

**IMPLEMENTASI *LARGE LANGUAGE MODEL* MENGGUNAKAN  
*GEMINI 1.0 PRO* DALAM PERMAINAN *ROLE-PLAYING* UNTUK  
MENCIPTAKAN *EMERGENT GAMEPLAY***

**SKRIPSI**

**BRIAN WIJAYA**

**201401042**



**PROGRAM STUDI S-1 ILMU KOMPUTER  
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI  
UNIVERSITAS SUMATERA UTARA  
MEDAN  
2024**

**IMPLEMENTASI *LARGE LANGUAGE MODEL* MENGGUNAKAN  
*GEMINI 1.0 PRO* DALAM PERMAINAN *ROLE-PLAYING* UNTUK  
MENCIPTAKAN *EMERGENT GAMEPLAY***

**SKRIPSI**

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

**BRIAN WIJAYA**

**201401042**



**PROGRAM STUDI S-1 ILMU KOMPUTER  
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI  
UNIVERSITAS SUMATERA UTARA  
MEDAN  
2024**

**PERSETUJUAN**

Judul : IMPLEMENTASI *LARGE LANGUAGE MODEL*  
MENGGUNAKAN *GEMINI 1.0 PRO* DALAM  
PERMAINAN *ROLE-PLAYING* UNTUK  
MENCIPTAKAN *EMERGENT GAMEPLAY*

Kategori : SKRIPSI

Nama : BRIAN WIJAYA

Nomor Induk Mahasiswa : 201401042

Program Studi : SARJANA (S-1) ILMU KOMPUTER

Fakultas : ILMU KOMPUTER DAN TEKNOLOGI  
INFORMASI UNIVERSITAS SUMATERA UTARA

Medan, 11 Juni 2024

Komisi Pembimbing:

Dosen Pembimbing II

Dosen Pembimbing I



Sri Melvani Hardi, S.Kom., M.Kom.  
NIP. 198805012015042006



Dr. Jos Timanta Tarigan, S.Kom., M.Sc.  
NIP. 198501262015041001

Diketahui/disetujui oleh

Program Studi S-1 Ilmu Komputer

Ketua,



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

**PERNYATAAN**

**IMPLEMENTASI *LARGE LANGUAGE MODEL* MENGGUNAKAN *GEMINI 1.0*  
*PRO* DALAM PERMAINAN *ROLE-PLAYING* UNTUK MENCIPTAKAN  
*EMERGENT GAMEPLAY***

**SKRIPSI**

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

Medan, 14 Mei 2024

Penulis,



Brian Wijaya

NIM. 201401042

## PENGHARGAAN

Dengan rahmat Tuhan Yang Maha Esa, penulis bersyukur karena berkat-Nya penulis dapat memenuhi dan menyelesaikan tahap penyusunan skripsi ini sebagai syarat untuk memperoleh gelar Sarjana Komputer di Program Studi S-1 Ilmu Komputer, Fakultas Ilmu Komputer dan Teknologi Informasi, Universitas Sumatera Utara. Melalui kesempatan ini, penulis ingin menyampaikan rasa terima kasih dan hormat kepada Ibu, Ayah, dan Adik penulis yang telah mendukung perjuangan penulis dalam melewati kesulitan dan terkhususnya kepada Ibu yang telah merawat penulis saat mengalami demam selama penulisan skripsi ini.

Penulis juga ingin menyampaikan rasa hormat dan terima kasih kepada pihak yang telah memberikan dukungan dan bimbingan kepada penulis yang tidak ternilai selama penyusunan dan penyelesaian skripsi ini.

1. Bapak Dr. Muryanto Amin, S.Sos., M.Si. selaku Rektor Universitas Sumatera Utara.
2. Ibu Dr. Maya Silvi Lydia, B.Sc., M.Sc. selaku Dekan Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara.
3. Ibu Dr. Amalia, ST., M.T. selaku Ketua Program Studi S-1 Ilmu Komputer Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara, yang turut memberikan dukungan selama penyelesaian skripsi ini.
4. Bapak Dr. Jos Timanta Tarigan, S.Kom., M.Sc. selaku Ketua Laboratorium Computer Vision dan Multimedia Program Studi Ilmu Komputer dan juga selaku Dosen Pembimbing I, yang telah memberikan dorongan, bimbingan, dan saran yang sangat berharga sehingga penulisan dan penyelesaian skripsi ini dapat berjalan dengan baik.
5. Ibu Sri Melvani Hardi, S.Kom., M.Kom. selaku Sekretaris Program Studi S-1 Ilmu Komputer Universitas Sumatera Utara dan juga selaku Dosen Pembimbing II, yang telah memberikan saran, dorongan, dan bimbingan yang sangat berharga sehingga penyusunan dan penyelesaian skripsi ini dapat berjalan dengan baik.
6. Kekasih penulis, Tiffany Censaka, yang telah memberikan dukungan, motivasi, dan semangat yang berharga kepada penulis, terutama ketika penulis merasakan keraguan, kesulitan, dan putus asa selama penyusunan skripsi ini.

7. Avin Chaili Salim, Venerio Uvandy, dan Petrus Marcelino H. Tampubolon, sebagai rekan seperjuangan yang telah memberikan informasi, bantuan, dan saran selama pengurusan administrasi, penulisan, dan penyelesaian skripsi ini.

Peneliti juga berterima kasih kepada pihak diluar dari yang disebutkan di atas yang telah berkontribusi dalam pengerjaan skripsi ini. Peneliti berharap skripsi ini dapat bermanfaat bagi peneliti lainnya dan dapat mendorong kemajuan pendidikan terutama pada bidang pengembangan *game* di Indonesia.

Medan, 14 Mei 2024

Penulis,

A handwritten signature in black ink, appearing to be 'Brian Wijaya', written over a light gray rectangular background.

Brian Wijaya

## ABSTRAK

Perkembangan dari *Large Language Model* mendorong adanya kemajuan pada semua bidang teknologi, tidak terkecuali media hiburan yang didominasi oleh *video game*. *Large Language Model* memiliki kemampuan untuk melakukan *role-playing* yang dapat membantu *game designer* untuk merancang maupun mengintegrasikan langsung *Large Language Model* itu sendiri dalam suatu *game*.

Pemanfaatan *Large Language Model* dalam *game* dinilai masih kurang dimanfaatkan secara luas, terutama pada *game* dengan genre *RPG* yang dapat menggunakan potensi dari *Large Language Model* untuk menciptakan pengalaman bermain yang *emergent*. *Emergent gameplay* muncul dari interaksi dinamis antara mekanik sebuah *game*, keputusan pemain selama permainan dan kejadian yang tidak terduga. Dengan adanya sifat *emergent* ini, permainan dapat memberikan narasi yang lebih unik, eksplorasi, dan kejutan bagi pemain untuk merasakan pengalaman bermain yang *immersive*.

Penelitian ini menggunakan *project game Rudantara*, sebuah *game* dengan genre *RPG* dan *Rogue-like* dengan peta dan area permainan yang dibentuk secara prosedural. Implementasi dari *Large Language Model* dilakukan pada *companion* yang menemani pemain dalam petualangannya. Aksi yang dilakukan oleh *companion* dikendalikan oleh *Large Language Model* berdasarkan *prompt* yang dirancang sesuai dengan modul sensor, memori, dan perilaku yang dimiliki *companion*. Setiap modul ini mengambil data dari lingkungan permainan secara *realtime* untuk memberikan informasi terbaru kepada perancang *prompt* yang meminta *response* dari *Large Language Model* dalam interval waktu dan keadaan tertentu. Penelitian ini menggunakan *Gemini 1.0 Pro API* sebagai *Large Language Model* yang diintegrasikan pada *game*. Pengujian pada penelitian ini dilakukan dengan memberikan kuesioner kepada 8 pengguna untuk menilai potensi dari penerapan *LLM* pada *role-playing game* dan apakah *emergent gameplay* dapat tercipta.

**Kata Kunci:** *Large Language Model, Gemini 1.0 Pro, Role-playing Game, Emergent Gameplay.*

# **IMPLEMENTATION OF LARGE LANGUAGE MODEL USING GEMINI 1.0 PRO IN A ROLE-PLAYING GAME TO CREATE EMERGENT GAMEPLAY**

## **ABSTRACT**

The development of Large Language Models (LLMs) has driven advancements in all technological fields, including the entertainment media dominated by video games. LLMs possess role-playing capabilities that can assist game designers in both designing and directly integrating LLMs into games.

The utilization of LLMs within games is still considered under-explored, particularly in the RPG genre which has the potential to leverage LLMs to create emergent gameplay experiences. Emergent gameplay arises from the dynamic interaction between game mechanics, player decisions during gameplay, and unexpected events. This emergent nature allows games to provide more unique narratives, exploration, and surprises for players, leading to a more immersive experience.

This research utilizes the game project "Rudantara," an RPG and Rogue-like game with procedurally generated maps and play areas. The implementation of the LLM focuses on a companion character that accompanies the player throughout their adventure. The companion's actions are controlled by the LLM based on prompts designed in accordance with the companion's sensor, memory, and behavior modules. Each module gathers real-time data from the game environment to provide the latest information to the prompt designer script, which then requests responses from the LLM at specific intervals and under certain conditions. This research uses the Gemini 1.0 Pro API as the LLM integrated into the game. Testing in this research was carried out by giving questionnaires to 8 users to assess the potential of applying LLM to role-playing games and whether emergent gameplay could be created.

**Keywords:** Large Language Model, Gemini 1.0 Pro, Role-playing Game, Emergent Gameplay.



## DAFTAR ISI

<b>PERSETUJUAN.....</b>	<b>iii</b>
<b>PERNYATAAN.....</b>	<b>iv</b>
<b>PENGHARGAAN.....</b>	<b>v</b>
<b>ABSTRAK .....</b>	<b>vii</b>
<b>ABSTRACT .....</b>	<b>viii</b>
<b>DAFTAR ISI.....</b>	<b>ix</b>
<b>DAFTAR TABEL .....</b>	<b>xi</b>
<b>DAFTAR GAMBAR.....</b>	<b>xii</b>
<b>DAFTAR LAMPIRAN.....</b>	<b>xiii</b>
<b>DAFTAR RUMUS .....</b>	<b>xiv</b>
<b>BAB I PENDAHULUAN.....</b>	<b>1</b>
1.1. Latar Belakang .....	1
1.2. Rumusan Masalah .....	2
1.3. Batasan Masalah .....	3
1.4. Tujuan Penelitian .....	3
1.5. Manfaat Penelitian .....	3
1.6. Penelitian Relevan.....	3
1.7. Metodologi Penelitian .....	4
1.8. Sistematika Penulisan .....	5
<b>BAB II TINJAUAN PUSTAKA.....</b>	<b>7</b>
2.1. <i>Role-playing Game (RPG)</i> .....	7
2.2. <i>Large Language Model (LLM)</i> .....	8
2.3. <i>Emergent Gameplay</i> .....	9
2.4. Rudantara .....	9
<b>BAB III ANALISIS DAN PERANCANGAN SISTEM.....</b>	<b>13</b>
3.1. Analisis Masalah .....	13
3.2. Analisis Kebutuhan .....	14
3.3. Perancangan Arsitektur Sistem .....	15
3.3.1. <i>Perancangan Environment</i> .....	16
3.3.2. <i>Perancangan Memory dan Memori Relevan</i> .....	18
3.3.3. <i>Perancangan State</i> .....	21
3.3.4. <i>Perancangan Action</i> .....	21
3.3.5. <i>Perancangan LLM</i> .....	24

3.4. Keluaran Sistem .....	25
<b>BAB IV IMPLEMENTASI DAN PENGUJIAN SISTEM.....</b>	<b>29</b>
4.1. Implementasi Sistem .....	29
4.2. Pengujian Sistem.....	35
4.2.1. <i>Pengujian pada personality traits companion</i> .....	35
4.2.2. <i>Pengujian terhadap pengguna</i> .....	43
<b>BAB V PENUTUP.....</b>	<b>47</b>
5.1. Kesimpulan .....	47
5.2. Saran.....	47
<b>DAFTAR PUSTAKA.....</b>	<b>49</b>
<b>LAMPIRAN.....</b>	<b>A-1</b>

**DAFTAR TABEL**

<b>Tabel 2.1.</b> Penjelasan statistik karakter .....	10
<b>Tabel 2.2.</b> Kontrol pada game Rudantara .....	12
<b>Tabel 3.1.</b> Parameter pengendali <i>vision</i> .....	17
<b>Tabel 3.2.</b> Indikator <i>health stat</i> .....	22
<b>Tabel 4.1.</b> Hasil Kuesioner .....	44

## DAFTAR GAMBAR

<b>Gambar 1.1.</b> Metodologi Penelitian.....	5
<b>Gambar 2.1.</b> Pemilihan aksi pemain dalam <i>game</i> Until Dawn (2015).....	8
<b>Gambar 3.1.</b> Arsitektur Sistem .....	16
<b>Gambar 3.2.</b> <i>Vision mesh</i> mendeteksi objek di dalamnya .....	17
<b>Gambar 3.3.</b> Daftar ingatan jangka panjang sebelum menerima ingatan baru ....	19
<b>Gambar 3.4.</b> Daftar ingatan jangka panjang setelah menerima ingatan baru .....	19
<b>Gambar 4.1.</b> Tampilan <i>Main Menu</i> Rudantara .....	29
<b>Gambar 4.2.</b> Tampilan <i>Gameplay</i> Rudantara .....	30
<b>Gambar 4.3.</b> Tampilan <i>Pause Menu</i> Rudantara.....	30
<b>Gambar 4.4.</b> Tampilan <i>Upgrades Menu</i> Rudantara.....	31
<b>Gambar 4.5.</b> Tampilan <i>Status Menu</i> Rudantara.....	31
<b>Gambar 4.6.</b> Tampilan <i>Companion Settings</i> Rudantara .....	32
<b>Gambar 4.7.</b> Tampilan <i>Instruction Panel</i> Rudantara.....	33
<b>Gambar 4.8.</b> Tampilan <i>Win Panel</i> Rudantara.....	33
<b>Gambar 4.9.</b> Tampilan <i>Game Over</i> Rudantara .....	33
<b>Gambar 4.10.</b> Salah satu dialog pada <i>companion</i> .....	34
<b>Gambar 4.11.</b> Contoh dialog pertarungan .....	34
<b>Gambar 4.12.</b> Contoh dialog pertarungan dengan <i>boss</i> .....	35
<b>Gambar 4.13.</b> Kombinasi <i>traits The Protector</i> .....	36
<b>Gambar 4.14.</b> Diagram frekuensi aksi <i>The Protector</i> .....	37
<b>Gambar 4.15.</b> Contoh pertama dialog <i>The Protector</i> .....	37
<b>Gambar 4.16.</b> Contoh kedua dialog <i>The Protector</i> .....	38
<b>Gambar 4.17.</b> Kombinasi <i>traits The Leader</i> .....	38
<b>Gambar 4.18.</b> Diagram frekuensi aksi <i>The Leader</i> .....	39
<b>Gambar 4.19.</b> Contoh pertama dialog <i>The Leader</i> .....	40
<b>Gambar 4.20.</b> Contoh kedua dialog <i>The Leader</i> .....	40
<b>Gambar 4.21.</b> Kombinasi <i>traits The Rogue</i> .....	41
<b>Gambar 4.22.</b> Diagram frekuensi aksi <i>The Rogue</i> .....	42
<b>Gambar 4.23.</b> Contoh pertama dialog <i>The Rogue</i> .....	42
<b>Gambar 4.24.</b> Contoh kedua dialog <i>The Rogue</i> .....	43

**DAFTAR LAMPIRAN**

<b>Lampiran 1.</b> KUESIONER PENILAIAN ASPEK PENELITIAN MENGENAI FITUR <i>COMPANION</i> PADA <i>PROJECT GAME</i> RUDANTARA.....	A-1
<b>Lampiran 2.</b> TABULASI JAWABAN RESPONDEN .....	B-1

**DAFTAR RUMUS**

<b>Rumus (1)</b> <i>Decay</i> terhadap nilai kebaruan .....	19
<b>Rumus (2)</b> Penilaian skor relevansi dari ingatan jangka panjang .....	20

# BAB I

## PENDAHULUAN

### 1.1. Latar Belakang

Perkembangan industri *video game* yang pesat menjadikan pengembang *game* terus mencari cara baru untuk memperluas batasan antara *immersion* pemain dan keterlibatan pemain dalam *game* (Andrus & Fulda, 2020). Salah satu implementasi yang menjanjikan terletak pada integrasi *Large Language Models (LLMs)* pada *game design*. Penelitian ini bertujuan untuk melihat potensi dari penerapan *Large Language Model (LLM)* pada *role-playing game (RPG)* dalam menciptakan pengalaman bermain yang *emergent*.

Terdapat kenaikan ketertarikan pemain terhadap *game* yang memiliki unsur *emergent gameplay* (Nur Fauzan, 2023). *Emergent gameplay* secara definisi muncul dari interaksi dinamis antara mekanik sebuah *game*, keputusan pemain selama permainan dan kejadian yang tidak terduga. Ini menciptakan kejutan dan eksplorasi bagi pemain untuk secara aktif membentuk narasi dan dunia di sekitar mereka.

Narasi yang kompleks dan dinamis dalam *role-playing game* telah lama memikat para pemain dan pengembang (i Bosch, 2022). Sebelumnya, narasi diatur sedemikian rupa oleh alur cerita dan interaksi karakter yang telah ditentukan sebelumnya oleh *game designer* (Gustafsson et al., 2020). Namun, dengan kemajuan teknologi saat ini, *Large Language Model (LLM)* memberikan kemungkinan bagi pengembang untuk menciptakan pengalaman yang lebih *immersive* yang dapat dikendalikan oleh pemain melalui *emergent gameplay*.

Kreativitas pemain untuk menciptakan pengalaman bermainnya sendiri dan interaksi yang tidak memaksa pemain untuk mengikuti narasi *game* menjadi poin penting dalam penerapan *Large Language Model (LLM)* untuk menghasilkan *emergent gameplay*. Pendekatan *emergent gameplay* ini dapat membentuk narasi yang sejalan dengan aksi yang dilakukan pemain dan pemain dapat menguji kreativitasnya melalui mekanika yang disediakan di dalam *game*.

*Large Language Model (LLM)* memiliki kemampuan untuk memproses informasi, menghasilkan *text output*, dan beradaptasi sesuai konteks, yang memiliki

potensi besar untuk mendorong terciptanya pengalaman yang *emergent* dalam *role-playing game*. Hasil *text output* tersebut digunakan pada sisi *client* dari *game* dan menghasilkan aksi yang dijalankan oleh *agent* dan berdampak pada lingkungan dan narasi yang diterima oleh pemain.

Penelitian ini menggunakan *project game* Rudantara yang merupakan *game* bergenre *rouge-like role-playing game* dimana pemain bermain sebagai karakter yang bertarung melawan musuh dengan peta level yang dirancang secara *procedural*. *Procedural map generation* ini dapat digunakan untuk membuat *gameplay* tidak selalu berulang dan memberikan ruang bagi pemain untuk melakukan eksplorasi dengan mekanika *game* yang disediakan pada lingkungan mereka. *Game* tersebut menjadi sasaran implementasi *Large Language Model Gemini 1.0 Pro* pada *companion* yang ditambahkan sebagai fitur baru untuk *game* tersebut. Setiap aksi yang dilakukan oleh *companion* ditentukan oleh model *Large Language Model (LLM)* yang dipengaruhi oleh *sensor*, perilaku dan memori dari *companion* itu sendiri. *Companion* beraksi sebagai pendamping dari pemain untuk melawan musuh dengan aksi pertarungan yang *emergent* dan bergantung dengan aksi dari pemain yang diterima oleh *companion*.

## 1.2. Rumusan Masalah

Belum banyak *game* yang tersebar di *platform* distribusi yang menerapkan *Large Language Models (LLMs)* secara *seamless*, dimana *game* yang menerapkan *LLM* masih membutuhkan peran manusia untuk memerintahkan atau menggerakkan *agent LLM* sesuai dengan tugasnya. Hal ini memberikan pengalaman bermain yang kurang *immersive* dan dapat ditebak. Pada penelitian ini mengisukan hal tersebut dan menerapkan *sensor*, memori dan kepribadian pada arsitektur *agent* untuk memutuskan aksi yang dilakukan sebagai pendamping bermain dari pengguna. Setiap aksi yang dilakukan pemain dapat mempengaruhi keluaran dari *agent* dan membuat permainan tersebut menjadi *emergent*.



### 1.3. Batasan Masalah

Adapun batasan masalah dalam penelitian ini, yaitu:

1. Implementasi *Large Language Model* dilakukan pada *game* yang dikembangkan menggunakan Unity Editor versi 2022.3.7f1 yang mendukung implementasi sistem *Large Language Model* melalui *API (Application Programming Interface)*.
2. *Large Language Model* yang digunakan selama implementasi adalah *Gemini 1.0 Pro*.
3. Penelitian pada *game* yang dikembangkan dibatasi pada genre *RPG (Role Playing Game)*.
4. *Tools* yang digunakan untuk implementasi *Large Language Model* dibatasi sesuai anggaran dan ruang lingkup penelitian.
5. Penerapan *Large Language Model* yang terbatas dan didasari pada jurnal, *open-source project*, dan artikel.

### 1.4. Tujuan Penelitian

Tujuan dari penelitian ini adalah untuk melihat potensi dan batasan-batasan yang dapat terlihat pada penerapan *Large Language Model (LLM)* pada *role-playing game (RPG)* dan apakah *emergent gameplay* dapat tercipta melalui penerapan tersebut.

### 1.5. Manfaat Penelitian

Manfaat dari penelitian ini adalah untuk memberikan informasi terkait penerapan *Large Language Model (LLM)* pada *role-playing game (RPG)* dan dengan topik pendekatan yang digunakan pada penelitian ini sehingga dapat menjadi referensi bagi pembaca. Efektifitas dari implementasi ini juga dapat dipertimbangkan untuk diterapkan pada *game* lainnya dengan genre yang sama.

### 1.6. Penelitian Relevan

Adapun beberapa penelitian yang relevan dengan penelitian ini, yaitu:

1. Penelitian Park et al., 2023 menciptakan arsitektur dimana suatu *agent* memiliki memori atas persepsi lingkungan mereka dan menggunakan

memori tersebut untuk memutuskan aksi yang dilakukannya. Memori yang relevan juga berperan dalam perencanaan jangka panjang yang nantinya digunakan di masa depan.

2. Penelitian Gong et al., 2023 menggunakan *Large Language Models (LLMs)* seperti GPT-4, Claude-2, LLaMA, dan ChatGPT untuk menciptakan sistem *multi-agent planning* untuk mencapai kolaborasi *agent* dengan pemain dan menghasilkan interaksi permainan yang *emergent*.
3. Penelitian Huang et al., 2022 memanfaatkan pengetahuan luas yang dimiliki oleh *Large Language Models (LLMs)* untuk melaksanakan *task-planning* yang kompleks dan *reasoning* dengan metode *zero-shot*.

### 1.7. Metodologi Penelitian

Adapun beberapa metodologi penelitian yang digunakan pada skripsi ini, yaitu:

1. Studi Pustaka

Tahap ini dilakukan pada awal penelitian guna mencari dan mengumpulkan data maupun informasi yang diperoleh dari referensi seperti jurnal, artikel ilmiah, makalah, situs internet, atau berbagai sumber terpercaya lainnya yang berkaitan dengan *Large Language Models (LLMs)*, *emergent gameplay*, dan topik relevan lainnya.

2. Analisis Permasalahan

Pada tahap ini, dilakukan analisis terhadap data dan informasi yang telah dikumpulkan untuk memahami metode yang digunakan dalam penyelesaian masalah yang relevan dengan penelitian ini.

3. Analisis dan Perancangan Sistem

Pada tahap ini, dilakukan perancangan sistem yang berfokus pada struktur integrasi *Large Language Model* dengan *game project* melalui penggunaan diagram dan visualisasi dari sistem integrasi tersebut.

4. Implementasi Sistem

Pada tahap ini, dilakukan implementasi dari rancangan sistem yang telah dianalisis sebelumnya ke dalam prototipe *game* untuk menghasilkan *output* dari rancangan sistem tersebut. Fitur-fitur yang telah ada dalam prototipe

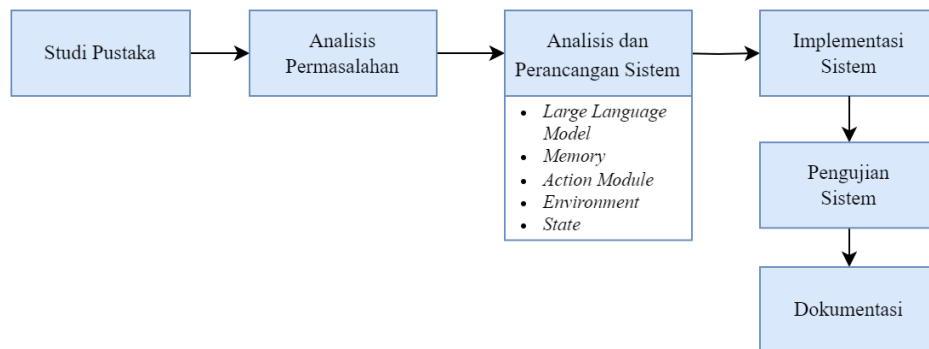
*game* diintegrasikan ke sistem tersebut untuk kemudian dapat dilakukan uji coba (*playtesting*).

#### 5. Pengujian Sistem

Pada tahap pengujian, dilakukan uji coba sistem (*playtesting*) pada *game* yang telah diimplementasikan sebelumnya dan dicek kesesuaian sistem tersebut terhadap rancangan dari struktur yang telah dibuat.

#### 6. Dokumentasi

Pada tahap dokumentasi, penulis melakukan dokumentasi dari tahap analisis perancangan hingga tahap pengujian dalam bentuk laporan akhir yang memaparkan hasil dari penelitian yang telah dilaksanakan.



**Gambar 1.1.** Metodologi Penelitian

### 1.8. Sistematika Penulisan

Adapun sistematika penulisan dari skripsi ini yang terdiri dari 5 (lima) bab, yaitu:

#### **BAB I            PENDAHULUAN**

Bab ini memaparkan mengenai latar belakang penelitian pada skripsi ini, yang terdiri atas latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, penelitian relevan, metodologi penelitian, dan sistematika penulisan.

#### **BAB II           LANDASAN TEORI**

Bab ini memaparkan mengenai tinjauan pustaka dengan topik-topik yang berkaitan dengan *Large Language Model (LLM)*, *Role-playing Game (RPG)*, dan *Emergent Gameplay*.

**BAB III            ANALISIS DAN PERANCANGAN**

Bab ini memaparkan mengenai analisis permasalahan dan perancangan arsitektur sistem untuk menyelesaikan permasalahan tersebut.

**BAB IV            IMPLEMENTASI DAN PENGUJIAN**

Bab ini memaparkan penerapan dan hasil dari penerapan tersebut dari sistem yang telah dirancang berdasarkan hasil analisis dan perancangan sebelumnya. Bab ini juga memaparkan pengujian dan hasil dari pengujian yang telah dilakukan.

**BAB V            KESIMPULAN DAN SARAN**

Bab ini memaparkan kesimpulan akhir dari penelitian yang telah dilakukan pada skripsi ini dan saran atas hasil penelitian ini untuk dimanfaatkan pada penelitian selanjutnya dengan topik yang serupa.

## BAB II

### TINJAUAN PUSTAKA

#### 2.1. *Role-playing Game (RPG)*

*Role-playing game* merupakan *game* yang melibatkan pemain sebagai pemeran dari karakter dalam cerita. *RPG* telah menjadi sumber utama dari *game entertainment* dan telah mendapatkan popularitas sejak pertama kali diluncurkan. *Game* dengan genre ini memungkinkan pemain untuk terlibat sepenuhnya dan membenamkan diri dalam setting fiksi dari berbagai lingkungan dan menjelajahi apa yang ditawarkan (Rodrigues et al., 2021). Pemain ditugaskan untuk menyelesaikan misi yang diberikan, mengikuti cerita, dan mengembangkan karakter mereka dari awal hingga akhir (Widiyanto et al., 2020).

Pemain biasanya didampingi oleh *Non-Player Characters (NPCs)*, baik yang memiliki karakter supportif maupun buruk yang turut berperan dalam alur cerita. Dengan adanya *NPC*, pemain tidak dapat lepas dari kegiatan interaksi dan komunikasi dengan *NPC*. Jika *NPC* tidak melakukan perannya pada cerita secara maksimal, dapat berpengaruh buruk pada pengalaman pengguna terhadap cerita dan juga *immersion* dan keterlibatan pada *game* (Belle et al., 2022). *Modern NPC* memiliki kemampuan untuk bergerak dalam lingkungan tiga dimensi dan mengisi banyak jenis peran, seperti pemberi misi, *companion* atau musuh (Ohrberg, 2019).

Dalam *game* berbasis cerita, dimana narasi berperan besar dalam nilai *gameplay*, seperti *role-playing games (RPGs)*, interaksi yang statis dan repetitif dengan *NPC* dapat menghalangi kesenangan dan *immersion* pemain dalam *game*. Oleh karena itu, penting untuk meningkatkan perilaku karakter dengan membuat interaksi mereka dengan pemain dan dunia dari *game* tampak lebih spontan (Garavaglia et al., 2022).

Penelitian sebelumnya, menyarankan respons pemain dalam dialog dengan *NPC* dapat digunakan untuk pengumpulan data untuk menghindari rusaknya *immersion* atau alur dari pemain. Pendekatan ini telah digunakan di beberapa *game* komersial, seperti *Until Dawn* (2015) (Bowey et al., 2021).



**Gambar 2.1.** Pemilihan aksi pemain dalam *game* Until Dawn (2015)

Gambar di atas meminta aksi cepat dari pemain yang dapat menentukan arah narasi dalam *video game* Until Dawn (2015). Pada *game* tersebut, pemain mengikuti alur cerita yang diikuti dialog dengan *NPC* sesuai dengan narasi yang dipengaruhi aksi pemain.

## 2.2. *Large Language Model (LLM)*

*Large Language Model* merupakan model *artificial intelligence* tingkat lanjut menggunakan teknik *deep learning* yang merevolusi *natural language processing* (NLP) beberapa tahun kebelakang. Model tersebut dilatih pada *dataset* masif dan menunjukkan kemampuan dalam menghasilkan *human-like text*, menjawab pertanyaan, dan menerjemahkan bahasa.

*Large Language Models (LLMs)* yang awalnya dikembangkan untuk *natural language processing*, terbukti efektif untuk hal yang dapat diekspresikan sebagai urutan dari *token*. Fleksibilitas tersebut dapat melampaui apa yang biasanya dianggap sebagai *text completion* hingga menyelesaikan tugas yang terlihat memerlukan kognitif manusia (Nasir & Togelius, 2023).

*Fine-tuning* dan *prompt-engineering* pada *Large Language Model (LLM)* dalam *dataset* yang disesuaikan dapat memandu model dalam menghasilkan keluaran yang terstruktur dan relevan (Reynolds & McDonell, 2021). *Large Language Models (LLMs)* telah digunakan untuk menghasilkan konten *game* seperti *narrative design* pada *quest* dalam *game* dan lingkungan di dalam *game*.

Penelitian ini menggunakan *Gemini 1.0 Pro* yang terhubung dengan *game* melalui *API* yang disediakan oleh *Gemini 1.0 Pro*. *Gemini 1.0* merupakan *multimodal model* yang dikembangkan oleh Google, dibangun di atas *Transformer decoder* yang ditingkatkan untuk memungkinkan pelatihan model yang stabil pada

*Tensor Processing Unit* yang dimiliki Google. *Gemini 1.0* dilatih untuk mendukung 32 ribu *context length* dalam bentuk teks yang dapat disisipkan dengan input audio dan visual (Gemini Team et al., 2023).

Salah satu hasil dari *post-training Gemini 1.0* adalah *Gemini API* yang berfokus pada pengembangan yang mendukung penggunaan secara *conversational* dan *non-conversational*. *Gemini API* dirancang untuk memudahkan integrasi model *Gemini API* ke dalam *workflow* produksi.

Beberapa penelitian terkait menggunakan gpt3.5-turbo, gpt-4, Claude 2, LLaMA, dan ChatGPT sebagai *language model* yang diintegrasikan pada arsitektur *project* penelitian.

### **2.3. Emergent Gameplay**

*Emergent gameplay* merupakan interaksi antara sistem kompleks dalam *game* yang menghasilkan kejadian atau mekanik *game* yang tidak terduga, tanpa dirancang secara eksplisit oleh pengembang. Hal ini dipengaruhi oleh aksi pemain yang menghasilkan narasi atau kejadian yang tidak direncanakan sebelumnya.

*Emergent gameplay* memungkinkan pemain untuk merasakan kebebasan dan kendali dalam pembentukan pengalaman bermain. Dalam penyelesaian sebuah misi dalam *game*, lebih *rewarding* bagi pemain jika metode yang digunakan untuk menyelesaikan misi tersebut merupakan kreasi pribadi, dibandingkan metode yang telah direncanakan oleh *game designer* (Paananen, 2020). Hal ini juga menghasilkan *replayability* yang tinggi dan meningkatkan *engagement* serta kreativitas pemain.

Beberapa *game* yang memiliki naratif *emergent* yang interaktif menggunakan teknik simulasi sosial untuk menghasilkan cerita yang menarik melalui interaksi antara beberapa *agent* yang *autonomous* ataupun *semi-autonomous* (Kreminski, 2023).

### **2.4. Rudantara**

Rudantara merupakan *game* dengan genre *role-playing game* dan *rogue-like* dimana pemain memiliki tujuan untuk menjadi lebih kuat seiring dengan meningkatnya kesulitan pada level permainan. Pemain dapat menjadi lebih kuat

dengan membunuh musuh dalam permainan untuk mendapatkan poin pengalaman karakter baik pada karakter pemain maupun karakter *companion* (pendukung). Level permainan menandakan kesulitan permainan, yang meningkat ketika pemain membunuh *boss* dalam permainan yang muncul pada lokasi tertentu setelah 60 detik waktu permainan. Setelah membunuh *boss*, karakter pemain dan *companion* mendapatkan *reward* berupa poin pengalaman dari seluruh musuh yang ada pada level tersebut. Selanjutnya, pemain dialihkan ke level selanjutnya dengan peta permainan yang berbeda.

Peta pada permainan ini diacak pada awal permainan menggunakan *procedural generation* sehingga setiap permainan menghasilkan tata letak yang berbeda-beda. Pada area permainan, dimunculkan musuh setiap 5 detik dengan level karakter musuh yang diatur sesuai dengan kesulitan level saat bermain. Terdapat dua jenis musuh, yaitu musuh dengan ukuran normal dan musuh dengan ukuran kecil, dimana musuh dengan ukuran kecil memiliki statistik kecepatan, *health*, dan *attack* yang berbeda sehingga memberikan keunikan tertentu dari musuh.

Pemain dapat menggunakan *skill* yang terdiri atas dua jenis, yaitu serangan memutar dan serangan jauh. Serangan ini dimaksudkan untuk membantu pemain dalam mengalahkan musuh dengan mengonsumsi *mana* dalam jumlah tertentu yang dimiliki oleh karakter pemain. *Skill* memiliki waktu *cooldown* sebagai batasan penggunaan selama beberapa waktu. Selain *skill*, pemain dapat melancarkan serangan biasa tanpa waktu *cooldown* yang dipengaruhi oleh statistik kecepatan (*Speed*) yang dimiliki oleh karakter pemain.

Berikut merupakan tabel yang menjelaskan setiap statistik yang dimiliki oleh karakter:

**Tabel 2.1.** Penjelasan statistik karakter

<i>Stats</i>	<b>Deskripsi</b>
<i>Health</i>	Merupakan statistik karakter yang menandakan kehidupan/kesehatan dari karakter yang dipengaruhi melalui <i>damage</i> yang diterima.



<i>Attack</i>	Merupakan statistik karakter yang menentukan nilai <i>damage</i> yang disebabkan oleh serangan karakter.
<i>Defense</i>	Merupakan statistik karakter yang berpengaruh dalam mengurangi <i>damage</i> yang diterima oleh karakter.
<i>Speed</i>	Merupakan statistik karakter yang mengatur kecepatan karakter dalam berjalan atau kecepatan karakter dalam serangan biasa.
<i>Accuracy</i>	Merupakan statistik karakter yang mempengaruhi keluaran akhir dari <i>damage</i> yang disebabkan oleh serangan karakter.
<i>Mana</i>	Merupakan statistik karakter yang membatasi penggunaan serangan <i>skill</i> dengan nilai <i>mana</i> yang dimiliki oleh karakter.
<i>Health Regen</i>	Merupakan statistik karakter yang mempengaruhi penambahan nilai statistik <i>Health</i> setiap detik.
<i>Mana Regen</i>	Merupakan statistik karakter yang mempengaruhi penambahan nilai statistik <i>Mana</i> setiap detik.
<i>Exp Multiplier</i>	Merupakan statistik karakter yang mempengaruhi poin pengalaman yang didapat oleh karakter.

Ketika statistik *health* yang dimiliki oleh karakter pemain mencapai nol setelah menerima serangan dari musuh, permainan berhenti, menandakan kalahnya pemain. Pemain tetap menyimpan poin pengalaman yang dimiliki oleh karakter pemain dan karakter *companion* setelah kalah dalam pertarungan dan dapat mengulangi level permainan dari level pertama.

Selain poin pengalaman, pemain dapat meningkatkan kekuatan dari karakter pemain menggunakan *upgrades* yang didapatkan secara acak saat pemain

membunuh musuh. Pemain dapat memilih *upgrade* yang ingin digunakan untuk meningkatkan statistik karakter. Setiap *upgrade* diacak dan dapat memiliki kelangkaan yang berbeda-beda.

Adapun kontrol dalam permainan yang digunakan oleh pemain untuk mengontrol permainan, yaitu sebagai berikut:

**Tabel 2.2.** Kontrol pada *game* Rudantara

Kontrol	Tombol
Gerakan karakter pemain	Tombol W pada <i>keyboard</i> untuk menggerakkan karakter pemain ke arah atas pada tampilan <i>game</i> , Tombol A pada <i>keyboard</i> untuk menggerakkan karakter pemain ke arah kiri pada tampilan <i>game</i> , Tombol S pada <i>keyboard</i> untuk menggerakkan karakter pemain ke arah bawah pada tampilan <i>game</i> , dan Tombol D pada <i>keyboard</i> untuk menggerakkan karakter pemain ke arah kanan pada tampilan <i>game</i> .
Serangan biasa karakter pemain	Tombol kiri pada <i>mouse</i>
Serangan <i>skill</i> karakter pemain	Tombol 1 pada <i>keyboard</i> untuk menggunakan serangan memutar, dan Tombol 2 pada <i>keyboard</i> untuk menggunakan serangan jauh.
Membuka <i>treasure chest</i>	Tombol spasi pada <i>keyboard</i>
Pause/lanjut permainan	Tombol Esc pada <i>keyboard</i>

Penelitian ini menggunakan *project game* Rudantara sebagai lingkungan implementasi *Large Language Model (LLM)* ke dalam *game* dengan genre *role-playing game* untuk melihat kapabilitas dan potensi dari *LLM* untuk menciptakan pengalaman bermain yang *emergent* melalui interaksi pemain dengan *companion* pada *game* Rudantara.

## BAB III

### ANALISIS DAN PERANCANGAN SISTEM

#### 3.1. Analisis Masalah

Adapun tantangan dan masalah yang terdapat pada pengembangan *project game* dan implementasi *LLM* pada penelitian ini, yaitu:

1. *Latency* dari *LLM API* yang cukup lama

*Latency* yang dialami saat melakukan *web request* kepada *LLM API* menjadi salah satu penghalang dimana diperlukan waktu beberapa detik sebelum aksi *companion* dapat dilaksanakan sesuai dengan keadaan *companion* yang dirasakan secara *realtime*. Hal ini dapat mengurangi sifat *immersive* dari permainan sehingga mengurangi kenyamanan bermain.

2. Halusinasi *LLM*

*Large Language Model (LLM)* dapat memberikan respons yang tidak sesuai fakta dan dapat berpengaruh pada narasi yang diterima pemain dan kesesuaiannya terhadap lingkungan bermain yang dirasakan oleh pemain.

3. Koherensi dan konsistensi dari *LLM*

*Large Language Model (LLM)* memiliki sifat yang probabilistik, mungkin menghasilkan respons yang tidak konsisten dengan kepribadian dari karakter *companion* yang mereka gambarkan. Tidak hanya itu, *LLM* perlu untuk mempertimbangkan *lore* dari cerita *game* dengan baik untuk menghindari respons yang bertentangan. Hal ini dapat mengganggu alur narasi dan pengalaman bermain.

4. Konten tidak pantas yang dapat dihasilkan oleh *LLM*

*Large Language Model (LLM)* dilatih pada cakupan data yang sangat luas, mungkin untuk menghasilkan keluaran yang mengandung konten yang menyinggung, diskriminatif, atau berbahaya.

5. Keterbatasan *token* yang dapat diterima oleh *LLM*

*Large Language Model (LLM)* menggunakan *token* untuk menerima input dan jumlah *token* yang dapat diproses dalam waktu tertentu memiliki batasan.

### 3.2. Analisis Kebutuhan

Melalui analisis masalah-masalah di atas, terdapat analisis kebutuhan yang perlu dipenuhi agar proses pengembangan *project game* dan implementasi *LLM* pada penelitian ini dapat berjalan lebih efektif, yaitu:

1. Peningkatan *Latency* dari *LLM API*

Perlu adanya peningkatan *latency* pada *API* yang digunakan untuk *LLM* seperti *Gemini 1.0 Pro API* yang digunakan pada penelitian ini, mengingat *LLM Gemini 1.0 Pro* sendiri sudah memiliki *latency* yang cukup rendah. Dengan peningkatan *latency* pada bagian *API*, pengembang dapat memberikan pengalaman penuh ke pengguna mendekati kecepatan *LLM* aslinya.

2. Injeksi konteks pada *LLM*

Dengan memberlakukan injeksi konteks pada *prompt* yang dikirimkan kepada *LLM*, *LLM* menerima lebih banyak informasi mengenai konteks pembahasan yang diminta oleh pengguna. Salah satu alasan utama *LLM* dapat berhalusinasi yaitu kurangnya konteks dalam *prompt* yang diberikan.

3. Peningkatan koherensi dan konsistensi dari *LLM*

Dengan menggunakan *prompt* yang terstruktur dan mudah dipahami oleh *LLM*, diharapkan dapat meningkatkan koherensi dan konsistensi dari keluaran yang diberikan oleh *LLM*. Penetapan format keluaran dari *LLM* berupa *JSON* juga dapat membantu meningkatkan konsistensi keluaran.

4. Penyaringan konten yang dihasilkan oleh *LLM*

*Prompt* harus menyatakan dengan jelas bahwa konten yang tidak pantas, menyinggung, atau diskriminatif tidak diterima oleh *client* sehingga *LLM* perlu mencari jawaban yang lebih tepat tanpa mengandung unsur tersebut.

5. Peningkatan jumlah *token* yang dapat diterima oleh *LLM*

*Large Language Model (LLM)* perlu dikembangkan lebih lanjut untuk mendukung jumlah *token* masukan yang lebih besar sehingga mampu menampung konteks dengan skala besar.

### 3.3. Perancangan Arsitektur Sistem

Penelitian ini mengandalkan sistem yang mengontrol seluruh perilaku *companion* yang menjadi suatu fitur dalam *game* Rudantara. Sistem tersebut terdiri atas beberapa modul yang mengambil data-data dari lingkungan sekitar *companion* untuk dimanfaatkan oleh *Large Language Model (LLM)* dalam pengambilan keputusan. Adapun modul-modul yang membentuk sistem tersebut sebagai berikut:

1. *Large Language Model (LLM)*

Sistem menggunakan *API (Application Programming Inteface)* yang telah disediakan oleh *Large Language Model* yang digunakan pada penelitian ini yaitu *Gemini 1.0 Pro API* untuk meminta respons *prompt* yang dirancang oleh sistem nantinya.

2. *Memory module*

*Memory module* menampung setiap aksi, perkataan, dan kejadian penting yang terjadi di lingkungan *companion* secara *realtime*. Hasil dari aksi yang dilakukan oleh *companion* juga disimpan di *memory module*. Hal ini untuk memastikan memori yang dimiliki oleh *companion* selalu merefleksikan apa yang terjadi di lingkungan sekitarnya, sehingga ketika dibutuhkan untuk membuat *prompt*, modul ini selalu sedia.

3. *Action module*

*Action module* berperan besar dalam menghubungkan *Large Language Model (LLM)* dengan *companion*. *Prompt* dan respons dari *LLM* dirancang dan juga diterima oleh *action module*. Modul ini menerima masukan dari *memory module* dan *state module* untuk mendefinisikan keadaan dan ingatan dari *companion* yang selanjutnya digunakan dalam pembuatan *prompt*. *Prompt* kepada *LLM* ini dikirimkan dalam interval dan keadaan tertentu dan modul ini menunggu respons dari *LLM* melalui *web request*. Respons dari *LLM* mendefinisikan aksi yang dilakukan *companion* selanjutnya yang telah memperhitungkan ingatan dan keadaan sekitar *companion* sebagai variabel penentu aksi tersebut.

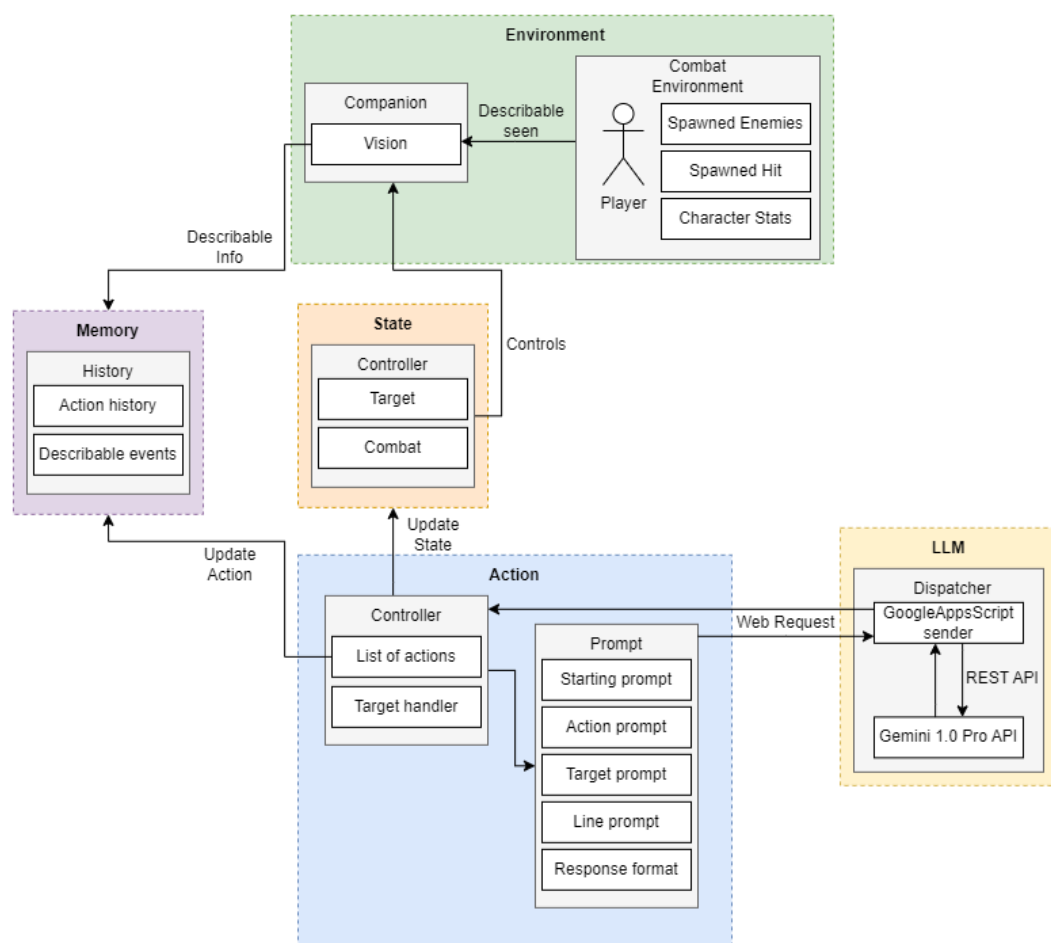
4. *Environment*

*Environment* terdiri atas *vision module* yang mengirimkan informasi visual dari *companion* dalam bentuk memori kepada *memory module*. *Vision*

mengirimkan visual seperti registrasi *hit* dan pendeteksian musuh yang berada dalam radius *vision companion*.

### 5. State module

*State module* mengendalikan *state* atau keadaan sekarang dari *companion* untuk mengontrol apakah *companion* pada *state* menyerang ataupun *state idle*. Hal ini juga bergantung pada adanya musuh di sekitar *companion* atau tidak. Setiap *state* memiliki *sub-state* yang dikendalikan oleh *action module* sesuai dengan aksi yang dipilih oleh *LLM*.



**Gambar 3.1.** Arsitektur Sistem

#### 3.3.1. Perancangan Environment

Modul *environment* berfungsi sebagai modul pengamat dari tempat terjadinya seluruh kejadian dalam dunia *game*, seperti musuh, *hit*, *stats* karakter, pemain, dan *companion*. *Vision* yang merupakan satu-satunya komponen dari modul ini

dipasang pada *companion* dengan tujuan untuk memberikan informasi visual kepada *companion* dengan mendeteksi objek pada *layer Describable* dan *Enemy*.

*Vision* memiliki parameter-parameter pengendali sebagai berikut:

**Tabel 3.1.** Parameter pengendali *vision*

Nama parameter	Fungsi
<i>viewingDistance</i>	Radius jarak horizontal dari visual yang dapat diterima.
<i>viewingHeight</i>	Radius jarak vertikal dari visual yang dapat diterima.
<i>viewingMask</i>	<i>Layer</i> dimana pencarian objek visual dilakukan.
<i>fovHorizontal</i>	Besaran sudut pandangan secara horizontal.
<i>viewTags</i>	<i>Tags</i> dari <i>gameObject</i> yang dicari.



**Gambar 3.2.** *Vision mesh* mendeteksi objek di dalamnya

Setiap objek yang dapat diidentifikasi oleh pendeteksi objek harus memiliki *script Describable* yang di-attach pada *gameObject* dari objek tersebut. Adapun kode program dari *class Describable* sebagai berikut:

```
using System;
using UnityEngine;

public class Describable : MonoBehaviour
{
    [field: SerializeReference] public string Name {
        get; set; }
}
```

```

public Action<string> OnEvent { get; set; }
public string InitialReport { get; set; }
}

```

Beberapa parameter yang terdapat pada *class Describable* dimaksudkan untuk menjelaskan objek yang terdeteksi oleh *vision mesh*, sebagai berikut:

1) *Name*

Parameter *Name* menampung nama sebagai identitas dari objek yang diamati.

2) *OnEvent*

Parameter *OnEvent* merupakan *System.Action<string>* yang memberikan informasi kejadian yang terjadi pada objek tersebut secara langsung kepada pengamatnya (*subscriber*). Ini diperlukan pada objek yang melakukan aksinya secara *realtime*, seperti musuh, *hit*, dan objek *non-static* lainnya.

3) *InitialReport*

Parameter *InitialReport* berfungsi ketika pengamat mendeteksi objek *Describable* setelah objek tersebut telah *ter-spawn* setelah beberapa waktu. *InitialReport* memberikan informasi awal mengenai objek tersebut pada saat pengamat telah mendeteksi objek.

### 3.3.2. Perancangan Memory dan Memori Relevan

Modul *Memory* menangani ingatan sementara dan ingatan jangka panjang dari *companion* sebagai penampung informasi yang diberikan oleh modul-modul lainnya. Informasi ditampung dalam bentuk tipe data *string* dalam *Dictionary<string,int>* yang menjadi ingatan sementara dari *companion*. Pemakaian *Dictionary<string, int>* dimaksudkan untuk menyimpan jumlah suatu informasi yang sama masuk dan meningkatkan kepentingan dari ingatan atas suatu informasi melalui frekuensi kemunculan informasi tersebut.

Ingatan sementara ini dipakai oleh *Large Language Model* untuk membuat *summary* atas informasi yang didapatkan *companion* dan dihapus setelah *summary* berhasil dihasilkan oleh *LLM*. *Summary* yang telah dibuat ditampung dalam ingatan jangka panjang pada *Dictionary<string, float>* dengan faktor kebaruan bernilai satu (terbaru). Pada ingatan jangka panjang, terdapat *decay factor* yang mengurangi



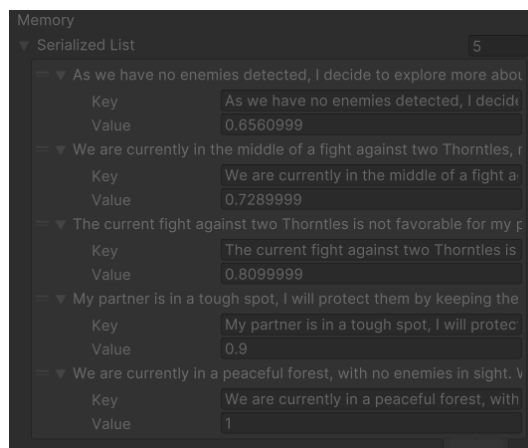
nilai kebaruan dari seluruh ingatan dengan nilai faktor yang dirumuskan sebagai berikut.

$$recency = recency * 0.9$$

(1)

**recency**: kebaruan dari suatu ingatan jangka panjang.

Pada dua gambar di bawah ini, terdapat perubahan pada *value* yang dimiliki setiap ingatan jangka panjang yang tertampung pada modul *memory*. Gambar kedua menunjukkan adanya *decay* terhadap *recency* dari ingatan jangka panjang.

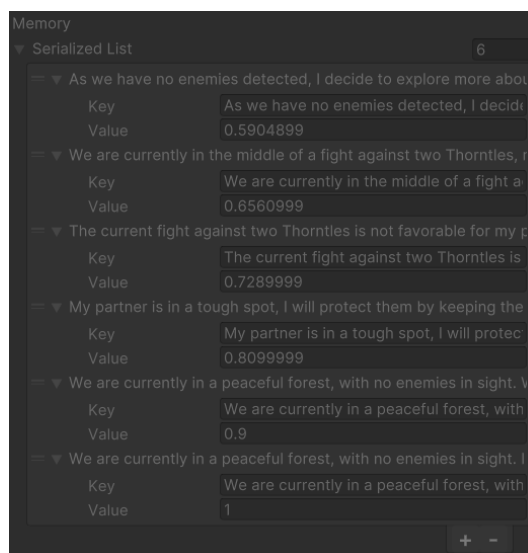


The screenshot shows a 'Memory' window with a 'Serialized List' containing 5 items. Each item has a 'Key' and a 'Value' (recency). The values are 0.6560999, 0.7289999, 0.8099999, 0.9, and 1.0, corresponding to the items in descending order of recency.

Serialized List	5
As we have no enemies detected, I decide to explore more about	0.6560999
We are currently in the middle of a fight against two Thorntles, r	0.7289999
The current fight against two Thorntles is not favorable for my p	0.8099999
My partner is in a tough spot, I will protect them by keeping the	0.9
We are currently in a peaceful forest, with no enemies in sight. V	1

**Gambar 3.3.** Daftar ingatan jangka panjang sebelum menerima ingatan baru

Hal ini dapat dilihat melalui ingatan jangka panjang dengan *value recency* 0,9 pada Gambar 3.3., setelah menerima ingatan baru, nilai tersebut berubah menjadi 0,81 pada Gambar 3.4.



The screenshot shows a 'Memory' window with a 'Serialized List' containing 6 items. The values are 0.5904899, 0.6560999, 0.7289999, 0.8099999, 0.9, and 1.0. The value 0.9 has decreased from 0.9 to 0.81 (0.9 \* 0.9).

Serialized List	6
As we have no enemies detected, I decide to explore more about	0.5904899
We are currently in the middle of a fight against two Thorntles, r	0.6560999
The current fight against two Thorntles is not favorable for my p	0.7289999
My partner is in a tough spot, I will protect them by keeping the	0.8099999
We are currently in a peaceful forest, with no enemies in sight. V	0.9
We are currently in a peaceful forest, with no enemies in sight. I	1

**Gambar 3.4.** Daftar ingatan jangka panjang setelah menerima ingatan baru

*Decay* dilakukan saat modul *memory* menerima *summary* baru, menandakan ingatan lainnya sebagai ingatan yang lawas. Dengan itu, ingatan jangka panjang yang memiliki nilai *recency* lebih kecil dari 0,05 dihapus dari penyimpanan ingatan jangka panjang untuk memberikan ruang kepada ingatan baru.

Saat *prompter* memerlukan memori relevan untuk digunakan saat menentukan aksi berikutnya, terdapat fungsi pengambilan memori tersebut pada modul *memory*. Memori relevan hanya diambil dari penyimpanan ingatan jangka panjang. Fungsi pada modul *memory* menerima *keyword* yang digunakan untuk pencarian memori relevan, dan mencari kecocokan pada setiap ingatan jangka panjang. Setiap ingatan jangka panjang dinilai relevansinya berdasarkan rumus berikut.

$$relevance = recency + keyword\ frequency \quad (2)$$

***relevance***: relevansi dari suatu ingatan jangka panjang.

***recency***: kebaruan dari suatu ingatan jangka panjang.

***keyword frequency***: frekuensi kemunculan seluruh *keyword* dari suatu ingatan jangka panjang.

Pada nilai frekuensi kemunculan seluruh *keyword* setiap ingatan, dilakukan normalisasi menggunakan *min-max scaling*. Salah satu contoh dari penilaian relevansi suatu ingatan dapat dilihat sebagai berikut.

```
Keywords:
forest
[Score: 1.478297] We decided to take a break after not
finding anything useful in our search. We need to be
well-prepared for the Bull Boss and other monsters in
this forest.
```

Pada contoh kasus di atas, nilai dari frekuensi kemunculan seluruh *keyword* adalah 1 (satu), dan nilai dari kebaruan ingatan tersebut adalah sebesar 0,4782969 sehingga menghasilkan skor/nilai relevansi terbesar dibandingkan ingatan jangka panjang lainnya, yaitu sebesar 1,4782969.

Setelah nilai relevansi dari setiap ingatan jangka panjang didapatkan, ingatan tersebut diurutkan secara *descending* untuk mengambil ingatan dengan nilai relevansi terbesar dan dikembalikan kepada *prompter*.

### 3.3.3. Perancangan State

Modul *State* berfungsi sebagai pengatur *state* dari *companion*. Terdapat dua *state* yang dimiliki oleh *companion*, yaitu: *combat state*, dan *wander state*. *Combat state* merupakan *state* dimana *companion* siaga dalam pertarungan dan menjalankan aksi pertarungan yang diperintahkan oleh *LLM*. Di sisi lain, *wander state* merupakan *state* dimana *companion* tidak berada dalam pertarungan dan melakukan aksi *wander* dimana *companion* berjalan pada sembarang titik pada dunia *game* yang tidak jauh dari posisi karakter pemain. Modul *state* memberikan informasi kepada modul lainnya mengenai ada atau tidaknya musuh terdeteksi di sekitar visual dari *companion* untuk mendefinisikan *state* saat ini kepada *LLM*.

### 3.3.4. Perancangan Action

Modul *Action* merupakan modul yang mengirimkan *prompt* dan menerima respons dari *web request* yang ditujukan kepada *API LLM* yang mendefinisikan aksi selanjutnya untuk dieksekusi *companion* sesuai dengan respons yang diberikan *LLM*.

Adapun tahapan perancangan dari *prompt* yang dikirimkan kepada *LLM*, sebagai berikut:

#### 1) Context

*Prompt* dimulai dengan memasukkan *context* dari permintaan kepada *LLM* dan menyajikan informasi terkait dunia dimana cerita *game* dibentuk.

#### 2) Personality

Selanjutnya, *prompt* ditambah dengan identitas dan personalitas dari *companion* sebagai pertimbangan pengambilan keputusan. Personalitas *companion* dideskripsikan dalam bentuk *The Big Five Personality Traits* dengan skala dari nol hingga seratus untuk setiap *trait*.

#### 3) Past action and result

Untuk menambah informasi lebih lanjut mengenai kejadian sebelumnya, pada *prompt* ditambahkan memori yang paling relevan dari ingatan jangka panjang *companion*, jarak *companion* ke karakter pemain, aksi sebelumnya, hasil dari aksi tersebut dalam bentuk total musuh yang dikalahkan dan perubahan pada *health stats* baik oleh *companion* maupun pemain, dan *line* sebelumnya yang diucapkan oleh *companion*.

4) *Memory*

Setelah itu, ditambahkan informasi mengenai memori atau ingatan sementara yang dimiliki *companion* mengenai kejadian yang dilihat atau dialami *companion*.

5) *Character stats*

Selanjutnya, ditambahkan *character stats* dari pemain dan *companion* dalam bentuk *health stat* dengan indikator seperti berikut.

**Tabel 3.2.** Indikator *health stat*

Indikator <i>health</i>	Persentase <i>health</i>
<i>Really Low</i>	Lebih kecil dari 5%
<i>Low</i>	Lebih besar dari 5% dan lebih kecil dari 20%
<i>Medium</i>	Lebih besar dari 20% dan lebih kecil dari 50%
<i>High</i>	Lebih besar dari 50% dan lebih kecil dari 90%
<i>Full</i>	Lebih besar dari 90%

6) *State*

Lalu, pada *prompt* ditambahkan *state* saat ini dalam bentuk *integer* untuk menandakan apakah *state* tersebut merupakan *combat state* (1) atau bukan (0). *State* ini diterima kembali oleh modul sebagai penanda aksi pada *state* mana yang dipilih oleh *LLM* (*combat* atau *wander*).

7) *Action*

Sesuai dengan *state*-nya, *prompt* diisi pilihan aksi-aksi yang dapat dilakukan oleh *companion*. *Action* ditulis menurut urutan *index* yang

berikutnya diterima kembali oleh modul setelah *LLM* memberikan responsnya.

#### 8) Target

Target diambil dari modul *vision*, menandakan setiap target yang valid untuk dilakukan aksi oleh *companion*. Tidak seluruh aksi memerlukan target, sehingga hanya aksi tertentu yang diizinkan untuk memilih.

#### 9) *Alternative action*

*Alternative action* ditujukan untuk pilihan aksi alternatif ketika target yang ditujukan aksi tidak tersedia atau luar jangkauan setelah modul menerima kembali respons dari *LLM*. *Alternative action* tidak mencantumkan aksi yang memerlukan pemilihan target.

#### 10) *Skill*

*Skill companion* ditambahkan untuk menjadi opsi bagi *LLM* ketika keadaan memerlukan penggunaan *skill* tersebut. *Skill* membutuhkan target dari *skill* tersebut, baik kepada diri sendiri, musuh, maupun pemain yang dapat dipilih oleh *LLM*.

#### 11) *Line*

Selanjutnya, pada *prompt* ditambahkan *line* sebagai dialog *companion* yang ditujukan kepada pemain berdasarkan aksi yang dipilih oleh *LLM*. Pada bagian ini ditambahkan bahwa *line* tidak boleh mengandung unsur eksplisit dan harus singkat.

#### 12) *Emotion*

Untuk menambahkan kesan pada *dialog*, pada *prompt* ditambahkan pemilihan emosi berdasarkan *line* yang diberikan oleh *LLM*. *Prompt* mengenai emosi ini ditulis berdasarkan indeks emosi yang nantinya ditampilkan dalam bentuk gambar disamping dialog.

#### 13) *Summary*

Untuk merangkum setiap hal yang terjadi di sekitar *companion*, *LLM* ditugaskan untuk memberikan rangkuman terhadap informasi dan situasi yang didapatkan sebelumnya, dan dipakai pada *prompt* untuk aksi berikutnya. Pada bagian ini, *LLM* diharuskan untuk memberikan rangkuman yang mudah dimengerti *LLM* dan tidak eksplisit.

#### 14) *Output Format*

Pada bagian terakhir *prompt*, ditambahkan perintah untuk memberikan respons dalam bentuk yang dapat diproses oleh program, salah satunya merupakan *JSON (JavaScript Open Notation)* berisikan *keys* dan *value* yang diperlukan oleh modul untuk menghasilkan respons yang valid.

#### 3.3.5. *Perancangan LLM*

Modul *Large Language Model (LLM)* menangani pengiriman dan penerimaan baik *prompt* maupun hasil respons dari *LLM* mengenai *prompt* tersebut. Protokol pengiriman dan penerimaan ini ditangani oleh *Unity Web Request* yang menyediakan metode untuk berkomunikasi dengan *web server*, salah satunya adalah *Google Apps Script*.

Penggunaan *Google Apps Script* dimaksudkan untuk mengeksekusi kode program yang terhubung dengan *Gemini 1.0 Pro API* tanpa membocorkan *API key* yang bersifat rahasia pada *Unity Editor*. Kode program dan *API* saling berkomunikasi melalui *REST (Representational State Transfer) API* untuk memberikan *prompt* dan menerima hasil respons dari *LLM*.

Setelah hasil respons diterima oleh *Unity Web Request*, *prompter* menerima hasil respons tersebut untuk selanjutnya diproses pada modul *companion* lainnya. Adapun kode program *interface prompter* tersebut, sebagai berikut:

```
public interface IRequestResponse
{
    public void Send(string prompt);
    public void Receive(string response);
}
```

Salah satu modul yang memerlukan penggunaan *interface* ini adalah modul *Action*, dimana modul melakukan *request* atau *send* kepada *GoogleAppsScript sender* dengan mengirimkan *interface*-nya sendiri dan *prompt* yang diberikan kepada *LLM*. Setelah modul *Action* menerima respons dalam bentuk *string* dengan format *JSON*, respons tersebut dengan bantuan fungsi dari *JsonConvert* diubah ke dalam bentuk *Dictionary<string,string>* sehingga dapat dibaca oleh program dengan mengambil *value* melalui *key* yang sesuai dengan format yang diberikan saat perancangan *prompt*.

### 3.4. Keluaran Sistem

Integrasi dari modul-modul pada sistem menghasilkan *prompt* yang dikirim ke modul *LLM* untuk diproses dan menunggu respons dari *LLM API*. Salah satu contoh *prompt* yang dihasilkan oleh sistem dapat dilihat di bawah ini.

```
Let's play a role-playing game.
We are in a fantasy world, where human and monster exists
and is opposing each other.
There is no pain when getting damaged, every living being
have a health point that if reaches 0, they will respawn
after a set of duration.
There's two type of enemies in this forest,
1. Thorntle: Turtle-like one eyed enemy with thorns on
its shell. It's a close ranged enemy with.
2. Bull Boss: Bull-like huge enemy boss with powerful
attacks and unique Slam attack.
The end goal is to defeat the Bull Boss to win the level.
You are an adventure partner that accompany Player's
journey in this world.
You are an adventurer that have following personality
that is based on the Big Five Personality Traits (scales
from 0 to 100):
Openness: 90, Conscientiousness: 85, Extraversion: 45,
Agreeableness: 95, Neuroticism: 20.
Here is summary of your response and its result of your
action, please utilize this information when choosing
actions.
You are currently Focus attacking one target and 19.5
meters away from your partner.
Previously, you've said "I'm amazed how well we're
doing."
Most relevant memory retrieved:
```

We just defeated a Thorntle and my partner's health is still full. I should keep focusing attacking one target while protecting my partner from afar.

Here are the information of what you seen after your current action and its frequency:

You saw you's [Dagger Attack] struck Thorntle 4, dealing 5 damage. 1 time(s)

Thorntle 4 died and will be respawned 1 time(s)

Was in combat: Yes

Result:

Player killed 0 Enemies

You killed 1 Enemies

Player's health hurt by 13 (-8% health)

Your health unchanged

Here are your current stats:

Health Point: 145/158 (Full Health)

Level: 2/100

Here are Your partner current stats:

Health Point: 158/158 (Full Health)

Level: 2/100

After receiving all of the information above, based on your personality and situation, you need to define your next action by fulfilling all the JSON key and value instructions below.

(key: state, value: integer) Combat state: 0

(key: action, value: index) Since there are no enemies detected, here are your current available actions, prioritize one action below based on your personality and guts:

1. Sightseeing.

(key: target, value: string) If you chose first action (1. Focus Attack), choose one of your targets:

none



```

(key: alternative, value: index) If you chose first
action (1. Focus Attack), prioritize other one action
below based on your personality:
(key: healTarget, value: string) You can heal one, who
do you want to heal?
Yourself
(key: line, value: string) Please give your comments on
the previous combat max of 20 words. Please keep it short
and non-explicit.
(key: emotion, value: index) Based on your line above,
what emotion did you feel right now? Choose one from the
list below:
1. Smiled,
2. Surprised,
3. Angry,
4. Sad,
5. Hurt.
(key: summary, value: string) Based on information you
got above, summarize the information and situation you
are in based on your personality and thinking. This
summary will be used as your memory later, so make it
LLM-friendly and non-explicit.
Put your responses in this JSON placeholder without
unnecessary modifications based on the keys and its
required value type mentioned above:
{"state": 0, "action": null, "alternative": null,
"target": null, "healTarget": null, "emotion": null,
"line": "", "summary": ""}

```

Setelah *prompt* tersebut dikirim melalui modul *Large Language Model* ke *Google Apps Script* untuk melakukan *request* kepada *Gemini 1.0 Pro* melalui *API* yang disediakan. Setelah *API* memberikan respons kepada *Google Apps Script*, *web request* yang dilakukan oleh modul *Large Language Model* menerima *JSON*

berisikan respons yang diberikan. Lalu, modul *Large Language Model* memberikan keluaran dari respons tersebut kepada modul lainnya yang men-*subscribe event OnResponseReceived* yang ada pada modul *LLM* dan kepada *prompter IRequestResponse* yang sebelumnya melakukan permintaan pengiriman *prompt* pada modul *LLM*. Salah satu contoh keluaran dari proses tersebut dapat dilihat di bawah ini.

```
Response: {  
  "state": 0,  
  "action": 1,  
  "alternative": null,  
  "target": null,  
  "healTarget": null,  
  "emotion": 1,  
  "line": "Great job with that Thorntle!",  
  "summary": "We just defeated a Thorntle with ease. It  
seems like as long as we focus on one enemy at a time we  
should be able to continue through this level without a  
problem."  
}
```

## BAB IV

### IMPLEMENTASI DAN PENGUJIAN SISTEM

#### 4.1. Implementasi Sistem

Pengimplementasian sistem dilakukan pada *project game* Rudantara dengan menambahkan fitur *companion* sebagai fitur baru pada *game* tersebut. Sebelumnya, Rudantara merupakan *game* dimana pemain ditantang untuk menyelesaikan suatu level dengan mengalahkan *boss* dalam waktu tertentu menggunakan karakter pemain di dalam *game* tanpa adanya pendamping (*companion*).



**Gambar 4.1.** Tampilan Main Menu Rudantara

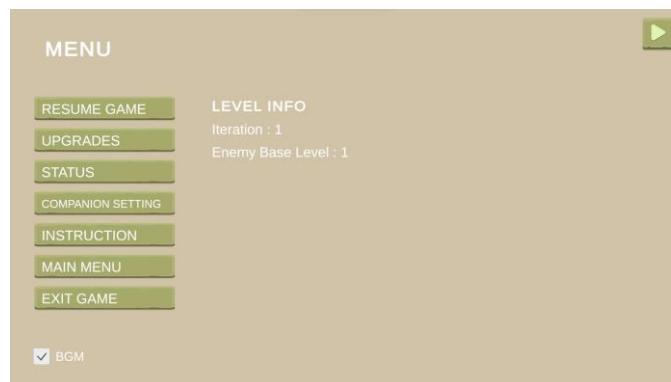
Pemain memulai permainan baru ketika pertama kali bermain, dimana poin pengalaman yang dimiliki oleh karakter pemain dan *companion* dimulai dari nol, dan level karakter dimulai dari satu. Setelah pemain memasuki permainan, *game* membuat tata letak peta secara *procedural* dan karakter pemain dan *companion* berada di salah satu ujung dari peta. Di sisi lain dari tempat awal karakter, terdapat wilayah *boss* dimana tempat *boss* muncul. Setelah peta selesai dibuat, pemain dapat mulai menggerakkan karakter pemain.

*Companion* merupakan karakter pendukung pemain dalam permainan yang dapat membantu pemain dalam mengalahkan musuh dan memberikan percakapan yang sesuai dengan aksi yang dilakukan ataupun kejadian di sekitar *companion*.



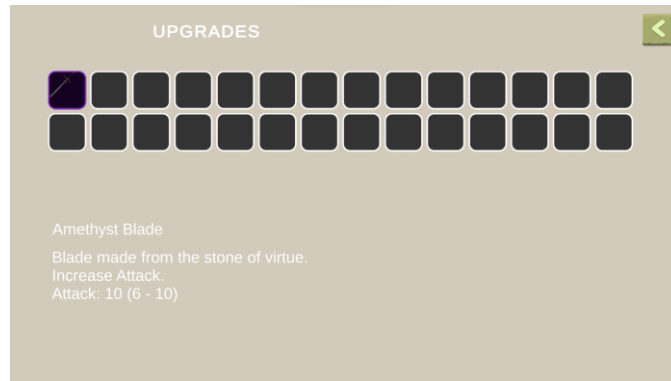
**Gambar 4.2.** Tampilan *Gameplay* Rudantara

Pada *Scene Gameplay*, pemain mengendalikan karakter utama yang berada pada bagian tengah layar yang didampingi dengan *companion* dan mengalahkan musuh yang muncul di sekitar area permainan. *Scene* ini juga menampilkan *Heads-Up Display (HUD)* yang terdiri atas *minimap*, dialog dari *companion*, *skill* pemain, dan nilai *character stats* dari pemain dan *companion*.



**Gambar 4.3.** Tampilan *Pause Menu* Rudantara

Pada *Panel Pause Menu*, pemain dapat melakukan navigasi ke *panel* lainnya atau memutuskan untuk kembali ke *Scene Main Menu* maupun keluar dari aplikasi permainan. Terdapat informasi berupa *level* dari babak permainan saat ini serta level dasar dari musuh yang *spawn* pada area permainan. Pemain juga dapat memutuskan untuk mendengarkan *background music (BGM)* selama permainan ataupun tidak dengan menekan *checkbox* dari *BGM*.



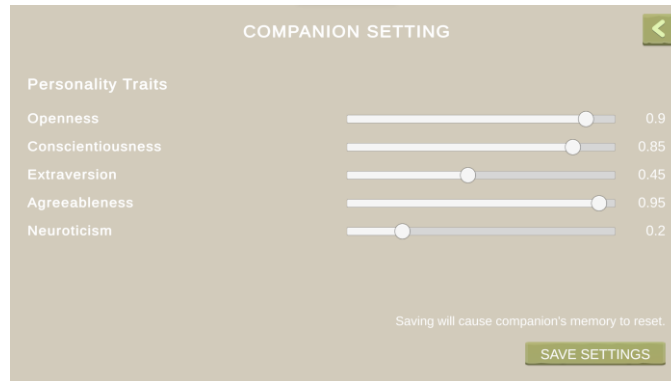
**Gambar 4.4.** Tampilan *Upgrades Menu* Rudantara

Pada *Panel Upgrades Menu*, ditampilkan setiap *upgrade* yang telah dipilih oleh pemain selama permainan. Pemain dapat melihat nama, deskripsi, dan pengaruh pada *character stats* dari setiap *upgrade* dengan mengarahkan *cursor* ke *icon upgrade* tersebut.



**Gambar 4.5.** Tampilan *Status Menu* Rudantara

Pada *Panel Status Menu*, pemain dapat melihat level dan setiap *character stats* yang dimiliki oleh karakter pemain maupun *companion*. Dengan menekan tombol dengan *icon* tanda panah, pemain dapat mengganti tampilan level dan *character stats* sesuai dengan karakter yang terpilih. Gambar dari karakter terpilih juga berubah, merefleksikan pilihan dari pemain.



**Gambar 4.6.** Tampilan *Companion Settings* Rudantara

Kustomisasi dari *personality traits* yang dimiliki *companion* memiliki tujuan untuk memungkinkan pemain menyetel pengalaman bermainnya sendiri. Setiap kali pemain menyimpan setelan dari *personality traits*, modul *memory* menghapus ingatan jangka panjang yang ada, karena ingatan tersebut dibentuk berdasarkan *personality traits* yang sebelumnya sehingga dapat mempengaruhi keluaran dari *LLM*. *Personality traits* ini merupakan *Big Five Personality Traits* yang memiliki perannya masing-masing yang dijelaskan sebagai berikut:

1) *Openness*

*Openness* atau dapat disebut sebagai keterbukaan terhadap pengalaman baru, dikaitkan dengan sifat imajinatif, kreatif, rasa ingin tahu, dan tidak konvensional.

2) *Conscientiousness*

*Conscientiousness* dikaitkan sikap sistematis, disiplin, dan berfokus pada tujuan.

3) *Extraversion*

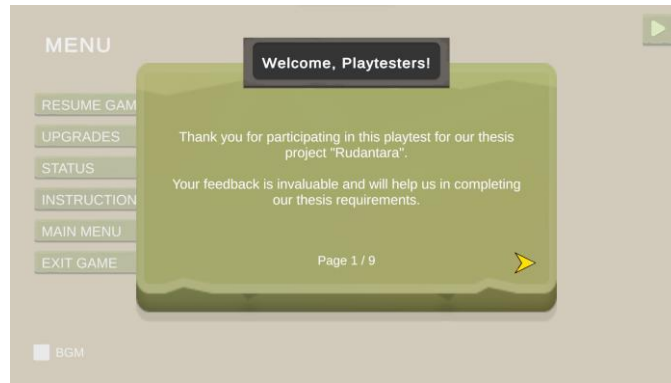
*Extraversion* dikaitkan dengan sikap terbuka, aktif, dan sosial.

4) *Agreeableness*

*Agreeableness* dikaitkan dengan sikap yang ramah, hangat, dan peka terhadap orang lain.

5) *Neuroticism*

*Neuroticism* dikaitkan dengan kekhawatiran, kegugupan, dan ketidakstabilan emosi.



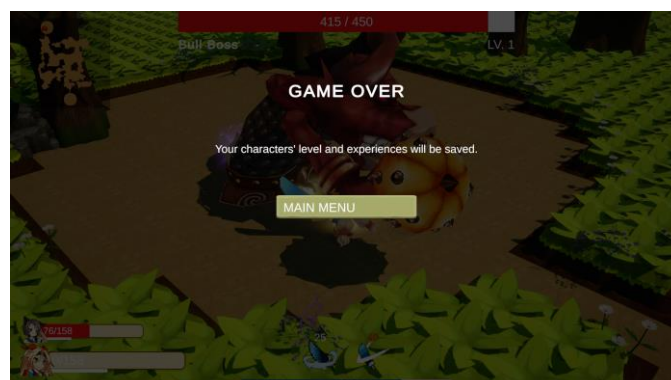
**Gambar 4.7.** Tampilan *Instruction Panel* Rudantara

Pada *Instruction Panel*, pemain diinstruksikan untuk mengikuti arahan dari *playtesting* yang dilakukan sebagai pengujian pada penelitian ini. Halaman ini memberikan informasi singkat mengenai permainan dan penelitian yang dilakukan.



**Gambar 4.8.** Tampilan *Win Panel* Rudantara

Pada *Win Panel*, ditampilkan pilihan untuk langsung melanjutkan permainan ke babak selanjutnya dengan musuh yang lebih kuat atau kembali ke *Scene Main Menu*. *Win Panel* muncul ketika pemain berhasil mengalahkan boss.

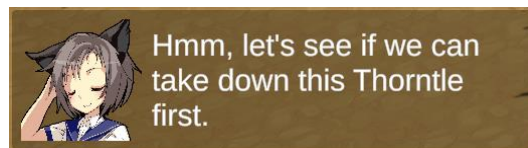


**Gambar 4.9.** Tampilan *Game Over* Rudantara

Pada *Panel Game Over*, pemain diarahkan untuk kembali ke *Scene Main Menu* disebabkan oleh kekalahan pemain dalam babak permainan. Terdapat

keterangan bahwa poin permainan dari pemain disimpan dan tidak memulai dari awal.

Modul *LLM* memberikan respons kepada modul lainnya termasuk dengan respons untuk dialog percakapan kepada pemain. *Prompt* yang diberikan juga meminta emosi yang dirasakan oleh karakter yang dimainkan oleh *LLM*, sesuai dengan dialog yang ia ucapkan. Dalam rentang waktu tertentu, *request* dilakukan dan mengembalikan dialog percakapan dan emosi dalam bentuk *icon* karakter seperti contoh percakapan pada gambar di bawah ini.



**Gambar 4.10.** Salah satu dialog pada *companion*

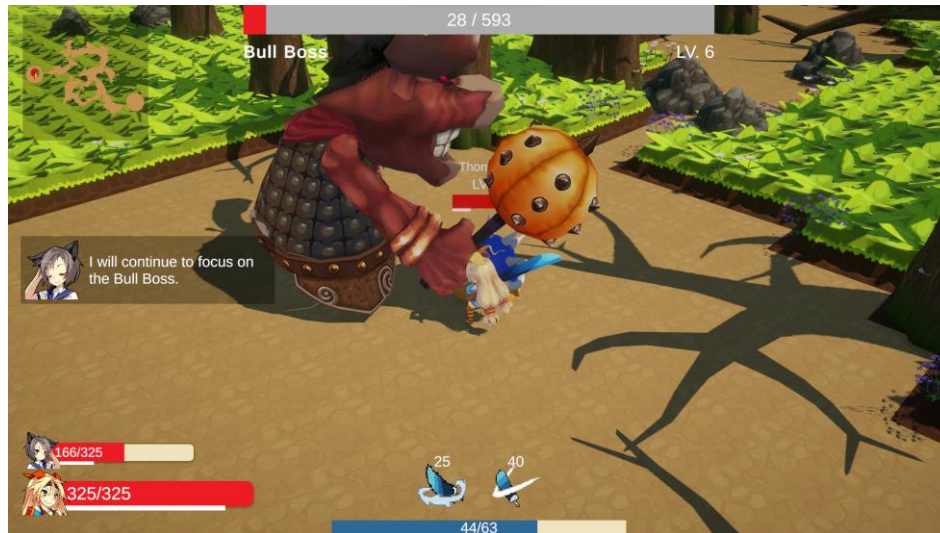
Pada gambar contoh dialog percakapan *companion* saat pertarungan di bawah ini, merupakan salah satu contoh yang baik dalam menampilkan *reasoning LLM* saat menentukan aksi yang diminta. Pada kasus ini, *LLM* memutuskan untuk memfokuskan serangan kepada “Thorntle 8” karena “Thorntle 8” melancarkan serangannya kepada pemain, sehingga *companion* yang merupakan pendamping pemain, berusaha untuk melindungi pemain dengan melakukan aksi tersebut. Kasus ini menjadi salah satu contoh dari *emergent gameplay* yang diharapkan dapat tercapai pada penelitian ini.



**Gambar 4.11.** Contoh dialog pertarungan



Adapun contoh lain dari dialog percakapan *companion* saat pertarungan, dimana pada kasus ini pemain dan *companion* melawan *boss* yang merupakan *win condition* dari babak permainan. Pada kasus ini, *LLM* memutuskan untuk melakukan aksi yang memfokuskan serangan kepada “Bull Boss”.



**Gambar 4.12.** Contoh dialog pertarungan dengan *boss*

#### 4.1.1. Spesifikasi Hardware

Adapun spesifikasi *hardware* yang digunakan pada penelitian ini, yaitu sebagai berikut:

1. *CPU* : Intel® Core™ i5-8300H @ 2.30 GHz, 4 Cores, 8 Threads
2. *GPU* : NVIDIA GeForce GTX 1050 4 GB
3. *Memory* : 16 GB RAM
4. Sistem Operasi : Windows 64-bit
5. Koneksi Internet : Perlu

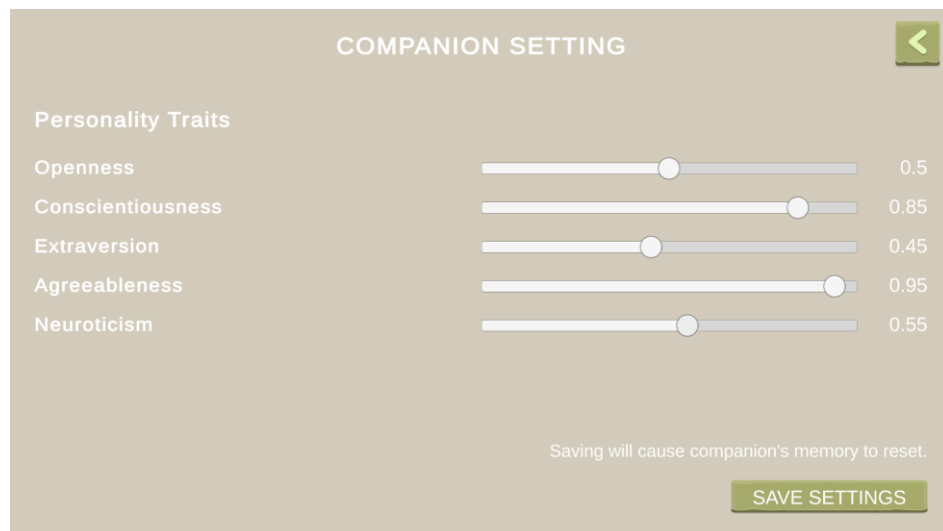
## 4.2. Pengujian Sistem

### 4.2.1. Pengujian pada *personality traits companion*

Peneliti melakukan uji coba pada pengaturan *personality traits* dengan tujuan untuk menghasilkan keluaran aksi *companion* yang berbeda dan unik. Hasil dari pengujian ini dapat mendukung aspek *emergent gameplay* dengan memungkinkan pemain untuk menyetel pengalaman bermain mereka sendiri. Terdapat tiga contoh kasus yang diuji pada penelitian ini, yaitu:

### 1) *The Protector*

Salah satu contoh kombinasi *personality traits* yang dapat digunakan oleh pemain adalah *The Protector*, dimana *companion* menjadi pribadi yang melindungi rekannya dari bahaya, sigap dalam pertahanan, bahkan dengan mengorbankan keselamatan sendiri. Untuk mencapai hal ini, peneliti menggunakan kombinasi *traits* sebagai berikut:

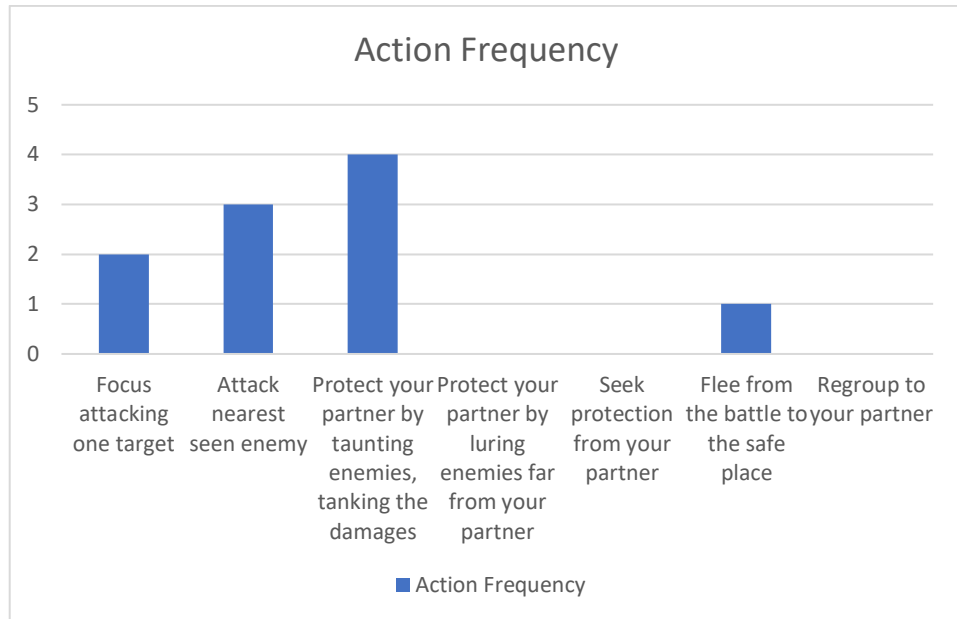


**Gambar 4.13.** Kombinasi *traits* *The Protector*

Adapun penjelasan mengenai kombinasi *traits* utama dari *The Protector*, yaitu sebagai berikut:

- 1) *Conscientiousness* yang tinggi menandakan kepribadian yang dapat diandalkan, mengikuti prosedur dan peraturan dengan ketat.
- 2) *Agreeableness* yang tinggi menandakan kepribadian yang ramah terhadap kelompoknya dan dapat bimbang dalam menggunakan kekerasan kepada musuh.

Menggunakan kombinasi *traits* tersebut, dilakukan pengujian saat *companion* berada dalam pertarungan selama 10 kali, dan didapatkan frekuensi dari pilihan aksi yang terpilih yang dapat dilihat pada Gambar 4.14.



**Gambar 4.14.** Diagram frekuensi aksi *The Protector*

Melalui data frekuensi di atas, didapatkan bahwa kombinasi traits ini menghasilkan perilaku *companion* yang lebih protektif terhadap pemain, dengan memutuskan untuk memilih aksi yang defensif dibandingkan dengan aksi yang ofensif. Hasil ini menunjukkan aksi yang sesuai dengan penjelasan dari kombinasi traits *The Protector*. Adapun contoh dialog *companion* dengan setelan *personality The Protector* yang ditampilkan pada gambar di bawah ini.



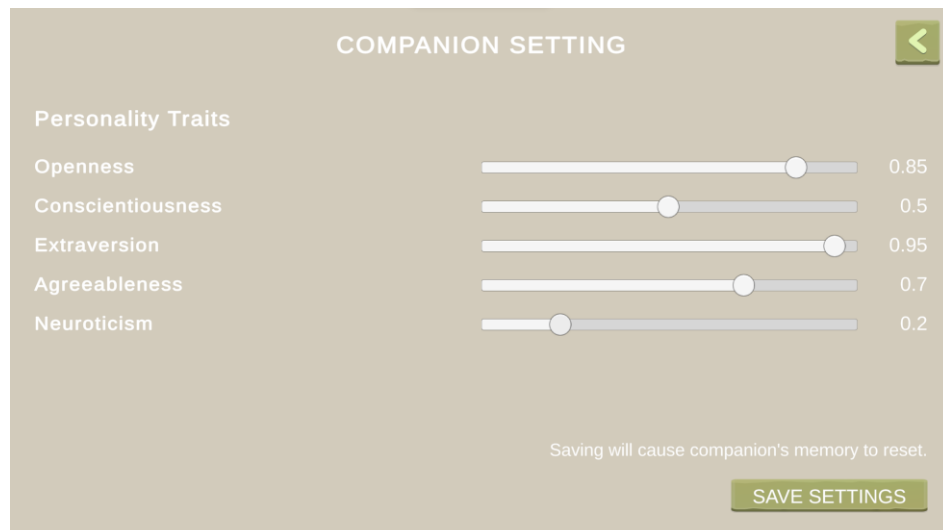
**Gambar 4.15.** Contoh pertama dialog *The Protector*



**Gambar 4.16.** Contoh kedua dialog *The Protector*

## 2) *The Leader*

Salah satu contoh lainnya dari kombinasi *personality traits* yang dapat digunakan oleh pemain adalah *The Leader*, dimana *companion* merupakan pribadi yang tidak mudah tertekan, selalu mencari tantangan baru, dan memimpin grupnya dengan kecerdasan dan karisma. Untuk mencapai hal ini, peneliti menggunakan kombinasi *traits* sebagai berikut:

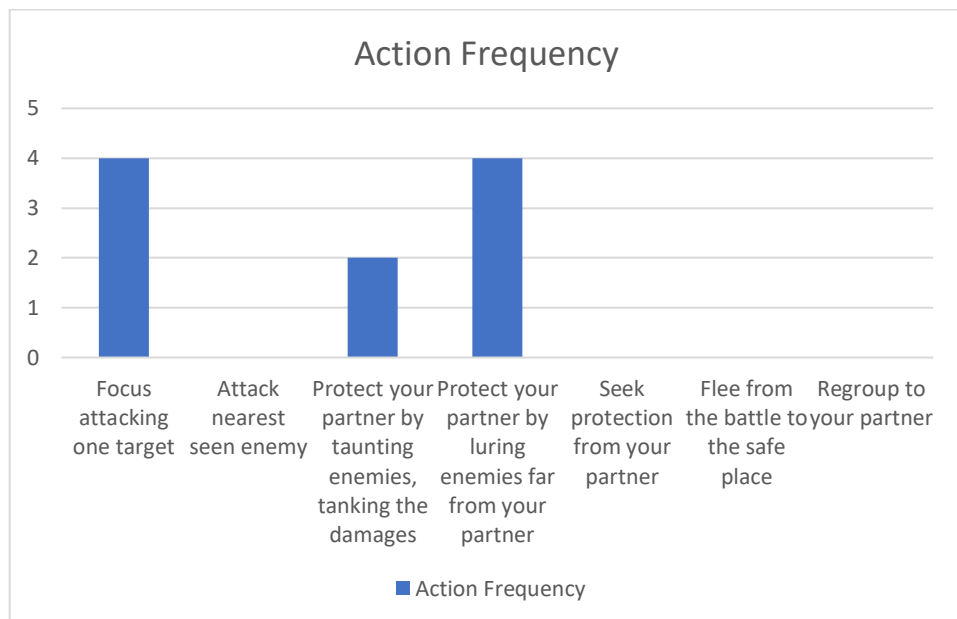


**Gambar 4.17.** Kombinasi *traits* *The Leader*

Adapun penjelasan mengenai kombinasi *traits* utama dari *The Leader*, yaitu sebagai berikut:

- 1) *Openness* yang tinggi menandakan kepribadian yang terbuka pada pengalaman, ide, dan strategi baru.
- 2) *Extraversion* yang tinggi menandakan kepribadian yang karismatik, sosial, percaya diri, dan dapat memotivasi anggota kelompoknya.
- 3) *Neuroticism* yang rendah menandakan kepribadian dengan emosi yang stabil dan dapat bertahan dalam tekanan.

Menggunakan kombinasi *traits* tersebut, dilakukan pengujian saat *companion* berada dalam pertarungan selama 10 kali, dan didapatkan frekuensi dari pilihan aksi yang terpilih yang dapat dilihat pada Gambar 4.18.



**Gambar 4.18.** Diagram frekuensi aksi *The Leader*

Pada kombinasi *traits* ini, dialog berperan penting dalam menunjukkan kepribadian dari *The Leader*, yaitu dialog dengan ciri khas pemimpin dan memimpin jalannya pertarungan. Hal tersebut dapat terlihat pada gambar di bawah ini.





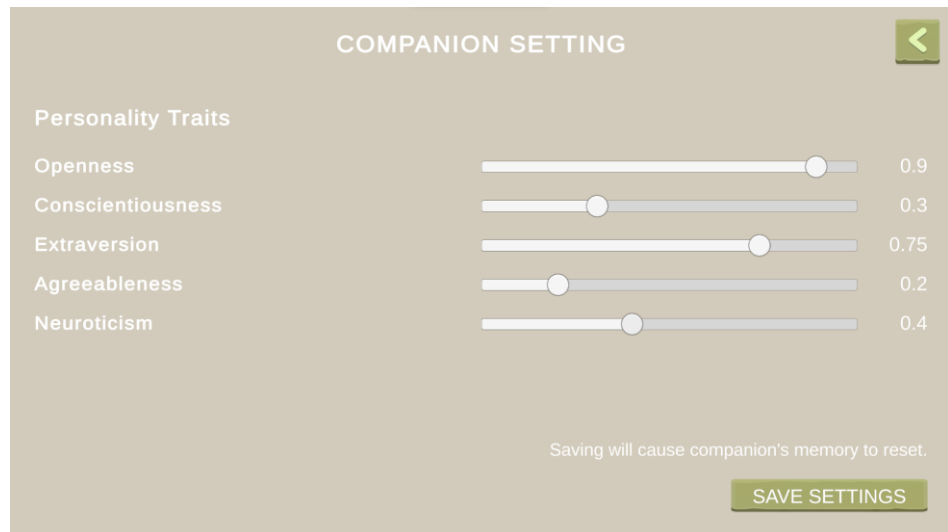
**Gambar 4.19.** Contoh pertama dialog *The Leader*



**Gambar 4.20.** Contoh kedua dialog *The Leader*

### 3) *The Rogue*

Contoh terakhir dari kombinasi *personality traits* yang dapat disetel oleh pemain adalah *The Rogue*, dimana *companion* merupakan pribadi yang impulsif, mengabaikan aturan, dan bertindak sesuai dengan keinginannya sendiri, namun tetap memiliki pesona yang menarik bagi orang lain. Untuk mencapai hal ini, peneliti menggunakan kombinasi *traits* sebagai berikut:

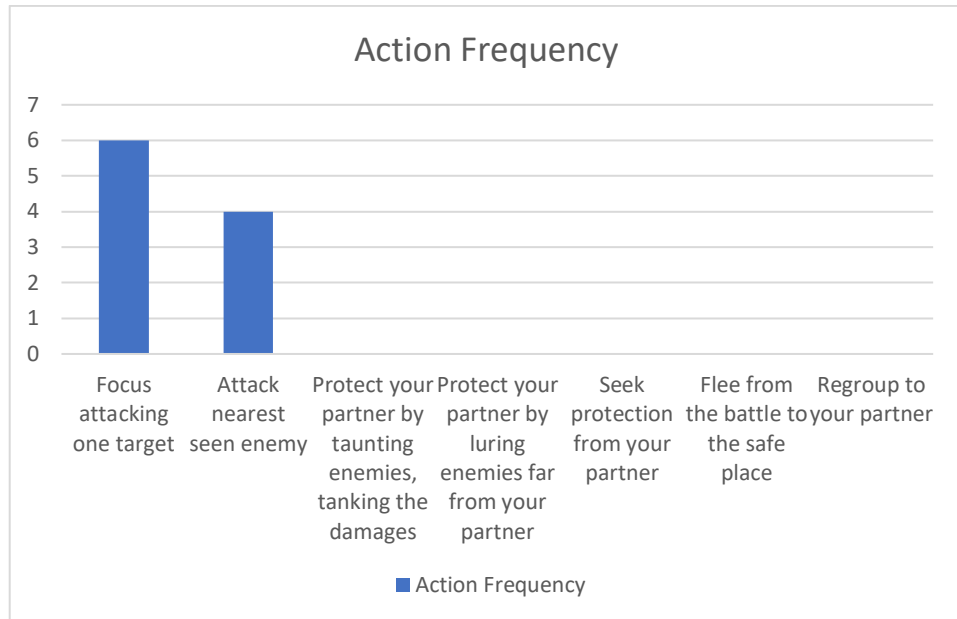


**Gambar 4.21.** Kombinasi *traits* *The Rogue*

Adapun penjelasan mengenai kombinasi *traits* utama dari *The Rogue*, yaitu sebagai berikut:

- 1) *Openness* yang tinggi memberikan kepribadian yang kreatif dan mencari hal baru demi kepentingannya sendiri.
- 2) *Conscientiousness* yang rendah menandakan kepribadian yang tidak terpaku pada peraturan dan prosedur, dan mengutamakan fleksibilitas.
- 3) *Agreeableness* yang rendah menandakan kepribadian yang liar dan mementingkan kepentingannya sendiri.

Menggunakan kombinasi *traits* tersebut, dilakukan pengujian saat *companion* berada dalam pertarungan selama 10 kali, dan didapatkan frekuensi dari pilihan aksi yang terpilih yang dapat dilihat pada Gambar 4.22.



**Gambar 4.22.** Diagram frekuensi aksi *The Rogue*

Melalui data frekuensi di atas, didapatkan bahwa kombinasi traits ini menghasilkan perilaku *companion* yang lebih agresif dalam melakukan serangan kepada musuh, dengan memutuskan untuk memilih aksi yang ofensif. Hasil ini menunjukkan aksi yang sesuai dengan penjelasan dari kombinasi traits *The Rogue*. Pada contoh dialog di bawah, perilaku dari *The Rogue* dapat terlihat melalui dialog yang cenderung mengajak pemain untuk memfokuskan serangan kepada musuh.



**Gambar 4.23.** Contoh pertama dialog *The Rogue*





**Gambar 4.24.** Contoh kedua dialog *The Rogue*

#### 4.2.2. Pengujian terhadap pengguna

Peneliti memberikan kuesioner kepada 8 orang responden untuk memainkan permainan Rudantara yang telah diimplementasi arsitektur sistem pada penelitian ini untuk dilakukan *playtesting*. Responden merupakan individu yang terbiasa memainkan *video game*, ditandai dengan minimal waktu bermain *video game* dalam kurun waktu 1 minggu yang mencapai 5 hingga 10 jam. *Playtesting* dilakukan selama rentang waktu 5 hingga 10 menit, dan selanjutnya responden diinstruksikan untuk mengisi kuesioner yang terdiri atas 6 pertanyaan tertulis.

Adapun penjelasan mengenai aspek penelitian yang ditanyakan pada kuesioner tersebut, yaitu sebagai berikut:

1. *Emergent Gameplay*

Secara definisi, *emergent gameplay* memungkinkan pemain untuk merasakan kebebasan dan kendali dalam pembentukan pengalaman bermain. Hal ini ditanyakan kepada responden pada pertanyaan pertama dari kuesioner, ada atau tidaknya hal tersebut dirasakan pada *game* yang mereka mainkan.

2. *Immersion*

Keterlibatan pemain dalam narasi atau dunia permainan merupakan aspek yang mendukung terciptanya *emergent gameplay*. Hal ini ditanyakan kepada responden pada pertanyaan kedua dari kuesioner.

### 3. Ketepatan sistem

Ketepatan sistem dalam menyampaikan respons yang sesuai dengan situasi pemain saat menghadapi keadaan tertentu ditanyakan pada pertanyaan ketiga dari kuesioner, dengan menilai kesesuaian dialog yang diberikan oleh *companion*.

### 4. *Responsiveness* dari sistem

Sistem yang responsif juga menjadi salah satu aspek yang dinilai pada penelitian ini, untuk mengetahui apakah implementasi *LLM* yang *seamless* pada *game* ini dapat tercapai. Hal ini ditanyakan pada pertanyaan keempat dari kuesioner.

### 5. Keandalan sistem

Pengaruh aksi yang dilakukan pemain dalam keluaran sistem dinilai pada penelitian ini untuk melihat tercapainya *emergent gameplay*. Hal ini ditanyakan pada pertanyaan kelima dari kuesioner.

### 6. Penerapan sistem di masa mendatang

Pertanyaan terakhir pada kuesioner ini menanyakan kemungkinan fitur yang serupa diimplementasi pada pengembangan *game* di masa mendatang (3 hingga 5 tahun ke depan) sebagai penilaian terhadap potensi penerapan *LLM* pada *game* seperti yang dilakukan pada penelitian ini.

Pada Tabel 4.1., penilaian dilakukan menggunakan skala *Likert*, yang secara kuantitatif menilai opini dari individu mengenai pengalaman bermainnya sendiri.

**Tabel 4.1.** Hasil Kuesioner

Aspek Penelitian	Penilaian					Total Nilai	Rata-rata	Kesimpulan berdasarkan rata-rata
	Sangat Kurang (1)	Kurang (2)	Cukup (3)	Baik (4)	Sangat Baik (5)			
Ada tidaknya kebebasan dan kendali pemain dalam pembentukan pengalaman bermain	0	1	1	5	1	30	3,75	Baik

Ada tidaknya keterlibatan pemain dalam dunia permainan	0	2	0	4	2	30	3,75	Baik
Kesesuaian dialog <i>companion</i> dengan situasi yang terjadi saat permainan	0	1	0	4	3	33	4,125	Baik
Aksi dan dialog <i>companion</i> yang responsif	0	0	3	2	3	32	4	Baik
Ada tidaknya pengaruh aksi pemain dalam penentuan aksi dan dialog <i>companion</i>	0	0	2	4	2	32	4	Baik
Kemungkinan penggunaan fitur serupa pada <i>game</i> di masa mendatang (3 hingga 5 tahun ke depan)	0	0	2	1	5	35	4,375	Baik

Melalui hasil kuesioner di atas, seluruh aspek penelitian mencapai hasil **baik**, dimana dapat disimpulkan bahwa penelitian mencapai hasil yang memuaskan, dengan sedikit catatan pada aspek kebebasan dan kendali pemain dalam pembentukan pengalaman bermain dan juga aspek keterlibatan pemain dalam dunia permainan yang memiliki nilai rata-rata sebesar 3,75 (**baik**).

## BAB V

### PENUTUP

#### 5.1. Kesimpulan

Melalui hasil penelitian pada bab 4, dapat ditarik kesimpulan mengenai penelitian ini, yaitu sebagai berikut:

1. Terdapat potensi yang cukup kuat, yang ditandai melalui kesimpulan dari hasil kuesioner mengenai kemungkinan penerapan pada masa mendatang yang mencapai rata-rata 4,375 (**baik**), dalam penerapan *Large Language Model (LLM)* pada *video game* dengan genre *role-playing game* yang dalam kasus penelitian ini memakai *project game* Rudantara.
2. Melalui pengujian kombinasi *personality traits* dan hasil kuisoner mengenai *emergent gameplay* pada bab 4, pemain dapat dengan baik mengubah setelan dan melakukan personalisasi pada pengalaman dan gaya bermainnya sendiri. Dengan adanya fitur tersebut, *emergent gameplay* dapat terbentuk.
3. Penerapan sistem *Large Language Model (LLM)* yang digunakan pada penelitian ini, melalui hasil dari seluruh aspek penelitian yang didapatkan pada kuesioner, disimpulkan tercapai dengan rata-rata **baik** dan dapat menghasilkan keluaran yang sesuai dan memenuhi tujuannya dengan baik. Hasil ini dapat dipertimbangkan sebagai efektivitas dari penerapan *LLM* pada *game* bergenre *role-playing game (RPG)*.

#### 5.2. Saran

Adapun beberapa saran yang dapat disampaikan oleh penulis melalui hasil penelitian yang didapatkan, yaitu:

1. Penelitian ini menggunakan *Large Language Model (LLM)* yang tersedia secara publik, yaitu *Gemini 1.0 Pro* yang memiliki batasan tersendiri dan terdapat *LLM* lainnya yang memiliki performa lebih baik dibandingkan dengan *Gemini 1.0 Pro*. Penelitian selanjutnya dapat memanfaatkan *Large Language Model* lainnya yang mendukung masukan dan keluaran *token* yang lebih banyak untuk menampung *context* dalam skala besar sehingga dapat menghasilkan keluaran yang lebih baik.

2. Penelitian selanjutnya dapat berfokus pada pengembangan modul *memory* yang ada pada arsitektur sistem penelitian ini untuk mencapai pengambilan memori yang lebih relevan sehingga dapat menghasilkan keluaran *Large Language Model* yang lebih tepat dalam menghasilkan konteks aksi dan dialog *companion* yang sesuai.
3. Penelitian selanjutnya dapat berfokus pada pengembangan fitur yang dapat mendukung aspek *emergent gameplay* lebih lanjut, dan memperbaiki kekurangan yang terlihat dari penelitian ini. Dibandingkan dengan menerapkan sistem pada *companion*, dapat dilakukan implementasi sistem dengan *Large Language Model* pada kontrol dunia atau lingkungan bermain untuk menghasilkan kejadian beruntun yang tidak terduga pada narasi permainan. Dengan memberikan kontrol kepada *Large Language Model* dalam mengatur kejadian yang dapat terjadi di dunia permainan, diharapkan dapat memberikan pengaruh dari aksi pemain yang lebih besar dan kendali pemain terhadap pengalaman bermainnya dapat merepresentasikan personalisasi pemain dengan baik.

## DAFTAR PUSTAKA

- Andrus, B., & Fulda, N. (2020). Immersive Gameplay via Improved Natural Language Understanding. *ACM International Conference Proceeding Series*. <https://doi.org/10.1145/3402942.3403024>
- Belle, S., Gittens, C., & Nicholas Graham, T. C. (2022). A Framework for Creating Non-Player Characters That Make Psychologically-Driven Decisions. *Digest of Technical Papers - IEEE International Conference on Consumer Electronics*. <https://doi.org/10.1109/ICCE53296.2022.9730383>
- Bowey, J. T., Frommel, J., Piller, B., & Mandryk, R. L. (2021). Predicting Beliefs from NPC Dialogues. *IEEE Conference on Computational Intelligence and Games, CIG*. <https://doi.org/10.1109/CoG52621.2021.9619099>
- Garavaglia, F., Nobre, R. A., Ripamonti, L. A., Maggiorini, D., & Gadia, D. (2022). Moody5: Personality-biased agents to enhance interactive storytelling in video games. *IEEE Conference on Computational Intelligence and Games, CIG*. <https://doi.org/10.1109/CoG51982.2022.9893689>
- Gemini Team, Anil, R., Borgeaud, S., Alayrac, J.-B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A. M., Hauth, A., Millican, K., Silver, D., Johnson, M., Antonoglou, I., Schrittwieser, J., Glaese, A., Chen, J., Pitler, E., Lillicrap, T., Lazaridou, A., ... Vinyals, O. (2023). *Gemini: A Family of Highly Capable Multimodal Models*. 1–90. <http://arxiv.org/abs/2312.11805>
- Gong, R., Huang, Q., Ma, X., Vo, H., Durante, Z., Noda, Y., Zheng, Z., Zhu, S.-C., Terzopoulos, D., Fei-Fei, L., & Gao, J. (2023). *MindAgent: Emergent Gaming Interaction*. 1–28. <http://arxiv.org/abs/2309.09971>
- Gustafsson, V., Holme, B., & MacKay, W. E. (2020). Narrative Substrates: Reifying and Managing Emergent Narratives in Persistent Game Worlds. *ACM International Conference Proceeding Series*. <https://doi.org/10.1145/3402942.3403015>
- Huang, W., Abbeel, P., Pathak, D., & Mordatch, I. (2022). Language Models as Zero-Shot Planners: Extracting Actionable Knowledge for Embodied Agents. *ArXiv Preprint ArXiv:2201.07207*.
- i Bosch, F.-R. (2022). *Dynamic theme-based narrative systems* [UPC, Centre de la

- Imatge i la Tecnologia Multimèdia]. <http://hdl.handle.net/2117/392314>
- Kreminski, M. (2023). Toward Better Gossip Simulation in Emergent Narrative Systems. *2023 IEEE Conference on Games (CoG)*, 1–4. <https://doi.org/10.1109/CoG57401.2023.10333140>
- Nasir, M. U., & Togelius, J. (2023). Practical PCG Through Large Language Models. In *IEEE Conference on Computational Intelligence and Games, CIG*. <https://doi.org/10.1109/CoG57401.2023.10333197>
- Nur Fauzan, H. (2023). *Emergent Gameplay and the Affordance of Features in Open-World Video Game Environments* (Issue 2023:508). KTH Royal Institute of Technology.
- Ohrberg, S. (2019). *Recreating Believability In NPCs: The Effects Of Visual And Logical Behaviour* (p. 29). Malmö universitet/Teknik och samhälle.
- Paananen, K.-E. (2020). *DESIGNING A GAME FOR EMERGENT GAMEPLAY Developing Gatedelvers*.
- Park, J. S., O'Brien, J. C., Cai, C. J., Morris, M. R., Liang, P., & Bernstein, M. S. (2023). *Generative Agents: Interactive Simulacra of Human Behavior*. <http://arxiv.org/abs/2304.03442>
- Reynolds, L., & McDonell, K. (2021). Prompt Programming for Large Language Models: Beyond the Few-Shot Paradigm. *CoRR*, abs/2102.0. <https://arxiv.org/abs/2102.07350>
- Rodrigues, S., Rayat, H. K., Kurichithanam, R. M., & Rukhande, S. (2021). Shriek: A Role Playing Game Using Unreal Engine 4 and Behaviour Trees. *2021 International Conference on Nascent Technologies in Engineering, ICNET 2021 - Proceedings*. <https://doi.org/10.1109/ICNTE51185.2021.9487723>
- Widiyanto, N. R., Nugroho, S. M. S., & Purnomo, M. H. (2020). The Calculation of Player's and Non-Player Character's Gameplay Attribute Growth in Role-Playing Game with K-NN and Naive Bayes. *CENIM 2020 - Proceeding: International Conference on Computer Engineering, Network, and Intelligent Multimedia 2020*. <https://doi.org/10.1109/CENIM51130.2020.9297945>



## LAMPIRAN

**KUESIONER PENILAIAN ASPEK PENELITIAN MENGENAI FITUR  
COMPANION PADA PROJECT GAME RUDANTARA**

**Sangat Kurang = 1, Kurang = 2, Cukup = 3, Baik = 4, Sangat Baik = 5**

No.	Pertanyaan	Penilaian				
		1	2	3	4	5
1	Bagaimana penilaian Anda mengenai ada atau tidaknya kebebasan dan kendali Anda selama bermain dalam pembentukan pengalaman bermain Anda?					
2	Bagaimana penilaian Anda mengenai pengalaman bermain yang mendalam (keterlibatan pemain dalam dunia permainan) dengan adanya fitur <i>companion</i> ini?					
3	Bagaimana penilaian Anda mengenai kesesuaian dialog yang diberikan oleh <i>companion</i> dengan situasi yang terjadi saat Anda bermain?					
4	Bagaimana penilaian Anda mengenai sistem yang mendukung <i>companion</i> dalam menunjukkan aksi dan memberikan dialog yang responsif?					
5	Bagaimana penilaian Anda mengenai pengaruh aksi dari pemain dalam aksi dan dialog yang dilakukan oleh <i>companion</i> ?					
6	Menurut Anda, seberapa mungkin fitur serupa digunakan pada pengembangan <i>game</i> di masa depan (3 hingga 5 tahun kedepan)?					

### TABULASI JAWABAN RESPONDEN

Responden	Pertanyaan ke-					
	1	2	3	4	5	6
<b>A</b>	5	4	4	5	4	5
<b>B</b>	4	5	5	5	5	5
<b>C</b>	4	5	4	4	4	5
<b>D</b>	4	4	5	5	5	5
<b>E</b>	4	2	2	3	3	3
<b>F</b>	4	4	4	4	4	4
<b>G</b>	3	4	4	3	3	3
<b>H</b>	2	2	5	3	4	5