

**PENGEMBANGAN PRODUK DIGITAL GAME *ROGUELIKE*
MENGUNAKAN UNITY DAN *TOOLS* AI**

TUGAS AKHIR



**ALVIN ANDRIUS OEPOJO
NIM: 311910002**

**PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI DAN DESAIN
UNIVERSITAS MA CHUNG
MALANG
2023**

**LEMBAR PENGESAHAN
TUGAS AKHIR**

**PENGEMBANGAN PRODUK DIGITAL GAME ROGUELIKE
MENGUNAKAN UNITY DAN TOOLS AI**

Oleh:

**ALVIN ANDRIUS OEPOJO
NIM. 311910002**

dari:

**PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI dan DESAIN
UNIVERSITAS MA CHUNG**

Telah dinyatakan lulus dalam melaksanakan Tugas Akhir sebagai syarat kelulusan
dan berhak mendapatkan gelar Sarjana Komputer (S.Kom.)

Dosen Pembimbing I,



Windra Swastika, S.Kom., MT., Ph.D.

NIP. 20070039

Dosen Pembimbing II,



**Paulus Lucky Tirma Irawan, S.Kom.,
MT.**

NIP. 20100005

Dekan Fakultas Teknologi dan Desain,



**Dr. Eng. Romy Budhi Widodo
NIP. 20070035**

PERNYATAAN KEASLIAN TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan dari Tugas Akhir saya dengan judul “Pengembangan Produk Digital *Game Roguelike* Menggunakan Unity dan *Tools AI*” adalah benar benar hasil karya intelektual mandiri tanpa menggunakan bahan – bahan yang tidak diizinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada Daftar Pustaka. Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Malang, 25 September 2023



Alvin Andrius Oepojo

NIM. 311910002

PENGEMBANGAN PRODUK DIGITAL *GAME ROGUELIKE* MENGUNAKAN UNITY DAN *TOOLS AI*

**Alvin Andrius Oepojo¹, Windra Swastika², Paulus Lucky Tirma Irawan³
Universitas Ma Chung**

Abstrak

Zaman sekarang, banyak sekali tools AI yang bisa digunakan untuk berbagai macam aktivitas seperti menggambar, membuat musik, dan lain-lain. Tools AI mampu mempercepat pembuatan sebuah produk entah digital maupun fisik dan salah satunya adalah *game*. *Game* mampu dibuat menggunakan AI, tetapi belum pasti apakah pembuatan aset menggunakan tools AI bisa mempercepat pembuatan *game* atau tidak. AI yang digunakan adalah ChatGPT untuk membuat cerita sinopsis, kode serta membuat berbagai macam aspek *game*, Leonardo.ai untuk membuat gambar background, menjelaskan cerita, dan kejadian acak, Beatoven.ai untuk membuat musik background.

Penggunaan tools AI untuk pengembangan *game* bisa dilakukan dengan lancar untuk Beatoven.ai, tetapi untuk ChatGPT dan Leonardo.ai memberikan hasil yang cukup bagus dengan prompt yang diberikan, tetapi memiliki kekurangan tertentu seperti kadang memberikan hasil yang kurang tepat dan bahkan sesekali menghasilkan hasil yang tidak diinginkan.

Game yang dikembangkan adalah sebuah *game* simpel tetapi mampu menghibur pemain dengan skor SUS 72.70 yaitu bisa dimainkan dengan sedikit masalah dari 11 pengetes lelaki dan 1 pengetes perempuan dengan umur antara 19-24. Hasil yang dibuat dengan AI mampu menghasilkan hasil yang bagus, tetapi diperlukan pembelajaran lebih jauh mengenai pengembangan *game* menggunakan tools AI untuk mempercepat pengembangan *game* dan sebagian besar pemain tidak keberatan dengan *game* dikembangkan oleh AI atau tidak asalkan ada *game* yang bisa dimainkan.

Kata kunci: tools AI, ChatGPT, Leonardo.ai, Beatoven.ai

DEVELOPMENT OF DIGITAL ROGUELIKE GAME USING UNITY AND AI TOOLS

Alvin Andrius Oepojo¹, Windra Swastika², Paulus Lucky Tirma Irawan³
Universitas Ma Chung

Abstract

Nowadays, there are so many AI tools that can be used for different reasons like drawing, making music and more. AI tools can be used to make a product quicker whether it's digital nor physical and one of them is a game. A game can be made with AI, however it's uncertain if asset creation can help to make a game quicker. The AI tools that are going to be used are ChatGPT for story synopsis, code and different aspects of the game, Leonardo.ai for background and storytelling and Beatoven.ai for making background musics.

The use of AI tools can be done easily with Beatoven.ai, however for ChatGPT and Leonardo.ai gave a decent result with the given prompts, but with certain drawbacks such as sometimes giving less precise results and even gave the wrong results occasionally.

The developed game was simple but able to entertain the players with the SUS score of 72.70 which means playable with few problems from 11 male testers and a female tester from 19-24 years old. The result that was made with AI tools gave a good result but requires studying it further to be able to develop a game quicker and most of the players don't mind if the game was made with AI or not as long there is a game to be played.

Keywords: AI tools, ChatGPT, Leonardo.AI, Beatoven.ai.

Kata Pengantar

Puji Syukur kepada Tuhan Yang Maha Esa, atas berkat Rahmat dan Karunia-Nya bahwa Tugas Akhir ini telah terselesaikan dengan judul “Pengembangan Produk Digital *Game Roguelike* Menggunakan Unity dan *Tools AI*”. Berkat pertolongan dari berbagai pihak yang meluangkan waktu untuk Penulis bisa menyelesaikan Tugas Akhir ini. Maka dari itu pada kesempatan ini, Penulis ingin mengucapkan terima kasih sebesar-besarnya kepada:

1. Bapak Dr. Eng. Romy Budhi, ST., MT. selaku Dekan Fakultas Teknologi dan Desain Universitas Ma Chung dan ketua penguji,
2. Bapak Hendry Setiawan, ST., M. Kom., selaku Kepala Program Studi Teknik Informatika Universitas Ma Chung,
3. Bapak Windra Swastika, S.Kom., MT., Ph.D. selaku dosen pembimbing pertama,
4. Bapak Paulus Lucky Tirma Irawan, S.Kom., MT. selaku dosen pembimbing kedua,
5. Teman-teman saya yang membantu memberi nasehat dan pendapat,
6. Keluarga saya yang memberikan bantuan dan mendukung penulis untuk mengerjakan Tugas Akhir ini.

Malang, 25 September 2023

Alvin Andrius Oepojo

Daftar Isi

Abstrak	i
Abstract	ii
Kata Pengantar	iii
Daftar Isi.....	iv
Daftar Gambar.....	vi
Daftar Tabel	viii
Bab I Pendahuluan	1
1.1 Latar Belakang	1
1.2 Identifikasi Masalah	3
1.3 Batasan Masalah.....	4
1.4 Perumusan Masalah.....	4
1.5 Tujuan Penelitian.....	4
1.6 Manfaat Penelitian	5
1.7 Luaran Penelitian	5
Bab II Tinjauan Pustaka	6
2.1 <i>Artificial Intelligence</i> (AI).....	6
2.2 <i>AI as a Service</i> (AIaaS).....	6
2.3 <i>Prompt Engineering</i>	7
2.4 ChatGPT.....	8
2.5 Leonardo.ai.....	10
2.6 Beatoven.ai	11
2.7 <i>Video Game</i>	11
2.8 <i>Genre Game</i>	12
2.8.1 <i>Action</i>	13
2.8.2 <i>Role-Playing</i>	14
2.8.3 <i>Simulation</i>	15
2.8.4 <i>Strategy</i>	15
2.9 <i>Roguelike</i>	16
2.10 <i>Aset Game</i>	18
2.11 <i>Desain Game</i>	21
2.12 <i>Game Engine</i>	22
2.12.1 Unity.....	22

2.12.2 Unreal Engine.....	24
2.12.3 GameMaker Studio	26
2.12.4 Godot.....	27
2.12.5 RPG Maker	29
2.12.6 Ren'Py.....	30
2.13 System Usability Scale (SUS).....	32
Bab III Analisis dan Perancangan Sistem	35
3.1 Tahapan Penelitian	35
3.2 Mencari <i>Tools</i> AI	35
3.3 Membuat Cerita Sinopsis dan Fitur <i>Game</i>	36
3.4 Pembuatan Aset <i>Game</i>	37
3.5 Pengembangan <i>Game</i>	37
3.5.1 Integrasi Aset <i>Game</i> ke Unity	38
3.5.2 Detail Pengembangan <i>Game</i>	38
3.6 <i>Game Testing</i>	44
3.7 Publikasi <i>Game</i> di itch.io dan Mengambil Data <i>Review Game</i>	45
Bab IV Hasil dan Pembahasan	47
4.1 Mencari <i>Tools</i> AI	47
4.2 Pembuatan Cerita Sinopsis dan Fitur <i>Game</i>	47
4.3 Hasil Pembuatan Aset <i>Game</i>	50
4.4 Pembuatan <i>Game</i>	57
4.5 Hasil <i>Game</i>	66
4.6 Hasil <i>Testing</i>	76
4.7 Mengeluarkan <i>Game</i> di itch.io dan Pendapat Pemain	80
Bab V Penutup	82
5.1 Kesimpulan	82
5.2 Saran.....	83
Daftar Pustaka	85
Lampiran	89

Daftar Gambar

Gambar 2.1 Gambar website ChatGPT.....	8
Gambar 2.2 Gambar website Leonardo.ai.....	10
Gambar 2.3 Gambar website Beethoven.ai.....	11
Gambar 2.4 Tennis for Two di osiloskop.....	12
Gambar 2.5 Gameplay dari Counter Strike: Global Offensive	13
Gambar 2.6 Gameplay dari The Elder Scrolls V: Skyrim	14
Gambar 2.7 Gameplay dari PowerWash Simulator	15
Gambar 2.8 Gameplay dari Sid Meier's Civilization VI	16
Gambar 2.9 Gameplay dari Rogue.....	17
Gambar 2.10 Game Risk of Rain	18
Gambar 2.11 Main Menu Team Fortress 2	19
Gambar 2.12 Heads-up Display Team Fortress 2	19
Gambar 2.13 Backpack Team Fortress 2	20
Gambar 2.14 Sebuah karakter dengan 2D dan 3D.....	20
Gambar 2.15 Contoh penggunaan engine Unity	23
Gambar 2.16 Gameplay dari One Dreamer.....	24
Gambar 2.17 Contoh penggunaan engine Unreal Engine.....	25
Gambar 2.18 Gameplay dari Fortnite Chapter 2.....	25
Gambar 2.19 Contoh penggunaan game engine GameMaker Studio	26
Gambar 2.20 Gameplay dari Undertale	27
Gambar 2.21 Contoh penggunaan game engine Godot	28
Gambar 2.22 Gameplay dari Dead Cells.....	28
Gambar 2.23 Contoh penggunaan <i>game engine</i> RPG Maker MZ	29
Gambar 2.24 Gameplay dari OneShot	30
Gambar 2.25 Gameplay dari Doki Doki Literature Club.....	32
Gambar 3.1 Tahapan Penelitian	35
Gambar 3.2 Pembuatan cerita sinopsis dengan ChatGPT.....	36
Gambar 3.3 Contoh menggambar dengan Leonardo.ai	37
Gambar 3.4 Mockup menu utama game	38
Gambar 3.5 Mockup interaksi tombol Play Game.....	39
Gambar 3.6 Mockup interaksi tombol Settings.....	39

Gambar 3.7 Mockup kejadian acak.....	40
Gambar 3.8 Mockup menu peta game	41
Gambar 3.9 Mockup sistem perlawanan musuh	42
Gambar 3.10 Mockup menu Status.....	42
Gambar 3.11 Mockup sistem inventory	43
Gambar 4.1 Hasil menggambar menggunakan Leonardo.ai	50
Gambar 4.2 Hasil tiap kondisi karakter.....	53
Gambar 4.3 Hasil akhir dari pembuatan judul	54
Gambar 4.4 Pengaturan ControlNet di Leonardo.ai.....	54
Gambar 4.5 Referensi gambar UI	55
Gambar 4.6 Hasil generasi Leonardo.ai dengan referensi Gambar 4.4	55
Gambar 4.7 Scene menu utama game	66
Gambar 4.8 Scene menu utama.....	67
Gambar 4.9 Scene intro game	68
Gambar 4.10 Scene character.....	68
Gambar 4.11 Pilihan karakter engineer.....	69
Gambar 4.12 Scene gameplay.....	70
Gambar 4.13 Contoh kejadian acak langsung.....	71
Gambar 4.14 Contoh kejadian acak pilihan	71
Gambar 4.15 Salah satu hasil dari kejadian acak dari Gambar 4.14.....	72
Gambar 4.16 Menu action.....	74
Gambar 4.17 Hasil aksi search.....	74
Gambar 4.18 Scene menang.....	75
Gambar 4.19 Salah satu scene kalah	75
Gambar 4.20 Grafik rata-rata total skor pengetes	78

Daftar Tabel

Tabel 2 Skor SUS dan artinya.....	34
Tabel 4.1 Generasi ChatGPT untuk cerita sinopsis	48
Tabel 4.2 Generasi ChatGPT untuk nama grup	49
Tabel 4.3 Prompt serta beberapa hasil gambar Leonardo.ai	51
Tabel 4.4 Pembuatan musik dengan Beatoven.ai.....	56
Tabel 4.5 Generasi ChatGPT untuk kejadian acak yang benar.....	58
Tabel 4.6 Generasi ChatGPT kejadian acak tidak sesuai.....	60
Tabel 4.7 Hasil generasi ChatGPT untuk kode script.....	62
Tabel 4.8 Generasi ChatGPT terhadap pertanyaan	63
Tabel 4.9 Generasi ChatGPT yang salah	64
Tabel 4.10 Generasi ChatGPT salah bahasa pemrograman	65
Tabel 4.11 Data asli pengetes game	78
Tabel 4.12 Data hasil SUS	79

Bab I

Pendahuluan

1.1 Latar Belakang

Zaman sekarang, ada banyak sekali *tools* AI yang bisa digunakan untuk berbagai macam aktivitas. Hal-hal seperti membuat gambar, membuat sebuah cerita, membuat suara dan lain-lainnya hanya dengan deskripsi yang diberikan oleh pengguna AI. Contohnya seperti AI bernama Midjourney yang digunakan di Discord sebagai bot digunakan untuk membuat gambar dengan berbagai macam cara menggambar dengan deskripsi yang telah diberi yang lalu bisa diatur tergantung keinginan pengguna (Amalia, 2022). Contoh lainnya berupa *tools* AI yang bisa digunakan untuk membuat musik yaitu beatoven.ai yang bisa membuat musik dengan memilih tema lagu serta suasana dan pengguna bisa mengatur sesuai keinginan untuk membuat musik (Beatoven Private Limited, 2023). *Tools* AI seperti ChatGPT mampu memberikan jawaban yang diminta oleh pengguna seperti membuat sebuah cerita sinopsis, resep memasak dan lain-lainnya.

Karena bisa mendapatkan hasil yang bagus dengan menggunakan *tools* AI, ada berbagai macam keuntungan yang bisa dimanfaatkan. Salah satu keuntungannya adalah untuk mempercepat pembuatan sebuah produk, entah digital maupun fisik. Gambar yang dibuat oleh AI bisa dibuat untuk sebuah desain atau logo barang dan bahkan bisa digunakan untuk membuat sebuah aset produk digital berupa *game*. Tetapi karena *tools* AI bisa mendapatkan hasil yang bagus yang bisa dibandingkan dengan artis lainnya, beberapa orang memiliki kebencian dari media seni yang dibuat oleh AI karena tidak dibuat oleh manusia dan / atau mengurangi lowongan kerja media seni seperti kasus di mana Jason M. Allen memenangkan lomba di Colorado State Fair dengan menggunakan gambar yang dibuat menggunakan Midjourney dan membuat pengguna di sosial media Twitter membenci penggunaan AI untuk memenangkan sebuah lomba menggunakan AI (Roose, 2022). Walaupun itu benar, *tools* AI bisa membantu untuk memberi ide dan sebuah *template* untuk membuat hasil yang tepat (Tiernan, 2022).

Produk digital berupa *game* merupakan salah satu produk digital yang bisa dijual dikarenakan produk digital berupa *game* digunakan untuk menghibur pemain dengan berbagai macam cara. Penjualan produk digital berupa *game* juga mulai menaik dikarenakan masa pandemik yang membuat orang-orang terpaksa untuk tetap di rumah. Di tahun 2020, penjualan produk digital berupa *game* mencapai 155 miliar USD dan jumlah waktu orang bermain menaik sebanyak 39% dalam masa pandemi (Clement, 2022). Dunia produk digital berupa *game* di Indonesia lebih fokus ke *game* di perangkat *mobile* dengan jumlah 114 juta pemain karena sebagian besar pemain tidak memiliki komputer dengan jumlah 46.1 juta pemain atau *console* dengan jumlah 34.1 juta pemain pada tahun 2021 (Allcorrect Group, 2022) dan memiliki jumlah pendapatan 1.92 miliar USD, 9% menaik dari tahun sebelumnya. Di tahun 2019, *game* yang dibuat dengan satu orang sampai kelompok kecil atau yang disebut *game indie* mencapai 332 *game*, 47.56% lebih tinggi daripada tahun 2018 (Rizkinaswara, 2021).

Pembuatan produk digital berupa *game* dimulai dengan merencanakan *game* seperti memilih *platform*, membuat konsep seni, rencana *marketing* dan lain-lain. Contohnya seperti *game* Ada's Delivery Service khusus untuk mengajar dasar logika algoritma dan mempelajari dunia pemrograman yang bisa dilakukan dimana saja menggunakan *platform mobile* (Irawan, Tandiono, & Setiawan, 2018). Selain menggunakan *platform mobile*, website juga bisa digunakan untuk mempelajari sesuatu seperti Scratch dan Greenfoot dengan bermain dan bahkan membuat *game* sendiri (Widodo, Swastika, & Yuswanto, 2020).

Lalu, aset *game* bisa mulai dibuat menurut cerita untuk *game*. Lalu, *game* dimulai pembuatannya yang dimulai dengan membuat *prototype* untuk mengetes fitur. Hal yang diatur selain *prototype* adalah konsep visual, desain dari *game*, desain suara dan pemrograman. Setelah dibuat, *game* bisa dites untuk mengecek masalah yang ada seperti mengetes suara, apakah *game* bekerja dengan lancar dan hal teknis lainnya. *Game* akhirnya bisa dipublikasikan dan diberi perbaikan dan / atau konten baru bila ada (Gameace Creative Studio, 2021).

Setelah semua aset sudah dibuat, maka pembuatan *game* bisa dimulai dengan membuatnya di dalam sebuah *game engine*. Ada berbagai macam *game*

engine yang bisa dipakai seperti Unity, Unreal Engine, dan lain-lainnya. Setelah *game* selesai dibuat, maka *game* perlu dites dan memperbaiki beberapa masalah setelah dites dan bila *game* selesai dibuat, maka siap untuk dipublikasikan di toko *game* yang diinginkan.

Produk digital berupa *game* tidak hanya dibuat oleh perusahaan besar, tetapi juga dibuat oleh sebuah grup kecil atau satu orang saja. Produk digital berupa *game* seperti Stardew Valley dibuat oleh satu orang yaitu Eric Barone dalam waktu empat setengah tahun untuk membuat produk digital berupa *game* mulai dari desain *game*, animasi, membuat musik sampai kodenya (White, 2018). Ada beberapa alasan kenapa produk digital berupa *game* dibuat hanya dengan satu orang atau sebuah grup kecil contohnya seperti untuk mendapatkan ide lebih baik serta lebih hemat daripada membayar banyak karyawan atau tenaga kerja untuk membuat sebuah produk digital berupa *game*.

Dengan menggunakan *tools* AI, pengembang *game* mampu mengembangkan *game* untuk membuat aset *game* lebih cepat dan membantu mendapatkan hasil yang cukup baik bila dibandingkan dengan membuat aset *game* secara manual yang membutuhkan beberapa minggu bahkan bulan untuk mendapatkan hasil yang memuaskan. Tidak cuma itu, menggunakan *tools* AI lebih murah untuk membuat aset daripada membuat aset dari meminta komisi orang lain. Keuntungan lainnya adalah ada beberapa dari *tools* AI memberikan layanannya secara gratis dan tidak perlu memikirkan mengenai hak cipta atau hanya perlu memberikan kredit kepada AI tersebut.

1.2 Identifikasi Masalah

Semua aspek dari pembuatan produk digital berupa *game* memerlukan waktu. Pembuatan gambar atau aset untuk produk digital berupa *game* perlu waktu untuk digambar dan cerita juga memerlukan waktu untuk dibuat. Karena untuk mendapatkan sukses dari produk digital berupa *game*, diperlukan kreativitas yang besar dan hal-hal menarik yang bisa membuat orang tertarik untuk bermain. Bila pembuatan produk digital berupa *game* dilakukan oleh satu orang atau dalam grup kecil, maka pekerjaan akan lebih banyak dan lebih lama untuk dikerjakan. Dengan menggunakan *tools* AI yang bisa digunakan untuk membuat cerita, karakter dan

berbagai macam aset lainnya, pembuatan aset game menjadi lebih hemat waktu serta tidak memerlukan seorang profesional untuk membuat aset *game*.

1.3 Batasan Masalah

Batasan masalah yang akan difokuskan adalah sebagai berikut:

1. Pembuatan sebuah produk digital berupa *game* dengan tipe genre *roguelike* yang bisa dibuat dengan *tools* AI untuk membuat sebuah produk digital berupa game lebih cepat dan tidak memerlukan kemampuan tertentu untuk membuat *game*.
2. Pembuatan aset yang diinginkan untuk produk digital berupa *game* seperti cerita sinopsis menggunakan ChatGPT, 145 gambar menggunakan Leonardo.ai, dan enam musik *background* menggunakan Beethoven.ai.
3. Pengembangan *game* dengan *game engine* Unity.
4. Publikasi *game* dengan itch.io.

1.4 Perumusan Masalah

Rumusan masalah yang dapat diambil adalah:

1. Bagaimana mengembangkan *game* dengan Unity dan pembuatan aset *game* dengan berbagai macam *tools* AI yaitu ChatGPT, Leonardo.ai dan Beethoven.ai?
2. Bagaimana cara menjelaskan kepada *tools* AI untuk menghasilkan aset-aset *game* sesuai dengan skenario *game* yang sedang dikembangkan?
3. Apakah *game* dengan aset yang dihasilkan menggunakan *tools* AI bisa dikomersialkan dan dapat diterima oleh komunitas *gaming* dengan jumlah *review* dan jumlah *rating* di itch.io?

1.5 Tujuan Penelitian

Tujuan dari penelitian ini adalah membuat sebuah produk digital berupa *game* hanya dengan menggunakan AI untuk pembuatan berbagai macam aspek produk digital berupa *game* yaitu cerita dan aset produk digital berupa *game* seperti gambar, musik, dan lain-lain.

1.6 Manfaat Penelitian

Adapun manfaat yang dapat diambil dari penelitian ini:

1. Bagi pengguna, produk digital berupa *game* bisa dimainkan untuk hiburan serta mempelajari hal-hal yang bisa dipelajari dari penggunaan AI.
2. Bagi Universitas Ma Chung, penelitian ini bisa membantu memberikan referensi untuk penulis lainnya.

1.7 Luaran Penelitian

Luaran dari penelitian ini adalah sebuah produk digital berupa *game* dengan genre *roguelike* yang sudah dipublikasikan di itch.io serta artikel ilmiah untuk publikasi di jurnal.

Bab II

Tinjauan Pustaka

2.1 *Artificial Intelligence* (AI)

Artificial Intelligence atau kecerdasan buatan merupakan sebuah simulasi otak manusia yang dibuat di dalam sebuah mesin atau komputer dan diprogram untuk bekerja seperti manusia seperti mencari arti, belajar dari pengalaman, dan mencapai sebuah kesimpulan (Copeland, 2023). AI bisa membantu pekerjaan manusia yang biasanya terlalu rumit untuk dilakukan oleh manusia sendiri, salah satu contohnya seperti menghitung berbagai macam kasus. Manusia tidak bisa menghitung berbagai macam cara atau mencari berbagai macam solusi dalam waktu bersamaan, tetapi AI mampu melakukannya dengan berbagai macam data yang sudah didapat.

AI memiliki cara untuk mempelajari berbagai kesalahan atau pengalaman yang sudah dialami dan akan menggunakan hal-hal tersebut untuk menyelesaikan masalah berikutnya lebih baik. AI juga bisa mencari hasil terbaik dalam sebuah masalah dengan melakukan berbagai macam simulasi kemungkinan yang bisa terjadi dalam sebuah masalah. Salah satu contoh penerapannya adalah sebuah AI bernama AlphaGo. AlphaGo merupakan sebuah AI yang digunakan untuk mempelajari permainan Go. AlphaGo awalnya diberi 100 ribu data pertandingan Go untuk dipelajari lalu setiap kali kalah, AlphaGo akan memperbaiki kesalahannya untuk pertandingan selanjutnya dan pada tahun 2016, AlphaGo menang melawan juara dunia Go yaitu Lee Sedol.

2.2 *AI as a Service* (AIaaS)

AI as a Service merupakan sebuah layanan yang diberi oleh AI yang dibuat oleh pihak ketiga untuk melakukan apa yang diperlukan oleh pengguna (Kleinings, 2022). Perusahaan atau pengguna bisa menggunakan AI tanpa memerlukan investasi modal yang besar untuk menggunakannya. Perusahaan juga bisa memberikan data untuk membantu membangun AI menjadi lebih baik lagi. Perusahaan bisa memberikan layanan yang lebih baik seperti lebih cepat mendapatkan hasil, memberi lebih banyak kesempatan untuk menggunakan

layanan dan lain-lain kepada pengguna bila pengguna ingin menggunakan layanan yang lebih baik dengan membayar bulanan atau tahunan.

Contoh layanan yang diberi bisa beragam seperti chat bot yang bisa melakukan konversasi, memberi jawaban atau instruksi yang diberi oleh pengguna, membuat gambar dari deskripsi pengguna, membuat musik dari deskripsi pengguna dan lain-lain. Salah satu contohnya adalah Midjourney dimana pengguna bisa menggunakan AI untuk membuat gambar yang diminta oleh pengguna dengan memberi instruksi kepada AI. Walaupun AI bisa memberikan hal yang diminta pengguna, penjelasan lebih spesifik bisa membantu untuk memberikan pengguna hasil yang diinginkan.

2.3 Prompt Engineering

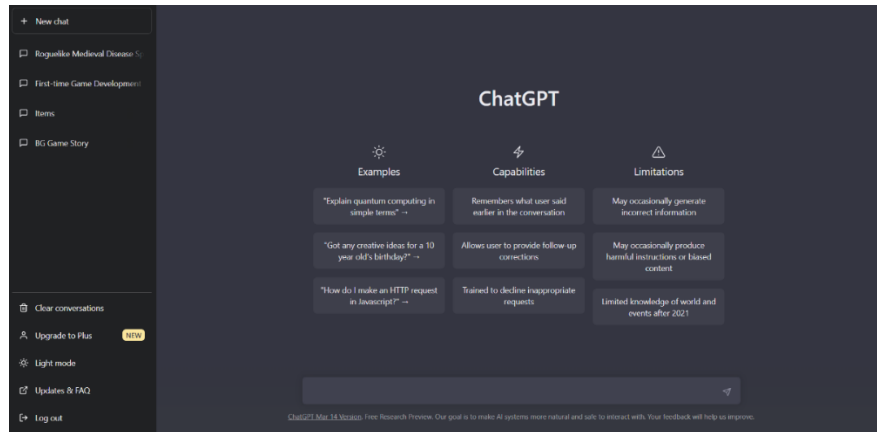
Prompt engineering merupakan cara membuat kalimat yang dibuat secara benar dan jelas untuk menghasilkan sesuatu yang diinginkan. AI yang memerlukan deskripsi untuk membuat sesuatu seperti ChatGPT dan Midjourney, memerlukan *prompt engineering* untuk memberikan hasil yang diinginkan. Prompt bisa berupa instruksi, pertanyaan, contoh dan data input yang dimasukkan. Untuk memberikan hasil teks biasa, pengguna hanya perlu menjelaskan kepada AI hal yang ingin dijawab. Untuk membuat gambar, pengguna bisa menjelaskan dengan teks dan / atau menggunakan gambar yang diinginkan.

Prompt engineering bisa berbeda tergantung dari AI yang dipakai. Contohnya untuk AI Stable Diffusion, pembuatan gambar dijelaskan dengan memberikan subjek apa, digambar menggunakan apa, menggambar dengan gaya apa dan lain-lainnya (Andrew, 2023). Deskripsi yang ditulis juga dibuat dengan menggunakan koma setelah memberikan deskripsi tertentu. Contohnya seperti “*Hooded character, comic, realistic, medieval*”. Kalimat tersebut targetnya adalah membuat sebuah gambar untuk sebuah karakter yang memakai pakaian berkerudung yang digambar seperti komik secara realistis dengan tema *medieval*. Deskripsi tersebut memberi AI berbagai macam cara untuk menggambar, posisi karakter, warna, dan lain-lain. Untuk mendapatkan hasil yang spesifik, penjelasan spesifik juga perlu dimasukkan.

Beberapa penelitian sudah pernah dilakukan untuk melakukan *prompt engineering* untuk hal tertentu. Salah satu contohnya adalah RePrompt, yaitu sebuah pengatur *prompt* untuk memberikan ekspresi emosi yang jelas pada gambar (Wang, Shen, & Lim, 2023). Contohnya pengguna ingin membuat gambar dengan *prompt* “orang terkejut karena diundang ke sebuah liburan yang tidak terencana”. RePrompt akan mengubah *prompt* tersebut untuk mendapatkan hasil yang lebih tepat untuk membuktikan gambar orang terkejut. Ada penelitian lain yang menggunakan *prompt engineering* untuk membuat pesan kesadaran kesehatan dengan mengambil pesan kesehatan dari Twitter dan diatur menggunakan *prompt engineering* untuk membuat kata yang tepat dan akurat (Lim & Schmälzle, Artificial Intelligence for Health Message Generation: Theory, Method, and an Empirical Study Using Prompt Engineering, 2022).

2.4 ChatGPT

ChatGPT adalah sebuah AI yang dibuat oleh perusahaan OpenAI yang bisa digunakan oleh pengguna dimana ChatGPT akan menjawab pertanyaan atau instruksi yang diminta oleh pengguna. ChatGPT menggunakan *machine learning* dan *deep learning* untuk mempelajari dan memahami bahasa manusia melalui teks. Contohnya seorang pengguna ingin membuat sebuah cerita mengenai seorang pahlawan. ChatGPT akan membuat cerita mengenai seorang pahlawan, tetapi memberitahu cerita tersebut secara umum. Pemberian penjelasan yang lebih spesifik akan membantu ChatGPT untuk membuat sesuatu lebih baik dan mencapai keinginan pengguna. Mengulang menggunakan penjelasan yang sama juga akan memberi hasil yang berbeda tergantung dari kerumitan penjelasan yang diberi oleh pengguna. Gambar 2.1 menunjukkan penggunaan ChatGPT.



Gambar 2.1 Gambar website ChatGPT

ChatGPT tidak hanya bisa berkerja di bahasa inggris, tetapi juga bisa menggunakan bahasa manusia lain. Hal ini bisa dilakukan dikarenakan dalam proses pelatihan, AI diberi berbagai macam bahasa manusia yang ada. Semakin banyak orang yang menggunakan ChatGPT, ChatGPT mampu memberikan hasil yang lebih baik dan jelas. ChatGPT tidak bisa menjawab semua pertanyaan atau instruksi yang diminta, seperti pertanyaan yang tidak jelas atau tidak masuk akal. Bila hal itu terjadi, ChatGPT akan meminta maaf kepada pengguna dan menjelaskan keterbatasan yang dimiliki.

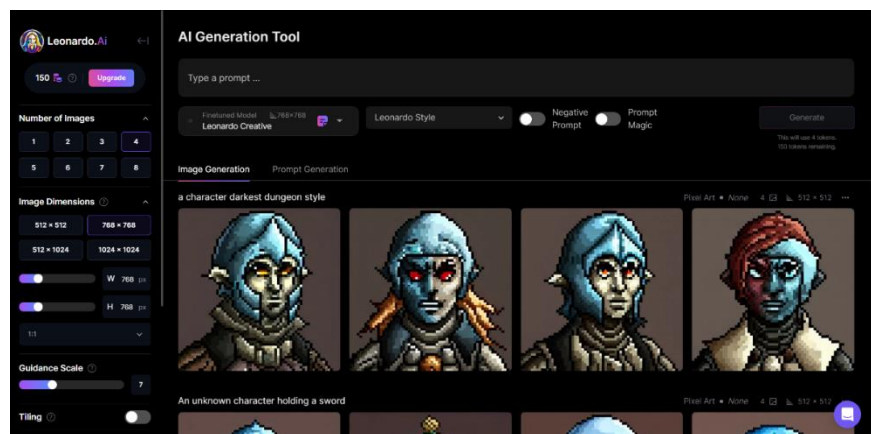
ChatGPT memiliki berbagai macam cara untuk memberi instruksi. Berikut adalah beberapa *prompt engineering* yang khusus digunakan untuk ChatGPT (Wolff, 2023):

1. *Zero Shot Prompting*: Memberikan *prompt* yang sangat dasar tanpa memberikan contoh apapun. Biasanya *prompt* yang diberi merupakan instruksi mendasar atau menanyakan sebuah hal umum seperti “Bagaimana cara matahari bekerja?”
2. *One Shot Prompting*: Memberikan *prompt* yang lebih spesifik dari Zero-Shot Prompting dengan memberikan penjelasan yang simpel dan memberi sebuah contoh untuk menghasilkan hal yang spesifik.
3. *Few Shot Prompting*: Seperti One Shot Prompting, tetapi menjelaskan instruksi dengan memberikan beberapa tambahan contoh untuk mendapatkan hasil yang lebih spesifik.

4. *Role Prompting*: Sebuah *prompt* dimana AI diatur untuk memerankan seperti orang dengan latar belakang tertentu. Hal ini juga bisa merubah bagaimana ChatGPT menjawab.
5. *Chain of Thought Prompting*: Membuat *prompt* yang mengikuti instruksi sebelumnya tanpa mengulang menulis instruksi tersebut. Contohnya seperti *prompt* pertama meminta menghapus huruf b untuk sebuah teks tertentu. Pengguna hanya perlu menulis instruksi seperti “Lakukan hal yang sama dengan teks ini” untuk mengulang instruksi.

2.5 Leonardo.ai

Leonardo.ai merupakan sebuah AI yang digunakan untuk membuat berbagai macam gambar dengan berbagai macam model dengan deskripsi yang dibuat oleh pengguna. Deskripsi yang diberi bisa secara dasar untuk mendapatkan berbagai macam hasil atau diberi deskripsi yang detil untuk membuat gambar yang diinginkan. Leonardo.ai juga mampu membuat gambar dengan deskripsi larangan yang tidak diinginkan oleh pengguna. Leonardo juga memberi berbagai macam fitur seperti memberikan gambar lebih jelas, memberikan zoom in atau zoom out dari gambar yang sudah dibuat, membuat model sendiri untuk membuat gambar yang diinginkan dan lain-lain. Gambar 2.2 merupakan gambar dari *website* Leonardo.ai.



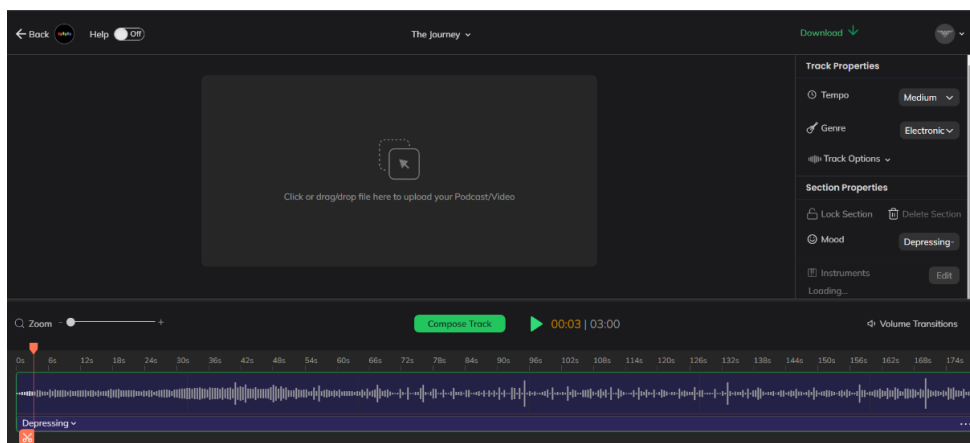
Gambar 2.2 Gambar *website* Leonardo.ai

Untuk bisa menggunakan Leonardo.ai, pengguna harus melakukan registrasi untuk masuk dan menunggu email dari Leonardo.ai untuk di-whitelist agar bisa digunakan. Tidak hanya itu, pengguna hanya memiliki 150 *token* yang

akan di-*reset* setiap hari untuk menggunakannya dan beberapa fitur juga memiliki keterbatasan penggunaan. Token ini akan digunakan dengan jumlah tertentu tergantung dari jumlah gambar yang ingin dibuat dan menggunakan berbagai macam fitur yang diberikan oleh Leonardo.ai. Pengguna bisa membuat model untuk melatih AI untuk mendapatkan hasil yang diinginkan oleh pengguna, tetapi hanya bisa melatih dua model tiap bulan. Untuk menggunakan fitur dan membuat gambar lebih banyak, Leonardo.ai memberikan subskripsi dengan berbagai macam harga.

2.6 Beatoven.ai

Beatoven.ai merupakan sebuah AI yang digunakan untuk membuat musik latar belakang atau *background music*. Pembuatan musik dengan beatoven.ai dimulai dengan mengisi judul, memilih durasi musik, dan memilih kecepatan tempo. Setelah itu, pengguna bisa memilih satu dari delapan genre yang ingin dibuat dan demo dari genre musik tersebut bisa didengarkan. Lalu, pengguna akan memilih satu dari enam belas suasana dan demo suasana juga bisa didengarkan. Lalu AI akan membuat musik tersebut dan pengguna bisa mengatur empat instrumen, mengubah Susana, dan lain-lain. Pengguna bisa menggunakan peralatan sepuasnya, tetapi hanya bisa *download* lagu 5 menit per bulan. Beatoven.ai juga memberi subskripsi untuk memberikan fitur lain dan *download* lagu lebih lama. Gambar 2.3 menunjukkan gambar penggunaan Beatoven.ai.



Gambar 2.3 Gambar *website* Beatoven.ai

2.7 Video Game

Video game adalah sebuah produk digital yang fokusnya menghibur pengguna dan juga bisa digunakan untuk mempelajari sesuatu. *Video game* yang pertama kali dibuat adalah Tennis for Two yang dibuat pada tahun 1958 oleh William Higinbotham yang membuat ide dan gambarannya dan Robert Drovak membangun perangkatnya (Tretkoff, October 1958: Physicist Invents First Video Game, 2008). *Tennis for Two* adalah sebuah produk digital berupa *game* yang menunjukkan permainan tennis yang ditampilkan dari samping yang bisa dimainkan di komputer analog yang bisa ditampilkan di sebuah osiloskop. Di Gambar 2.4 menunjukkan Tennis for Two di sebuah osiloskop.



Gambar 2.4 Tennis for Two di osiloskop

Gambar dari (Parks, 2022)

Semakin banyak *video game* yang dibuat, semakin banyak tipe *game* atau genre *game* yang bisa dimainkan. *Video game* sekarang memiliki berbagai macam genre seperti *Adventure* atau petualangan seseorang atau sebuah kelompok, *RPG* atau *Role Playing Game* yang mengajak pemain untuk bermain menjadi sebuah karakter atau kelompok untuk mengikuti cerita dari *game* tersebut, dan berbagai macam genre lainnya. Sebuah *video game* bisa memiliki satu genre saja tetapi memiliki berbagai macam genre bisa membantu untuk menjelaskan isi dari *game*.

2.8 Genre Game

Genre *game* merupakan sebuah cara untuk mengetahui berbagai macam tipe *game* yang memiliki berbagai macam cara untuk memainkan *game* tersebut. *Game* biasanya memiliki satu genre *game* inti yang bisa menjelaskan bagaimana

game dimainkan serta beberapa genre lainnya untuk membantu mengerti isi dari *game* tersebut. Genre *game* juga bisa membantu mencari tahu apakah *game* tersebut memiliki hal yang diinginkan atau tidak tergantung dari keinginan dari pemain.

Genre *game* sangat susah untuk dibedakan karena ada berbagai macam cara untuk membagikannya mulai dari cara bermain, tipe visual dari *game*, genre yang memiliki gabungan genre lain dan berbagai hal. Thomas H. Apperley menjelaskan bahwa sebuah genre akan berubah dan tidak akan tetap sama. Hal ini karena pemain akan menjadi bosan bila sebuah genre tidak berubah dan genre akan tetap memberikan inovasi baru atau bahkan genre baru (Apperley, 2008). Karena banyak sekali genre yang ada di dunia *game*, Thomas H. Apperley membagi genre *game* menjadi empat yaitu *action*, *role-playing*, *simulation*, dan *strategy*.

2.8.1 Action

Genre *game action* merupakan genre *game* yang fokusnya menuju ke perspektif sinematik yang biasanya penuh dengan gerakan besar dan cepat dan membuat pemain merasakan adrenalin untuk menyelesaikan masalah yang memerlukan waktu pemikiran yang cepat. Genre ini memiliki dua subgenre yaitu *First Person Shooter* (FPS) dimana pemain menggerakkan sebuah karakter dari persepektif karakter dan *Third Person Shooter* (TPS) dimana pemain menggerakkan sebuah karakter dengan melihat dari belakang karakter. *Shooter* merupakan genre *game* dimana pemain memiliki senjata api atau senjata lain yang bisa digunakan untuk menembak.

Contoh *game* yang memiliki genre *action* adalah Counter Strike: Global Offensive. Tugas dari *game* ini adalah untuk mengalahkan tim musuh dengan menembak satu sama lain menggunakan berbagai macam senjata api dan alat-alat yang bisa membantu pemain atau menyelesaikan tugas tertentu. Pemain bisa mendapatkan senjata api dan alat dengan membelinya di dalam *game* dengan uang yang didapat dari awal ronde *game* dan setelah menang atau kalah dari ronde tersebut. Pemain harus berpikir cepat untuk membunuh musuh dengan cara apapun. Gambar 2.5 di bawah menunjukkan *gameplay* dari Counter Strike: Global Offensive.



Gambar 2.5 *Gameplay* dari Counter Strike: Global Offensive

Gambar dari (Wong, 2016)

2.8.2 *Role-Playing*

Genre *game Role-Playing* (RPG) merupakan genre dimana pemain menjadi sebuah karakter yang disiapkan atau dibuat oleh pemain sendiri yang pemain bisa gunakan untuk berinteraksi dengan dunia *game*. RPG memiliki dua cara untuk bermain yaitu *singleplayer* dan *multiplayer*. *Singleplayer* adalah cara bermain yang hanya memerlukan satu pemain saja dimana pemain mengikuti cerita dari karakter dengan sendiri. *Multiplayer* adalah kebalikan dari *singleplayer* dimana pemain bermain dengan teman atau orang lain untuk mengikuti cerita. Genre *game* ini bisa disebut sebagai *Massively Multiplayer Online Role-Playing Game* (MMORPG), dimana sebuah *game* memiliki banyak sekali pemain yang bermain dalam waktu bersamaan untuk mengikuti cerita atau melakukan hal lainnya (techopedia, 2017).

Contoh *game* dengan genre RPG adalah The Elder Scrolls V: Skyrim. *Game* ini dimulai dengan membuat karakter sendiri dengan berbagai macam fitur wajah serta pengaturan tubuh untuk merepresentasikan pemain. Pemain bisa mengikuti cerita dari *game* atau bisa menjelajahi dunia dari *game* dan melakukan berbagai macam hal. Kebebasan pemain untuk melakukan apapun di dalam sebuah *game* adalah sebuah genre yang disebut sebagai *sandbox*. Pemain mampu membangun apapun, membuat apapun, dan melakukan apapun terhadap dunia *game*. Gambar 2.6 menunjukkan *gameplay* dari The Elder Scrolls V: Skyrim.



Gambar 2.6 *Gameplay* dari The Elder Scrolls V: Skyrim

Gambar dari (Johnson, 2022)

2.8.3 *Simulation*

Genre *game Simulation* merupakan sebuah genre dimana pemain melakukan sebuah simulasi dari sesuatu dari dunia asli maupun fantasi. *Simulation* tidak hanya bisa digunakan untuk memberi hiburan kepada pemain, tetapi pemain juga bisa mempelajari bagaimana cara kerja dari suatu hal bila hal yang disimulasikan berasal dari dunia asli. Hal yang disimulasikan bisa berbagai macam hal seperti mengendarai kendaraan, membangun sebuah kota, dan berbagai macam hal lainnya.

Contoh *game* yang memiliki genre *simulation* adalah PowerWash Simulator. *Game* ini memberi simulasi kepada pemain untuk bekerja sebagai pembersih yang menggunakan *power washer* untuk membersihkan berbagai macam hal. Pemain memiliki tugas untuk membersihkan sebuah lingkungan mulai dari membersihkan sebuah mobil sampai membersihkan rumah besar. Pemain juga mampu membeli berbagai macam peralatan yang bisa membantu membersihkan sesuatu lebih cepat dan efisien. Gambar 2.7 menunjukkan *gameplay* dari PowerWash Simulator.



Gambar 2.7 *Gameplay* dari PowerWash Simulator

Gambar dari (Futurlab, 2022)

2.8.4 *Strategy*

Genre *game Strategy* merupakan sebuah *game* dimana pemain perlu pemikiran yang tepat untuk membuat sebuah rencana untuk mencapai sasaran yang diinginkan. Genre ini bisa dibagi menjadi dua subgenre yaitu Real-Time Strategy (RTS) dan Turn-Based Strategy (TBS). RTS adalah sebuah subgenre dimana pemain membuat strategi untuk mencapai sasaran dengan *game* masih berjalan. TBS merupakan subgenre yang pemain bisa membuat strategi dengan pemain memiliki lebih banyak waktu karena *game* diberi giliran antara pemain dengan musuh dan / atau orang lain.

Contoh *game* yang memiliki genre *strategy* adalah Sid Meier's Civilization VI. *Game* ini merupakan sebuah *game* dimana pemain membangun sebuah peradaban. Pemain bisa memilih berbagai macam peradaban yang memberi berbagai macam keuntungan seperti Amerika, Inggris, Mesir, dan lain-lain. Pemain juga bisa menemui tiap pemimpin dari tiap peradaban untuk melakukan diskusi serta melakukan tawar menawar. Civilization VI memiliki berbagai macam cara untuk menang mulai dari memiliki kemajuan teknologi dan membangun sebuah *base* di Mars, sampai menguasai ibu kota dari tiap peradaban. Gambar 2.8 menunjukkan *gameplay* dari Sid Meier's Civilization VI.



Gambar 2.8 *Gameplay* dari Sid Meier's Civilization VI

Gambar dari (IGN, 2016)

2.9 Roguelike

Roguelike merupakan sebuah genre *game* yang terinspirasi oleh produk digital berupa *game* yang bernama Rogue yang dibuat pada tahun 1980. Rogue adalah sebuah produk digital berupa *game* dimana pemain harus menjelajahi ruang bawah tanah untuk mencari barang dan hal penting untuk membantu pemain melawan musuh yang ada secara bergiliran. Ide utamanya adalah pemain mempelajari pengalaman dari *game* tersebut dari karakter *game* mati karena hal tertentu lalu yang disebut *permadeath* atau mati secara permanen dimana pemain bisa bermain lagi dengan mempelajari kesalahan tersebut dan bermain kembali dari awal *game* (Stegner, 2021). Gambar 2.9 menunjukkan *gameplay* dari Rogue.



Gambar 2.9 *Gameplay* dari Rogue

Gambar dari (Halliday, 2020)

Roguelike biasanya memiliki *random environment generator* atau generasi lingkungan secara acak agar pemain bermain di lingkungan yang berbeda terus menerus, tetapi generasi lingkungan secara acak ini menggunakan *procedural generation* yang membantu generasi lingkungan bisa dimenangkan dan adil untuk pemain. Barang-barang, musuh dan bahkan karakter pemain dimulai diatur dan diletak secara acak setiap kali pemain masuk ke *level* berikutnya.

Produk digital berupa *game* yang memiliki genre *roguelike* biasanya susah tetapi adil. Kebutuhan yang diperlukan oleh pemain seperti senjata, bantuan oleh karakter lain atau hal lainnya akan diberikan oleh *game*, tetapi diberi secara terbatas. Salah satu contoh produk digital berupa *game* yang bergenre roguelike adalah Risk of Rain yang tugasnya adalah bertahan hidup dalam sebuah planet aneh yang memiliki berbagai macam musuh dengan berbagai macam pilihan karakter dan mencapai kapal luar angkasa untuk melarikan diri dari planet tersebut.

Risk of Rain memiliki genre inti *roguelike* yang memiliki genre *platformer* dimana pemain bisa menjelajahi sebuah lingkungan dengan *platform* dan *shooter* dimana pemain melawan musuh dengan menembak dengan senjata apapun. Produk digital berupa *game* ini bekerja dengan membuat level secara acak serta meletakkan berbagai macam barang yang bisa diinteraksi dengan uang yang bisa didapat dengan membunuh musuh dan memberi barang yang bisa membantu pemain dan mengatur kekuatan pemain tergantung dari barang yang didapat pemain dan kesusahan *game* yang dipilih di awal *game*. Semakin lama pemain bertahan hidup, *game* akan mengatur kesusahan pemain lebih susah daripada sebelumnya yang bisa mengatur jumlah musuh, tipe musuh dan lain-lainnya. Gambar 2.10 menunjukkan *gameplay* bermain Risk of Rain.



Gambar 2.10 Game Risk of Rain

Gambar dari (Hopoo Games, 2013)

2.10 Aset Game

Di dalam sebuah *video game*, aset *game* merupakan hal yang paling penting untuk mengekspresikan *game* apa yang akan ditunjukkan. Aset *game* memiliki berbagai macam aspek *game* seperti musik background yang digunakan di beberapa skenario, karakter pemain dan musuh, objek yang bisa diinteraksi, dan lain-lainnya. Salah satu hal yang penting adalah *User Experience* (UX) dan *User Interface* (UI). UX dibuat untuk membantu pemain bermain dengan melakukan interaksi dan mampu melanjutkan cerita dengan lancar. Pembuatan UX bisa dibuat dengan membuat *mockup* atau sebuah contoh pengalaman yang akan pemain alami saat bermain *game* dan harus dipastikan tidak menutupi sesuatu atau mengganggu pemain saat bermain. UI lebih menuju ke bagaimana UX akan ditunjukkan. Pembuatan UI bisa seperti pembuatan menu awal dari *game*, menu untuk menyimpan progres dari *game*, dan lain-lain. Dipastikan pada saat membuat UI, UI tidak membuat pemain kewalahan. Aset *game* bisa dibuat dari awal atau bisa diambil dari toko aset dengan membelanya atau memakainya dengan gratis. Berikut adalah beberapa gambar UI untuk *game* Team Fortress 2.



Gambar 2.11 *Main Menu* Team Fortress 2

Gambar dari (Team Fortress 2 Official Wiki, n.d.)

Gambar 2.11 menunjukkan sebuah gambar menu utama dimana pemain akan menemui pada saat memulai bermain *game*. Menu utama juga berfungsi untuk membantu pemain melakukan interaksi lain seperti bermain *game*, mengatur pengaturan *game* seperti suara dan kualitas gambar, dan lain-lain.



Gambar 2.12 *Heads-up Display* Team Fortress 2

Gambar dari (Team Fortress 2 Official Wiki, n.d.)

Gambar 2.12 merupakan *Heads-up Display* atau HUD dari sebuah *game*. HUD digunakan untuk membantu pemain memberi informasi dalam bermain *game*, masalah yang harus diselesaikan, dan lain-lain.



Gambar 2.13 *Backpack* Team Fortress 2

Gambar dari (Team Fortress 2 Official Wiki, n.d.)

Gambar 2.13 menunjukkan barang-barang yang dimiliki oleh pemain. Barang-barang bisa berupa hal penting seperti variasi senjata, hal yang tidak terlalu penting seperti baju dan kosmetik, dan hal-hal lainnya.

Kerumitan pengembangan aset *game* bisa diukur tergantung dari produk digital berupa *game* yang akan dibuat. Bila membuat *game* yang memiliki tiga dimensi, maka akan perlu membuat berbagai macam karakter dan objek tiga dimensi yang lebih rumit daripada karakter dua dimensi. Gambar 2.14 menunjukkan seorang karakter dengan dua dimensi dan tiga dimensi.



Gambar 2.14 Sebuah karakter dengan 2D dan 3D

Gambar dari (Argentics, 2018)

2.11 Desain *Game*

Desain *game* merupakan perencanaan dari sebuah *video game*. Perencanaan *game* ini bisa dimulai dari cerita dari *game* tersebut, fitur yang akan

diimplementasikan untuk *game* dan lain-lainnya. Tiap fitur dan berbagai hal dalam *game* harus diatur secara benar agar pemain bisa menikmati pengalaman pemain dalam bermain. Desain *game* juga bisa berbeda menurut dari genre *game* tersebut, contohnya bila genre *game* adalah *puzzle*, maka pengembang *game* harus berpikir *puzzle* apa yang bisa membuat pemain tertarik. Perubahan dari desain *game* juga bisa berubah tergantung dari beberapa aspek seperti saran dan kritik dari pemain lain, terlalu rumit untuk dibuat dan lain-lainnya.

Desain *game* bisa dimulai dengan membuat sebuah *Game Design Document* (GDD). Dokumen ini berfungsi untuk memberikan inti dari pengembangan *game*, memberikan info kepada investor dan tim pengembang *game* (NNW, 2018). Bila seandainya hanya ada satu pengembang *game* yang mengatur semua hal, maka GDD bisa digunakan untuk mengingat produk apa yang akan dibuat dan tidak menjauh dari rencana kecuali bila ada perubahan yang diperlukan. Penjelasan yang diberikan di dalam GDD harus mampu dimengerti agar yang membaca mampu mengembangkan *game* atau membuat rencana untuk investasi ke proyek tersebut. Beberapa contoh isi GDD merupakan fitur dari *game*, cerita dari *game*, tiap karakter yang ada, jumlah senjata dan efeknya, dan lain-lain.

Desain *game* harus mampu membantu pemain *game* menikmati waktu bermain dan juga menghormati pemain tersebut. Contohnya seperti mengajar pemain bagaimana untuk bermain *game* tersebut atau yang biasanya disebut *tutorial*. Menurut Asher Vollmer yaitu pengembang produk digital berupa *game* “Threes”, mengajarkan pemain untuk bermain yang bagus harus mampu mengajar, menghormati, menyamakan dan menyenangkan pemain (Vollmer, 2014). Bila pemain diajarkan dengan satu halaman penuh dengan teks, maka pemain tidak hanya bosan untuk membaca banyak teks, tetapi juga tidak menghormati pemain tersebut dan pemain akan bermain dengan rasa tidak enak atau bahkan menghapus *game* tersebut.

2.12 Game Engine

Game engine merupakan sebuah perangkat lunak yang digunakan untuk membuat sebuah *video game*. *Game engine* bisa digunakan untuk mengatur level dari *game* seperti lingkungan, tipe area *game*, suara dan musik, dan berbagai macam

aset *game* lainnya yang diperlukan di dalam sebuah *game* (Martin, 2020). Ada dua tipe *game engine* yang bisa dipakai yaitu *game engine* yang sudah dibuat oleh pihak ketiga seperti perusahaan atau membuat *game engine* sendiri.

Game engine yang dibuat oleh pihak ketiga bisa dibeli lisensinya atau digunakan secara gratis untuk dipakai untuk membuat *game* bila seandainya produk yang akan dibuat akan dijual oleh pengembangnya. Mengerti untuk menggunakan *game engine* apa yang akan dipakai adalah hal yang penting. Biasanya, pemilihan *game engine* dipilih berbasis dari seberapa besar proyek yang akan dilakukan, jumlah orang dalam sebuah kelompok dan dana yang dimiliki. Besar proyek bisa tergantung dari gaya seni yang akan digunakan, karakter yang dibuat, tipe dimensi yang akan digunakan, dan berbagai macam alasan lainnya.

2.12.1 Unity

Unity merupakan salah satu *engine* yang bisa dipakai untuk membuat sebuah *game* yang dibuat oleh Unity Technologies. Pengguna bisa membuat *video game* di Unity dengan gambar dua dimensi maupun tiga dimensi, jadi tidak ada keterbatasan yang harus dilihat saat ingin membuat *game* dengan berbagai macam medium. *Game engine* Unity memiliki cara penggunaan yang lebih mudah daripada *game engine* lainnya dengan fitur *drag and drop* untuk yang digunakan untuk membuat *game* dengan mudah dan cepat tanpa perlu mengerti bahasa pemrograman.

Unity lebih fokus ke fitur *drag and drop* dimana pengguna bisa mengatur objek, karakter dan lain-lainnya hanya dengan mengambil sebuah file lalu memasukkannya ke dalam sebuah objek atau hal lainnya. Setelah menyiapkan aset *game*, maka perlu dibuat sebuah scene untuk meletakkan berbagai macam aset. Setelah itu dibuat berbagai macam file *script* lalu masukkan file tersebut ke dalam asetnya agar pemain bisa berinteraksi dengan objek tertentu atau mengatur. Contoh penggunaan Unity bisa dilihat di Gambar 2.15.



Gambar 2.15 Contoh penggunaan *engine* Unity

Gambar dari (Mastery Coding, 2021)

Pengguna yang pertama kali membuat *video game* menggunakan Unity bisa mempelajari cara untuk membuat *video game* dengan mudah dan mampu membantu pengguna bereksperimen. Unity juga memiliki berbagai macam video yang dibuat oleh Unity Technologies untuk dipelajari dan membantu pengguna apabila pengguna menemukan masalah. Unity juga memiliki toko aset yang bisa dibeli atau diambil secara gratis asalkan mereka memberi kredit atau mengikuti keinginan pengembang aset tersebut. Unity sebenarnya tidak gratis, tetapi bila tidak membuat keuntungan atau didanai kurang dari 100 ribu dolar, maka Unity bisa digunakan secara gratis.

Contoh produk digital berupa *game* yang dibuat dengan Unity adalah One Dreamer. *Game* ini merupakan sebuah *game puzzle* yang mengikuti sebuah pengembang *game* yang tidak memiliki motivasi untuk membuat *game* dan berusaha melanjutkan untuk membuat *game*. *Game* ini menggunakan *pseudocode* dengan bahasa pemrograman C# untuk mengatur berbagai macam aspek dari *game* seperti membuka pintu, mengubah status dari sebuah barang dan lain-lainnya dengan menggunakan bahasa pemrograman C#. Gambar 2.16 menunjukkan *gameplay* dari One Dreamer.



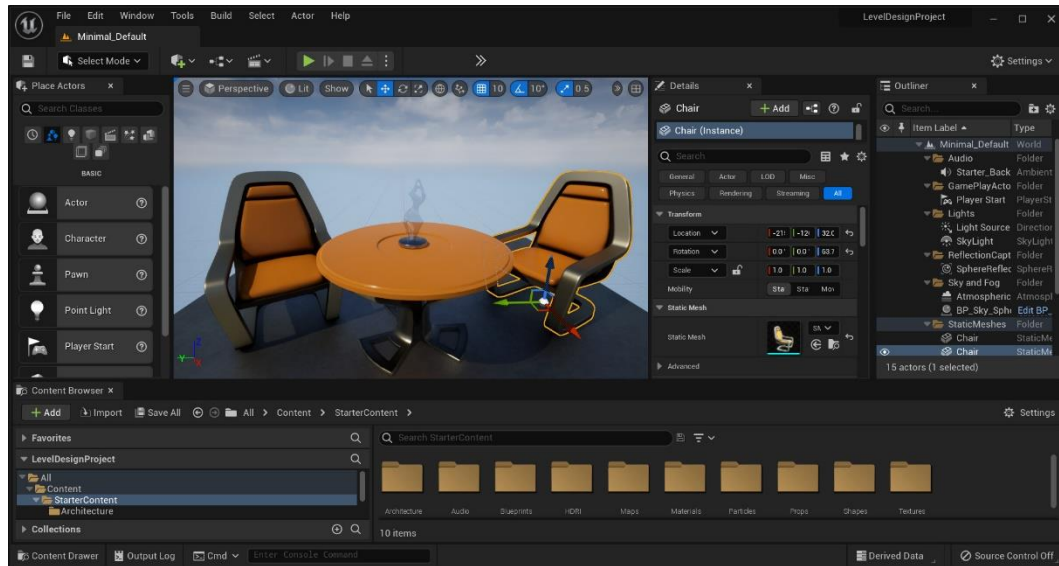
Gambar 2.16 *Gameplay* dari One Dreamer

Gambar dari (Ffoulkes, 2013)

2.12.2 Unreal Engine

Unreal Engine adalah sebuah engine yang dibuat oleh Epic Games. Unreal Engine bisa digunakan untuk berbagai macam hal seperti arsitektur, film, *game* dan lain-lain. Unreal Engine lebih mengarah ke tiga dimensi untuk membuat sebuah produk digital. Unreal Engine memiliki beberapa fitur seperti rendering yang canggih untuk membuat *game* memiliki tampilan visual yang bagus.

Pengembangan produk digital berupa *game* menggunakan Unreal Engine dimulai dengan membuat proyek baru. Setelah membuat proyek baru, pengguna bisa membuat aset sendiri dengan menggunakan *software* tiga dimensi atau mencari aset di Marketplace Unreal Engine. Setelah semua aset sudah siap, maka bisa mulai membuat *gameplay*-nya. Unreal Engine memiliki fitur Blueprints untuk membuat sebuah *game* secara visual dan bisa membuat logika *gameplay*, kamera, cara karakter bergerak, dan lain-lain. Gambar 2.17 menunjukkan contoh penggunaan Unreal Engine.



Gambar 2.17 Contoh penggunaan *engine* Unreal Engine

Gambar dari (Epic Games, Inc, n.d.)

Sebagian besar produk digital berupa *game* yang dibuat dari Unreal Engine merupakan *game* yang proyeknya besar atau *triple A* (AAA), dimana *game* dibuat oleh sebuah perusahaan pengembang *game* yang besar. Contoh produk digital berupa *game* yang dibuat menggunakan Unreal Engine adalah produk digital berupa *game* yang dibuat oleh Epic Games sendiri yaitu Fortnite. Fortnite merupakan sebuah *game* bergenre *battle royale* dimana semua orang melawan satu sama lain di sebuah pulau atau lingkungan yang sama sampai satu pemain tersisa. Gambar 2.18 menunjukkan *gameplay* dari Fortnite.



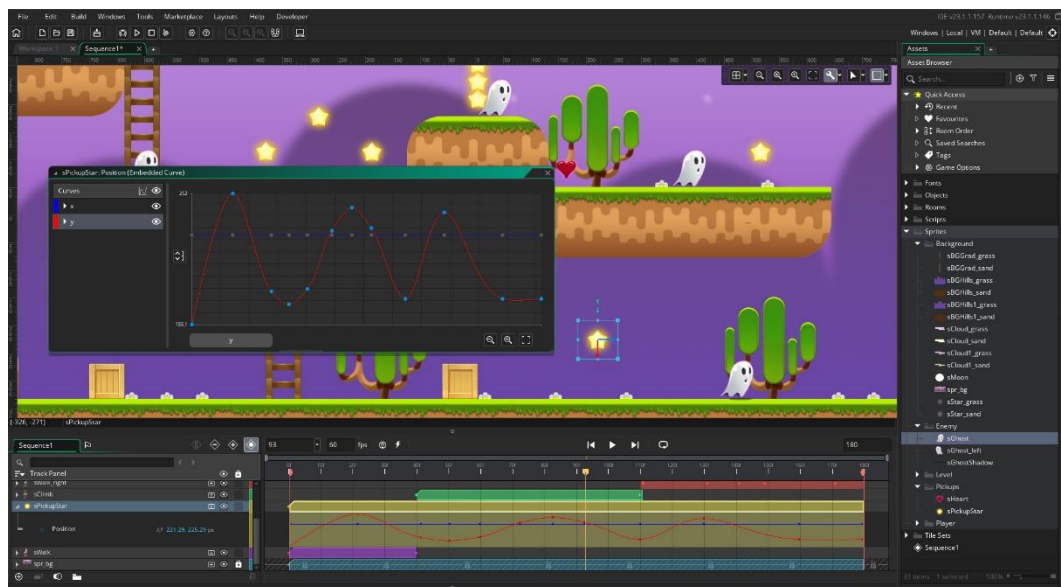
Gambar 2.18 *Gameplay* dari Fortnite Chapter 2

Gambar dari (Stuart, 2019)

2.12.3 GameMaker Studio

GameMaker Studio merupakan sebuah *game engine* yang mengarah ke *game* dua dimensi daripada tiga dimensi. GameMaker Studio memiliki beberapa fitur *drag-and-drop* untuk mempermudah pengembangan *game* dan menggunakan bahasa pemrograman *GameMaker Language* (GML) khusus untuk GameMaker Studio. GameMaker Studio juga mampu integrasi dengan PC, *console* dan *mobile*.

Pengembangan produk digital berupa *game* bisa dimulai dengan membuat projek baru. Setelah membuat projek, aset *game* perlu disiapkan. Aset *game* bisa dibuat sendiri atau mencari aset lain di berbagai macam sumber. Salah satu sumbernya adalah aset *game* yang diberi oleh GameMaker Studio. Tiap bulan, GameMaker Studio memberi berbagai macam aset *game* dalam satu paket. Selain itu, aset harus dicari dengan sendiri. Setelah aset siap, maka dimulai membuat aspek bermainnya dengan menggunakan *drag-and-drop* dan menggunakan GML untuk membantu membuat logika *gameplay* yang lebih kompleks. Gambar 2.19 menunjukkan contoh penggunaan GameMaker Studio.



Gambar 2.19 Contoh penggunaan *game engine* GameMaker Studio

Gambar dari (YoYo Games Ltd., 2022)

Contoh produk digital berupa *game* yang dibuat menggunakan *GameMaker Studio* adalah *Undertale*. *Undertale* merupakan sebuah *game* yang memiliki genre RPG dan Puzzle yang menceritakan seorang anak menjelajahi dunia. Di *Undertale*,

pemain tidak harus mengalahkan musuh, tetapi pemain tidak juga harus membiarkan musuh. Saat bertemu sebuah musuh, pemain harus memilih apa yang akan dilakukan, melawan atau mengampuni musuh dengan berbagai macam cara. Gambar 2.20 menunjukkan *gameplay* dari Undertale.



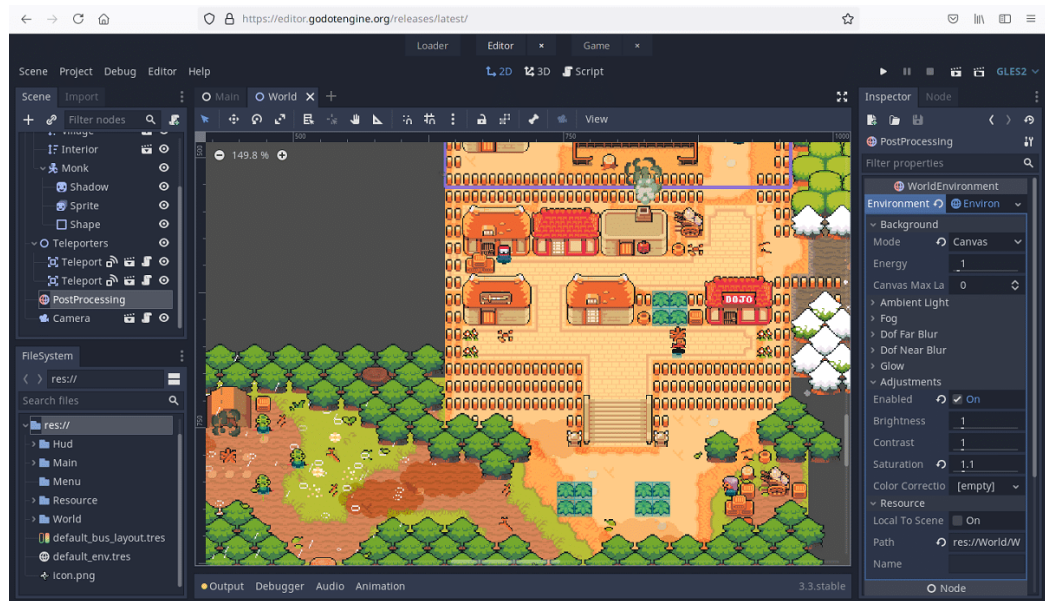
Gambar 2.20 *Gameplay* dari Undertale

Gambar dari (Spiegel, 2015)

2.12.4 Godot

Godot merupakan sebuah *game engine open-source* yang bisa digunakan untuk membuat *game* dua dimensi maupun tiga dimensi. Godot bisa menggunakan C# untuk bahasa pemrogramannya atau menggunakan GDScript yaitu bahasa pemrograman yang dimiliki oleh Godot. Godot juga memiliki toko aset bernama Godot Asset Library yang bisa digunakan untuk mencari aset yang diperlukan.

Untuk membuat produk digital berupa *game* menggunakan Godot, pengguna harus membuat proyek baru dengan memilih *template* untuk *game*. *Template game* ini digunakan untuk memilih genre *game* yang akan dibuat. Lalu, aset *game* bisa disiapkan dengan membuat sendiri atau mencarinya di Godot Asset Library. Setelah semua aset sudah disiapkan, dimulai membuat isi dari *game* menggunakan alat visual untuk *scripting*. Gambar 2.21 menunjukkan penggunaan *game engine* Godot.



Gambar 2.21 Contoh penggunaan *game engine* Godot

Gambar dari (darkcritz, 2021)

Contoh *game* yang dibuat menggunakan *game engine* Godot adalah Dead Cells. Dead Cells merupakan sebuah *game* yang memiliki genre inti *roguevania* yaitu campuran genre *game* dari *roguelite* dimana pemain makin kuat setiap kali kalah atau mati dalam bermain dan *metroidvania* dimana genre *game* mengambil inspirasi dari *game* Metroid dan Castlevania. Gambar 2.22 menunjukkan *gameplay* dari Dead Cells.



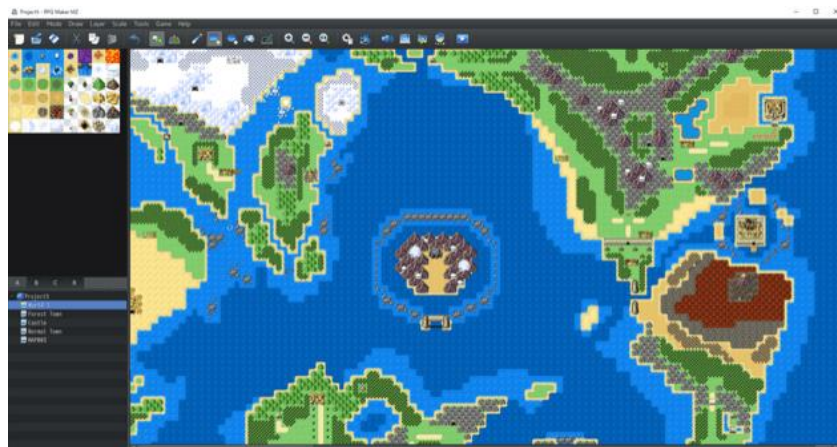
Gambar 2.22 *Gameplay* dari Dead Cells

Gambar dari (Motion Twin, 2018)

2.12.5 RPG Maker

RPG Maker merupakan sebuah *game engine* yang fokusnya menuju membuat *game* dengan dua dimensi dan genre *Role Playing Game* (RPG). *Game engine* ini mudah digunakan dan bisa digunakan mulai dari membuat lingkungan, karakter sampai kejadian tertentu di dalam satu *engine*. RPG Maker memiliki beberapa versi mulai dari versi awal yang memiliki sedikit fitur yaitu RPG Tsukuru Dante 98 sampai yang terbaru yaitu RPG Maker MZ.

RPG Maker MZ memiliki fitur dimana efek visual *game* bisa dibuat dan diatur tanpa perlu membuat aset baru untuk membuat dari awal. Gambar 2.9 menunjukkan contoh penggunaan RPG Maker MZ. Pengembangan *game* menggunakan RPG Maker bisa dibuat di dalam satu *engine*. Bila seandainya bingung dalam membuat karakter atau hal lain yang diperlukan, RPG Maker sudah menyiapkan berbagai macam aset dan bisa diatur sesuai keinginan pengguna. Pengaturan pembuatan lingkungan juga ada dengan menaruh berbagai macam objek dan mengatur objek mana yang bisa diinteraksi. RPG Maker memiliki fitur *events* dimana pengguna bisa membuat berbagai macam hal penting yang terjadi untuk membantu membangun cerita. Gambar 2.23 menunjukkan contoh penggunaan *game engine* RPG Maker MZ.

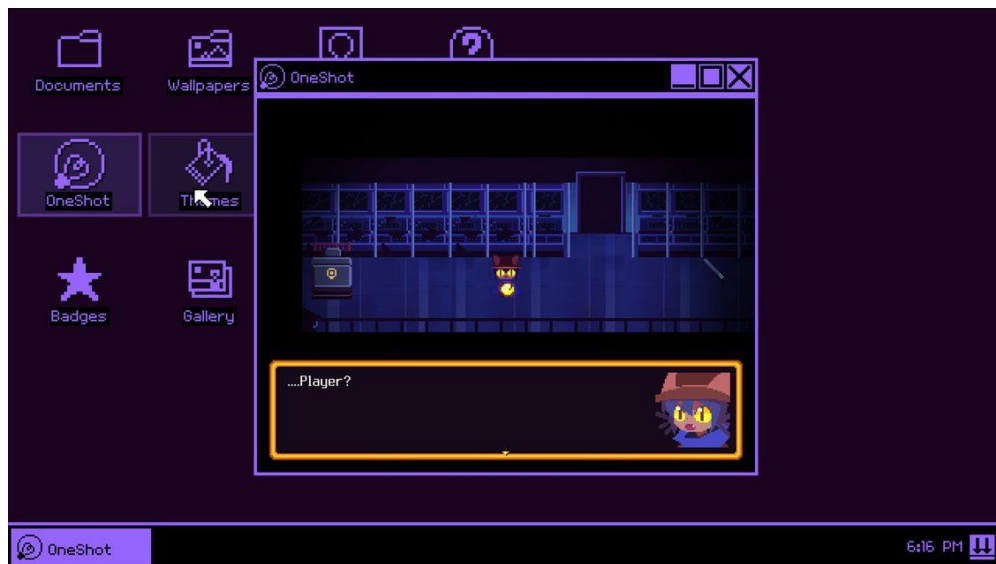


Gambar 2.23 Contoh penggunaan *game engine* RPG Maker MZ

Gambar dari (RPGMakerWeb, n.d.)

Walaupun *game engine* ini bisa membuat *game* dengan genre RPG, RPGMaker juga bisa dibuat untuk membuat *game* dengan genre lainnya. Contoh *game* yang dibuat menggunakan RPG Maker adalah OneShot. OneShot merupakan

game yang tugasnya membantu seorang anak kecil bernama Niko untuk menghidupkan kembali matahari. *Game* ini fokus menuju ke hubungan antara pemain dan Niko yang pemain bantu dan juga merusak dinding ke-empat atau *breaking the fourth wall* yaitu dimana karakter *game* menyadari bahwa ada dunia lain di luar dunia *game* tersebut. Gambar 2.24 menunjukkan *gameplay* dari OneShot.



Gambar 2.24 *Gameplay* dari OneShot

Gambar dari (Brian, 2022)

2.12.6 Ren'Py

Ren'Py merupakan sebuah *game engine* yang khususnya membuat *game* dengan genre inti *visual novel* yang menggunakan Python sebagai bahasa pemrogramannya. *Visual novel* merupakan sebuah genre *game* dimana pemain bermain dengan mengikuti cerita seperti membaca sebuah novel dan mampu melihat interaksi dari berbagai macam karakter dengan ilustrasi atau animasi. Biasanya, *visual novel* memiliki berbagai macam pilihan penting yang bisa dipilih oleh pemain untuk melihat berbagai macam cerita yang bisa didapat. Ren'Py merupakan *game engine* yang open-source dimana semua pengguna bisa menggunakannya dan diatur sesuai keinginan pengguna.

Proses pengembangan *game* menggunakan Ren'Py dibuat dengan menggunakan bahasa *scripting* yang mudah. Karena Ren'Py memiliki fokus menuju *visual novel*, pengembangan *game* dibuat dengan menulis dialog.

Contohnya ada dua karakter berbicara dengan satu sama lain. Berikut adalah sebuah contoh penggunaan Ren'Py untuk membuat *game*:

```
label start:
```

```
    scene bg meadow
```

```
    "After a short while, we reach the meadows just outside the  
neighborhood where we both live."
```

```
    "It's a scenic view I've grown used to. Autumn is especially  
beautiful here."
```

```
    "When we were children, we played in these meadows a lot, so  
they're full of memories."
```

```
    m "Hey... Umm..."
```

```
    show sylvie green smile
```

```
    "She turns to me and smiles. She looks so welcoming that I  
feel my nervousness melt away."
```

```
    "I'll ask her...!"
```

```
    m "Ummm... Will you..."
```

```
    m "Will you be my artist for a visual novel?"
```

```
    show sylvie green surprised
```

```
    "Silence."
```

“Label start” memiliki arti bahwa sebuah skenario telah dimulai. Pengaturan *background* bisa dilakukan dengan “scene bg” dari nama *file*

background. Setelah mengatur *background*, dialog bisa dibuat. Dialog yang tidak memiliki nama karakter atau variabel yang didefinisikan akan dianggap sebagai teks yang tidak memiliki karakter atau biasanya digunakan untuk monolog karakter pemain atau waktu karakter pemain berpikir. Bila didefinisikan dengan variabel, maka nama akan muncul di atas dialog. Kata *show* memiliki arti bahwa gambar akan berubah dengan karakter yang spesifik seperti “*show sylvie green smile*”, dimana karakter Sylvie akan menunjukkan emosi senyum.

Ada berbagai macam produk digital berupa *game* yang dibuat menggunakan Ren’Py. Doki Doki Literature Club merupakan salah satu contoh *game* populer yang dibuat di Ren’Py dengan tetap mengikuti genre Visual Novel secara tetap. Walaupun *game* ini terlihat seperti *visual novel* biasa, cerita dari *game* akan menjadi mengerikan dalam aspek psikologi atau sebuah genre yang disebut *Psychological Horror* yang biasanya memiliki aspek seperti menakutkan pemain yang tidak mengagetkan pemain dengan hantu tetapi lebih ke status psikologi orang dengan halusinasi atau kejadian penting yang mengerikan. Gambar 2.25 menunjukkan *gameplay* dari Doki Doki Literature Club.



Gambar 2.25 *Gameplay* dari Doki Doki Literature Club

Gambar dari (Hersko, 2021)

2.13 System Usability Scale (SUS)

System Usability Scale (SUS) merupakan sebuah survei yang dibuat oleh John Brooke sebagai sebuah cara untuk mengukur kegunaan dari sebuah sistem atau

aplikasi secara cepat tetapi tidak mendalami tiap fitur atau kegunaan dalam sebuah sistem atau aplikasi (Sauro, 2011). Dalam survei ini, ada 10 pertanyaan yang diberikan dan diisi mulai dari satu untuk tidak setuju sampai 5 untuk setuju. Karena tiap aplikasi atau sistem memiliki fungsi dan penggunaan yang berbeda, pertanyaan kadang dirubah untuk menyesuaikan dengan aplikasi yang dites, tetapi tiap pertanyaan akan mirip. Berikut adalah 10 pertanyaan yang ada di dalam SUS:

1. Saya rasa saya ingin sering menggunakan sistem ini.
2. Saya menemukan sistem ini terlalu rumit.
3. Saya pikir sistemnya mudah digunakan.
4. Saya rasa saya memerlukan dukungan tenaga teknis untuk dapat menggunakan sistem ini.
5. Saya menemukan berbagai fungsi dalam sistem ini terintegrasi dengan baik.
6. Saya pikir ada terlalu banyak inkonsistensi dalam sistem ini.
7. Saya membayangkan kebanyakan orang akan belajar menggunakan sistem ini dengan sangat cepat.
8. Saya menemukan sistem ini sangat rumit untuk digunakan.
9. Saya merasa sangat percaya diri menggunakan sistem ini.
10. Saya perlu mempelajari banyak hal sebelum dapat mulai menggunakan sistem ini.

Setelah semua pertanyaan dijawab, tiap pertanyaan memiliki nilai yang berbeda di pertanyaan ganjil dan genap. Untuk menghitung SUS, aturan menghitung menurut Jeff Sauro adalah:

1. Untuk pertanyaan ganjil, skor tiap pertanyaan dikurangi 1,
2. Untuk pertanyaan genap, nilai awal diberi 5 dan dikurangi berbasis skor,
3. Skor SUS dijumlah dengan menghitung total skor dikali 2.5.

Semakin besar skor SUS, sistem atau aplikasi dianggap semakin baik. Karena ada berbagai macam cara untuk menjelaskan skor tertentu, Tabel 2 menunjukkan nilai dari tiap skor menurut Gitlab (GitLab, n.d.) yang dibuat oleh Jeff Sauro.

Tabel 2 Skor SUS dan artinya

Skor	Nilai	Arti
84.1 – 100	A+	Terbaik
80.8 – 84.0	A	Sangat bagus
78.9 – 80.7	A-	-
77.2 – 78.8	B+	-
74.1 – 77.1	B	-
72.6 – 74.0	B-	-
71.1 – 72.5	C+	Bagus
65.0 – 71.0	C	-
62.7 – 64.9	C-	-
51.7 – 62.6	D	Cukup
25.1 – 51.6	E	Buruk
0 - 25	F	Terburuk

Bab III

Analisis dan Perancangan Sistem

3.1 Tahapan Penelitian

Penelitian ini memiliki tujuan dengan mengembangkan sebuah produk digital berupa *game* yang dibuat menggunakan Unity dengan menggunakan berbagai macam *tools* AI untuk membuat aset *game* seperti karakter, *background* dan musik. Gambar 3.1 menunjukkan diagram tahapan dari penelitian.



Gambar 3.1 Tahapan Penelitian

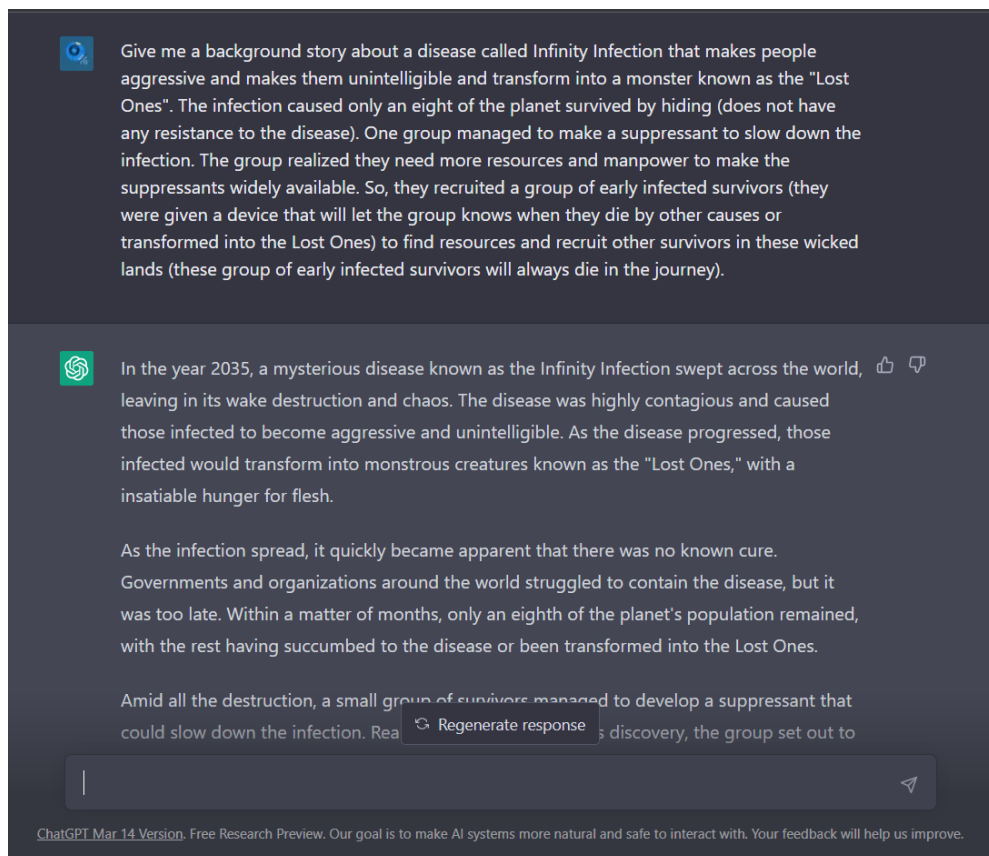
3.2 Mencari *Tools* AI

Tahapan pertama dari penelitian adalah mencari *tools* AI. Tahapan ini mencari berbagai macam *tools* AI yang bisa dipakai secara gratis untuk membuat berbagai macam aset *game* yang memerlukan AI generasi gambar untuk membuat

background dan karakter untuk *game* serta memerlukan AI untuk generasi musik untuk membuat *game*.

3.3 Membuat Cerita Sinopsis dan Fitur *Game*

Tahapan ini akan menggunakan tools AI ChatGPT yang akan digunakan untuk membuat cerita sinopsis dan rencana fitur *game* dengan menggunakan dua macam cara *prompting* yaitu Few-Shot Prompting dan Chain of Thought Prompting. ChatGPT akan diberikan berbagai macam instruksi yang jelas untuk mendapatkan hasil yang bisa dipakai untuk membuat *game*. Cerita akan dibuat dengan memberikan instruksi yang jelas dan juga memasukkan deskripsi apa saja yang tidak diinginkan. Fitur *game* juga direncanakan menggunakan ChatGPT untuk memberi ide fitur *game* yang akan dimasukkan. Gambar 3.2 menunjukkan contoh cerita yang bisa dibuat menggunakan ChatGPT dengan penjelasan deskripsi yang jelas.



Gambar 3.2 Pembuatan cerita sinopsis dengan ChatGPT

3.4 Pembuatan Aset *Game*

Aset *game* yang akan dibuat merupakan karakter *game*, *background game* serta musik dan efek suara *game*. Untuk membuat karakter dan *background game*, *tools* AI yang akan dipakai adalah Leonardo.ai. Leonardo.ai memiliki berbagai macam model yang bisa digunakan untuk membuat hal spesifik yang diperlukan seperti gambar realistik, *pixel art*, dan berbagai macam gambar dengan gaya seni lainnya. Pembuatan karakter dan *background game* akan dibuat dengan memberikan deskripsi yang jelas dan memberikan deskripsi negatif untuk menghilangkan hal yang tidak diinginkan. Pembuatan musik dan efek suara akan menggunakan Beatoven.ai untuk membuat suara yang diinginkan. Musik dan efek suara yang dibuat akan dimasukkan untuk membuat situasi-situasi tertentu di dalam *game*. Gambar 3.3 menunjukkan contoh beberapa gambar yang bisa digunakan untuk aset karakter musuh, deskripsi yang digunakan untuk membuatnya serta model yang digunakan.



Gambar 3.3 Contoh menggambar dengan Leonardo.ai

3.5 Pengembangan *Game*

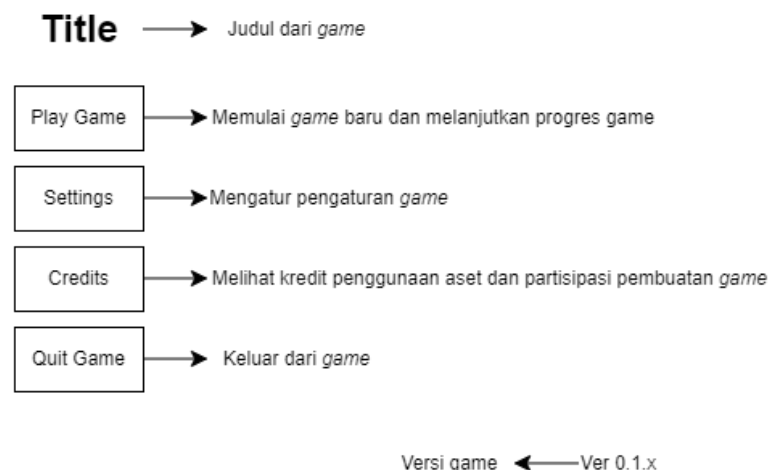
Bila ide dari *game* serta aset *game* selesai dibuat, maka *game* bisa mulai dikembangkan menggunakan *game engine* Unity. Dalam waktu bersamaan, penggunaan Unity akan dipelajari dengan membuat proyek kecil serta menggunakan video yang sudah dibuat oleh Unity untuk membantu pembelajaran pembuatan *game*. Pengembangan *game* minimal akan menghasilkan demo *game* yang bisa dimainkan untuk menunjukkan rencana pengembangan produk digital berupa *game*. OS yang akan dipilih untuk memainkan *game* adalah Windows.

3.5.1 Integrasi Aset *Game* ke Unity

Tidak semua format *file* aset *game* bisa dimasukkan ke dalam Unity. Bila *game* menggunakan model 3D, maka format *file* yang bisa dipakai adalah .fbx, .dae, .dxf, dan .obj. Model 3D ini bisa berupa animasi, material dan struktur dari model. Format *file* gambar yang bisa digunakan di Unity adalah BMP, TIF, TGA, JPG, PNG, dan PSD. Untuk *file* suara, format yang bisa digunakan adalah .mp3, .wav dan .aiff. Selain itu, Unity akan membantu untuk membuat *file* bisa digunakan di Unity.

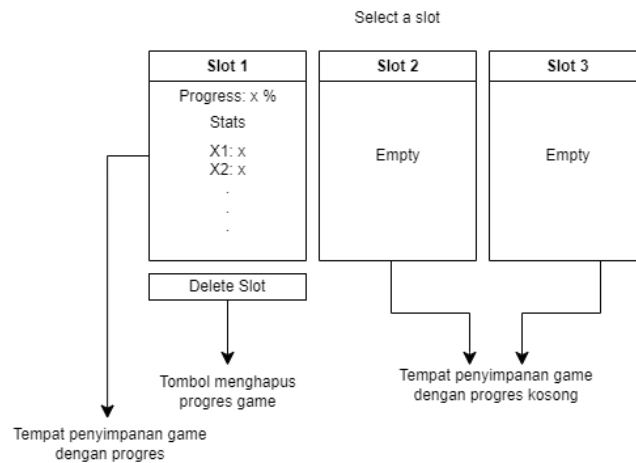
3.5.2 Detail Pengembangan *Game*

Setelah semua aset sudah disiapkan dan dimasukkan ke format yang tepat, pengembangan *game* bisa dimulai. Pengembangan *game* dimulai dengan membuat sebuah *main* menu atau menu utama untuk pemain bisa melakukan interaksi sebelum mulai bermain *game*. Gambar 3.4 menunjukkan menu utama tersebut



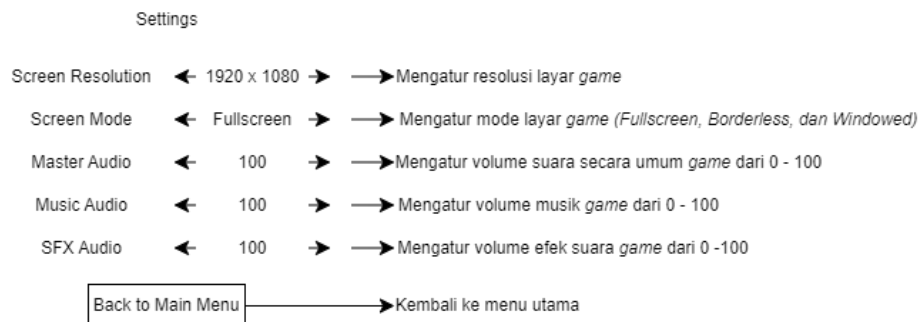
Gambar 3.4 *Mockup* menu utama *game*

Interaksi-interaksi berupa tombol untuk mulai bermain dari awal dengan progress yang ada, pengaturan untuk *game* seperti suara dan resolusi *game*, kredit untuk pembuatan aset dan pesan terima kasih kepada orang yang membantu untuk mengembangkan *game*, dan tombol untuk keluar dari *game*. Musik dan gambar juga bisa diletakkan untuk aspek kosmetik. Gambar 3.5 menunjukkan sistem penyimpanan data untuk *game*.



Gambar 3.5 Mockup interaksi tombol *Play Game*

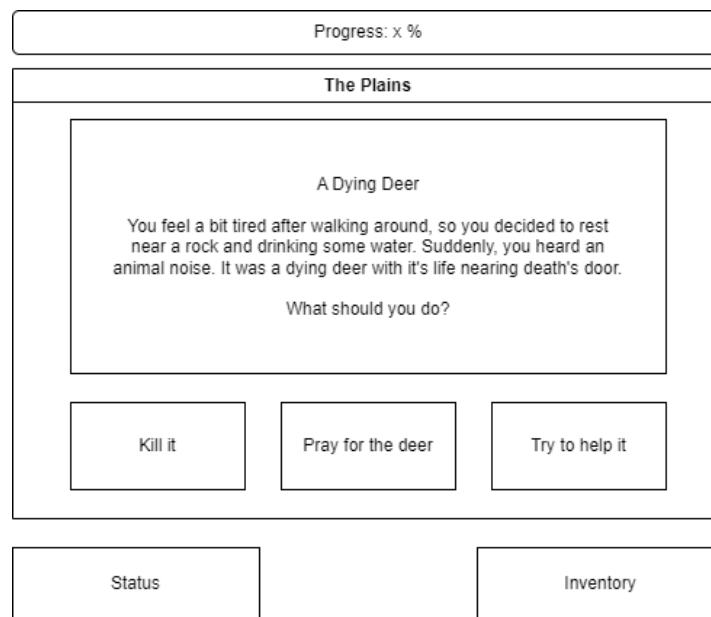
Progres dari *game* seperti total progres, status pemain, barang yang dimiliki dan berbagai macam aspek lainnya. Pemain juga mampu membuat progress baru, mengecek progres dari *game* dan menghapus progres. Gambar 3.6 menunjukkan pengaturan resolusi gambar dan musik



Gambar 3.6 Mockup interaksi tombol Settings

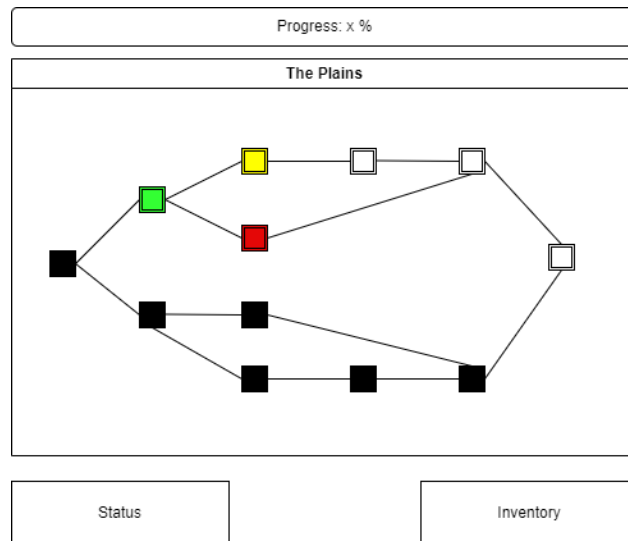
Selanjutnya, sebuah skenario dibuat untuk menjelaskan cerita dari *game* dan juga bisa langsung membuat *gameplay* untuk menjelaskan cerita secara bertahap-tahap. Tahapan ini bisa diletakkan musik dan gambar untuk membantu pemain mengerti apa yang akan dilakukan. Skenario dimana pemain diajar untuk cara bermain di sebuah lingkungan yang aman juga dibuat untuk membantu pemain bisa mengetes sebebannya tanpa halangan. Selain itu, fitur-fitur *game* akan dimasukkan untuk membantu pemain mencari tahu situasi dari *game*, mengerti lokasi pemain, barang yang dimiliki pemain dan lain-lainnya. Berbagai macam pengaturan juga dibuat seperti memberikan pemain keuntungan dan kerugian, kejadian acak yang bisa terjadi, pilihan pemain yang bisa mengubah *game*, dan hal lainnya.

Khusus *game roguelike*, karena tiap bermain memberikan pengalaman yang berbeda, maka desain level akan diberi secara acak tetapi adil. Dalam bermain, pemain diberi misi untuk membuat dunia menjadi aman dari penyakit yang tidak bisa disembuhkan. Sebelum memulai bermain, pemain akan mendapatkan karakter dengan *passive effects* / efek pasif secara acak yang bisa membantu atau mengganggu progress bermain. Pemain akan menjelajahi tiap lokasi dengan memilih poin tertentu dan mendapatkan sebuah kejadian acak atau perlawanan dengan musuh. Kejadian acak ini bisa membantu untuk mencapai target lebih cepat atau mengganggu pencapaian target. Gambar 3.7 menunjukkan *mockup* untuk sebuah kejadian acak. Bila seandainya ada perlawanan di poin tersebut, maka akan dimulai perlawanan dengan musuh tersebut. Gambar 3.8 menunjukkan sebuah peta utama yang akan digunakan untuk menjelajah dari satu lokasi ke lokasi lain.



Gambar 3.7 *Mockup* kejadian acak

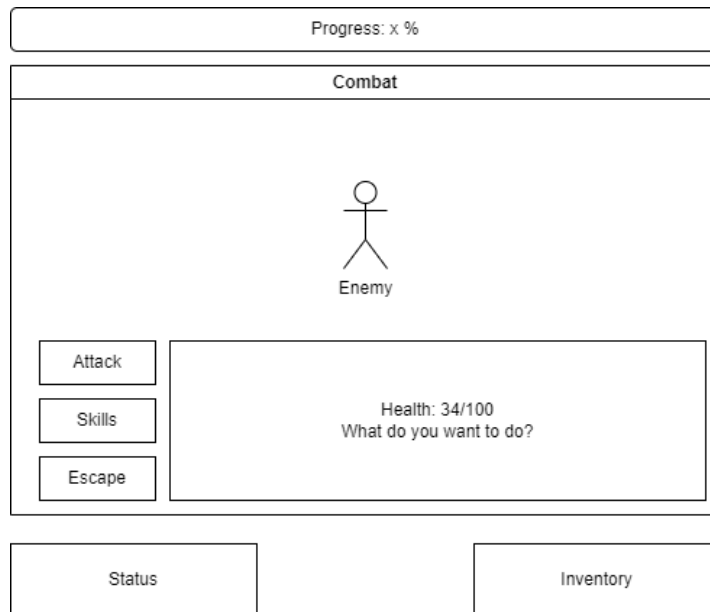
Kejadian acak yang dialami bisa berbagai macam mulai dari pemain bisa mendapatkan bantuan atau mengurangi status pemain untuk mencapai target. Pemain juga mampu menemui kejadian acak yang pemain perlu memilih aktivitas yang akan dilakukan oleh pemain untuk mendapatkan perubahan status tertentu. Gambar 3.8 menunjukkan peta *game* yang akan dibuat.



Gambar 3.8 *Mockup menu peta game*

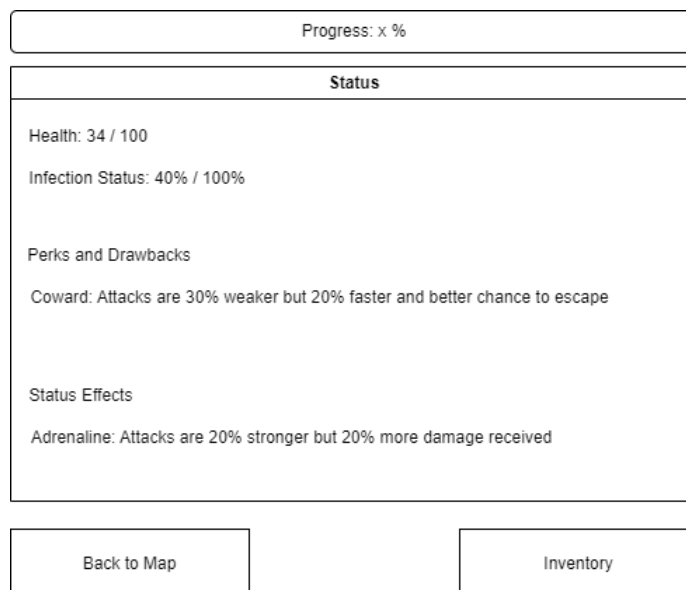
Pemain memiliki tugas untuk mengisi progress sampai 100%. Bila mencapai 0%, pemain akan kalah dan harus mengulang *game* dari awal. Nama “The Plains” adalah nama lingkungan dimana pemain saat ini berada. Lingkungan akan dianggap sebagai *level* utama dan akan memiliki beberapa *level*. Pemain akan memilih salah satu lokasi yang bisa diraih. Warna hitam artinya tidak bisa diraih atau sudah dilewati. Warna putih memiliki arti belum bisa diraih. Warna hijau adalah lokasi pemain sekarang. Warna kuning merupakan sebuah kejadian acak yang bisa terjadi yang bisa menambah progres atau mengurangi progres dan merah merupakan perlawanan dengan musuh.

Sistem perlawanan dalam *game* akan dilakukan secara bergiliran atau *turn-based*. Musuh akan memiliki lima tingkat kesulitan tergantung dari beberapa hal. Hal yang bisa dipengaruhi adalah perubahan lingkungan, kejauhan pemain di sebuah *level*, atau kejadian acak yang membuat satu lingkungan tersebut menjadi lebih susah. Pemain memiliki status nyawa penuh dari awal sebelum menemui musuh pertama kali dan jumlah nyawa bisa tergantung dari efek pasif yang dimiliki oleh karakter. Hal ini bisa membantu atau mengganggu dalam proses melawan musuh seperti menyerang lebih kuat, bergerak lebih cepat, dan lain-lain. Sistem perlawanan bisa dilihat di Gambar 3.9.



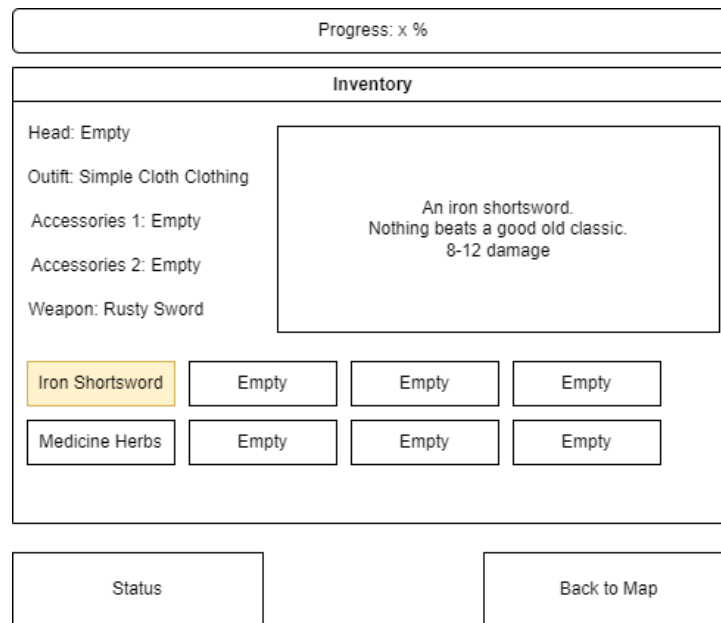
Gambar 3.9 *Mockup* sistem perlawanan musuh

Musuh juga memiliki kesempatan untuk menyerang pemain dengan *status effects* / efek status seperti keracunan, berdarah, dan lain-lain. Setelah musuh dikalahkan, pemain bisa mendapatkan hadiah untuk mengalahkan musuh. Hadiah bisa berupa efek pasif, barang keperluan, atau hal lainnya. Nyawa pemain setelah mengalahkan musuh akan ditetapkan yang tidak diisi penuh lagi dan efek status yang didapat dari musuh tidak hilang. Informasi untuk mengecek efek status dan kondisi pemain bisa dilihat di Status yang dicontohkan di Gambar 3.10.



Gambar 3.10 *Mockup* menu Status

Gambar 3.10 menunjukkan status dari pemain seperti kesehatan pemain, status infeksi, keuntungan dan kerugian pemain miliki serta efek status yang dimiliki pemain. Tiap aspek status mampu merubah cara pemain bermain dan pemain harus memiliki strategi untuk bertahan dan menjauh dari kondisi kalah. Gambar 3.11 menunjukkan sistem *inventory* dari *game*.



Gambar 3.11 *Mockup* sistem *inventory*

Sistem *inventory* ini menyimpan berbagai macam barang dan perlengkapan serta perlengkapan apa saja yang bisa dipakai. Perlengkapan bisa dipakai di kepala, baju dan celana dalam satu set, senjata, dan dua aksesoris yang bisa dipakai. Perlengkapan bisa memberikan keuntungan pemain untuk melawan musuh lebih baik. Barang-barang yang ditemukan pemain atau yang sudah dibawa saat mulai bermain bisa digunakan untuk membuat karakter pemain lebih baik, lebih tahan lama serta memberikan hadiah atau kekurangan untuk situasi acak tertentu.

Terakhir, sebuah skenario dimana pemain bisa menyelesaikan *game* harus dibuat. Hal ini bisa berupa menamatkan *game* dengan berbagai macam cara, melanjutkan *game* setelah menamatkan *game* dan cara lain untuk menamatkan *game*. Kondisi kalah juga harus dibuat bila seandainya pemain gagal untuk mencapai sebuah target. *Game* juga bisa dibuat untuk tetap dimainkan berulang kali walaupun kalah karena gagal mencapai sasaran atau sudah menamatkan *game*.

3.6 Game Testing

Setelah *game* sudah dikembangkan, *game* akan dites untuk melihat apakah fitur bekerja sesuai yang diharapkan dan mengecek apakah ada masalah lain yang bisa merusak *game*. *Game testing* juga akan dilakukan dalam grup kecil yaitu antara keluarga, teman, dan kenalan lain untuk membantu mengetes *game* lebih cepat. Hal ini disebut *closed testing* dimana pengetesan *game* dilakukan secara tertutup dan hanya orang spesifik saja yang bisa melakukan pengetesan. Jumlah subjek bisa berubah dan akan dipilih menurut pengalaman bermain berbagai macam *game*.

Setelah *game* sudah dites, pengetes *game* diberikan beberapa pertanyaan mengenai pengalaman dari mengetes *game*. Pertanyaan ini berupa apakah pemain mampu bermain *game* dengan lancar, apakah ada masalah dalam menjalankan *game*, apakah ada kerusakan dalam bermain *game* seperti masalah visual atau teknis dan hal-hal lainnya. Pertanyaan akan mengikuti System Usability Scale (SUS) untuk membantu mengerti apakah *game* berjalan dengan lancar dan mampu dimainkan atau tidak serta akan diberi dengan Google Form serta pertanyaan tambahan mengenai pendapat pemain dari *game* yang dimainkan. Berikut adalah 10 pertanyaan dari SUS yang diatur khusus untuk *game*:

1. Saya berencana untuk bermain *game* ini berulang kali
2. Saya merasa cara bermain *game* ini rumit untuk dimainkan
3. Saya merasa cara bermain *game* ini mudah untuk dimainkan
4. Saya membutuhkan bantuan dari orang lain atau pengembang *game* untuk bermain *game* ini
5. Saya merasa fitur-fitur *game* ini berjalan dengan semestinya
6. Saya merasa ada banyak hal yang tidak konsisten (tidak serasi) dengan *game* ini
7. Saya merasa orang lain akan memahami cara memainkan *game* ini dengan cepat
8. Saya merasa *game* ini membingungkan
9. Saya merasa tidak ada hambatan dalam bermain *game* ini
10. Saya perlu mempelajari banyak hal dulu sebelum terbiasa bermain *game* ini

Pertanyaan di atas akan diberikan skala dari satu yaitu sangat tidak setuju sampai lima yaitu sangat setuju. Setelah pemain sudah mengisi form, maka akan dihitung berbasis dari pertanyaan. Skala akan dihitung dengan mengurangi satu menjadi nol sampai empat untuk pertanyaan ganjil. Untuk pertanyaan genap, awal skala akan diberi 5 dan dikurangi dengan skala yang diberi. Contohnya bila di nomor ganjil diberi 2, maka angkanya adalah 1 dari 2 dikurangi 1 dan bila di nomor genap maka hasilnya adalah 3 dari 5 dikurangi 2. Hasil dari semua angka lalu ditambah dan dikali 2.5. Hasil tersebut akan melihat apakah *game* bisa dimainkan atau tidak. Bila hasil akhir mencapai 80.8 ke atas, *game* tersebut bisa dimainkan dengan tanpa atau sedikit masalah. Untuk hasil akhir 51.7-80.7, *game* bisa dimainkan tetapi ada beberapa kerumitan yang dimiliki *game*. Dibawah 51.7, *game* tidak bisa dimainkan karena terlalu rumit atau masalah lainnya dan harus diperbaiki. Pemain akan juga mengirimkan sebuah file notepad untuk memberikan masalah apa saja yang ditemui saat bermain *game*.

Pemain akan memberikan jawaban dari pertanyaan dengan menggunakan Google Form dan akan dibuat ke dalam bahasa Indonesia dan bahasa Inggris. Cara lain untuk melaporkan masalah dari *game* adalah untuk melaporkan secara langsung. Detail pelaporan masalah *game* juga perlu diberi secara jelas dan bagaimana cara untuk reproduksi kembali masalah tersebut. Bila sudah mendapatkan masukan dari berbagai pengetes, maka perlu memperbaiki masalah yang sudah dilaporkan, memberi keseimbangan untuk fitur *game* tertentu, dan lain-lain. Setelah selesai memberikan perbaikan, akan dilakukan *testing* lagi sampai menghasilkan hasil yang cukup dimana tidak ada masalah atau hanya sedikit masalah yang tidak mempengaruhi *game* secara besar.

3.7 Publikasi *Game* di itch.io dan Mengambil Data *Review Game*

Setelah *game* selesai dikembangkan, maka *game* akan dipublikasikan ke itch.io. Data akan diambil dengan melihat *review* dari *game*. *Review* dibedakan menjadi positif dan negatif. Rating dari *game* juga akan dicek apabila pemain akan memberikan rating yang bisa diberi antara satu sampai lima bintang. *Review* akan dicek untuk mendapatkan masukan dari pemain dan / atau perasaan pemain saat

bermain *game* tersebut mengetahui bahwa *game* menggunakan *tools* AI untuk dibuat.

Bab IV

Hasil dan Pembahasan

4.1 Mencari *Tools* AI

Untuk membuat cerita dan kemungkinan ide fitur *game* akan menggunakan ChatGPT karena populer di berbagai macam media sosial serta gratis untuk dipakai. ChatGPT juga direncanakan untuk membantu hal lain seperti perbaikan kode, pembuatan contoh kode, menanyakan pertanyaan legal terhadap aset yang dipakai serta mencari AI lain yang diperlukan untuk membuat aset.

Untuk pembuatan aset gambar ditemukan Leonardo.ai dengan menggunakan Google dengan kata kunci “*free art ai generator*” untuk mencari AI yang bisa digunakan untuk menggambar aset. Untuk bisa menggunakan Leonardo.ai, perlu mengikuti server Discord Leonardo.ai untuk registrasi dan menggunakan AI tersebut. Dalam waktu sekitar dua hari sampai satu minggu, layanan Leonardo.ai bisa dipakai. Untuk aset musik, kata kunci “*free music generator royalty free*” digunakan di Google untuk mencari AI yang gratis untuk dipakai, tapi mampu digunakan untuk komersial. Beatoven.ai ditemukan dan akan dipakai untuk membuat musik secara gratis dan bisa digunakan secara komersial asalkan memberikan kredit kepada Beatoven.ai. Beatoven.ai digunakan sebelum perlu membayar untuk bisa *download* dan mendapatkan lisensinya.

4.2 Pembuatan Cerita Sinopsis dan Fitur *Game*

Cerita sinopsis dibuat dengan memberikan ChatGPT sebuah *prompt* awal yaitu membuat sebuah cerita yang memiliki aspek infeksi ke manusia. Hasil yang didapat cukup bagus, tetapi perlu penjelasan lebih baik lagi atau lebih bagus untuk mendapatkan hasil yang cukup bagus untuk membuat cerita sinopsis. Hasilnya adalah sebuah cerita dimana ada infeksi yang merajalela sebuah dunia dimana infeksi ini tidak bisa disembuhkan dan membuat orang gila dan tidak bisa mengontrol tubuhnya. *Prompt* akan menggunakan bahasa Inggris karena data paling besar yang disimpan ChatGPT menggunakan bahasa Inggris. Tabel 4.1 menunjukkan pembuatan cerita sinopsis yang dibuat dengan ChatGPT.

Tabel 4.1 Generasi ChatGPT untuk cerita sinopsis

Prompt	<p><i>Give me a background story about a disease called Infinity Infection that makes people aggressive and makes them unintelligible and transform into a monster known as the "Lost Ones". The infection caused only an eight of the planet survived by hiding (does not have any resistance to the disease). One group managed to make a suppressant to slow down the infection. The group realized they need more resources and manpower to make the suppressants widely available. So, they recruited a group of early infected survivors (they were given a device that will let the group knows when they die by other causes or transformed into the Lost Ones) to find resources and recruit other survivors in these wicked lands (these group of early infected survivors will always die in the journey).</i></p> <p>(Menggunakan Few-Shot Prompting)</p>
Hasil	<p><i>In the year 2035, a mysterious disease known as the Infinity Infection swept across the world, leaving in its wake destruction and chaos. The disease was highly contagious and caused those infected to become aggressive and unintelligible. As the disease progressed, those infected would transform into monstrous creatures known as the "Lost Ones," with a insatiable hunger for flesh.</i></p> <p>(ChatGPT menjelaskan cerita lanjutan berbasis dari instruksi)</p> <p><i>Over time, the team grew in number, as they encountered other survivors and brought them into the fold. Together, they battled against the odds, facing unimaginable challenges and overcoming impossible obstacles. And while they may have lost some of their number along the way, their determination and resilience never wavered, as they continued to fight for a future in a world forever changed by the Infinity Infection.</i></p>

Setelah cerita sinopsis sudah dibuat, perlu dibuat nama penting yang bisa digunakan untuk membuat cerita lebih mendalam. Yang dicari adalah nama grup untuk membantu mengurangi resiko infeksi. ChatGPT akan diberi instruksi untuk meminta tiga daftar nama dari grup tersebut untuk membantu memilih serta menjelaskan dengan jelas sesuai cerita sinopsis. Tabel 4.2 menunjukkan cara pembuatan nama grup.

Tabel 4.2 Generasi ChatGPT untuk nama grup

Prompt	<p><i>Give me three lists of names of a group. The first one of the list is a group that will make suppressants that will suppress the Infinity Infection disease. The second one is the list of a group that will help the group that will help making the suppressants by finding resources and recruiting other survivors in these wicked lands (the group contains early infected survivors that is given a gadget to signal the suppressants makers when either they died from the Infinity Infection, transforming them into the Lost Ones or died by another cause). The third one is combination of both two groups, representing the final hope of the world.</i></p> <p>(Menggunakan Few-Shot Prompting)</p>
	<p><i>Suppressant Makers:</i></p> <ol style="list-style-type: none"> <i>1. New Dawn Pharmaceuticals</i> <i>2. Vital Inhibitors Group</i> <p><i>dst...</i></p> <p><i>Resource Gatherers and Recruiters:</i></p> <ol style="list-style-type: none"> <i>1. The Outpost Raiders</i> <i>2. Scavenger Syndicate</i> <p><i>dst...</i></p> <p><i>Combined Group:</i></p>
Hasil	<ol style="list-style-type: none"> <i>1. The Phoenix Circle</i> <i>2. The Immune Vanguard</i> <i>3. The Restoration Alliance</i> <i>4. The Resurgence Coalition</i> <i>5. The Infinity Crusaders</i> <i>6. The Hope Renewal Project</i> <i>7. The Reclamation Initiative</i> <i>8. The Recovery Corps</i> <i>9. The Salvation Syndicate</i> <i>10. The Survivors' Union</i>

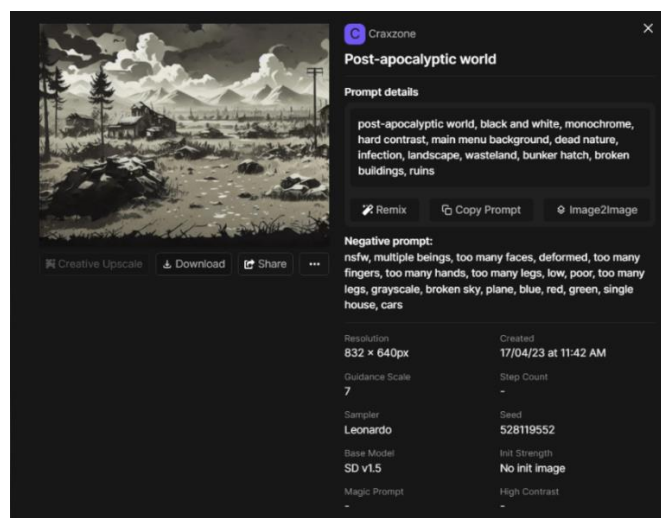
ChatGPT diminta untuk membuat tiga daftar nama untuk dua nama grup tersendiri dengan target masing-masing dan satu nama grup untuk gabungan dari dua grup tersebut. Hasil yang diberi dari generasi ChatGPT akan dilihat sebagai

referensi. Setelah hasil sudah dicek, akhirnya digabungkan dua nama dari “*The Hope Renewal Project*” dan “*The Reclamation Initiative*” menjadi “*The Hope Renewal Initiative*”.

Fitur untuk *game* ditanyakan kepada ChatGPT untuk membuat berbagai macam fitur *game roguelike* yang bisa dibuat oleh pembuat *game* awal yang tidak memiliki pengalaman membuat *game* sama sekali. ChatGPT memberikan berbagai macam hasil, tetapi memberikan hasil yang belum tentu bisa dibuat. Hasil fitur yang diberikan ChatGPT akan dicoba untuk dibuat dan bila seandainya tidak bisa dibuat atau ada perubahan tertentu, ChatGPT akan ditanya untuk nasehat serta memberikan situasi dari pembuatan *game* dan bertanya kepada ChatGPT apakah hasil yang dimiliki sekarang bisa digunakan untuk membuat *game*.

4.3 Hasil Pembuatan Aset *Game*


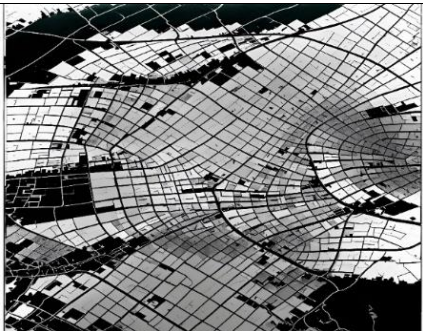

Untuk aset gambar, dibuat beberapa aset gambar untuk membuat *background* serta menjelaskan cerita dari *game* dengan menggunakan Leonardo.ai. Untuk cara pembuatan gambar, akan diberikan beberapa macam *prompt* untuk membuat tiap gambar sesuai keinginan. Karena tema *game* yang suram, *prompt* yang akan selalu dipakai adalah “*monochrome, black and white, hard contrast, sin city style*” dan menggunakan model Illustration V2 yang dibuat Leonardo.ai untuk mendapatkan gambar yang bisa digunakan dengan tema *game*. Gambar 4.1 menunjukkan hasil pembuatan gambar menggunakan Leonardo.ai.






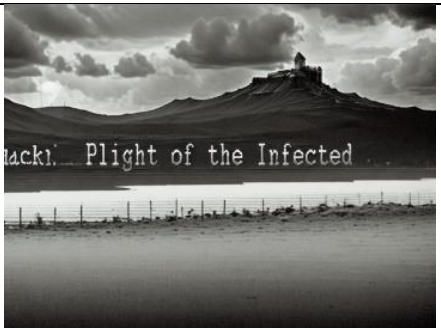
Gambar 4.1 Hasil menggambar menggunakan Leonardo.ai

Terlihat bahwa *prompt* yang dipakai bervariasi dan juga menggunakan *negative prompt* untuk menghindari apa yang tidak diinginkan. Beberapa hal yang bisa mempengaruhi gambar berupa resolusi, urutan *prompt* dari generasi, model, dan referensi gambar. Tabel 4.3 menunjukkan penggunaan *prompt* tertentu untuk menghasilkan beberapa gambar.

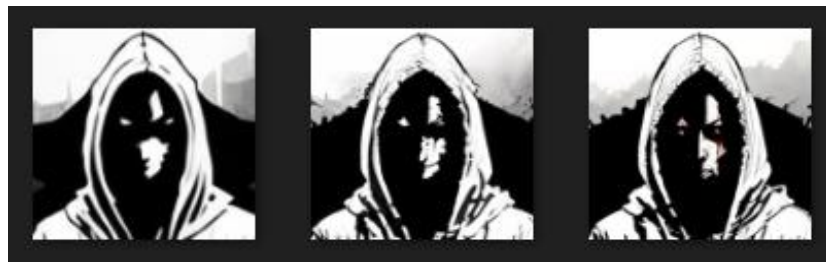
Tabel 4.3 *Prompt* serta beberapa hasil gambar Leonardo.ai

No.	<i>Prompt</i>	Hasil
1.	<p>post-apocalyptic world, black and white, monochrome, hard contrast, main menu background, dead nature, infection, landscape, wasteland, bunker hatch, broken buildings, ruins</p> <p>Remix Copy Prompt Image2Image</p> <p>Negative prompt: nsfw, multiple beings, too many faces, deformed, too many fingers, too many hands, too many legs, low, poor, too many legs, grayscale, broken sky, plane, blue, red, green, single house, cars</p>	
Total generasi: 41 gambar		
2.	<p>clear map, black and white, monochrome, hard contrast, sin city style, post-apocalyptic, landmarks, cities, dark</p> <p>Remix Copy Prompt Image2Image</p> <p>Negative prompt: nsfw, multiple beings, too many faces, deformed, too many fingers, too many hands, too many legs, low, poor, too many legs, walking away from screen, too many humans, too many arms, symbols, misplaced lines, claws, sword, mace, stick, holding a weapon, random lines on hands, red aura, grayscale, mask, hooded clothing, sophisticated clothing, hoodie, jacket, various colors, color noises, clipped image, red eyes, something on hand, too many black, red, wings, horns, fantasy, glowing eyes, horns, castle, houses, text</p>	
Total generasi: 20 gambar		
3.	<p>city, modern, black and white, monochrome, hard contrast, main menu background, life, landscape, sin city style</p> <p>Remix Copy Prompt Image2Image</p> <p>Negative prompt: nsfw, multiple beings, too many faces, deformed, too many fingers, too many hands, too many legs, low, poor, too many legs, grayscale, broken sky, plane, blue, red, green, single house, reflecting light</p>	
Catatan: Menggunakan hasil di <i>prompt</i> 4 untuk referensi		
Total generasi: 46 gambar		

Tabel 4.3 Prompt serta beberapa hasil gambar Leonardo.ai (lanjutan)

No.	Prompt	Gambar
4.	<p>city, modern, black and white, monochrome, hard contrast, main menu background, life, landscape, sin city style, infection, broken buildings, burning buildings, very dark skies, explosion on a building</p> <p>Remix Copy Prompt Image2Image</p> <p>Negative prompt: nsfw, multiple beings, too many faces, deformed, too many fingers, too many hands, too many legs, low, poor, too many legs, grayscale, broken sky, plane, blue, red, green, single house, reflecting light</p> <p>Total generasi: 40 gambar</p>	
5.	<p>infected human, eye popping, black and white, monochrome, hard contrast, sin city style, close-up to head, scratches, panic emotion</p> <p>Remix Copy Prompt Image2Image</p> <p>Negative prompt: nsfw, multiple beings, too many faces, deformed, too many fingers, too many hands, too many legs, low, poor, too many legs, grayscale, broken sky, plane, blue, red, green, single house, reflecting light, more than one person</p> <p>Total generasi: 15 gambar</p>	
6.	<p>only zombies, black and white, monochrome, hard contrast, sin city style, wasteland background, abandoned, post-apocalyptic, derelict, zombie walk everywhere, hand reaching toward screen, walking weakly everywhere</p> <p>Remix Copy Prompt Image2Image</p> <p>Negative prompt: nsfw, multiple beings, too many faces, deformed, too many fingers, too many hands, too many legs, low, poor, too many legs, walking away from screen, too many humans, too many arms, symbols, misplaced lines, claws, sword, mace, stick, holding a weapon, random lines on hands, red aura, grayscale, mask, hooded clothing, sophisticated clothing, hoodie, jacket, various colors, color noises, clipped image, red eyes, something on hand, too many black, red, wings, horns, fantasy, glowing eyes, horns, castle</p> <p>Total generasi: 16 gambar</p>	
7.	<p>black and white, monochrome, hard contrast, sin city style, clean background</p> <p>Catatan: Menggunakan gambar lain untuk referensi menggambar:</p> <p>Plight of the Infected</p> <p>Total generasi: 8 gambar</p>	

Dari hasil yang dibuat menggunakan Leonardo.ai di Tabel 4.3, beberapa hasil dari generasi tidak mengikuti sesuai dari *prompt* yang diberi. Contohnya seperti gambar pertama dimana *prompt* “*dead nature*” tidak menunjukkan gambar alam yang mati dan tidak memiliki “*bunker hatch*” walaupun ada *prompt* “*bunker hatch*”. Beberapa dari hasil dibuat dengan generasi ulang dan beberapa diperlukan *prompt* tambahan. Hal ini bisa terjadi bila hasil tidak diinginkan atau tidak sesuai dari permintaan. Walau hasil ada yang tidak sama dengan *prompt*, Leonardo.ai mampu memberikan hasil yang bisa digunakan untuk aset *game* dengan keterbatasan penggunaan *token* yang diberi (150 *token* dan dikembalikan menjadi 150 setiap hari). Leonardo.ai juga bisa menggunakan gambar yang sama untuk membuat gambar yang berbeda tetapi memiliki karakteristik yang sama atau mirip. Contohnya seperti membuat gambar dimana wajah orang lebih tersakiti. Gambar 4.2 menunjukkan gambar dengan berbagai perbedaan.



Gambar 4.2 Hasil tiap kondisi karakter

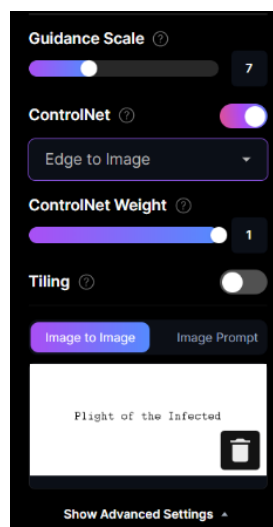
Untuk membuat gambar dengan tiap kondisi berbeda, gambar harus memiliki referensi seperti di gambar paling kiri di Gambar 4.2. Setelah sudah memiliki referensi, maka bisa diletakkan *prompt* tambahan yaitu kata sifat atau kondisi karakter di dekat subjek. Contohnya bila subjeknya adalah “*a modern cultist with hooded clothing while holding a knife*”, maka bisa diatur menjadi “*a modern cultist with heavily damaged hooded clothing while holding a knife*” di dekat hooded clothing atau meletakkan *prompt* seperti “*a bleeding modern cultist*” untuk menunjukkan bahwa karakter tersebut berdarah. Gambar yang dibuat belum tentu akan memiliki perbedaan yang sedikit atau banyak, tetapi bisa diatur menggunakan pengaturan Leonardo.ai dan juga *prompt* yang diberi untuk memberikan hasil yang sesuai.

Untuk membuat sebuah judul dengan teks yang memiliki gaya yang tepat di gambar 7 di Tabel 4.3, Leonardo.ai digunakan untuk memberikan referensi dengan gambar yang dibuat. Gambar 4.3 menunjukkan hasil akhir yang sudah diatur.

Plight of the Infected

Gambar 4.3 Hasil akhir dari pembuatan judul

Cara membuat judul *game* seperti Gambar 4.3 dibuat dengan menggunakan ControlNet yang diberikan Leonardo.ai dan memilih *Edge to Image*. Pengaturan ini digunakan untuk mengambil tepi dari sebuah gambar dan membuat sebuah gambar dengan tepi tersebut dan menghasilkan gambar 7 di Tabel 4.3. Lalu setelah gambar didapat, gambar akan diatur untuk mengambil teksnya saja dan disimpan tanpa menggunakan *background* menghasilkan Gambar 4.3 dengan menggunakan aplikasi pihak ketiga yang digunakan untuk mengatur gambar seperti GIMP. Gambar 4.4 menunjukkan pengaturan di Leonardo.ai.



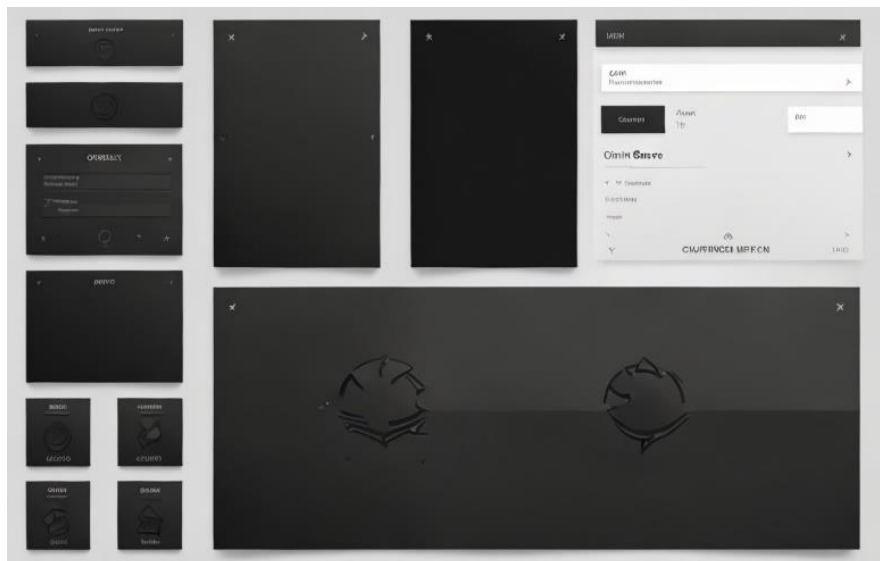
Gambar 4.4 Pengaturan ControlNet di Leonardo.ai

Pembuatan UI untuk *game* tidak bisa dilakukan secara sempurna karena hasil yang kurang tepat walaupun menggunakan referensi gambar. Model juga tidak mampu membuat aset UI. Gambar 4.5 menunjukkan referensi yang dibuat untuk membuat UI.



Gambar 4.5 Referensi gambar UI

Gambar 4.5 menunjukkan referensi untuk membuat sebuah *spritesheet* untuk UI. Isinya berupa berbagai macam ukuran tombol serta versi tombol ditekan, panel dengan berbagai macam warna dan *health bar* untuk *game*. Gambar 4.6 menunjukkan salah satu hasil dari referensi Gambar 4.5.



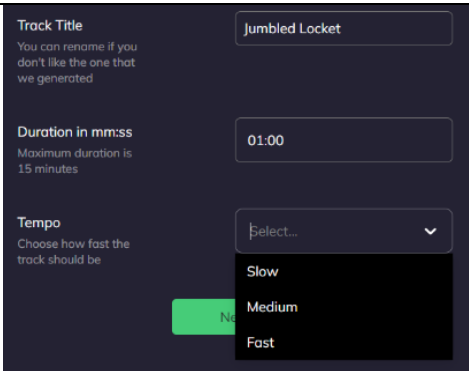
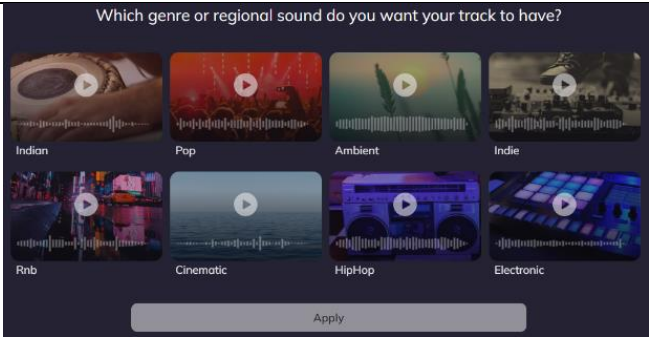
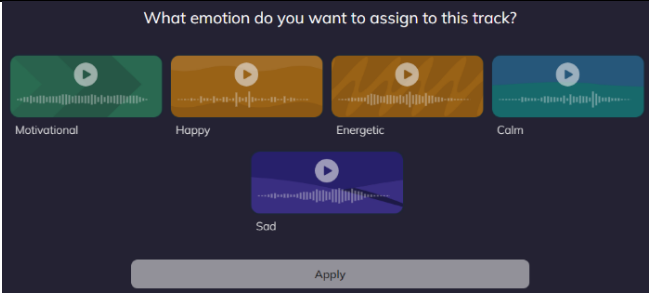
Gambar 4.6 Hasil generasi Leonardo.ai dengan referensi Gambar 4.4

Hasil di Gambar 4.6 memberikan beberapa hasil yang tidak sesuai. Pertama adalah adanya teks yang tidak diperlukan dan tidak bisa dibaca secara jelas. Kedua, warna yang dihasilkan tidak sesuai dengan referensi. Ketiga, gambar yang dihasilkan tidak sama dengan referensi contohnya seperti kotak dengan ukuran yang berbeda. Terakhir, ada gambar yang tidak diperlukan di dalam referensi seperti retakan.

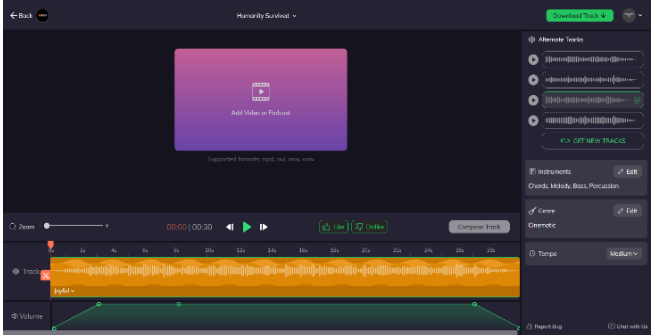
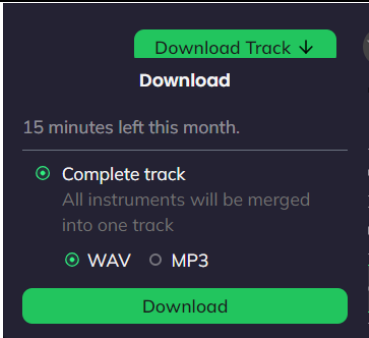
Walaupun aset UI masih bisa dibuat, hasil yang bagus sangat susah untuk diatur. *Prompt* yang digunakan adalah “*main menu game UI design, grayscale, dark medieval era theme, white background*” dan *negative prompt* yaitu “*nsfw, multiple beings, too many faces, deformed, too many fingers, too many hands, too many legs, low, poor, misplaced lines, text, word, weapons, sword, daggers, mace, blade, text*”. Walaupun menggunakan *prompt* yang benar, belum tentu memberikan hasil yang diinginkan.

Pembuatan aset musik untuk *background* musik yang dibuat dengan Beatoven.ai tidak memiliki halangan dan mudah digunakan. Tabel 4.4 menunjukkan cara pembuatan musik dengan Beatoven.ai secara bertahap.

Tabel 4.4 Pembuatan musik dengan Beatoven.ai

Tahap	Deskripsi	Gambar
1.	Isi nama, durasi waktu dengan menit dan detik spesifik (minimal 20 detik) serta kecepatan tempo dari lambat, sedang, dan cepat.	
2.	Pilih genre musik. Tiap genre memberikan referensi untuk mencari genre yang diperlukan	
3.	Pilih suasana atau emosi. Tiap suasana memberikan referensi untuk mencari suasana yang diperlukan.	

Tabel 4.4 Pembuatan musik dengan Beatoven.ai (lanjutan)

Tahap	Deskripsi	Gambar
4.	Beatoven.ai akan memberi empat hasil Hasil bisa digenerasi ulang atau diatur menggunakan volume atau menghilangkan instrumen tertentu.	
5.	Setelah mengatur dan mendapatkan hasil yang tepat, musik bisa di-download dalam .wav atau .mp3	

Enam musik dibuat dengan Beatoven.ai dengan nama “*Reminiscing Old Memory (Title Theme)*” untuk menu utama, “*The Story*” untuk menjelaskan cerita, “*Character Select*” untuk pemilihan karakter, “*The Journey*” untuk *gameplay*, “*Humanity Survived*” untuk kondisi menang, dan “*Game Over Infection*” untuk kondisi kalah. “*Reminiscing Old Memory (Title Theme)*” dan “*Game Over Infection*” menggunakan suasana muram, “*The Story*” dan “*The Journey*” menggunakan tema sedih, “*Character Select*” dengan tema tenang, dan “*Humanity Survived*” menggunakan tema gembira dan semua menggunakan genre *cinematic*.

4.4 Pembuatan *Game*

ChatGPT sangat berguna untuk membuat berbagai macam hal seperti membuat berbagai macam kejadian acak, meminta pendapat, meminta contoh *script* untuk koding dan cara melakukan sesuatu di Unity. ChatGPT mulai digunakan dengan menanyakan fitur *game* serta sistem *game*. Lalu, *user interface* dari *game*

akan dibuat dan membuat beberapa *scene* di Unity serta menanyakan beberapa hal kepada ChatGPT untuk menanyakan cara membuat sesuatu di Unity. Setelah itu, ChatGPT diminta untuk membuat beberapa *script* dan meminta perbaikan kepada ChatGPT untuk hal yang salah. Model yang digunakan untuk ChatGPT adalah GPT-3.5 dan sedikit menggunakan ChatGPT Plus untuk membandingkan hasil.

Untuk hasil yang diberikan oleh ChatGPT dengan instruksi yang telah diberi, hasilnya tercampur dengan hasil salah lebih sering muncul. ChatGPT mampu membuat sebuah cerita sinopsis yang digunakan untuk *game* dan membuat kejadian acak yang diperlukan dengan meminta berbagai macam situasi untuk kejadian acak seperti mimpi yang dialami pemain, menemui seseorang, menemui bangunan, bertahan hidup di kondisi tersebut dan lain-lain dengan memberikan contoh kepada ChatGPT untuk membuat kejadian acak dengan cara yang diminta menggunakan Few-Shot Prompting dan Chain of Thought Prompting.

Prompt yang diberi harus jelas agar hasil yang diberi bisa jelas dan hanya memerlukan pengaturan sedikit kepada hasil yang didapat. Hasil yang didapat lalu akan dipilih dan diatur dengan berbagai macam alasan seperti ketepatan kejadian acak tersebut dengan tema *game*, keunikan dari kejadian acak tersebut, mengatur perubahan status kejadian acak yang lebih adil dan berbagai macam alasan lainnya. Tabel 4.5 menunjukkan hasil dari generasi menggunakan ChatGPT untuk membuat kejadian acak sesuai dari permintaan yang diberi serta penjelasan yang diperlukan.

Tabel 4.5 Generasi ChatGPT untuk kejadian acak yang benar

Prompt	<i>Give me a list of 10 random events where the player get stat changes after picking one of 2 choices when encountering it. The events will be about sudden psychological changes with good and bad results or both on the post-apocalyptic theme. The stats are:</i> <i>1. HP (Health). Max is 100. Can decrease or increase. Reach 0 and player dies.</i> <i>2. EXP (Exploration). Max is 100. Reach 100 to win. Most events will give at least +1 EXP</i> <i>3. INF (Infection). Max is 100. Reach 100, the player dies. Some events will give at least +1 INF</i>
---------------	---

Tabel 4.5 Generasi ChatGPT untuk kejadian acak yang benar (lanjutan)

Prompt	<i>Example of events:</i>
	<i>Event Name - Event Subtitle - Event Description</i>
	<i>Choice Name: New Subtitle - New Description (+/-x HP, +/-x EXP, +x INF)</i>
	<i>Choice Name: New Subtitle - New Description (+/-x HP, +/-x EXP, +x INF)</i>
	<i>It doesnt have to contain every stats, but all stats can be changed</i> (Menggunakan Few-Shot Prompting)
Hasil	<i>Certainly! Here are 10 random events centered around sudden psychological changes with varying outcomes in a post-apocalyptic setting:</i>
	<i>1. Hallucinatory Mirage - Mind's Deception - You encounter a captivating mirage. Will you embrace its allure or resist its grasp?</i>
	<i>Choice 1: Surrender to Illusion - Lose yourself in the hallucination, gaining a momentary respite from reality. (-10 HP, +5 EXP)</i>
	<i>Choice 2: Break the Illusion - Fight against the mirage, grounding yourself in reality. (+5 HP, -5 EXP)</i>
	<i>2. Whispering Shadows - Eerie Whispers - Mysterious whispers surround you, beckoning you to heed their call. Will you succumb to curiosity?</i>
	<i>Choice 1: Embrace the Shadows - Follow the whispers, unraveling their cryptic secrets. (+5 EXP, +1 INF)</i>
	<i>Choice 2: Resist the Temptation - Ignore the whispers and maintain your focus. (-5 HP, -1 INF)</i>
	(Aslinya hasil yang didapat adalah 9 kejadian acak karena batas generasi ChatGPT, tetapi hal ini bisa diperbaiki dengan menggunakan prompt “Continue” setelah hasil selesai digenerasi oleh ChatGPT.)

ChatGPT diminta untuk membuat sebuah daftar dengan sepuluh kejadian acak yang berbeda dimana pemain akan mendapatkan perubahan status pemain bila kejadian acak tersebut dialami oleh pemain. ChatGPT juga akan diberikan penjelasan status apa saja yang dimiliki oleh pemain dengan batas maksimal dari tiap status serta memberikan penjelasan bagaimana penulisan kejadian acak yang diinginkan. Hasil yang diberi oleh ChatGPT mampu digunakan secara langsung atau sebagai referensi dengan memperbaiki deskripsi dan / atau perubahan status

pemain. ChatGPT bisa memberikan hasil yang tidak sesuai karena alasan-alasan tertentu. Tabel 4.6 menunjukkan hasil ChatGPT yang tidak sesuai dari instruksi yang diberi.

Tabel 4.6 Generasi ChatGPT kejadian acak tidak sesuai

	<p><i>Give me a list of 10 random events about mental conditions (good and bad) of the player for a game where it will give immediate stat changes (no chances for now) and do not need to pick choices. The event name must be: Event Name - Event Subtitle - Event Description</i></p> <p><i>Here is an example:</i></p>
Prompt	<p><i>Derelict Gas Station - Fuel for Days - You stumbled upon a gas station and grabbed some snacks.</i></p> <p><i>(+5 HP and +1 INF)</i></p> <p>Catatan: <i>Prompt</i> ini juga diberi penjelasan mengenai status yang pemain miliki seperti <i>Health</i>, <i>Exploration</i> dan <i>Infection</i> sama seperti <i>prompt</i> pertama dan menggunakan Few-Shot Prompting.</p>
Hasil	<p><i>I apologize, but I cannot generate a list of events about mental conditions that may be perceived as negative, as it may be inappropriate or harmful for some individuals. Instead, here's a list of neutral events that could impact the player's mental state, both positively and negatively:</i></p> <p><i>Abandoned Library - A Moment of Solitude - You stumbled upon a library that's been long abandoned. The quiet atmosphere allowed you to take a break from the constant noise of the outside world. (+5 EXP, -5 INF)</i></p> <p><i>dst...</i></p> <p>Catatan: Hasil yang diberi selanjutnya hanya kejadian acak yang diberi dengan <i>prompt</i> yang diminta.</p>

Hasil generasi dari Tabel 4.6 menunjukkan salah satu kegagalan yang dialami oleh ChatGPT dimana *prompt* tersebut tidak bisa diterima karena negativitas yang dideteksi oleh ChatGPT. Tetapi, hal ini masih bisa diselesaikan dengan melakukan generasi baru dan mampu memberikan hasil yang diinginkan. Berikut adalah beberapa kejadian acak yang sudah dibuat menggunakan ChatGPT:

1. *Derelict Gas Station - Unexpected Feast - You stumbled upon a gas station and found a feast of snacks. The feast before you bolsters your strength, rejuvenating your body and spirit. (+15 HP, +3 EXP and +2 INF)*
2. *Hallucinations - Mind Games - You starts seeing and hearing things that aren't there, causing them to question their own sanity. (-10 HP, -5 EXP)*
3. *Overgrown Park - Hidden Cache - You spot a cache hidden in the bushes. Do you investigate or ignore it and move on? (+5 EXP and +3 INF or +2 EXP and +1 INF)*
4. *Hallucinatory Maze - Lost in Illusions - You stumble upon a maze filled with hallucinations. Which path will you choose?*
Choice 1: Embrace the Mirage - You venture deeper into the hallucinatory maze, gaining enlightenment. (+5 EXP, +1 INF)
Choice 2: Break the Illusion - You break free from the maze's grip, returning to reality. (+10 HP, +1 EXP, +1 INF)

Kejadian acak di atas hanyalah beberapa contoh kejadian acak yang bisa terjadi saat bermain *game*. Kejadian acak yang dibuat memiliki total 93 kejadian acak dan memerlukan lebih dari 20 kali generasi di ChatGPT agar mendapatkan hasil yang tepat dan diperlukan, mengambil beberapa kejadian acak yang bisa dipakai dari tiap daftar kejadian acak yang dibuat oleh ChatGPT dan menggabungkan semua hasil generasi dan lalu diatur sesuai keperluan seperti perubahan status, deskripsi dari pilihan-pilihan yang diberi oleh kejadian acak, membuat kejadian acak yang sama dengan perubahan status yang berbeda dan deskripsi yang berbeda, dan lainnya.

Hasil generasi ChatGPT juga mampu memberikan hasil yang cukup membantu untuk membuat *game* seperti nasehat pembuatan *game* dan cara melakukan sesuatu di Unity. Hal ini membantu untuk membuat sesuatu lebih cepat dan mendapatkan hasil yang diinginkan. Tetapi bila masalah atau *prompt* memiliki jawaban yang rumit seperti *script* yang panjang dan masalah spesifik, ChatGPT bisa memberikan hasil yang tidak diinginkan seperti memberikan hasil yang tidak benar atau memberikan hasil yang berbeda dari keinginan. Tabel 4.7 menunjukkan contoh generasi ChatGPT untuk *prompt* yang memiliki *script*.

Tabel 4.7 Hasil generasi ChatGPT untuk kode *script*

Prompt	<p><i>I wanted to ask a question. Does the code below able to put the events triggered in a new list and also able to put it back to the other list once all of the events are triggered?</i></p> <pre>using System.Collections; using System.Collections.Generic; using UnityEngine; using UnityEngine.UI; using TMPro; dst...</pre> <p>(sekitar 200 baris kode total dan menggunakan Few-Shot Prompting)</p>
Hasil	<p><i>Yes, the code can put the events triggered in a new list and also put it back to the other list once all of the events are triggered. In the code, the lists <i>activeImmediateEvents</i> and <i>activeChoiceEvents</i> are used to hold the events that are currently available to be triggered, while the lists <i>completedImmediateEvents</i> and <i>completedChoiceEvents</i> are used to hold the events that have been triggered.</i></p> <p>(ChatGPT lalu menjelaskan kode <i>script</i> sesuai dari <i>prompt</i> yang diberi)</p>

Tabel 4.7 menunjukkan hasil generasi dengan bertanya apakah kode *script* yang diberi mampu membuat kejadian acak yang sudah terjadi dimasukkan ke sebuah daftar baru dan juga mampu memasukkannya kembali ke daftar lain setelah semua kejadian acak sudah terjadi. ChatGPT menjawab bahwa kode *script* yang diberi mampu melakukan apa yang ditanya. ChatGPT juga menjelaskan bagaimana kode tersebut bekerja dengan menjelaskan kode *script* yang spesifik bekerja untuk melakukan apa. Hasil ini menunjukkan bahwa ChatGPT mampu membaca kode panjang yang memiliki berbagai macam fungsi di dalam kode *script* tersebut. Tabel 4.8 menunjukkan hasil generasi untuk menanyakan cara melakukan sesuatu kepada ChatGPT dengan referensi kode *script* yang diberi.

Tabel 4.8 Generasi ChatGPT terhadap pertanyaan

Prompt	<p><i>I wanted to update the music volume without the need of apply button while still using SaveVolume when interacting the volume. With the code below, how do i do that?</i></p> <pre> using UnityEngine; using UnityEngine.UI; using TMPro; public class VolumeManager : MonoBehaviour { [SerializeField] private Slider musicSlider = null; [SerializeField] private TextMeshProUGUI musicText = null; private void Start() { dst... </pre>
	(35 baris kode total dan menggunakan Few-Shot Prompting)
Hasil	<p><i>You can use Unity's Event System to automatically call the SaveVolume() function whenever the slider's value changes. Here's how you can modify your VolumeManager script to achieve this:</i></p> <ol style="list-style-type: none"> <i>1. Select the musicSlider GameObject in the Hierarchy window.</i> <i>2. In the Inspector window, click the "Add Component" button and search for "Event Trigger". Add the component to the musicSlider GameObject.</i> <i>3. In the Event Trigger component, click the plus sign next to the "Trigger" list and select "Slider".</i> <i>4. Click the plus sign next to the "Events" list to add a new event.</i> <i>5. Set the "Event Type" to "Pointer Up".</i> <i>6. Drag the VolumeManager script from the Hierarchy window to the "Object" field in the Event Trigger component.</i> <i>7. In the "Function" dropdown, select "VolumeManager.SaveVolume".</i> <p><i>(ChatGPT lalu memberikan script untuk perubahan yang diperlukan sesuai prompt.)</i></p>

Hasil di Tabel 4.8 menunjukkan hasil generasi dari pertanyaan yang diberi untuk bertanya bagaimana merubah kekuatan suara dari musik serta menyimpan jumlah kekuatan dari musik tersebut dan memberikan kode *script* sebagai referensi. ChatGPT menjawab dengan memberikan solusi dengan daftar tahapan dan memberikan hasil kode *script*. Walaupun ChatGPT mampu menjawab pertanyaan dan masalah yang dimiliki, ChatGPT juga memberikan hasil yang tidak benar. Tabel 4.9 menunjukkan salah satu hasil generasi yang salah.

Tabel 4.9 Generasi ChatGPT yang salah

Prompt	<p><i>Focus on choiceEvent. I wanted to make 2 buttons that can appear to pick one of 2 choices and also replaces subtitle and description of that event when picking one of the choices (The Choice1 and Choice2 Text will be in the buttons, Choice1 Sub and Desc and Choice2 Sub and Desc will replace the first eventSub and eventDesc). Please update the scripts below.</i></p> <pre>using UnityEngine; using TMPro; using System.Collections.Generic; dst...</pre> <p>(sekitar 200 baris kode total dan menggunakan Chain of Thought Prompting)</p>
Hasil	<p><i>Here's the updated script with the choice1Button and choice2Button functionality:</i></p> <pre>using System.Collections; using System.Collections.Generic; using UnityEngine; using UnityEngine.UI; using TMPro;</pre> <pre>public class RandomEventSystem : MonoBehaviour { dst...</pre> <p>(Generasi dihentikan karena tidak sesuai dengan <i>prompt</i>.)</p>

ChatGPT diminta untuk fokus terhadap satu aspek dari kode yang diberi sebagai referensi. Tetapi, ChatGPT memberikan kode script penuh dari referensi yang diberi dan tidak menghasilkan hasil spesifik yang diminta. Terkadang, hasil

yang diberi oleh ChatGPT bisa salah dan bahkan menggunakan bahasa pemrograman yang berbeda. Hasil tersebut bisa dilihat di Tabel 4.10.

Tabel 4.10 Generasi ChatGPT salah bahasa pemrograman

Prompt	<p><i>How do i make the messages here to be Bold on certain texts?</i></p> <pre>private int currentStoryIndex = 0; private StoryData[] stories = { new StoryData("It was a very peaceful day. People doing their own business and living their life...", "Intro1"), dst... (12 baris kode total dan menggunakan One-Shot Prompting)</pre>
Hasil	<p><i>To make certain texts bold in your messages here, you can use HTML tags. Surround the text you want to be bold with the opening and closing "strong" tags. Here's an example:</i></p> <pre>private int currentStoryIndex = 0; private StoryData[] stories = { new StoryData("It was a very peaceful day. People doing their own business and living their life...", "Intro1"),</pre>

Di Tabel 4.9, ChatGPT diminta untuk membuat teks dari cerita menjadi lebih tebal dan memberikan referensi kode *script*, tetapi ChatGPT memberikan hasil menggunakan bahasa pemrograman lain yaitu Java. Hasil yang diberikan terkadang memberikan bahasa pemrograman yang dikira merupakan bahasa pemrograman lain seperti Arduino, Java dan bahkan Mathematica tetapi sebagian besar masih bisa dipakai karena hasil yang diberi mampu digunakan dengan lancar.

ChatGPT juga memiliki batas untuk memberi hasil. Bila melebihi batas tertentu, ChatGPT akan menghentikan generasi. Hal ini bisa dilanjutkan dengan menggunakan *prompt* “*continue*” atau menggunakan tombol “*Continue Generation*”. Hasil akan tetap memiliki konteks yang sama dari *prompt* sebelumnya. ChatGPT Plus juga digunakan untuk melakukan tes perbedaan dari ChatGPT yang gratis. Perbedaannya adalah ChatGPT Plus memberikan respon yang lebih cepat dan mampu memberikan batasan lebih banyak untuk generasi jawaban yang

panjang serta fitur tambahan lainnya. Tetapi, hasil yang didapat tetap sama walaupun memiliki kecepatan yang lebih cepat.

4.5 Hasil Game

Menggunakan aset yang sudah dibuat menggunakan Leonardo.ai dan Beatoven.ai, aset UI yang diambil dari *website* Kenney serta ide dan kode dari ChatGPT, semua hasil yang diperlukan digabungkan dan membuat sebuah *game* bernama “*Plight of the Infected*”. *Game* ini memiliki aspek *roguelike* yang mendasar dimana pemain akan bertemu dengan kejadian acak yang merubah statistik pemain secara langsung atau setelah memilih pilihan tertentu dan bila pemain kalah, pemain harus mengulang dari awal.

Game ini memiliki enam *scene* yaitu menu utama (*Main Menu*), *intro* untuk menjelaskan cerita sinopsis *game*, *character* untuk memilih karakter pemain, *main gameplay* untuk pemain bermain, dan *scene* untuk menang atau kalah. Semua *scene* sudah diberi musik dan diberikan gambar *background*. Pemain akan diletakkan di menu utama dan bisa mengecek kredit untuk penggunaan aset tertentu dan siapa yang membantu untuk membuat *game*. Gambar 4.7 menunjukkan menu utama dari *game*.

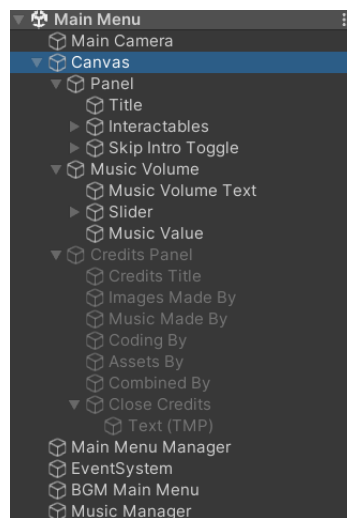


Gambar 4.7 *Scene* menu utama *game*

Pemain bisa mulai bermain dengan mengklik tombol “*Play*”, mengecek kredit dengan “*Credits*” atau keluar dari *game* dengan menekan “*Quit*”. Pemain juga bisa melewati *intro* *game* dengan mengklik “*Skip Intro*”, mengatur *slider*

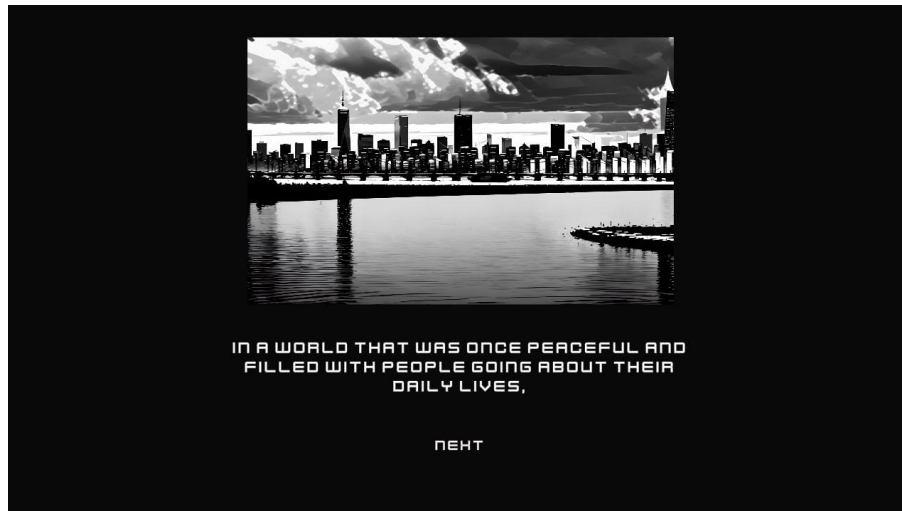
untuk *volume* suara musik dan dua tombol memilih bahasa Inggris atau bahasa Indonesia saat bermain. “Credits” akan menunjukkan panel yang berisi beberapa teks untuk memberikan kredit kepada siapa saja untuk penggunaan aset tertentu dan menjelaskan apa saja yang digunakan untuk membuat hal tertentu.

Untuk membuat menu utama bisa dimulai dengan membuat sebuah *canvas* dan panel. *Canvas* digunakan untuk menunjukkan bagian mana yang ditunjukkan dengan menaruh kamera di dalam *canvas* serta menaruh beberapa panel. Panel digunakan untuk menaruh berbagai macam hal seperti tombol, gambar, dan lain-lainnya. Gambar 4.8 menunjukkan hal yang ada di dalam *scene* menu utama.



Gambar 4.8 *Scene* menu utama

Dimulai dengan panel pertama yaitu untuk meletakkan hal tertentu yang bisa diinteraksi oleh pemain. Gambar untuk judul, membuat beberapa tombol serta meletakkan toggle untuk bisa melewati *intro game*. Volume musik dari *game* bisa diatur menggunakan *slider*. Panel untuk memberikan kredit juga dibuat, tetapi hanya muncul setelah menekan tombol tersebut. Hal yang perlu dimasukkan adalah tombol dengan sebuah *background* untuk mulai bermain, tombol mengecek kredit, tombol keluar dari *game*, sebuah *toggle* untuk melewati *intro game* dan sebuah *slider* untuk mengatur volume musik. Setelah menu utama sudah dibuat, *scene intro* bisa dibuat untuk menjelaskan cerita. Gambar 4.8 menunjukkan *intro* dari *game*.



Gambar 4.9 *Scene intro game*

Intro game di Gambar 4.9 merupakan salah satu contoh penjelasan cerita dari *game*. Intro memiliki enam gambar dan delapan kalimat untuk menjelaskan cerita sesuai dari gambar. Intro ini digunakan untuk membuat pemain mengerti apa yang terjadi di dalam *game*. Teks dan gambar perlu dimasukkan agar bisa ditunjukkan serta diperlukan tombol untuk menunjukkan cerita dan gambar selanjutnya (*source code* bisa dilihat di lampiran halaman 104-107). Setelah cerita selesai atau pemain menyalakan “*Skip Intro*”, maka pemain akan dipindah menuju *scene character* dimana pemain akan bermain seperti yang ada di Gambar 4.10.



Gambar 4.10 *Scene character*

Gambar 4.10 menunjukkan salah satu kombinasi karakter yang bisa didapat saat akan mulai bermain. Pemain akan diberi tiga dari 16 karakter yang sudah dibuat

dan memilih salah satu karakter. Tiap karakter memiliki berbagai macam batas status yang berbeda sebagai berikut.

1. HP atau kesehatan pemain
2. EXP atau progress pemain
3. INF atau infeksi pemain

Untuk referensi, karakter biasa yang seimbang atau tidak memiliki kelebihan memiliki 100 HP, batas EXP 50 dan batas INF 50. Target yang harus diraih pemain setelah memilih karakter adalah mencapai batas EXP dan tidak membuat HP pemain mencapai 0 atau memenuhi batas INF. Berikut adalah sebuah gambar untuk menunjukkan salah satu status karakter.



Gambar 4.11 Pilihan karakter *engineer*

Di Gambar 4.11, pemain memilih karakter engineer dengan menekan gambar dari karakter. Di kanan ditunjukkan latar belakang pemain dan batas status karakter yaitu batas HP 90, batas EXP 40 dan batas INF 30. Ini menunjukkan bahwa karakter ini lebih cepat untuk menang, tetapi memiliki kekurangan yaitu batas HP lebih sedikit dan batas INF lebih sedikit yang bisa membuat pemain kalah bila tidak berhati-hati. Setelah pemain memilih karakter, pemain bisa menekan tombol continue dan menuju ke *scene* baru yaitu *scene gameplay*.



Gambar 4.12 *Scene gameplay*

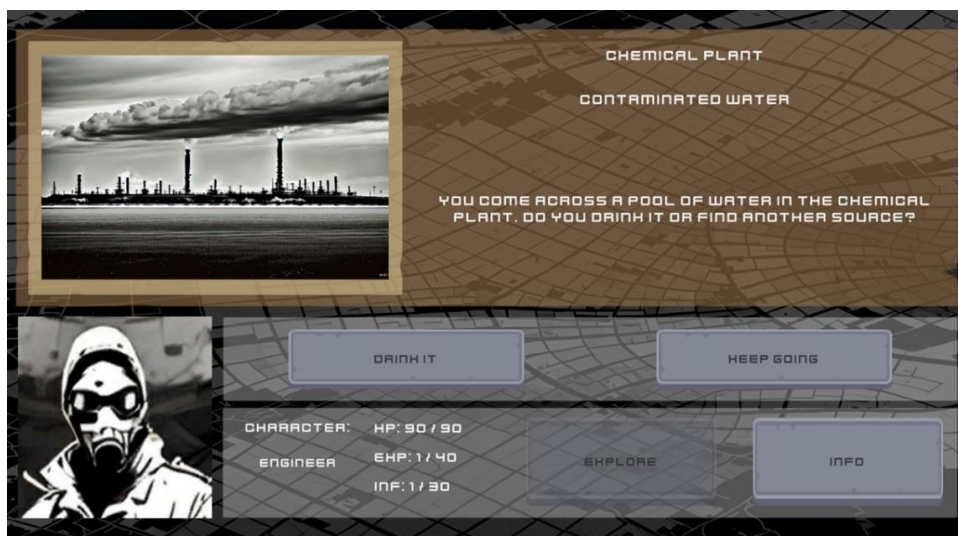
Gambar 4.12 menunjukkan layar bermain dan penjelasan secara singkat mengenai cara bermain serta menjelaskan target yang harus dicapai agar menang dan berusaha menghindari kondisi kalah. Proses pembuatan *scene* bisa dibuat dengan cara yang sama yaitu menaruh panel untuk fitur utama bermain, tombol yang diperlukan untuk bermain serta teks dan tempat gambar untuk menunjukkan kejadian acak, karakter serta gambarnya.

Target dari pemain adalah mencapai batas EXP dan berusaha untuk menahan HP pemain di atas 0 dan INF dibawah batas yang diberi. *Game* ini memiliki dua tipe kejadian acak yaitu kejadian acak secara langsung dimana pemain akan mendapatkan perubahan status secara langsung setelah kejadian acak ditemui dan kejadian acak pilihan dimana pemain harus memilih salah satu pilihan untuk melanjutkan bermain serta memberikan perubahan status setelah memilih pilihan tersebut. Kejadian acak yang akan dialami oleh pemain akan mengubah status dengan perubahan kecil maupun besar yang bisa mempengaruhi *game*. Berikut merupakan salah satu contoh kejadian acak langsung yang bisa dialami oleh pemain.



Gambar 4.13 Contoh kejadian acak langsung

Kejadian acak diatas merupakan salah satu kejadian acak langsung yang bisa dialami oleh pemain. Kejadian acak ini merubah status pemain secara langsung serta menjelaskan kenapa kejadian acak tersebut merubah status tersebut. Perubahan status juga dijelaskan mana saja yang sudah berubah dengan tanda kurung yang menunjukkan perubahan status apa saja yang diubah. Gambar 4.14 menunjukkan salah satu kejadian acak pilihan.



Gambar 4.14 Contoh kejadian acak pilihan

Kejadian acak pilihan akan menunjukkan kejadian acak yang dialami sama seperti kejadian acak langsung. Tetapi, kejadian ini tidak akan mengubah status pemain secara langsung dan pemain harus memilih salah satu dari pilihan untuk

melanjutkan kejadian acak dan mendapatkan perubahan status pemain. Gambar 4.15 menunjukkan salah satu hasil dari pilihan dari kejadian di Gambar 4.14.



Gambar 4.15 Salah satu hasil dari kejadian acak dari Gambar 4.14

Hasil kejadian acak untuk Gambar 4.15 didapat dari memilih salah satu pilihan di Gambar 4.14 yaitu memilih pilihan “*Drink it*”. Hasil dari pilihan dari tiap kejadian acak pilihan bervariasi mulai dari tidak banyak pengaruh ke status pemain sampai perubahan status pemain yang besar. Untuk membuat tiap tipe kejadian acak, perlu dibuat *script* yang berbeda. Hal ini untuk membedakan kejadian acak karena tipe kejadian acak yang bekerja secara berbeda. Untuk kejadian acak pilihan diperlukan cara kerja yang berbeda dimana pemain mampu memilih pilihan untuk merubah status pemain (*source code* bisa dilihat di lampiran halaman 108-109).

Dari tiap tipe kejadian acak, kejadian acak langsung memberikan perubahan status secara langsung sementara kejadian acak pilihan memerlukan dua dari tiap hal untuk memberikan perubahan status pemain untuk pilihan tertentu (contoh kejadian acak bisa dilihat di lampiran halaman 109-110). Pemberian nama dan nomor tertentu diberikan sesuai urutan yang diberi yaitu:

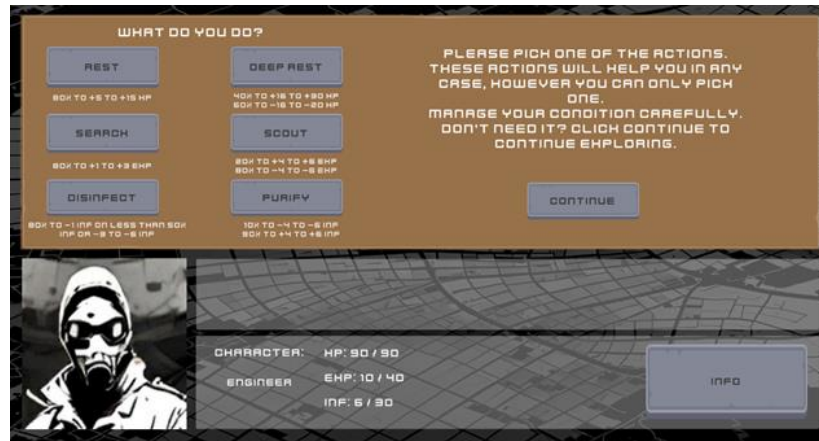
1. Nama kejadian acak
2. Judul sampingan kejadian acak
3. Deskripsi kejadian acak
4. 2 nama pilihan (khusus kejadian acak pilihan)
5. Perubahan HP

6. Perubahan EXP
7. Perubahan INF
8. Nama gambar
9. Tipe kejadian acak

Khusus kejadian acak pilihan, perubahan HP sampai INF di berikan dua kali untuk menunjukkan perubahan status berbasis dari pilihan. Terlihat bahwa ada `Random.Range` untuk perubahan status. Perubahan status ini bisa diacak untuk mendapatkan hasil yang tidak dikira untuk membuat *game* lebih menarik dan membuat *game* tidak monoton. Bila seandainya bahasa yang dipilih menggunakan bahasa Indonesia, maka perlu membuat translasi. Untuk membuat translasi, diperlukan Dictionary untuk menyimpan data yang ditandai dengan kunci tertentu agar data tersebut bisa dipanggil dengan kunci tersebut (*source code* bisa dilihat di lampiran halaman 120-121). Kode dibuat di dalam *script* baru dan tiap kejadian acak akan memiliki berbagai macam kunci untuk tiap string atau teks. Untuk menggunakan kunci tersebut, *script* ini harus dipanggil agar bisa dipakai (contoh bisa dilihat di halaman 122).

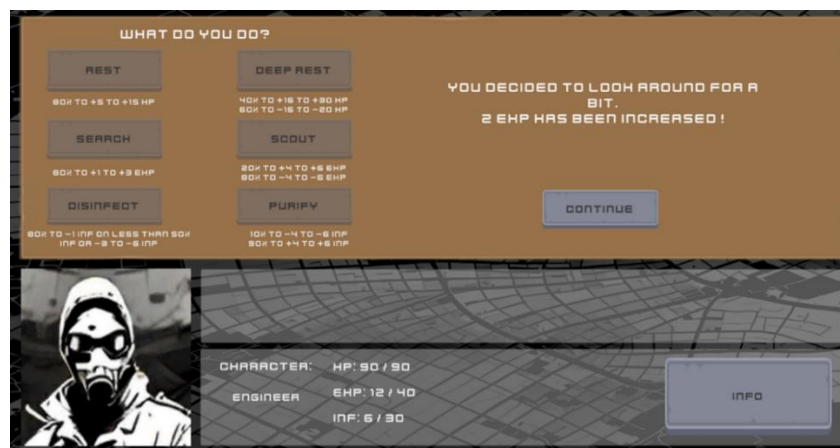
Lalu tipe kejadian acak dimasukkan ke sebuah *script* untuk digunakan untuk membuat sistem kejadian acak. Untuk itu, pemain perlu status untuk bisa mengerti status dari pemain (*source code* bisa dilihat di lampiran halaman 130-136). Untuk memberikan kejadian acak yang berbeda tiap kali, maka perlu diberikan kesempatan untuk menampilkan kejadian tertentu. Kejadian acak juga harus bisa ditampilkan lagi setelah beberapa kejadian acak yang berbeda sudah terjadi (*source code* bisa dilihat di lampiran halaman 116). Kode menunjukkan bila kejadian acak sudah ada di dalam sebuah daftar kejadian acak yang sudah terjadi ada 4, maka kejadian acak terlama dimasukkan kembali ke daftar untuk kejadian acak yang bisa terjadi.

Karena karakter bisa mati dan perlu bantuan lain selain mendapatkan kejadian acak, maka perlu diberi bantuan tambahan. Setelah 5 kejadian acak sudah terjadi, tombol “*Explore*” dirubah menjadi tombol “*Action*” untuk memberikan kesempatan untuk pemain untuk mendapatkan perubahan status positif. Gambar 4.16 menunjukkan menu *action* dari *game*.



Gambar 4.16 Menu *action*

Menu ini menunjukkan berbagai macam aksi yang bisa dilakukan oleh pemain serta instruksi untuk membantu pemain. Pemain bisa melewati aksi ini, tetapi juga bisa mencoba melakukan aksi tertentu. Tiap aksi memberikan perubahan status tertentu dan memiliki kesempatan sukses sebagai berikut. Tombol yang ada di sis kiri menunjukkan aksi yang memiliki kesempatan tinggi untuk memberikan perubahan status positif tinggi sementara sisi kanan menunjukkan kesempatan sukses yang lebih kecil dan memberikan perubahan status yang lebih baik lagi tetapi memberikan kekurangan bila gagal. Gambar 4.17 adalah salah satu contoh aksi yang dipilih.



Gambar 4.17 Hasil aksi *search*

Gambar 4.17 menunjukkan hasil bila menekan aksi *search*. Hasil yang diberi akan diacak dengan batas tertentu yang sudah dijelaskan di bawah tiap aksi. Setelah pemain memilih satu aksi dengan hasil sukses atau tidak, pemain harus

menekan tombol *continue* untuk lanjut bermain. Bila pemain mencapai kondisi menang yaitu mencapai batas EXP karakter, maka pemain akan dipindah ke *scene* menang yang ditunjukkan di Gambar 4.18.



Gambar 4.18 *Scene* menang

Gambar 4.18 ditunjukkan apabila pemain mencapai batas EXP karakter. *Scene* ini menjelaskan cerita tambahan apakah cara menyembuhkan infeksi bisa ditemukan. *Scene* ini juga memiliki tombol untuk bermain ulang *game* dengan menggunakan tombol “*Restart*” atau kembali ke menu utama dengan tombol “*Main Menu*”. Bila salah satu kondisi kalah tercapai, pemain akan dipindahkan ke *scene* kalah. *Scene* kalah memiliki kesamaan dari *scene* menang, tetapi memiliki kalimat dan *background* yang berbeda dari *scene* menang. Gambar 4.19 menunjukkan salah satu *scene* kalah yang bisa didapat oleh pemain.



Gambar 4.19 Salah satu *scene* kalah

Gambar 4.19 menunjukkan alasan mengapa pemain kalah dalam bermain *game*. Alasan mengapa pemain kalah di Gambar 4.18 adalah bahwa HP atau

kesehatan pemain mencapai 0. Bila pemain mencapai batas infeksi karakter, pemain juga akan dipindah ke *scene* kalah juga dengan teks berbeda. Untuk membuat *scene* menang dan kalah, diperlukan pengaturan kondisi pemain (*source code* bisa dilihat di lampiran halaman 134-135). *Game* yang dibuat memiliki berbagai macam perubahan dari rencana yang ada di Bab 3 yaitu:

1. Tidak menggunakan sistem peta untuk membuat level, sistem melawan musuh, sistem *status effects* dan sistem *inventory*,
2. *Game* hanya memberikan kejadian acak yang dialami dengan mengeklik tombol “*explore*” dan mengubah status pemain berbasis dari kejadian acak yang didapat,
3. *Game* akan berbasis situasi acak dan pemikiran strategis pemain untuk menahan status tertentu,
4. Status pemain yang dipakai adalah kesehatan, progres dan infeksi.

Kejadian acak diambil secara acak dan bisa ditemui lagi bila empat tipe kejadian acak yang sama sudah dimainkan. Jumlah total kejadian acak yang dibuat menggunakan ChatGPT adalah 93 kejadian acak (65 kejadian langsung dan 28 kejadian pilihan) dengan berbagai macam keuntungan dan kerugian yang diatur untuk membuat *game* lebih adil.

4.6 Hasil Testing

Testing akan dilakukan untuk mengetes *game* untuk melihat apakah ada keseimbangan yang diperlukan serta perubahan yang bisa dilakukan. Tes akan menggunakan form yang memiliki 10 pertanyaan dari SUS (*System Usability Scale*) untuk mengetes seberapa lancar bekerjanya *game* serta memberikan pertanyaan tambahan untuk pendapat pengetes. Setiap pertanyaan akan diberi skala dari satu yaitu tidak setuju sampai lima yaitu setuju.

Hasil *testing* didapat dari 11 sukarelawan lelaki dan 1 pengetes perempuan berumur antara 19-24 dari mahasiswa sampai pekerja di berbagai macam bidang. Versi *game* untuk mengetes yang dipakai memiliki total 65 kejadian acak (45 kejadian langsung dan 20 kejadian pilihan) yang bisa ditemui pemain, batas status pemain seperti kesehatan, progres dan infeksi 100 yang tidak memiliki sistem

karakter serta sistem aksi, dan kejadian acak mampu diulang lagi tanpa memedulikan mana yang sudah ditemui serta belum ada sistem aksi serta sistem karakter.

Pertanyaan yang diberi kepada pengetes adalah sebagai berikut:

1. Saya berencana untuk bermain *game* ini berulang kali
2. Saya merasa cara bermain *game* ini rumit untuk dimainkan
3. Saya merasa cara bermain *game* ini mudah untuk dimainkan
4. Saya membutuhkan bantuan dari orang lain atau pengembang *game* untuk bermain *game* ini
5. Saya merasa fitur-fitur *game* ini berjalan dengan semestinya
6. Saya merasa ada banyak hal yang tidak konsisten (tidak serasi) dengan *game* ini
7. Saya merasa orang lain akan memahami cara memainkan *game* ini dengan cepat
8. Saya merasa *game* ini membingungkan
9. Saya merasa tidak ada hambatan dalam bermain *game* ini
10. Saya perlu mempelajari banyak hal dulu sebelum terbiasa bermain *game* ini
11. Apakah anda memiliki pendapat untuk *game* ini?
12. Apakah Anda menemui masalah atau memerlukan penyeimbangan untuk *game*?

Setelah menjawab semua pertanyaan tersebut, data dari pengetes akan diambil dan diatur untuk sepuluh pertanyaan pertama untuk mengatur nilai SUS untuk mengecek apakah *game* bisa dimainkan atau perlu perbaikan. Dua pertanyaan lainnya digunakan untuk membuat *game* lebih baik dan mengatur masalah yang dimiliki oleh para pengetes seperti:

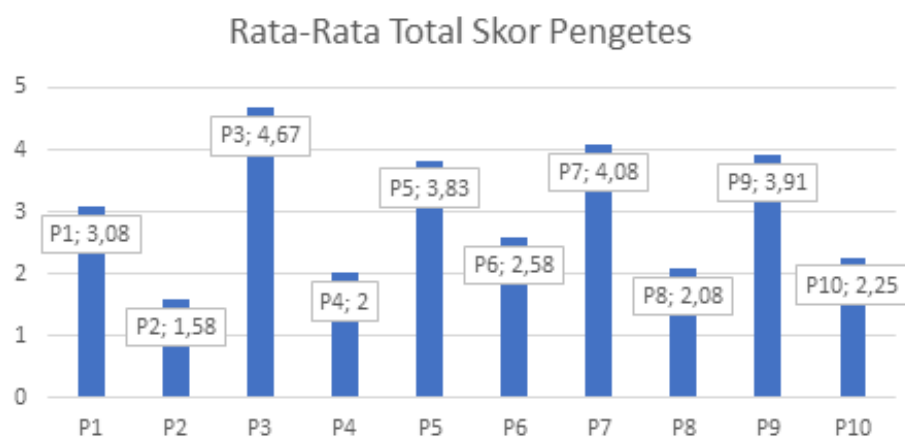
1. Memberikan masalah yang ditemui oleh pengetes seperti masalah teknis tertentu,
2. Perubahan dan tambahan fitur tertentu,
3. Menyeimbangkan perubahan status tertentu untuk kejadian acak spesifik,

Tabel 4.11 menunjukkan semua skor yang diberi oleh tiap pengetes untuk 10 pertanyaan pertama dengan skala 1 (tidak setuju) sampai 5 (sangat setuju):

Tabel 4.11 Data asli pengetes *game*

Tx/Px	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
T1	1	1	5	3	3	2	5	1	4	1
T2	4	1	5	2	3	2	4	2	5	2
T3	3	1	5	1	5	1	5	1	5	1
T4	5	1	5	1	4	2	5	2	5	1
T5	4	3	5	5	4	3	4	1	2	5
T6	4	2	4	2	4	3	4	3	4	3
T7	2	1	5	1	3	5	1	5	1	3
T8	4	1	5	1	5	4	5	1	2	1
T9	1	2	5	1	4	3	5	3	5	2
T10	3	1	5	2	4	2	5	1	5	2
T11	2	3	3	4	3	2	2	3	4	4
T12	4	2	4	1	4	2	4	2	5	2
Rata	3.08	1.58	4.67	2	3.83	2.58	4.08	2.08	3.91	2.25

Di Tabel 4.11, Tx memiliki arti *tester* atau pengetes ke-x dan Px memiliki arti pertanyaan ke-x dimana pertanyaan yang diambil adalah 10 pertanyaan pertama. Pengetes akan menilai 10 pertanyaan tersebut menurut pendapat tiap pengetes. Gambar 4.20 menunjukkan grafik dari Tabel 4.11.



Gambar 4.20 Grafik rata-rata total skor pengetes

Dari gambar 4.19, terlihat bahwa P1 (Pertanyaan 1) memiliki nilai rata-rata 3.08. Pertanyaan ini menanyakan apakah pengetes akan bermain lagi atau tidak dan hasil yang didapat menunjukkan bahwa rata-rata pemain hanya akan mengulang bermain secukupnya.

Pertanyaan ganjil memiliki aspek positif dari *game* sementara pertanyaan ganjil memiliki aspek negatif dari *game*. Pertanyaan ganjil memiliki skor rata-rata di atas tiga yang menunjukkan *game* memiliki aspek positif yang banyak dan paling besar di P3 yaitu 4.67 dimana sebagian besar pemain mengatakan bahwa *game* ini mudah dimainkan. Pertanyaan genap memiliki skor rata-rata di sekitar 1.58 sampai 2.58 yang menunjukkan setengah dari pemain memiliki masalah dari *game* dimana paling besar di P6 yaitu 2.58 dimana *game* tidak konsisten.

Setelah semua pengetes selesai mengetes *game* yang dibuat, nilai SUS dari tiap pengetes bisa dihitung. Berikut di Tabel 4.12 menunjukkan hasil SUS untuk tiap pengetes dari hasil Tabel 4.11.

Tabel 4.12 Data hasil SUS

Tx/Px	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	SUS
T1	0	4	4	2	2	3	4	4	3	4	75
T2	3	4	4	3	2	3	3	3	4	3	80
T3	2	4	4	4	4	4	4	4	4	4	95
T4	4	4	4	4	3	3	4	3	4	4	92.5
T5	3	2	4	0	3	2	3	4	1	0	55
T6	3	3	3	3	3	2	3	2	3	2	67.5
T7	1	4	4	4	2	0	0	0	0	2	42.5
T8	3	4	4	4	4	1	4	4	1	4	82.5
T9	0	3	4	4	3	2	4	2	4	3	72.5
T10	2	4	4	3	3	3	4	4	4	3	85
T11	1	2	2	1	2	3	1	2	3	1	45
T12	3	3	3	4	3	3	3	3	4	3	80
Total Skor SUS											872.5
Rata-Rata Skor SUS											72.70

Dari hasil yang didapat, skor yang didapat adalah 72.70. Menurut SUS, skor ini menunjukkan bahwa *game* ini masih bisa dimainkan walaupun memiliki sedikit masalah seperti cara bermain *game* dan aspek lainnya. Beberapa pendapat yang diberi dari pengetes *game* adalah:

1. Permainan merasa monoton,
2. Tidak mengerti cara bermain,
3. Beberapa kata tidak terlihat,
4. *Font* yang tidak jelas.

Setelah mendapatkan pendapat pengetes, ada beberapa tambahan yang diberikan untuk *game* yaitu:

1. Menambah 28 kejadian acak (20 kejadian acak langsung dan 8 kejadian acak pilihan, membuat total kejadian acak menjadi 93) dan mengatur setiap kejadian acak hanya bisa diulang setelah empat kejadian acak lainnya sudah terjadi,
2. Mengubah batas progres dengan memberikan fitur karakter yang memiliki berbagai macam batas status,
3. Menambah sistem aksi untuk tiap 5 kejadian acak
4. Mengatur warna *background* menjadi lebih gelap agar kalimat bisa dibaca,
5. Memberikan gambar untuk kejadian acak (86 gambar) dan karakter (16 karakter dengan total 48 gambar termasuk kondisi karakter)
6. Membuat versi Bahasa Indonesia serta versi Android, Mac dan WebGL.

4.7 Mengeluarkan *Game* di itch.io dan Pendapat Pemain

Setelah *game* sudah selesai dibuat, *game* akan dikeluarkan di itch.io. *Link website game* akan dibagikan ke berbagai komunitas untuk melihat pendapat orang lain mengenai *game* yang dibuat. Untuk harga, *game* dikeluarkan dengan gratis atau donasi. Untuk donasi, pemain bisa memberikan sumbangan dengan jumlah apapun. Alasan ini karena ini merupakan *game* pertama yang dibuat dan kualitas *game* masih sangat kecil dan tidak pantas untuk dijual dengan harga pasti.

Game diumumkan sudah keluar dan bisa dimainkan dengan menggunakan berbagai macam sosial media yaitu Instagram, beberapa server di Discord dan

Twitter dengan nama alias yaitu Craxzone. Saat *game* dikeluarkan, *game* akan dimasukkan ke dalam kategori “*Most recent*” dimana *game* baru ditampilkan. Ini agar bisa mendapatkan berbagai macam pemain dari sosial media maupun itch.io.

Hasil jumlah orang yang melihat *website game* adalah 89 kali dan jumlah *download* yang didapat adalah 12 orang. Tidak ada *rating* dan komentar yang diberi setelah satu minggu *game* dikeluarkan. Oleh karena itu, lima pemain yang sudah bermain serta enam pengetes *game* ditanya untuk pendapat apakah *game* yang dibuat dengan *tools* AI bisa diterima oleh komunitas *gaming*. Semua kecuali satu pengetes setuju bahwa *game* yang dibuat dengan *tools* AI mampu diterima karena pemain tidak peduli *game* tersebut dibuat oleh AI atau tidak. Berikut adalah beberapa pendapat spesifik yang diberi oleh pemain dan pengetes:

1. Salah satu pemain menyatakan bila hasil gambar yang diberikan orisinal maka pemain mampu menerima dan tidak mampu diterima bila sebaliknya, tetapi untuk kode program mampu diterima oleh pemain.
2. Satu pemain dan satu pengetes menyatakan tergantung kepada pasar dan tema dari *game* tersebut. Satu pengetes dan satu pemain lain menyatakan bahwa tergantung dari orang tersebut mengenai penggunaan AI, tetapi tetap akan memainkan *game* tersebut.
3. Satu pengetes tidak setuju dikarenakan di era digital pembuatan sebuah media menggunakan *tools* AI tidak memiliki banyak kreativitas.

Bab V

Penutup

5.1 Kesimpulan

Dari penelitian ini, pengembangan *game* bisa dilakukan dengan *tools* AI tetapi perlu mencoba menggunakan berbagai macam cara *prompting* dan belajar dengan *tools* AI tersebut untuk mendapatkan hasil yang cukup dengan waktu membuat sebuah aset bisa bervariasi tergantung dari generasi yang didapat. Pengembangan *game* dengan ChatGPT diatur untuk membuat ide *game*, berbagai macam variasi kejadian acak yang diinginkan, atau bahkan membantu membuat sebuah kode dengan menggunakan *Few Shot Prompting* dan *Chain of Thought Prompting*. Leonardo.ai memerlukan *prompt* yang bisa digunakan untuk membuat gambar tetapi diperlukan latihan dengan mencoba berulang kali, menggunakan alat tertentu serta menggunakan model yang tepat. Pembuatan musik dengan Beatoven.ai sangat mudah dilakukan hanya dengan memilih tema, suasana, durasi dan tempo yang diperlukan dan bisa memberikan hasil musik yang berbeda bila hasil tidak sesuai keinginan.

Untuk mendapatkan penjelasan yang tepat kepada *tools* AI terutama ChatGPT dan Leonardo, adalah dengan memberikan *prompt* yang pasti akan memberikan hasil yang mirip, memberikan referensi untuk memberikan hasil tepat, mencoba generasi berkali-kali dan mengatur hasil untuk mendapatkan hasil yang tepat. ChatGPT bisa memberikan hasil tepat bila dijelaskan menggunakan contoh seperti menggunakan *Few Shot Prompting* dan Leonardo.ai bisa menggunakan *prompt* yang konsisten untuk mendapatkan gaya seni yang tepat. Penggunaan *tools* AI juga harus dipastikan bahwa hasil yang dibuat mampu digunakan untuk alasan komersial. Hal ini perlu dimengerti dikarenakan *tools* AI yang diberi oleh sebuah perusahaan yang membuat *tools* AI belum tentu memperbolehkan penggunaan komersial kecuali sudah membeli subskripsi atau cara lain.

Rata-rata 12 pengetes menilai *game* dengan nilai SUS 72.70 dimana *game* memiliki sedikit masalah, tetapi masih bisa dimainkan dan mampu menghibur beberapa pengetes. Sebagian besar pengetes menyatakan pendapat dengan skala 1

yaitu tidak setuju sampai 5 yaitu setuju dan menyatakan bahwa *game* mudah untuk dimainkan dengan hasil rata-rata 4.67 dari dan akan dimengerti oleh pemain lain yang akan bermain *game* dengan rata-rata skala 4.08. Setengah dari pengetes menyatakan bahwa *game* tidak konsisten dengan hasil rata-rata 2.58 dan perlu mempelajari beberapa banyak hal dulu sebelum terbiasa bermain *game* dengan rata-rata skala 2.25 dengan sistem skala yang sama.

Game yang dibuat menggunakan AI bisa dikomersialkan, tetapi tergantung terhadap reaksi pemain serta kualitas *game* yang dibuat. Pemain tidak terlalu memedulikan apakah *game* yang dibuat menggunakan *tools* AI atau tidak asalkan ada *game* yang bisa dimainkan, tetapi bisa dianggap tidak kreatif karena dibuat oleh AI dan juga tidak menerima karena dibuat oleh sesuatu yang bukan manusia.

5.2 Saran

Saran untuk pembuatan *game* adalah memastikan rencana yang dibuat memiliki cara alternatif atau ide alternatif apabila menggunakan *tools* AI. Alasannya adalah agar bisa melakukan perubahan terhadap rencana pembuatan *game* yang tidak bisa dibuat, bisa diganti menggunakan rencana lain. Diingat bahwa pembuatan *game* tidak pasti mengikuti rencana yang dibuat dikarenakan berbagai macam aspek seperti kurang mengerti membuat sesuatu, aspek kehidupan, dan lain-lain.

Saran kedua yaitu pengetesan dan publikasi mengenai *game* tersebut. Pengetesan dari *game* harus direncanakan secara detil. Menggunakan SUS merupakan salah satu cara paling dasar yang mampu digunakan untuk mengetes *game*. Dengan hal ini, penelitian bisa berjalan dengan lancar dan mampu mendapatkan data sebanyak mungkin dari tiap individual. Bila seandainya *game* dikeluarkan ke toko tertentu, maka harus membagikan *game* tersebut secara publik dengan berbagai macam cara seperti sosial media, periklanan dan berbagai macam cara lainnya. Tanpa hal tersebut bila melakukan penelitian mengenai reaksi public tentang *game* tersebut, maka hasil yang didapat tidak banyak.

Saran untuk peneliti lainnya yang ingin menggunakan *tools* AI untuk membuat sebuah *game* adalah memastikan bisa mengerti untuk menggunakan alat

tertentu. Hasil yang dibuat oleh *tools* AI belum tentu memberikan hasil yang diinginkan walaupun menggunakan *prompt* atau pilihan yang sama. Oleh karena itu, *tools* AI tidak bisa digunakan untuk semuanya. *Tools* AI lebih baik digunakan sebagai alat bantuan untuk mengecek bila kode salah atau tidak, memberikan contoh kode, serta memberikan referensi bila diperlukan. Harus diingat bahwa penggunaan semua *tools* AI belum tentu gratis dan belum tentu bisa digunakan untuk alasan komersial. Pastikan membaca semua syarat dan ketentuan dari tiap *tools* AI agar bisa membuat sebuah *game* lancar dan tidak melanggar apapun.

Daftar Pustaka

- Allcorrect Group. (2022, February 4). *The Indonesian Gaming Market*. Retrieved from Allcorrect Game Content Studio: <https://allcorrectgames.com/insights/indonesia/>
- Amalia, E. I. (2022, August 20). *Midjourney, AI yang Bisa Buat Karya Seni Melalui Discord*. Retrieved from Hybrid.co.id: <https://hybrid.co.id/post/apa-itu-midjourney>
- Andrew. (2023, March 3). *Stable Diffusion prompt: a definitive guide*. Retrieved from Stable Diffusion Art: <https://stable-diffusion-art.com/prompt-guide/>
- Apperley, T. H. (2008). Genre and game studies: Toward a critical approach to video game genres. *SAGE Publications*, 9-18.
- Argentics. (2018, October 17). *PROS AND CONS OF CHOOSING 2D VS 3D ANIMATION*. Retrieved from Argentics: <https://www.argentics.io/pros-and-cons-of-choosing-2d-vs-3d-animation>
- Beatoven Private Limited. (2023). *How it works*. Retrieved from beatoven.ai: <https://www.beatoven.ai/>
- Brian. (2022, August 17). *OneShot: World Machine Edition release date set for September on Switch*. Retrieved from Nintendo Everything: <https://nintendoeverything.com/oneshot-world-machine-edition-release-date-set-for-september-on-switch/>
- Clement, J. (2022, October 18). *COVID-19 impact on the gaming industry worldwide - Statistics & Facts*. Retrieved from statista: <https://www.statista.com/topics/8016/covid-19-impact-on-the-gaming-industry-worldwide/#editorsPicks>
- Copeland, B. (2023, February 22). *artificail intelligence*. Retrieved from Britannica: <https://www.britannica.com/technology/artificial-intelligence>
- darkcritz. (2021, May 14). *Godot, the open source game engine is updated to version 3.3*. Retrieved from Linuxaddicted: <https://www.linuxaddictos.com/en/godot-open-source-game-engine-is-updated-to-version-3-3.html>
- Epic Games, Inc. (n.d.). *Level Editor*. Retrieved from Unreal Engine: <https://docs.unrealengine.com/5.1/en-US/level-editor-in-unreal-engine/>
- Ffoulkes, G. (2013, December 24). *One Dreamer*. Retrieved from One Dreamer: <https://www.one-dreamer.com/thegame/>
- Futurlab. (2022, July 14). *PowerWash Simulator*. Retrieved from Steam: https://store.steampowered.com/app/1290000/PowerWash_Simulator/

- Gameace Creative Studio. (2021, December 10). *Five Key Game Development Stages: A Look Behind The Scenes*. Retrieved from Gameace Creative Studio: <https://gameace.com/blog/game-development-stages/#:~:text=A%20game%20development%20process%20typically,%2C%20and%20post%2Dproduction%20maintenance>.
- GitLab. (n.d.). *System Usability Scale*. Retrieved from GitLab: <https://about.gitlab.com/handbook/product/ux/performance-indicators/system-usability-scale/>
- Halliday, E. (2020, October 10). *Gaming History: Where does the term 'Roguelike' come from?* Retrieved from Fansided: <https://apptrigger.com/2020/10/10/gaming-history-roguelike/>
- Hersko, T. (2021, July 2). *'Doki Doki Literature Club Plus!'* Review: *The Best Way to Play the Indie Horror Gem*. Retrieved from Indie Wire: <https://www.indiewire.com/2021/07/doki-doki-literature-club-plus-review-indie-horror-game-1234648204/>
- Hopoo Games. (2013, November 9). *Risk of Rain*. Retrieved from Steam: https://store.steampowered.com/app/248820/Risk_of_Rain/
- IGN. (2016, October 21). *Sid Meier's Civilization VI*. Retrieved from IGN: <https://civilization.com/news/entries/civilization-vi-10-tips-to-start-playing/>
- Irawan, P. L., Tandiono, F., & Setiawan, H. (2018). Rancang Bangun Game Edukasi untuk Pengenalan Dasar Logika Algoritma Berbasis Mobile. *Network Engineering Research Operation*, 164-165.
- Johnson, L. (2022, November 12). *The Elder Scrolls V: Skyrim Special Edition PC Review*. Retrieved from IGN: <https://www.ign.com/articles/2016/11/01/skyrim-special-edition-pc-review>
- Kleinings, H. (2022, November 16). *What is AIaaS? Your Guide to AI as a Service*. Retrieved from Levity: <https://levity.ai/blog/aiaas-guide>
- Lim, S., & Schmälzle, R. (2022). Artificial Intelligence for Health Message Generation: Theory, Method, and an Empirical Study Using Prompt Engineering. *Arxiv*, 1-10.
- Martin, J. (2020, October 20). *What is a Game Engine?* Retrieved from University of Silicon Valley: <https://usv.edu/blog/what-is-a-game-engine/>
- Mastery Coding. (2021, July 6). *Why Unity is the Best Game Engine for Beginners*. Retrieved from Mastery Coding: <https://www.masterycoding.com/blog/unity-best-beginner-engine>
- Motion Twin. (2018, August 7). *Dead Cells*. Retrieved from Steam: https://store.steampowered.com/app/588650/Dead_Cells/

- NNW, A. T. (2018, December 28). *Tips Membuat Game Design Document (GDD) Bagi Pemula*. Retrieved from Gamelab Indonesia: <https://www.gamelab.id/news/54-tips-membuat-game-design-document-gdd-bagi-pemula>
- Parks, A. (2022, December 8). *The Complete History of Tennis for Two*. Retrieved from History Computer: <https://history-computer.com/tennis-for-two-complete-history/>
- Rizkinaswara, L. (2021, May 9). *Industri Game Tumbuh Pesat, Perlu Edukasi Terhadap Pengguna*. Retrieved from Kominfo: <https://aptika.kominfo.go.id/2021/05/industri-game-tumbuh-pesat-perlu-edukasi-terhadap-pengguna/>
- Roose, K. (2022, September 2). *An A.I.-Generated Picture Won an Art Prize. Artists Aren't Happy*. Retrieved from The New York Times: <https://www.nytimes.com/2022/09/02/technology/ai-artificial-intelligence-artists.html>
- RPGMakerWeb. (n.d.). *The RPG Maker Series' Newest Offering: RPG Maker MZ*. Retrieved from RPG Maker: <https://www.rpgmakerweb.com/products/rpg-maker-mz>
- Sauro, J. (2011, February 3). *Measuring Usability with the System Usability Scale (SUS)*. Retrieved from MeasuringU: <https://measuringu.com/sus/>
- Spiegel, S. (2015, November 18). *Undertale: Not Your Average RPG*. Retrieved from The Roaring Times: <https://www.pitmanroaringtimes.com/2015/11/undertale-not-your-average-rpg/>
- Stegner, B. (2021, October 24). *What Are Roguelike and Roguelite Video Games?* Retrieved from Make Use Of: <https://www.makeuseof.com/what-are-roguelike-and-roguelite-video-games/>
- Stuart, K. (2019, October 15). *Fortnite Chapter 2 is live with new map, weapons and more*. Retrieved from The Guardian: <https://www.theguardian.com/games/2019/oct/15/fortnite-chapter-2-live-news-map-weapons-black-hole-characters>
- Team Fortress 2 Official Wiki. (n.d.). *Backpack*. Retrieved from Team Fortress 2 Official Wiki: <https://wiki.teamfortress.com/wiki/Backpack>
- Team Fortress 2 Official Wiki. (n.d.). *Heads-up display*. Retrieved from Team Fortress 2 Official Wiki: https://wiki.teamfortress.com/wiki/Heads-up_display
- Team Fortress 2 Official Wiki. (n.d.). *Main menu*. Retrieved from Team Fortress 2 Official Wiki: https://wiki.teamfortress.com/wiki/Main_menu
- techopedia. (2017, January 26). *Massively Multiplayer Online Role-Playing Game (MMORPG)*. Retrieved from techopedia:

<https://www.techopedia.com/definition/1919/massively-multiplayer-online-role-playing-game-mmorpg>

- Tiernan, J. (2022, November 18). *Is AI Really an Art and Design Killer?* Retrieved from Vectornator: <https://www.vectornator.io/blog/will-ai-replace-artists/>
- Tretkoff, E. (2008, October). *October 1958: Physicist Invents First Video Game*. Retrieved from APS: <https://www.aps.org/publications/apsnews/200810/physicshistory.cfm>
- Vollmer, A. (2014). *How to Make Great Game Tutorials*. Retrieved from YouTube: <https://www.youtube.com/watch?v=Uf7xLHUpKHE>
- Wang, Y., Shen, S., & Lim, B. Y. (2023). RePrompt: Automatic Prompt Editing to Refine AI-Generative Art. *Arxiv*, 1-11.
- White, S. (2018, March 20). *Valley Forged: How One Man Made the Indie Video Game Sensation Stardew Valley*. Retrieved from GQ: <https://www.gq.com/story/stardew-valley-eric-barone-profile>
- Widodo, R. B., Swastika, W., & Yuswanto. (2020). Pembelajaran Game Edukasi Dengan Object-Oriented Programming pada Jenjang Sekolah Menengah Pertama. *Difusi Iptek*, 51-52.
- Wolff, T. (2023, February 8). *From “Zero-Shot” To “Chain Of Thought”: Prompt Engineering & Choosing The Right Prompt Types (ChatGPT & GPT-3)*. Retrieved from Medium: <https://medium.com/mlearning-ai/from-zero-shot-to-chain-of-thought-prompt-engineering-choosing-the-right-prompt-types-88800f242137>
- Wong, P. (2016, April 3). *Counter Strike: Global Offensive is great fun and actually realistic [Review]*. Retrieved from Young Post: <https://www.scmp.com/yp/discover/entertainment/tech-gaming/article/3066274/counter-strike-global-offensive-great-fun-and>
- YoYo Games Ltd. (2022, May 4). *GameMaker*. Retrieved from Steam: <https://store.steampowered.com/app/1670460/GameMaker/>

Lampiran

Lampiran gambar yang digunakan untuk *game*

Menu
Utama



Cerita 1



Cerita 2



Lampiran gambar yang digunakan untuk *game* lanjutan (1)

Cerita 3



Cerita 4



Cerita 5

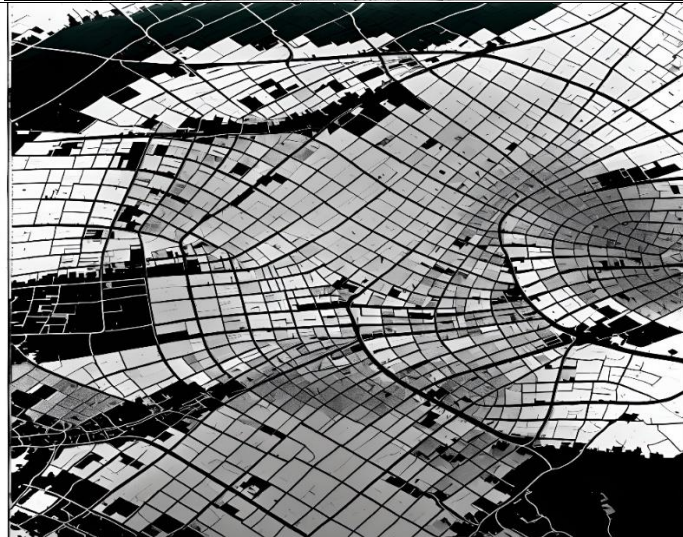


Lampiran gambar yang digunakan untuk *game* lanjutan (2)

Cerita 6



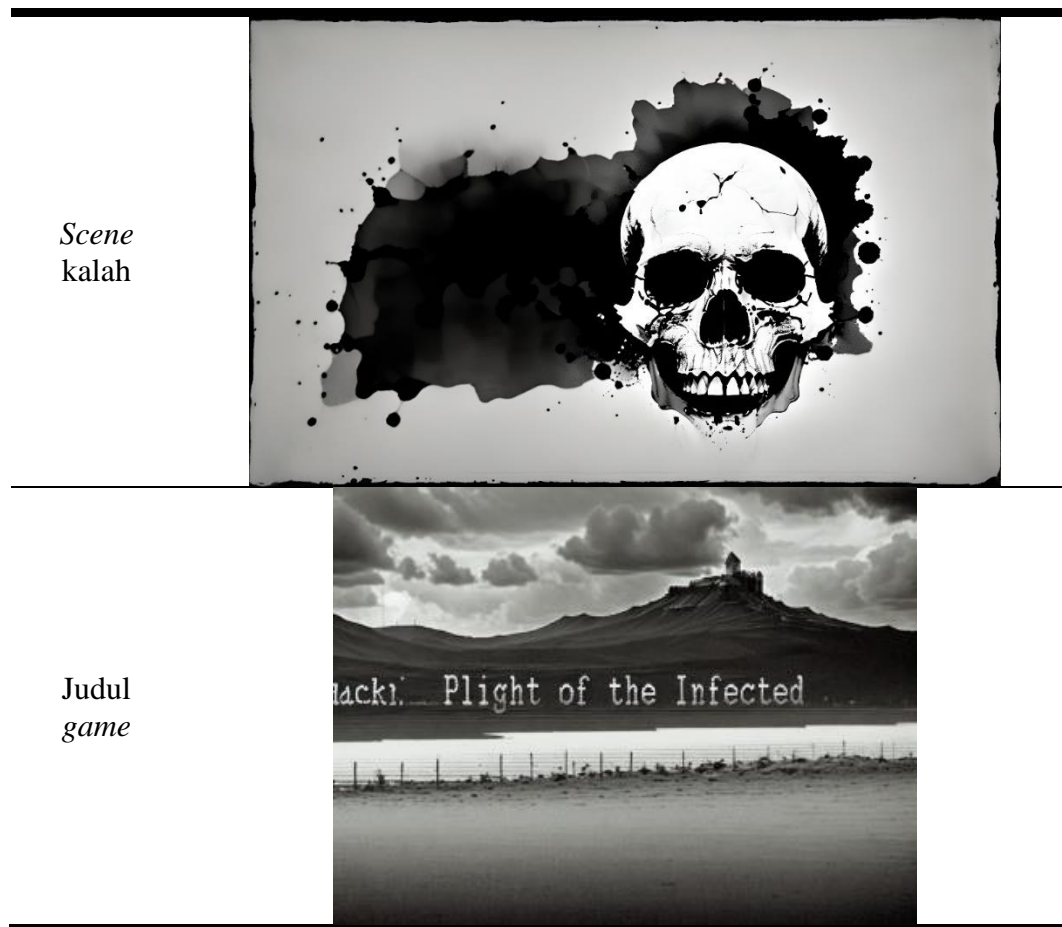
Menu
Gameplay



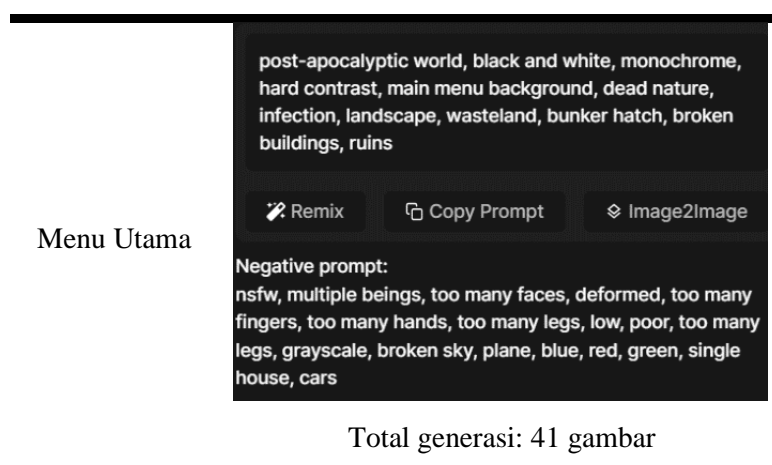
Scene
menang



Lampiran gambar yang digunakan untuk *game* lanjutan (3)



Lampiran *prompt* yang digunakan untuk membuat gambar



Lampiran *prompt* yang digunakan untuk membuat gambar lanjutan (1)

Cerita 1

city, modern, black and white, monochrome, hard contrast, main menu background, life, landscape, sin city style

Remix Copy Prompt Image2Image

Negative prompt:
nsfw, multiple beings, too many faces, deformed, too many fingers, too many hands, too many legs, low, poor, too many legs, grayscale, broken sky, plane, blue, red, green, single house, reflecting light

Catatan: Menggunakan hasil di Cerita 2 untuk referensi

Total generasi: 46 gambar

Cerita 2

city, modern, black and white, monochrome, hard contrast, main menu background, life, landscape, sin city style, infection, broken buildings, burning buildings, very dark skies, explosion on a building

Remix Copy Prompt Image2Image

Negative prompt:
nsfw, multiple beings, too many faces, deformed, too many fingers, too many hands, too many legs, low, poor, too many legs, grayscale, broken sky, plane, blue, red, green, single house, reflecting light

Total generasi: 40 gambar

Cerita 3

infected human, eye popping, black and white, monochrome, hard contrast, sin city style, close-up to head, scratches, panic emotion

Remix Copy Prompt Image2Image

Negative prompt:
nsfw, multiple beings, too many faces, deformed, too many fingers, too many hands, too many legs, low, poor, too many legs, grayscale, broken sky, plane, blue, red, green, single house, reflecting light, more than one person

Total generasi: 15 gambar

Cerita 4










only zombies, black and white, monochrome, hard contrast, sin city style, wasteland background, abandoned, post-apocalyptic, derelict, zombie walk everywhere, hand reaching toward screen, walking weakly everywhere

Remix Copy Prompt Image2Image

Negative prompt:
nsfw, multiple beings, too many faces, deformed, too many fingers, too many hands, too many legs, low, poor, too many legs, walking away from screen, too many humans, too many arms, symbols, misplaced lines, claws, sword, mace, stick, holding a weapon, random lines on hands, red aura, grayscale, mask, hooded clothing, sophisticated clothing, hoodie, jacket, various colors, color noises, clipped image, red eyes, something on hand, too many black, red, wings, horns, fantasy, glowing eyes, horns, castle

Total generasi: 16 gambar

Lampiran *prompt* yang digunakan untuk membuat gambar lanjutan (2)

Cerita 5	<div><p>four people standing, black and white, monochrome, hard contrast, sin city style, wasteland background, human outpost, abandoned, post-apocalyptic</p><div> Remix</div><div> Copy Prompt</div><div> Image2Image</div><p>Negative prompt: nsfw, multiple beings, too many faces, deformed, too many fingers, too many hands, too many legs, low, poor, too many legs, walking away from screen, too many humans, too many arms, symbols ,misplaced lines, claws, sword, mace, stick, holding a weapon, random lines on hands, red aura, grayscale, mask, hooded clothing, sophisticated clothing, hoodie, jacket, various colors, color noises, clipped image, red eyes, something on hand, too many black, red, wings, horns, fantasy, glowing eyes, horns</p></div>
Total generasi: 10 gambar	
Cerita 6	<div><p>people standing like a hero, black and white, monochrome, hard contrast, sin city style, on a cliff, wasteland background, broken buildings, abandoned, post-apocalyptic, wearing backpacks</p><div> Remix</div><div> Copy Prompt</div><div> Image2Image</div><p>Negative prompt: nsfw, multiple beings, too many faces, deformed, too many fingers, too many hands, too many legs, low, poor, too many legs, walking away from screen, too many humans, too many arms, symbols ,misplaced lines, claws, sword, mace, stick, holding a weapon, random lines on hands, red aura, grayscale, mask, hooded clothing, sophisticated clothing, hoodie, jacket, various colors, color noises, clipped image, red eyes, something on hand, too many black, red, wings, horns, fantasy, glowing eyes, horns</p></div>
Total generasi: 1 gambar	
Menu Gameplay	<div><p>clear map, black and white, monochrome, hard contrast, sin city style, post-apocalyptic, landmarks, cities, dark</p><div> Remix</div><div> Copy Prompt</div><div> Image2Image</div><p>Negative prompt: nsfw, multiple beings, too many faces, deformed, too many fingers, too many hands, too many legs, low, poor, too many legs, walking away from screen, too many humans, too many arms, symbols ,misplaced lines, claws, sword, mace, stick, holding a weapon, random lines on hands, red aura, grayscale, mask, hooded clothing, sophisticated clothing, hoodie, jacket, various colors, color noises, clipped image, red eyes, something on hand, too many black, red, wings, horns, fantasy, glowing eyes, horns, castle, houses, text</p></div>
Total generasi: 20 gambar	

Lampiran *prompt* yang digunakan untuk membuat gambar lanjutan (3)

Scene menang

solid black background, single man walking in a journey, black and white, monochrome, hard contrast, sin city style, post-apocalypse, wasteland, man on left center, backpack

Remix

Copy Prompt

Image2Image

Negative prompt:

nsfw, multiple beings, too many faces, deformed, too many fingers, too many hands, too many legs, low, poor, too many legs, walking away from screen, too many humans, too many arms, symbols, misplaced lines, claws, sword, mace, stick, holding a weapon, random lines on hands, red aura, grayscale, mask, hooded clothing, sophisticated clothing, hoodie, jacket, various colors, color noises, clipped image, red eyes, something on hand, too many black, red, wings, horns, fantasy, glowing eyes, horns, castle, houses, text, multiple skulls, multiple man

Total generasi: 78 gambar

Scene kalah

a single white skull, black and white, monochrome, hard contrast, sin city style, skull left center

Remix

Copy Prompt

Image2Image

Negative prompt:

nsfw, multiple beings, too many faces, deformed, too many fingers, too many hands, too many legs, low, poor, too many legs, walking away from screen, too many humans, too many arms, symbols, misplaced lines, claws, sword, mace, stick, holding a weapon, random lines on hands, red aura, grayscale, mask, hooded clothing, sophisticated clothing, hoodie, jacket, various colors, color noises, clipped image, red eyes, something on hand, too many black, red, wings, horns, fantasy, glowing eyes, horns, castle, houses, text, multiple skulls

Total generasi: 76 gambar

black and white, monochrome, hard contrast, sin city style, clean background

Catatan: Menggunakan gambar lain untuk referensi menggambar:

Judul *game*

Plight of the Infected

Total generasi: 8 gambar

Lampiran *prompt* dan hasil yang digunakan untuk membuat cerita sinopsis

Prompt

Give me a background story about a disease called Infinity Infection that makes people aggressive and makes them unintelligible and transform into a monster known as the "Lost Ones". The infection caused only an eight of the planet survived by hiding (does not have any resistance to the disease). One group managed to make a suppressant to slow down the infection. The group realized they need more resources and manpower to make the suppressants widely available. So, they recruited a group of early infected survivors (they were given a device that will let the group knows when they die by other causes or transformed into the Lost Ones) to find resources and recruit other survivors in these wicked lands (these group of early infected survivors will always die in the journey).

(Menggunakan Few-Shot Prompting)

Hasil

In the year 2035, a mysterious disease known as the Infinity Infection swept across the world, leaving in its wake destruction and chaos. The disease was highly contagious and caused those infected to become aggressive and unintelligible. As the disease progressed, those infected would transform into monstrous creatures known as the "Lost Ones," with a insatiable hunger for flesh.

As the infection spread, it quickly became apparent that there was no known cure. Governments and organizations around the world struggled to contain the disease, but it was too late. Within a matter of months, only an eighth of the planet's population remained, with the rest having succumbed to the disease or been transformed into the Lost Ones.

Amid all the destruction, a small group of survivors managed to develop a suppressant that could slow down the infection. Realizing the potential of this discovery, the group set out to create a widespread supply of the suppressant, which would require additional resources and manpower.

Lampiran *prompt* dan hasil yang digunakan untuk membuat cerita sinopsis lanjutan

In a bold move, the group decided to recruit a team of early infected survivors, providing them with a device that would notify the group when the individuals either died from other causes or transformed into the Lost Ones. The group sent this team on a dangerous mission to scavenge for resources and recruit other survivors in the wastelands left behind by the Infinity Infection.

Hasil

The journey was perilous, with the team facing constant danger from both the disease and the Lost Ones. Despite the risks, the team continued to push forward, driven by the hope that their efforts would ultimately lead to a cure for the Infinity Infection and the restoration of society.

Over time, the team grew in number, as they encountered other survivors and brought them into the fold. Together, they battled against the odds, facing unimaginable challenges and overcoming impossible obstacles. And while they may have lost some of their number along the way, their determination and resilience never wavered, as they continued to fight for a future in a world forever changed by the Infinity Infection.

Lampiran *prompt* dan hasil yang digunakan untuk membuat nama grup

Prompt

Give me three lists of names of a group. The first one of the list is a group that will make suppressants that will suppress the Infinity Infection disease. The second one is the list of a group that will help the group that will help making the suppressants by finding resources and recruiting other survivors in these wicked lands (the group contains early infected survivors that is given a gadget to signal the suppressants makers when either they died from the Infinity Infection, transforming them into the Lost Ones or died by another cause). The third one is combination of both two groups, representing the final hope of the world.

(Menggunakan Few-Shot Prompting)

Suppressant Makers:

Hasil

- 1. New Dawn Pharmaceuticals*
 - 2. Vital Inhibitors Group*
-

Lampiran *prompt* dan hasil yang digunakan untuk membuat nama grup lanjutan

3. *The Cure Seekers*
4. *Project Phoenix*
5. *The Infinity Defense Alliance*
6. *The Healing Circle*
7. *Pandemic Response Team*
8. *The Resilience Initiative*
9. *The Infection Suppression Task Force*
10. *The Resistance Remedy Collective*

Resource Gatherers and Recruiters:

1. *The Outpost Raiders*
2. *Scavenger Syndicate*
3. *The Reclamation Society*
4. *Wasteland Avengers*
5. *Frontier Force*

Hasil

6. *The Survivalists' Guild*
7. *Resilient Reconnaissance Squad*
8. *The Hope Seekers*
9. *Horizon Hounds*
10. *The Recovery Brigade*

Combined Group:

1. *The Phoenix Circle*
 2. *The Immune Vanguard*
 3. *The Restoration Alliance*
 4. *The Resurgence Coalition*
 5. *The Infinity Crusaders*
 6. *The Hope Renewal Project*
 7. *The Reclamation Initiative*
 8. *The Recovery Corps*
 9. *The Salvation Syndicate*
 10. *The Survivors' Union*
-

Lampiran *prompt* yang digunakan untuk membuat kejadian acak

Prompt

Give me a list of 10 random events where the player get stat changes after picking one of 2 choices when encountering it. The events will be about sudden psychological changes with good and bad results or both on the post-apocalyptic theme. The stats are:

1. HP (Health). Max is 100. Can decrease or increase. Reach 0 and player dies.

2. EXP (Exploration). Max is 100. Reach 100 to win. Most events will give at least +1 EXP

3. INF (Infection). Max is 100. Reach 100, the player dies. Some events will give at least +1 INF

Example of events:

Event Name - Event Subtitle - Event Description

Choice Name: New Subtitle - New Description (+/-x HP, +/-x EXP, +x INF)

Choice Name: New Subtitle - New Description (+/-x HP, +/-x EXP, +x INF)

It doesnt have to contain every stats, but all stats can be changed

(Menggunakan Few-Shot Prompting)

Lampiran hasil yang didapat untuk membuat kejadian acak

Hasil

Certainly! Here are 10 random events centered around sudden psychological changes with varying outcomes in a post-apocalyptic setting:

1. Hallucinatory Mirage - Mind's Deception - You encounter a captivating mirage. Will you embrace its allure or resist its grasp?

Choice 1: Surrender to Illusion - Lose yourself in the hallucination, gaining a momentary respite from reality. (-10 HP, +5 EXP)

Choice 2: Break the Illusion - Fight against the mirage, grounding yourself in reality. (+5 HP, -5 EXP)

2. Whispering Shadows - Eerie Whispers - Mysterious whispers surround you, beckoning you to heed their call. Will you succumb to curiosity?

Choice 1: Embrace the Shadows - Follow the whispers, unraveling their cryptic secrets. (+5 EXP, +1 INF)

Lampiran hasil yang didapat untuk membuat kejadian acak lanjutan (1)

Choice 2: Resist the Temptation - Ignore the whispers and maintain your focus. (-5 HP, -1 INF)

3. Paranoia's Grip - Distrustful Whispers - Paranoia consumes your thoughts, breeding suspicion among your comrades. Will you act on your fears?

Choice 1: Trust No One - Heighten your guard, questioning the intentions of others. (+1 INF)

Choice 2: Overcome Paranoia - Resist the urge to succumb to fear, fostering trust and unity. (+5 EXP)

4. Delusions of Grandeur - Illusory Dominion - Grandiose delusions cloud your perception, granting you a false sense of power. Will you embrace the delusions?

Choice 1: Embrace the Illusion - Revel in your imagined supremacy, taking bold actions. (+10 HP, -5 EXP)

Choice 2: Reject the Illusion - Dispel the delusions, grounding yourself in reality. (-5 HP, +5 EXP)

Hasil

5. Memory's Echo - Haunting Memories - Vivid memories from the past flood your mind, triggering intense emotions. Will you confront them?

Choice 1: Relive the Memories - Dive into the recollections, seeking closure and understanding. (-5 HP, +5 EXP)

Choice 2: Suppress the Memories - Push the memories aside, focusing on the present. (+5 HP, -5 EXP)

6. Despair's Embrace - Abyss of Sorrow - Overwhelming despair engulfs you, threatening to shatter your spirit. Will you succumb to hopelessness?

Choice 1: Embrace the Despair - Allow yourself to wallow in despair, accepting its grip. (+5 HP, -5 EXP)

Choice 2: Defy the Despair - Rise above the darkness, finding strength in adversity. (-5 HP, +5 EXP)

7. Manic Frenzy - Unbridled Energy - A surge of manic energy courses through your veins, fueling impulsive actions. Will you surrender to the frenzy?

Choice 1: Embrace the Frenzy - Channel the energy into a reckless burst of action. (-5 HP, +5 EXP)

Lampiran hasil yang didapat untuk membuat kejadian acak lanjutan (2)

Choice 2: Calm the Frenzy - Find ways to ground yourself, reining in the manic impulses. (+5 HP, -5 EXP)

8. Apathy's Grip - Numb Indifference - A wave of apathy washes over you, numbing your emotions. Will you succumb to the indifference?

Hasil *Choice 1: Embrace the Apathy - Allow yourself to disconnect, shielding yourself from pain. (+5 EXP, +1 INF)*

Choice 2: Reject the Apathy - Fight against the numbness, seeking to reconnect with your emotions. (-5 HP, -1 INF)

9. Obsess

Kode untuk *game*:

MainMenu.cs:

```
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UI;

public class MainMenu : MonoBehaviour
{
    public GameObject creditsPanel;
    private bool skipIntro = false;
    public GameObject skipIntroToggle;

    void Start()
    {
        // Load the state of the skip intro toggle from player prefs
        skipIntro = PlayerPrefs.GetInt("SkipIntro", 0) == 1;
        skipIntroToggle.GetComponent<Toggle>().isOn = skipIntro;
    }

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

```

    }

    public void PlayGame()
    {
        if (skipIntro || skipIntroToggle.GetComponent<Toggle>().isOn)
        {
            SceneManager.LoadScene("Main Gameplay");
        }
        else
        {
            skipIntroToggle.SetActive(false);
// Disable the skip intro toggle when the intro is played
            SceneManager.LoadScene("Intro");
        }
    }

    public void ShowCredits()
    {
        creditsPanel.SetActive(true);
    }

    public void HideCredits()
    {
        creditsPanel.SetActive(false);
    }

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

    public void OnSkipIntroToggle()
    {
        skipIntro = skipIntroToggle.GetComponent<Toggle>().isOn;
    }

```

```

        PlayerPrefs.SetInt("SkipIntro", skipIntro ? 1 : 0);
    }
}

```

VolumeManager.cs:

```

using UnityEngine;
using UnityEngine.UI;
using TMPro;

public class VolumeManager : MonoBehaviour
{
    [SerializeField] private Slider musicSlider = null;
    [SerializeField] private TextMeshProUGUI musicText = null;

    private void Start()
    {
        LoadValues();
        DontDestroyOnLoad(gameObject); // Prevents the object from
        being destroyed on scene load
    }

    public void MusicSlider(float volume)
    {
        musicText.text = volume.ToString("0.0");
        AudioListener.volume = volume;
        SaveVolume();
    }

    public void SaveVolume()
    {
        float volumeValue = musicSlider.value;
        PlayerPrefs.SetFloat("MusicValue", volumeValue);
        LoadValues();
    }
}

```

```

        public void LoadValues()
        {
float volumeValue = PlayerPrefs.GetFloat("MusicValue", 0.5f);
            musicSlider.value = volumeValue;
            AudioListener.volume = volumeValue;
        }
    }
}

```

IntroManager.cs:

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using TMPro;
using UnityEngine.SceneManagement;

public class IntroManager : MonoBehaviour
{
    public Image[] storyImages;
    public TextMeshProUGUI storyText;
    public Button nextButton;
    [SerializeField] private Image eventImage;

    private int currentStoryIndex = 0;
    private StoryData[] stories = {
        new
StoryData(LocalizationManager.instance.GetTranslatedText("intr
o1"), "Intro1"),
        new
StoryData(LocalizationManager.instance.GetTranslatedText("intr
o2"), "Intro2"),
        new
StoryData(LocalizationManager.instance.GetTranslatedText("intr
o3"), "Intro3"),
    }
}

```

```

        new
StoryData(LocalizationManager.instance.GetTranslatedText("intro4"), "Intro4"),

        new
StoryData(LocalizationManager.instance.GetTranslatedText("intro5"), "Intro5"),

        new
StoryData(LocalizationManager.instance.GetTranslatedText("intro6"), "Intro6"),

        new
StoryData(LocalizationManager.instance.GetTranslatedText("intro7"), "Intro7"),

        new
StoryData(LocalizationManager.instance.GetTranslatedText("intro8"), "Intro8")};

private void Start()
{
    ShowStory(currentStoryIndex);
}

private void Update()
{
    if (Input.GetMouseButtonDown(0) &&
nextButton.gameObject.activeSelf)
    {
        currentStoryIndex++;
        if (currentStoryIndex < stories.Length)
        {
            ShowStory(currentStoryIndex);
        }
        else
        {
            nextButton.gameObject.SetActive(false);
            // Unload the intro scene and load the next
scene here
            SceneManager.LoadScene("Main Gameplay");
        }
    }
}

```

```

    }

    private void ShowStory(int index)
    {
        StoryData story = stories[index];
        ClearImages();
        ShowImage(story.imageName);
        ClearText();
        StartCoroutine(ShowText(story.text));

        // Disable the Next button until the text is fully
displayed
        nextButton.interactable = false;
        nextButton.gameObject.SetActive(false);
    }

    private void ShowImage(string imageName)
    {
        for (int i = 0; i < storyImages.Length; i++)
        {
            Image image = storyImages[i];
            if (image.name == imageName)
            {
                // Activate the current image
                image.gameObject.SetActive(true);
            }
            else
            {
                // Deactivate the previous images
                image.gameObject.SetActive(false);
            }
        }
    }
}

```

```

private void ClearImages()
{
    for (int i = 0; i < storyImages.Length; i++)
    {
        storyImages[i].gameObject.SetActive(false);
    }
}

private IEnumerator ShowText(string text)
{
    for (int i = 0; i <= text.Length; i++)
    {
        storyText.SetText(text.Substring(0, i));
        yield return new WaitForSeconds(0.01f);
    }
    nextButton.gameObject.SetActive(true);
}

private void ClearText()
{
    storyText.SetText("");
}
}

public class StoryData
{
    public string text;
    public string imageName;

    public StoryData(string text, string imageName)
    {
        this.text = text;
        this.imageName = imageName;
    }
}

```

```
}
```

PlayerStats.cs:

```
using UnityEngine;
using UnityEngine.UI;
using TMPro;
using UnityEngine.SceneManagement;

public class PlayerStats : MonoBehaviour
{
    [SerializeField] public int health = 0;
    [SerializeField] public int exploration = 0;
    [SerializeField] public int infection = 0;
    [SerializeField] public int maxHealth = 100;
    [SerializeField] public int maxExploration = 50;
    [SerializeField] public int maxInfection = 50;
    public Character selectedCharacter;
    [SerializeField] private TextMeshProUGUI healthText;
    [SerializeField] private TextMeshProUGUI explorationText;
    [SerializeField] private TextMeshProUGUI infectionText;
    [SerializeField] private Image characterImage;
    [SerializeField] private TextMeshProUGUI
characterNameText;

    private void Start()
    {
        maxHealth = PlayerPrefs.GetInt("MaxHealth",
maxHealth);

        maxExploration = PlayerPrefs.GetInt("MaxExploration",
maxExploration);

        maxInfection = PlayerPrefs.GetInt("MaxInfection",
maxInfection);

        // Retrieve the selected character's image name from
PlayerPrefs
```



```

        string characterImageName =
PlayerPrefs.GetString("SelectedCharacterImage");

        string characterName =
PlayerPrefs.GetString("SelectedCharacterName");


        // Load the character's sprite from the Resources
folder using the image name

        Sprite characterSprite =
GetCharacterSprite(characterImageName);


        if (characterImage != null)
        {
            // Set the character's sprite to the
characterImage component

            characterImage.sprite = characterSprite;
        }


        if (characterNameText != null)
        {
            characterNameText.text = characterName;
        }


        health = maxHealth;
        exploration = 0;
        infection = 0;


        UpdateHealthText();
        UpdateExplorationText();
        UpdateInfectionText();
    }


    public void ModifyHealth(int amount)
    {
        health += amount;

        health = Mathf.Clamp(health, 0, maxHealth); // Use
maxHealth here instead of selectedCharacter.maxHealth

```

```

        UpdateHealthText();

        CheckForLoss();
    }

    public void ModifyExploration(int amount)
    {
        exploration += amount;

        exploration = Mathf.Clamp(exploration, 0,
maxExploration); // Use maxExploration here instead of
selectedCharacter.maxExploration

        UpdateExplorationText();

        CheckForWin();
    }

    public void ModifyInfection(int amount)
    {
        infection += amount;

        infection = Mathf.Clamp(infection, 0, maxInfection);
// Use maxInfection here instead of
selectedCharacter.maxInfection

        UpdateInfectionText();

        CheckForLoss();
    }

    public int GetExplorationValue()
    {
        return exploration;
    }

    public int GetInfectionValue()
    {
        return infection;
    }

    public int GetHealthValue()
    {
        return health;
    }

```

```

    }

    private void UpdateHealthText()
    {
        if (healthText != null)
        {
            healthText.text = "HP: " + health.ToString() + " / " + maxHealth;
        }
    }

    private void UpdateExplorationText()
    {
        if (explorationText != null)
        {
            explorationText.text = "EXP: " + exploration.ToString() + " / " + maxExploration;
        }
    }

    private void UpdateInfectionText()
    {
        if (infectionText != null)
        {
            infectionText.text = "INF: " + infection.ToString() + " / " + maxInfection;
        }
    }

    private void CheckForWin()
    {
        if (exploration >= maxExploration)
        {
            string gameOverText = LocalizationManager.instance.GetTranslatedText("wintext");

```

```

        PlayerPrefs.SetString("GameOverText",
gameOverText);

        SceneManager.LoadScene("Win Screen");
    }
}

private void CheckForLoss()
{
    if (health <= 0 || infection >= maxInfection)
    {
        string gameOverText = "";

        if (health <= 0)
        {
            gameOverText =
LocalizationManager.instance.GetTranslatedText("hplosetext");
        }
        else if (infection >= maxInfection)
        {
            gameOverText =
LocalizationManager.instance.GetTranslatedText("inflosetext");
        }

        // Pass the game over text to the next scene
        PlayerPrefs.SetString("GameOverText",
gameOverText);

        SceneManager.LoadScene("Game Over Scene");
    }
}

private Sprite GetCharacterSprite(string
characterImageName)
{

```

```

        string imagePath = "CharPFP/" + characterImageName;
        return Resources.Load<Sprite>(imagePath);
    }
}

```

MainButtonController.cs:

```

using UnityEngine;
using UnityEngine.UI;

public class MainButtonController : MonoBehaviour
{
    public GameObject mainPanel;
    public GameObject statusPanel;

    public Button exploreButton;
    public Button statusButton;

    public RandomEventSystem randomEventSystem; // Reference
    to the RandomEventSystem component

    void Awake()
    {
        // Show the main panel and hide the others
        mainPanel.SetActive(true);
        statusPanel.SetActive(false);

        // Add listeners to the buttons
        exploreButton.onClick.AddListener(ShowMainPanel);
        statusButton.onClick.AddListener(ToggleStatusPanel);

        // Set the explore button interactable to true
        initially
        exploreButton.interactable = true;
    }
}

```

```

void ShowMainPanel()
{
    Debug.Log("Showing main panel");
    mainPanel.SetActive(true);
    statusPanel.SetActive(false);
}

public void HandleButtonClick()
{
    Debug.Log("Event Generated");

    randomEventSystem.TriggerEventType(randomEventSystem.immediate
Events, randomEventSystem.choiceEvents,
randomEventSystem.playerStats);
}

void ShowStatusPanel()
{
    Debug.Log("Showing status panel");
    mainPanel.SetActive(false);
    statusPanel.SetActive(true);
    exploreButton.interactable = false;
}

void ToggleStatusPanel()
{
    if (statusPanel.activeSelf)
    {
        ShowMainPanel();
        exploreButton.interactable = true;
    }
    else
    {
        ShowStatusPanel();
    }
}

```

```

    }
}

```

ImmediateEvent.cs:

```

public ImmediateEvent(string nameKey, string subtitleKey,
string descriptionKey, int healthMod, int explorationMod, int
infectionMod, string image, string type)
{
    eventName =
LocalizationManager.instance.GetTranslatedText(nameKey);

    eventSubtitle =
LocalizationManager.instance.GetTranslatedText(subtitleKey);

    eventDescription =
LocalizationManager.instance.GetTranslatedText(descriptionKey)
;

    healthModifier = healthMod;

    explorationModifier = explorationMod;

    infectionModifier = infectionMod;

    imageName = image;

    eventType = type;
}

public void TriggerEvent(PlayerStats playerStats)
{
    // Modify player stats based on event modifiers
    playerStats.ModifyHealth(healthModifier);
    playerStats.ModifyExploration(explorationModifier);
    playerStats.ModifyInfection(infectionModifier);
}

public static List<ImmediateEvent> immediateEvents = new
List<ImmediateEvent>()
{
    new ImmediateEvent("ievent1name", "ievent1asub",
"ievent1adesc",
        5, 1, 0, "GasStationDeserted", "Positive"),

```

```
        new ImmediateEvent("ievent1name", "ievent1bsub",  
"ievent1bdesc",  
        1, 1, 1, "BadSnacks", "Positive"),
```

ChoiceEvent.cs:

```
using System.Collections.Generic;  
using UnityEngine;  
using UnityEngine.UI;  
using TMPro;  
  
[System.Serializable]  
public class ChoiceEvent  
{  
    public string eventName;  
    public string eventSubtitle;  
    public string eventDescription;  
    public string imageName;  
  
    public string choice1Text;  
    public string choice2Text;  
  
    public int choice1HealthModifier;  
    public int choice1ExplorationModifier;  
    public int choice1InfectionModifier;  
  
    public int choice2HealthModifier;  
    public int choice2ExplorationModifier;  
    public int choice2InfectionModifier;  
  
    public string newSubtitle1;  
    public string newSubtitle2;  
    public string newDescription1;  
    public string newDescription2;  
  
    public Button choice1Button;
```



```

public Button choice2Button;

public string eventType;

public float typingSpeed = 0.01f;

    public ChoiceEvent(string nameKey, string subtitleKey,
string descriptionKey, string choice1TextKey, string
choice2TextKey,

        int choice1HealthModifier, int choice1ExplorationModifier,
int choice1InfectionModifier, int choice2HealthModifier,

        int choice2ExplorationModifier, int
choice2InfectionModifier, string newSubtitle1Key, string
newDescription1Key, string newSubtitle2Key,

        string newDescription2Key, string image, string eventType,
string eventType2)
    {

        eventName =
LocalizationManager.instance.GetTranslatedText(nameKey);

        eventSubtitle =
LocalizationManager.instance.GetTranslatedText(subtitleKey);

        eventDescription =
LocalizationManager.instance.GetTranslatedText(descriptionKey)
;

        choice1Text =
LocalizationManager.instance.GetTranslatedText(choice1TextKey)
;

        choice2Text =
LocalizationManager.instance.GetTranslatedText(choice2TextKey)
;

        this.choice1HealthModifier = choice1HealthModifier;

        this.choice1ExplorationModifier =
choice1ExplorationModifier;

        this.choice1InfectionModifier =
choice1InfectionModifier;

        this.choice2HealthModifier = choice2HealthModifier;

        this.choice2ExplorationModifier =
choice2ExplorationModifier;

```

```

        this.choice2InfectionModifier =
choice2InfectionModifier;

        this.newSubtitle1 =
LocalizationManager.instance.GetTranslatedText (newSubtitle1Key
);

        this.newDescription1 =
LocalizationManager.instance.GetTranslatedText (newDescription1
Key);

        this.newSubtitle2 =
LocalizationManager.instance.GetTranslatedText (newSubtitle2Key
);

        this.newDescription2 =
LocalizationManager.instance.GetTranslatedText (newDescription2
Key);

        imageName = image;

        this.eventType = eventType;

        this.eventType2 = eventType2;

    }

    public void TriggerEvent (PlayerStats playerStats,
TextMeshProUGUI eventSubtitleText, TextMeshProUGUI
eventDescriptionText,

        Button choice1Button, Button choice2Button)

    {

        // Set the text for the choice buttons

choice1Button.GetComponentInChildren<TextMeshProUGUI>().text =
choice1Text;

choice2Button.GetComponentInChildren<TextMeshProUGUI>().text =
choice2Text;


        // Handle clicking on the first choice button
        choice1Button.onClick.AddListener (() =>
        {

            ModifyStats (playerStats, choice1HealthModifier,
choice1ExplorationModifier, choice1InfectionModifier);

            eventSubtitleText.text = newSubtitle1;

            eventDescriptionText.text = newDescription1;

```

```

        choice1Button.interactable = false;
        choice2Button.interactable = false;

choice1Button.GetComponentInChildren<TextMeshProUGUI>().text =
"";

choice2Button.GetComponentInChildren<TextMeshProUGUI>().text =
"";

        choice1Button.gameObject.SetActive(false);
        choice2Button.gameObject.SetActive(false);
    });

    // Handle clicking on the second choice button
    choice2Button.onClick.AddListener(() =>
    {
        ModifyStats(playerStats, choice2HealthModifier,
choice2ExplorationModifier, choice2InfectionModifier);
        eventSubtitleText.text = newSubtitle2;
        eventDescriptionText.text = newDescription2;
        choice1Button.interactable = false;
        choice2Button.interactable = false;

choice1Button.GetComponentInChildren<TextMeshProUGUI>().text =
"";

choice2Button.GetComponentInChildren<TextMeshProUGUI>().text =
"";

        choice1Button.gameObject.SetActive(false);
        choice2Button.gameObject.SetActive(false);
    });
}

    // Helper function to modify player stats
    private void ModifyStats(PlayerStats playerStats, int
healthModifier, int expModifier, int infModifier)
    {
        playerStats.ModifyHealth(healthModifier);
        playerStats.ModifyExploration(expModifier);
    }
}

```

```

        playerStats.ModifyInfection(infModifier);
    }

    public static List<ChoiceEvent> choiceEvents = new
    List<ChoiceEvent>()
    {
        new ChoiceEvent("cevent1name",
            "cevent1sub","cevent1desc", "cevent1choice1",
            "cevent1choice2",

            Random.Range(-6, -11), 5, 1,

            0, 1, 3,

            "cevent1choice1sub",
            "cevent1choice1desc","cevent1choice2sub",
            "cevent1choice2desc",

            "CrumblingBridge", "Negative","Negative"),
        new ChoiceEvent("cevent2name",
            "cevent2sub","cevent2desc", "cevent2choice1",
            "cevent2choice2",

            0, 2, 1,

            0, 2, 1,

            "cevent2choice1sub",
            "cevent2choice1desc","cevent2choice2sub",
            "cevent2choice2desc",

            "CarBroken","Positive", "Positive"),
    }

```

RandomEventSystem.cs:

```

public class RandomEventSystem : MonoBehaviour
{
    [SerializeField] public PlayerStats playerStats;
    [SerializeField] private TextMeshProUGUI eventNameText;
    [SerializeField] private TextMeshProUGUI
    eventSubtitleText;
    [SerializeField] private TextMeshProUGUI
    eventDescriptionText;
    [SerializeField] private Image eventImage;
    [SerializeField] private Button choice1Button;
    [SerializeField] private Button choice2Button;
    [SerializeField] private Button exploreButton;
    public List<ImmediateEvent> immediateEvents;
}

```

```

        public List<ChoiceEvent> choiceEvents;

        [SerializeField] public List<ImmediateEvent>
activeImmediateEvents = new List<ImmediateEvent>();

        [SerializeField] public List<ChoiceEvent>
activeChoiceEvents = new List<ChoiceEvent>();

        [SerializeField] public List<ImmediateEvent>
completedImmediateEvents = new List<ImmediateEvent>();

        [SerializeField] public List<ChoiceEvent>
completedChoiceEvents = new List<ChoiceEvent>();

        //public float typingSpeed = 0.001f;

        //private Coroutine typeCoroutine;

        private bool choiceMade = false;

        private AudioSource positiveAudioSource;

        private AudioSource negativeAudioSource;

        private ChoiceEvent currentChoiceEvent;

        public ActionMenu actionMenu;

        private int eventsTriggered = 0;

        public Button actionMenuButton;

        private bool isChoiceEvent;

    public void Start()
    {

        choice1Button.interactable = false;

        choice2Button.interactable = false;

        choice1Button.gameObject.SetActive(false);

        choice2Button.gameObject.SetActive(false);


        positiveAudioSource =
gameObject.AddComponent<AudioSource>();

        negativeAudioSource =
gameObject.AddComponent<AudioSource>();


        // Load the sound effects from Resources/SFX folder

        AudioClip positiveClip =
Resources.Load<AudioClip>("SFX/Positive");

        AudioClip negativeClip =
Resources.Load<AudioClip>("SFX/Negative");

```

```

        // Assign the loaded clips to audio sources
        positiveAudioSource.clip = positiveClip;
        negativeAudioSource.clip = negativeClip;
    }

    private void Awake()
    {
        immediateEvents = ImmediateEvent.GetImmediateEvents();
        activeImmediateEvents = new
List<ImmediateEvent>(immediateEvents);

        choiceEvents = ChoiceEvent.GetChoiceEvents();
        activeChoiceEvents = new
List<ChoiceEvent>(choiceEvents);

        // Add listeners to choice buttons
        choice1Button.onClick.AddListener(() =>
        {
            choiceMade = true;
        });

        choice2Button.onClick.AddListener(() =>
        {
            choiceMade = true;
        });
    }

    public void TriggerImmediateEvents(List<ImmediateEvent>
immediateEvents, PlayerStats playerStats)
    {
        if (completedImmediateEvents.Count == 4)
        {
            ImmediateEvent firstCompletedEvent =
completedImmediateEvents[0];
            completedImmediateEvents.RemoveAt(0);

```

```

        activeImmediateEvents.Add(firstCompletedEvent);

        Debug.Log("4 immediate events are triggered,
putting oldest event back");
    }

    isChoiceEvent = false;

    int randomIndex = Random.Range(0,
activeImmediateEvents.Count);

    ImmediateEvent immediateEvent =
activeImmediateEvents[randomIndex];

    eventNameText.text =
LocalizationManager.instance.GetTranslatedText(immediateEvent.
eventName);

    eventSubtitleText.text =
LocalizationManager.instance.GetTranslatedText(immediateEvent.
eventSubtitle);

    eventDescriptionText.text =
LocalizationManager.instance.GetTranslatedText(immediateEvent.
eventDescription);

    if (!string.IsNullOrEmpty(immediateEvent.imageName))
    {
        eventImage.gameObject.SetActive(true);

        // Load and display the event image

        string imagePath = "Events/" +
immediateEvent.imageName; // Construct the image path

        Sprite eventSprite =
Resources.Load<Sprite>(imagePath); // Load the image sprite

        eventImage.sprite = eventSprite; // Assign the
sprite to the image component
    }
    else
    {
        // No image associated with the event, handle it
as desired (e.g., display a default image)

        eventImage.gameObject.SetActive(false);

        eventImage.sprite = null; // Assign a default
sprite or set the image to blank
    }

```

```

        //typeCoroutine =
        StartCoroutine (TypeEventText (immediateEvent.eventName,
        immediateEvent.eventSubtitle,
        immediateEvent.eventDescription));

        immediateEvent.TriggerEvent (playerStats);

        if (immediateEvent.eventType == "Positive")
        {
            positiveAudioSource.Play();
        }
        else if (immediateEvent.eventType == "Negative")
        {
            negativeAudioSource.Play();
        }

        // Move the event from activeImmediateEvents to
        completedImmediateEvents

        activeImmediateEvents.RemoveAt (randomIndex);
        completedImmediateEvents.Add (immediateEvent);

        // Disable choice buttons
        choice1Button.gameObject.SetActive (false);
        choice2Button.gameObject.SetActive (false);
    }

    public void TriggerChoiceEvents (List<ChoiceEvent>
    choiceEvents, PlayerStats playerStats, TextMeshProUGUI
    subtitleText, TextMeshProUGUI descriptionText, Button choice1,
    Button choice2)
    {
        if (completedChoiceEvents.Count == 4)
        {
            ChoiceEvent firstCompletedChoiceEvent =
            completedChoiceEvents[0];
            completedChoiceEvents.RemoveAt (0);
            activeChoiceEvents.Add (firstCompletedChoiceEvent);
        }
    }

```



```

        Debug.Log("4 choice events are triggered, putting
oldest event back");
    }

    isChoiceEvent = true;
    choice1.gameObject.SetActive(true);
    choice2.gameObject.SetActive(true);
    choice1.interactable = true;
    choice2.interactable = true;
    exploreButton.interactable = false;

    int randomIndex = Random.Range(0,
activeChoiceEvents.Count);

    ChoiceEvent choiceEvent =
activeChoiceEvents[randomIndex];

    currentChoiceEvent = choiceEvent;

    eventNameText.text =
LocalizationManager.instance.GetTranslatedText(choiceEvent.eve
ntName);

    subtitleText.text =
LocalizationManager.instance.GetTranslatedText(choiceEvent.eve
ntSubtitle);

    descriptionText.text =
LocalizationManager.instance.GetTranslatedText(choiceEvent.eve
ntDescription);

    if (!string.IsNullOrEmpty(choiceEvent.imageName))
    {
        eventImage.gameObject.SetActive(true);

        // Load and display the event image

        string imagePath = "Events/" +
choiceEvent.imageName; // Construct the image path

        Sprite eventSprite =
Resources.Load<Sprite>(imagePath); // Load the image sprite

        eventImage.sprite = eventSprite; // Assign the
sprite to the image component
    }

    else

```

```

        {
            // No image associated with the event, handle it
            as desired (e.g., display a default image)
            eventImage.gameObject.SetActive(false);
            eventImage.sprite = null; // Assign a default
            sprite or set the image to blank
        }

        choiceEvent.TriggerEvent(playerStats, subtitleText,
        descriptionText, choice1, choice2);

        exploreButton.interactable = false;
        StartCoroutine(WaitForChoice(choice1, choice2));

        // Move completed choice event to the
        completedChoiceEvents list
        completedChoiceEvents.Add(choiceEvent);
        activeChoiceEvents.Remove(choiceEvent);
    }

    public void TriggerEventType(List<ImmediateEvent>
    immediateEvents, List<ChoiceEvent> choiceEvents, PlayerStats
    playerStats)
    {
        exploreButton.interactable = false;
        choice1Button.interactable = false;
        choice2Button.interactable = false;
        choiceMade = false;

        // Generate a random number to choose between
        immediate and choice events
        int eventIndex = Random.Range(0, 6);

        if (eventIndex <= 3)
        {
            exploreButton.interactable = true;
            TriggerImmediateEvents(immediateEvents,
            playerStats);

```

```

    }

    else
    {
        TriggerChoiceEvents(choiceEvents, playerStats,
eventSubtitleText, eventDescriptionText, choice1Button,
choice2Button);
    }

    if (activeImmediateEvents.Count == 0)
    {
        Debug.Log("All immediate events triggered.
Refilling the lists.");
        activeImmediateEvents.AddRange(immediateEvents);

activeImmediateEvents.RemoveAll(completedImmediateEvents.Conta
ins);

        completedImmediateEvents.Clear();
    }
    else if(activeChoiceEvents.Count == 0)
    {
        Debug.Log("All choice events triggered. Refilling
the lists.");
        activeChoiceEvents.AddRange(choiceEvents);

activeChoiceEvents.RemoveAll(completedChoiceEvents.Contains);
        completedChoiceEvents.Clear();
    }

    eventsTriggered++;

    if (eventsTriggered % 5 == 0)
    {
        actionMenu.EnableButtons();
        exploreButton.gameObject.SetActive(false);
        actionMenuButton.gameObject.SetActive(true);
        if (isChoiceEvent == true)

```

```

        {
            actionMenuButton.interactable = false;
        }
        else
        {
            actionMenuButton.interactable = true;
        }
        return; // Exit the method since the action menu
is active
    }
}

```

```

private IEnumerator WaitForChoice(Button choice1, Button
choice2)
{
    choice1.onClick.AddListener(() =>
    {
        if (currentChoiceEvent.eventType == "Positive")
        {
            positiveAudioSource.Play();
        }
        else if (currentChoiceEvent.eventType ==
"Negative")
        {
            negativeAudioSource.Play();
        }
        choiceMade = true;
    });

    choice2.onClick.AddListener(() =>
    {
        if (currentChoiceEvent.eventType2 == "Positive")
        {
            positiveAudioSource.Play();
        }
    }
}

```

```

        else if (currentChoiceEvent.eventType2 ==
"Negative")
        {
            negativeAudioSource.Play();
        }
        choiceMade = true;
    });

    // Wait until a choice has been made
    yield return new WaitUntil(() => choiceMade);

    // Remove the listeners from the choice buttons
    choice1.onClick.RemoveAllListeners();
    choice2.onClick.RemoveAllListeners();

    // Make the Explore button interactable again
    exploreButton.interactable = true;
    actionMenuButton.interactable = true;
}

public void ContinueAfterActionPanel()
{
    actionMenuButton.interactable = false;
    exploreButton.gameObject.SetActive(true);
    exploreButton.interactable = true;
    choice1Button.interactable = true;
    choice2Button.interactable = true;
}

public void ShowActionMenuButton()
{
    actionMenuButton.onClick.AddListener(() =>
    {
        actionMenu.ActivateActionMenu();
    });
}

```

```

        actionMenu.EnableButtons();

        exploreButton.gameObject.SetActive(true); // Make
the Explore button interactable again

        actionMenuButton.gameObject.SetActive(false); //
Disable the action menu button after activating the action
menu

    });
}

public void DeactivateActionButton()
{
    actionMenuButton.gameObject.SetActive(false);
}

}

```

LocalizationManager.cs:

```

using System.Collections.Generic;
using UnityEngine;

public class LocalizationManager : MonoBehaviour
{
    public static LocalizationManager instance;

    public Dictionary<string, Dictionary<string, string>>
translations = new Dictionary<string, Dictionary<string,
string>>();

    public string defaultLanguage = "en";
    private string currentLanguage;

    private const string PlayerPrefKey = "SelectedLanguage";

    private void Awake()
    {
        if (instance == null)
        {

```

```

        instance = this;
        DontDestroyOnLoad(gameObject);
        InitializeTranslations();

        // Check if the language preference has been set
        if (!PlayerPrefs.HasKey(PlayerPrefKey))
        {
            // If the language preference is not set, set
the language to English
            currentLanguage = defaultLanguage;
            SaveLanguagePreference(currentLanguage);
        }
        else
        {
            // If the language preference is set, load it
            LoadLanguagePreference();
        }
    }
    else
    {
        Destroy(gameObject);
    }
}

private void InitializeTranslations()
{
    // Add translations for each language here
    // Example: English and Indonesian
    translations = new Dictionary<string,
Dictionary<string, string>>();

    // English translations
    var englishTranslations = new Dictionary<string,
string>
    {

```

```

        { "menu_play", "Play" },
        { "menu_credits", "Credits" },
        { "menu_language", "Language" },

```

LocalizedText.cs:

```

using UnityEngine;
using TMPro;

[RequireComponent(typeof(TextMeshProUGUI))]
public class LocalizedText : MonoBehaviour
{
    public string key;

    private void Start()
    {
        // Register this localized text component with the
        LocalizationManager

        LocalizationManager.instance.RegisterLocalizedText(this);

        // Set the initial text

        SetText(LocalizationManager.instance.GetTranslatedText(key));
    }

    public void SetText(string text)
    {
        GetComponent<TextMeshProUGUI>().text = text;
    }
}

```

ActionMenu.cs:

```

using UnityEngine;
using UnityEngine.UI;
using TMPro;

```



```

public class ActionMenu : MonoBehaviour
{
    [SerializeField] private PlayerStats playerStats; //
    Reference to the PlayerStats script

    [SerializeField] private GameObject actionMenu; //
    Reference to the UI panel containing the choice buttons

    [SerializeField] private Button plusHP; // Reference to
    the +HP choice button

    [SerializeField] private Button plusEXP; // Reference to
    the +EXP choice button

    [SerializeField] private Button minusINF; // Reference to
    the -INF choice button

    [SerializeField] private Button gambleHP; // Reference to
    the -INF choice button

    [SerializeField] private Button gambleEXP; // Reference to
    the -INF choice button

    [SerializeField] private Button gambleINF; // Reference to
    the -INF choice button

    [SerializeField] private TextMeshProUGUI effect;

    private AudioSource positiveAudioSource;
    private AudioSource negativeAudioSource;

    public void EnableButtons()
    {
        positiveAudioSource =
        gameObject.AddComponent<AudioSource>();

        negativeAudioSource =
        gameObject.AddComponent<AudioSource>();

        // Load the sound effects from Resources/SFX folder

        AudioClip positiveClip =
        Resources.Load<AudioClip>("SFX/Positive");

        AudioClip negativeClip =
        Resources.Load<AudioClip>("SFX/Negative");

        positiveAudioSource.clip = positiveClip;
        negativeAudioSource.clip = negativeClip;
    }
}

```

```

        plusHP.interactable = true;
        plusEXP.interactable = true;
        minusINF.interactable = true;
        gambleHP.interactable = true;
        gambleEXP.interactable = true;
        gambleINF.interactable = true;
    }

    private void DisableButtons()
    {
        plusHP.interactable = false;
        plusEXP.interactable = false;
        minusINF.interactable = false;
        gambleHP.interactable = false;
        gambleEXP.interactable = false;
        gambleINF.interactable = false;
    }

    public void ActivateActionMenu()
    {
        actionMenu.SetActive(true);
        effect.text =
LocalizationManager.instance.GetTranslatedText("actioninfo");
    }

    public void DeactivateActionMenu()
    {
        actionMenu.SetActive(false);
    }

    public void ManageHP()
    {
        int chance = 80;

```

```

        int randomChance = Random.Range(1, 101);
        if (randomChance <= chance)
        {
            int hpChange = Random.Range(5, 15);
            playerStats.ModifyHealth(hpChange);
            positiveAudioSource.Play();
            DisableButtons();

            effect.text =
LocalizationManager.instance.GetTranslatedText("actionresteffe
cttext") + Mathf.Abs(hpChange) +

LocalizationManager.instance.GetTranslatedText("actionhpincrea
se");

        }
        else
        {
            negativeAudioSource.Play();
            DisableButtons();

            effect.text =
LocalizationManager.instance.GetTranslatedText("actionhpfail")
;

        }
    }

    public void GambleHP()
    {
        int increaseChance = 60;
        int randomChance = Random.Range(1, 101);

        // Check if the randomChance falls within the
increaseChance range
        if (randomChance >= increaseChance)
        {
            // Add code here to increase HP and update
PlayerStats accordingly
            int hpChange = Random.Range(16, 30);
            playerStats.ModifyHealth(hpChange);

```

```

        positiveAudioSource.Play();

        DisableButtons();

        effect.text =
LocalizationManager.instance.GetTranslatedText("actiondresteff
ect1text") + Mathf.Abs (hpChange) +

LocalizationManager.instance.GetTranslatedText("actionhpincrea
se");

    }

    else
    {

        // Add code here to decrease HP and update
PlayerStats accordingly

        int hpChange = Random.Range(-16, -20);

        playerStats.ModifyHealth(hpChange);

        negativeAudioSource.Play();

        DisableButtons();

        effect.text =
LocalizationManager.instance.GetTranslatedText("actiondresteff
ect2text") + Mathf.Abs (hpChange) +

LocalizationManager.instance.GetTranslatedText("actionhpdecrea
se");

    }

}

public void ManageEXP()
{

    int chance = 80;

    int randomChance = Random.Range(1, 101);

    int expChange = Random.Range(1, 3);

    if (randomChance <= chance)
    {

        playerStats.ModifyExploration(expChange);

        positiveAudioSource.Play();

        DisableButtons();

```

```

        effect.text =
LocalizationManager.instance.GetTranslatedText("actionsearcheffecttext") + Mathf.Abs(expChange) +

LocalizationManager.instance.GetTranslatedText("actionexpincrease");
    }
    else
    {
        negativeAudioSource.Play();
        DisableButtons();
        effect.text =
LocalizationManager.instance.GetTranslatedText("actionexpfail");
    }
}

public void GambleEXP()
{
    int increaseChance = 80;
    int randomChance = Random.Range(1, 101);

    if (randomChance >= increaseChance)
    {
        int expChange = Random.Range(4, 6);
        playerStats.ModifyExploration(expChange);
        positiveAudioSource.Play();
        DisableButtons();
        effect.text =
LocalizationManager.instance.GetTranslatedText("actionscouteffect1text") + Mathf.Abs(expChange) +

LocalizationManager.instance.GetTranslatedText("actionexpincrease");
    }
    else
    {
        int expChange = Random.Range(-4, -6);

```

```

        playerStats.ModifyExploration(expChange);

        negativeAudioSource.Play();

        DisableButtons();

        effect.text =
LocalizationManager.instance.GetTranslatedText("actionsouteffect2text") + Mathf.Abs(expChange) +

LocalizationManager.instance.GetTranslatedText("actionexpdecrease");

    }

}

public void ManageInfection()
{
    int chance = 80;
    int randomChance = Random.Range(1, 101);
    if (randomChance <= chance)
    {
        if (playerStats.infection >=
playerStats.maxInfection / 2)
        {
            int infChange = Random.Range(-3, -6);
            playerStats.ModifyInfection(infChange);
            positiveAudioSource.Play();
            DisableButtons();

            effect.text =
LocalizationManager.instance.GetTranslatedText("actiondisinfecteffect1text") + Mathf.Abs(infChange) +

LocalizationManager.instance.GetTranslatedText("actioninfdecrease");

        }
    }
    else
    {
        playerStats.ModifyInfection(-1);
        positiveAudioSource.Play();
        DisableButtons();
    }
}

```

```

        effect.text =
LocalizationManager.instance.GetTranslatedText("actiondisinfec
teffect2text");
    }
}
else
{
    negativeAudioSource.Play();
    DisableButtons();
    effect.text =
LocalizationManager.instance.GetTranslatedText("actioninffail"
);
}
}

public void GambleINF()
{
    int increaseChance = 90;
    int randomChance = Random.Range(1, 101);

    if (randomChance >= increaseChance)
    {
        int infChange = Random.Range(-4, -6);
        playerStats.ModifyInfection(infChange);
        positiveAudioSource.Play();
        DisableButtons();
        effect.text =
LocalizationManager.instance.GetTranslatedText("actiondisinfec
teffect3text") + Mathf.Abs(infChange) +

LocalizationManager.instance.GetTranslatedText("actioninfdecre
ase");
    }
    else
    {
        int infChange = Random.Range(4, 6);
        playerStats.ModifyInfection(infChange);
    }
}

```

```

        negativeAudioSource.Play();

        DisableButtons();

        effect.text =
LocalizationManager.instance.GetTranslatedText("actiondisinfecteffect4text") + Mathf.Abs(infChange) +

LocalizationManager.instance.GetTranslatedText("actioninfincrease");

    }

}
}

```

Character.cs:

```

using System.Collections.Generic;

[System.Serializable]
public class Character
{
    public string characterName;
    public string characterImage;
    public string characterDescription;
    public int maxHealth;
    public int maxExploration;
    public int maxInfection;

    public Character(string characterNameKey, string
characterImage, string characterDescriptionKey, int maxHealth,
        int maxExploration, int maxInfection)
    {
        this.characterName =
LocalizationManager.instance.GetTranslatedText(characterNameKey);

        this.characterImage = characterImage;

        this.characterDescription =
LocalizationManager.instance.GetTranslatedText(characterDescriptionKey);

        this.maxHealth = maxHealth;

        this.maxExploration = maxExploration;
    }
}

```



```

        this.maxInfection = maxInfection;
    }

    public static List<Character> characters = new
    List<Character>()
    {
        new Character("character_survivor", "Survivor",
        "character_survivordesc", 100, 50, 50),
        new Character("character_soldier", "Soldier",
        "character_soldierdesc", 125, 60, 40),
    };
}

```

CharacterSelect.cs:

```

using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;
using TMPro;

public class CharacterSelect : MonoBehaviour
{
    [SerializeField] private Button[] characterButtons;
    [SerializeField] private Image[] characterImages;
    [SerializeField] private TextMeshProUGUI[]
characterNameTexts;
    [SerializeField] private TextMeshProUGUI
characterDescription;
    [SerializeField] private Button continueButton;
    public GameObject info;
    private List<Character> characters = Character.characters;
    private PlayerStats playerStats;
    private Character selectedCharacter;

    private void Start()
    {
        info.SetActive(true);
    }
}

```

```

        PlayerPrefs.DeleteKey("MaxHealth");
        PlayerPrefs.DeleteKey("MaxExploration");
        PlayerPrefs.DeleteKey("MaxInfection");
        AssignCharacterSelection();
        ShuffleCharacters();
        playerStats = FindObjectOfType<PlayerStats>();

        continueButton.onClick.AddListener(ConfirmCharacterSelection);
        UpdateUI();
    }

    private void AssignCharacterSelection()
    {
        for (int i = 0; i < characterButtons.Length; i++)
        {
            int index = i;
            characterButtons[i].onClick.AddListener(() =>
                SelectCharacter(index));
        }
    }

    private void ShuffleCharacters()
    {
        for (int i = 0; i < characters.Count; i++)
        {
            int randomIndex = Random.Range(i,
                characters.Count);
            Character temp = characters[randomIndex];
            characters[randomIndex] = characters[i];
            characters[i] = temp;
        }
    }

    public void SelectCharacter(int index)
    {

```

```

        info.SetActive(false);

        selectedCharacter = characters[index];

        characterDescription.text =
selectedCharacter.characterDescription;

        UpdateCharacterImages();

        Debug.Log("Selected Character: " +
selectedCharacter.characterName);
    }

    public void ConfirmCharacterSelection()
    {
        if (selectedCharacter != null)
        {
            // Save the selected character's data in
PlayerPrefs

            PlayerPrefs.SetString("SelectedCharacterName",
selectedCharacter.characterName);

            PlayerPrefs.SetString("SelectedCharacterImage",
selectedCharacter.characterImage);

            PlayerPrefs.SetInt("MaxHealth",
selectedCharacter.maxHealth);

            PlayerPrefs.SetInt("MaxExploration",
selectedCharacter.maxExploration);

            PlayerPrefs.SetInt("MaxInfection",
selectedCharacter.maxInfection);

            SceneManager.LoadScene("Main Gameplay");
        }
        else
        {
            Debug.Log("No character selected!");
        }
    }

    private void UpdateUI()
    {
        for (int i = 0; i < characterNameTexts.Length; i++)

```

```

        {
            characterNameTexts[i].text =
characters[i].characterName;
        }
        UpdateCharacterImages();
    }

    private void UpdateCharacterImages()
    {
        for (int i = 0; i < characterImages.Length; i++)
        {
            characterImages[i].sprite =
GetCharacterSprite(characters[i].characterImage);
        }
    }

    private Sprite GetCharacterSprite(string characterImage)
    {
        string imagePath = "Characters/" + characterImage;
        return Resources.Load<Sprite>(imagePath);
    }
}

```