController.instance.FadeTo
Black();

        yield return new
WaitForSeconds((1f /
LSUIController.instance.fadeSp
eed) + .25f);


SceneManager.LoadScene(thePlay
er.currentPoint.levelToLoad);
    }
}

public class LSPlayer :
MonoBehaviour
{
    public MapPoint
currentPoint;

    public float moveSpeed =
10f;

    private bool levelLoading;

    public LSManager
theManager;

    // Start is called before
the first frame update
    void Start()
    {

    }

    // Update is called once
per frame
    void Update()
    {
        transform.position =
Vector3.MoveTowards(transform.
position,
currentPoint.transform.positio
n, moveSpeed *
Time.deltaTime);

        if
(Vector3.Distance(transform.po
sition,
currentPoint.transform.positio
n) < .1f && !levelLoading)
        {

            if
(Input.GetAxisRaw("Horizontal"
) > .5f)
            {
                if
(currentPoint.right != null)
                {

SetNextPoint(currentPoint.righ
t);
                }
            }

            if
(Input.GetAxisRaw("Horizontal"
) < -.5f)
            {
                if
(currentPoint.left != null)
                {

SetNextPoint(currentPoint.left
);
                }
            }

            if
(Input.GetAxisRaw("Vertical")
> .5f)
            {
                if
(currentPoint.up != null)
                {

SetNextPoint(currentPoint.up);
                }
            }
```

```
            if
(Input.GetAxisRaw("Vertical")
< -.5f)
            {
                if
(currentPoint.down != null)
                {

SetNextPoint(currentPoint.down
);
                }
            }


if(currentPoint.isLevel &&
currentPoint.levelToLoad != ""
&& !currentPoint.isLocked)
            {

LSUIController.instance.ShowIn
fo(currentPoint);


if(Input.GetButtonDown("Jump")
)
                {

levelLoading = true;


theManager.LoadLevel();
                }
            }
        }



    }

    public void
SetNextPoint(MapPoint
nextPoint)
    {
        currentPoint =
nextPoint;

LSUIController.instance.HideIn
fo();


AudioManager.instance.PlaySFX(
5);
    }
}

public class LSUIController :
MonoBehaviour
{
    public static
LSUIController instance;

    public Image fadeScreen;
    public float fadeSpeed;
    private bool
shouldFadeToBlack,
shouldFadeFromBlack;

    public GameObject
levelInfoPanel;

    public Text levelName,
gemsFound, gemsTarget,
bestTime, timeTarget;

    private void Awake()
    {
        instance = this;
    }

    // Start is called before
the first frame update
    void Start()
    {
        FadeFromBlack();
    }

    // Update is called once
per frame
    void Update()
    {
        if (shouldFadeToBlack)
        {
            fadeScreen.color =
new Color(fadeScreen.color.r,
fadeScreen.color.g,
fadeScreen.color.b,
Mathf.MoveTowards(fadeScreen.c
olor.a, 1f, fadeSpeed *
Time.deltaTime));
            if
(fadeScreen.color.a == 1f)
            {

shouldFadeToBlack = false;
            }
        }

        if
(shouldFadeFromBlack)
        {
            fadeScreen.color =
new Color(fadeScreen.color.r,
fadeScreen.color.g,
```

```
fadeScreen.color.b,
Mathf.MoveTowards(fadeScreen.c
olor.a, 0f, fadeSpeed *
Time.deltaTime));
            if
(fadeScreen.color.a == 0f)

            {

shouldFadeFromBlack = false;
            }
        }
    }

    public void FadeToBlack()
    {
        shouldFadeToBlack =
true;
        shouldFadeFromBlack =
false;
    }

    public void
FadeFromBlack()
    {
        shouldFadeFromBlack =
true;
        shouldFadeToBlack =
false;
    }

    public void
ShowInfo(MapPoint levelInfo)
    {
        levelName.text =
levelInfo.levelName;

        gemsFound.text =
"FOUND: " +
levelInfo.gemsCollected;
        gemsTarget.text = "IN
LEVEL: " +
levelInfo.totalGems;

        timeTarget.text =
"TARGET: " +
levelInfo.targetTime + "s";

        if(levelInfo.bestTime
== 0)
        {
            bestTime.text =
"BEST: ----";
        } else
        {
            bestTime.text =
"BEST: " +
levelInfo.bestTime.ToString("F
2") + "s";
        }


levelInfoPanel.SetActive(true)
;
    }

    public void HideInfo()
    {

levelInfoPanel.SetActive(false
);
    }
}

public class MainMenu :
MonoBehaviour
{
    public string startScene,
continueScene, aboutScene;

    public GameObject
continueButton;

    // Start is called before
the first frame update
    void Start()
    {
        if
(PlayerPrefs.HasKey(startScene
+ "_unlocked"))
        {

continueButton.SetActive(true)
;
        } else
        {

continueButton.SetActive(false
);
        }


    }

    // Update is called once
per frame
    void Update()
    {

    }
```

```
    public void StartGame()
    {

SceneManager.LoadScene(startSc
ene);


PlayerPrefs.DeleteAll();
    }

    public void ContinueGame()
    {

SceneManager.LoadScene(continu
eScene);
    }

    public void AboutGame()
    {

SceneManager.LoadScene(aboutSc
ene);


PlayerPrefs.DeleteAll();
    }

      public void QuitGame()
    {
        Application.Quit();
        Debug.Log("Quitting
Game");
    }
}

public class MapPoint :
MonoBehaviour
{
    public MapPoint up, right,
down, left;
    public bool isLevel,
isLocked;
    public string levelToLoad,
levelToCheck, levelName;

    public int gemsCollected,
totalGems;
    public float bestTime,
targetTime;

    public GameObject
gemBadge, timeBadge;

    // Start is called before
the first frame update
    void Start()
    {
        if(isLevel &&
levelToLoad != null)
        {

if(PlayerPrefs.HasKey(levelToL
oad + "_gems"))
            {
                gemsCollected
=
PlayerPrefs.GetInt(levelToLoad
+ "_gems");
            }


if(PlayerPrefs.HasKey(levelToL
oad + "_time"))
            {
                bestTime =
PlayerPrefs.GetFloat(levelToLo
ad + "_time");
            }

            if(gemsCollected
>= totalGems)
            {

gemBadge.SetActive(true);
            }

            if(bestTime <=
targetTime && bestTime != 0)
            {

timeBadge.SetActive(true);
            }

            isLocked = true;

            if(levelToCheck !=
null)
            {

if(PlayerPrefs.HasKey(levelToC
heck + "_unlocked"))
                {

if(PlayerPrefs.GetInt(levelToC
heck + "_unlocked") == 1)
                    {

isLocked = false;
                    }
                }
            }
```

```
            if(levelToLoad ==
levelToCheck)
            {
                isLocked =
false;
            }
        }
    }

    // Update is called once
per frame
    void Update()
    {

    }
}

public class PauseMenu :
MonoBehaviour
{
    public static PauseMenu
instance;
    public string levelSelect,
mainMenu;

    public GameObject
pauseScreen;
    public bool isPaused;

    private void Awake()
    {
        instance = this;
    }

    // Start is called before
the first frame update
    void Start()
    {

    }

    // Update is called once
per frame
    void Update()
    {

if(Input.GetKeyDown(KeyCode.Es
cape))
        {
            PauseUnpause();
        }
    }

    public void PauseUnpause()
    {
        if(isPaused)
        {
            isPaused = false;

pauseScreen.SetActive(false);
            Time.timeScale =
1f;
        } else
        {
            isPaused = true;

pauseScreen.SetActive(true);
            Time.timeScale =
0f;
        }
    }

    public void LevelSelect()
    {

PlayerPrefs.SetString("Current
Level",
SceneManager.GetActiveScene().
name);


SceneManager.LoadScene(levelSe
lect);
        Time.timeScale = 1f;
    }

    public void MainMenu()
    {

SceneManager.LoadScene(mainMen
u);
        Time.timeScale = 1f;
    }
}

public class Pickup :
MonoBehaviour
{
    public bool isGem, isHeal;

    private bool isCollected;

    public GameObject
pickupEffect;

    // Start is called before
the first frame update
    void Start()
    {

    }
```

```
    // Update is called once
per frame
    void Update()
    {

    }

    private void
OnTriggerEnter2D(Collider2D
other)
    {

if(other.CompareTag("Player")
&& !isCollected)
        {
            if(isGem)
            {

LevelManager.instance.gemsColl
ected++;

                isCollected =
true;

Destroy(gameObject);


Instantiate(pickupEffect,
transform.position,
transform.rotation);


UIController.instance.UpdateGe
mCount();


AudioManager.instance.PlaySFX(
6);
            }

            if(isHeal)
            {

if(PlayerHealthController.inst
ance.currentHealth !=
PlayerHealthController.instanc
e.maxHealth)
                {

PlayerHealthController.instanc
e.HealPlayer();


isCollected = true;

Destroy(gameObject);


Instantiate(pickupEffect,
transform.position,
transform.rotation);


AudioManager.instance.PlaySFX(
7);
                }
            }
        }
    }
}

public class PlayerController
: MonoBehaviour
{
    public static
PlayerController instance;

    public float moveSpeed;
    public Rigidbody2D theRB;
    public float jumpForce;

    private bool isGrounded;
    public Transform
groundCheckPoint;
    public LayerMask
whatIsGround;

    private bool
canDoubleJump;

    private Animator anim;
    private SpriteRenderer
theSR;

    public float
knockBackLength,
knockBackForce;
    private float
knockBackCounter;

    public float bounceForce;

    public bool stopInput;

    private void Awake()
    {
        instance = this;
    }

    // Start is called before
the first frame update
    void Start()
```

```
    {
        anim =
GetComponent<Animator>();
        theSR =
GetComponent<SpriteRenderer>()
;
    }

    // Update is called once
per frame
    void Update()
    {
        if
(!PauseMenu.instance.isPaused
&& !stopInput)
        {
            if
(knockBackCounter <= 0)
            {

                theRB.velocity
= new Vector2(moveSpeed *
Input.GetAxis("Horizontal"),
theRB.velocity.y);

                isGrounded =
Physics2D.OverlapCircle(ground
CheckPoint.position, .2f,
whatIsGround);

                if
(isGrounded)
                {

canDoubleJump = true;
                }

                if
(Input.GetButtonDown("Jump"))
                {
                    if
(isGrounded)
                    {

theRB.velocity = new
Vector2(theRB.velocity.x,
jumpForce);

AudioManager.instance.PlaySFX(
10);
                    }
                    else
                    {
                        if
(canDoubleJump)
                        {

theRB.velocity = new
Vector2(theRB.velocity.x,
jumpForce);

canDoubleJump = false;

AudioManager.instance.PlaySFX(
10);
                        }
                    }
                }

                if
(theRB.velocity.x < 0)
                {

theSR.flipX = true;
                }
                else if
(theRB.velocity.x > 0)
                {

theSR.flipX = false;
                }
            }
            else
            {

knockBackCounter -=
Time.deltaTime;
                if
(!theSR.flipX)
                {

theRB.velocity = new Vector2(-
knockBackForce,
theRB.velocity.y);
                }
                else
                {

theRB.velocity = new
Vector2(knockBackForce,
theRB.velocity.y);
                }
            }

        }


anim.SetFloat("moveSpeed",
Mathf.Abs( theRB.velocity.x));

anim.SetBool("isGrounded",
isGrounded);
```

```
    }

    public void KnockBack()
    {
        knockBackCounter =
knockBackLength;
        theRB.velocity = new
Vector2(0f, knockBackForce);


anim.SetTrigger("hurt");
    }

    public void Bounce()
    {
        theRB.velocity = new
Vector2(theRB.velocity.x,
bounceForce);

AudioManager.instance.PlaySFX(
10);
    }
}

public class
PlayerHealthController :
MonoBehaviour
{
    public static
PlayerHealthController
instance;

    public int currentHealth,
maxHealth;

    public float
invincibleLength;
    private float
invincibleCounter;

    private SpriteRenderer
theSR;

    public GameObject
deathEffect;

    private void Awake()
    {
        instance = this;
    }

    // Start is called before
the first frame update
    void Start()
    {
        currentHealth =
maxHealth;

        theSR =
GetComponent<SpriteRenderer>()
;
    }

    // Update is called once
per frame
    void Update()
    {
        if(invincibleCounter >
0)
        {
            invincibleCounter
-= Time.deltaTime;


if(invincibleCounter <= 0)
            {
                theSR.color =
new Color(theSR.color.r,
theSR.color.g, theSR.color.b,
1f);
            }
        }
    }

    public void DealDamage()
    {
        if (invincibleCounter
<= 0)
        {
            //currentHealth -=
1;
            currentHealth--;

            if (currentHealth
<= 0)
            {
                currentHealth
= 0;


//gameObject.SetActive(false);


Instantiate(deathEffect,
transform.position,
transform.rotation);


LevelManager.instance.RespawnP
layer();
            }
```

```
            else
            {

invincibleCounter =
invincibleLength;
                theSR.color =
new Color(theSR.color.r,
theSR.color.g, theSR.color.b,
.5f);


PlayerController.instance.Knoc
kBack();


AudioManager.instance.PlaySFX(
9);
            }


UIController.instance.UpdateHe
althDisplay();
        }
    }

    public void HealPlayer()
    {
        //currentHealth =
maxHealth;

        currentHealth++;
        if(currentHealth >
maxHealth)
        {
            currentHealth =
maxHealth;
        }


UIController.instance.UpdateHe
althDisplay();
    }

    private void
OnCollisionEnter2D(Collision2D
other)
    {

if(other.gameObject.tag ==
"Platform")
        {
            transform.parent =
other.transform;
        }
    }

    private void
OnCollisionExit2D(Collision2D
other)
    {

if(other.gameObject.tag ==
"Platform")
        {
            transform.parent =
null;
        }
    }
}

public class UIController :
MonoBehaviour
{
    public static UIController
instance;

    public Image heart1,
heart2, heart3;

    public Sprite heartFull,
heartEmpty, heartHalf;

    public Text gemText;

    public Image fadeScreen;
    public float fadeSpeed;
    private bool
shouldFadeToBlack,
shouldFadeFromBlack;

    public GameObject
levelCompleteText;

    private void Awake()
    {
        instance = this;
    }

    // Start is called before
the first frame update
    void Start()
    {
        UpdateGemCount();
        FadeFromBlack();
    }

    // Update is called once
per frame
    void Update()
    {
        if(shouldFadeToBlack)
        {
```

```
            fadeScreen.color =
new Color(fadeScreen.color.r,
fadeScreen.color.g,
fadeScreen.color.b,
Mathf.MoveTowards(fadeScreen.c
olor.a, 1f, fadeSpeed *
Time.deltaTime));

if(fadeScreen.color.a == 1f)
            {

shouldFadeToBlack = false;
            }
        }


if(shouldFadeFromBlack)
        {
            fadeScreen.color =
new Color(fadeScreen.color.r,
fadeScreen.color.g,
fadeScreen.color.b,
Mathf.MoveTowards(fadeScreen.c
olor.a, 0f, fadeSpeed *
Time.deltaTime));
            if
(fadeScreen.color.a == 0f)
            {

shouldFadeFromBlack = false;
            }
        }
    }

    public void
UpdateHealthDisplay()
    {

switch(PlayerHealthController.
instance.currentHealth)
        {
            case 6:
                heart1.sprite
= heartFull;
                heart2.sprite
= heartFull;
                heart3.sprite
= heartFull;

                break;

            case 5:
                heart1.sprite
= heartFull;
                heart2.sprite
= heartFull;
                heart3.sprite
= heartHalf;

                break;

            case 4:
                heart1.sprite
= heartFull;
                heart2.sprite
= heartFull;
                heart3.sprite
= heartEmpty;

                break;

            case 3:
                heart1.sprite
= heartFull;
                heart2.sprite
= heartHalf;
                heart3.sprite
= heartEmpty;

                break;

            case 2:
                heart1.sprite
= heartFull;
                heart2.sprite
= heartEmpty;
                heart3.sprite
= heartEmpty;

                break;

            case 1:
                heart1.sprite
= heartHalf;
                heart2.sprite
= heartEmpty;
                heart3.sprite
= heartEmpty;

                break;

            case 0:
                heart1.sprite
= heartEmpty;

                heart2.sprite
= heartEmpty;
                heart3.sprite
= heartEmpty;
```

```
                break;

            default:
                heart1.sprite
= heartEmpty;
                heart2.sprite
= heartEmpty;
                heart3.sprite
= heartEmpty;

                break;
        }
    }

    public void
UpdateGemCount()
    {
        gemText.text =
LevelManager.instance.gemsColl
ected.ToString();
    }

    public void FadeToBlack()
    {
        shouldFadeToBlack =
true;
        shouldFadeFromBlack =
false;
    }

    public void
FadeFromBlack()
    {
        shouldFadeFromBlack =
true;
        shouldFadeToBlack =
false;
    }
}

public class Victory :
MonoBehaviour
{
    public string mainMenu;

    // Start is called before
the first frame update
    void Start()
    {

    }

    // Update is called once
per frame
    void Update()
    {

    }

    public void MainMenu()
    {

SceneManager.LoadScene(mainMen
u);
    }
}
```