

SriRama: Dungeon Adventure Game



Kelompok 5 - 2023E:

1. Gerry Moeis M.D.P - 23091397164
2. Umar Faruq - 23091397157
3. Yosion Besty Marpaung - 23091397168

<https://github.com/gerrymoeis/SriRama>

D4 Manajemen Informatika

Universitas Negeri Surabaya

Daftar Isi

BAB I Pendahuluan

1.1 Latar Belakang
1.2 Rumusan Masalah
1.3 Tujuan Proyek
1.4 Manfaat Proyek
1.5 Ruang Lingkup Proyek

BAB II Perancangan Sistem

2.1 Analisis Kebutuhan
2.2 Konsep Game <i>SriRama: Dungeon Adventure</i>
2.3 Struktur Folder dan Modularisasi
2.4 Perancangan Objek dan Class Diagram

BAB III Implementasi dan Penjelasan Kode

3.1 Penggunaan Pygame dan Alasan Pemilihan
3.2 Implementasi Modul Utama
3.3 Implementasi OOP pada Game
3.4 Penjelasan Mekanik Utama

BAB IV Hasil dan Pembahasan

4.1 Hasil Pengembangan Game
4.2 Evaluasi Gameplay
4.3 Pembahasan Teknis

BAB V Penutup

5.1 Kesimpulan
5.2 Evaluasi Keseluruhan Proyek
5.3 Saran dan Rencana Pengembangan Selanjutnya

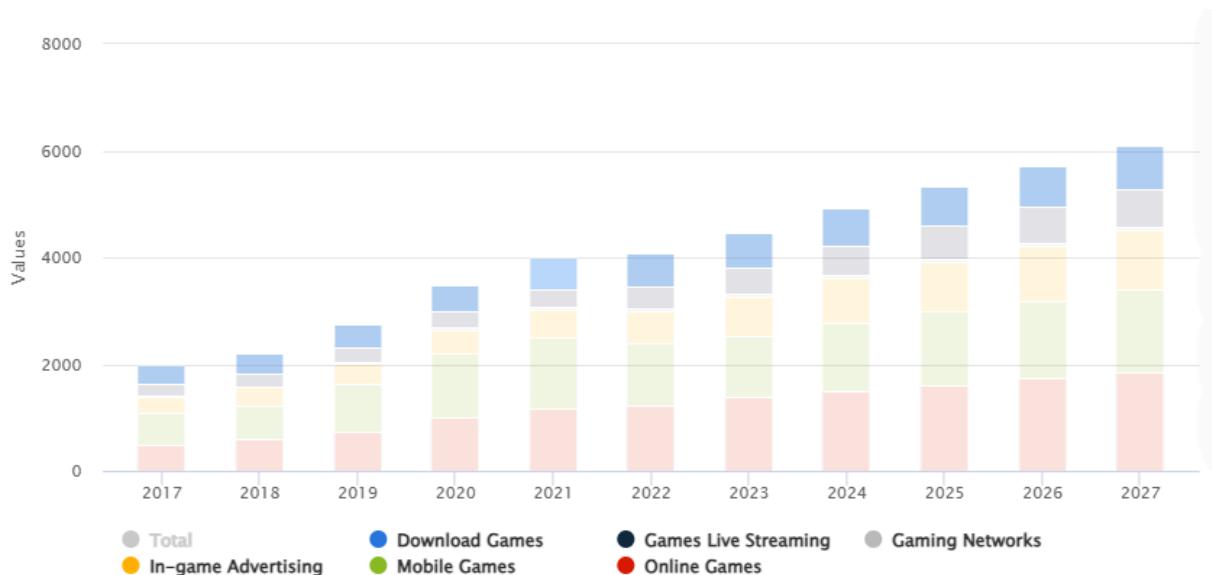
Daftar Pustaka

BAB I

Pendahuluan

1.1 Latar Belakang

Industri game di Indonesia telah mengalami pertumbuhan yang signifikan dalam beberapa tahun terakhir. Menurut data dari Statista, pendapatan pasar video game di Indonesia diproyeksikan mencapai US\$1,23 miliar pada tahun 2024, dengan tingkat pertumbuhan tahunan sebesar 7,31% hingga mencapai US\$1,52 miliar pada tahun 2027. Jumlah pengguna diperkirakan mencapai 53,8 juta pada tahun 2027, dengan tingkat penetrasi pengguna meningkat dari 16,7% pada tahun 2024 menjadi 18,4% pada tahun 2027 (Statista, 2024).



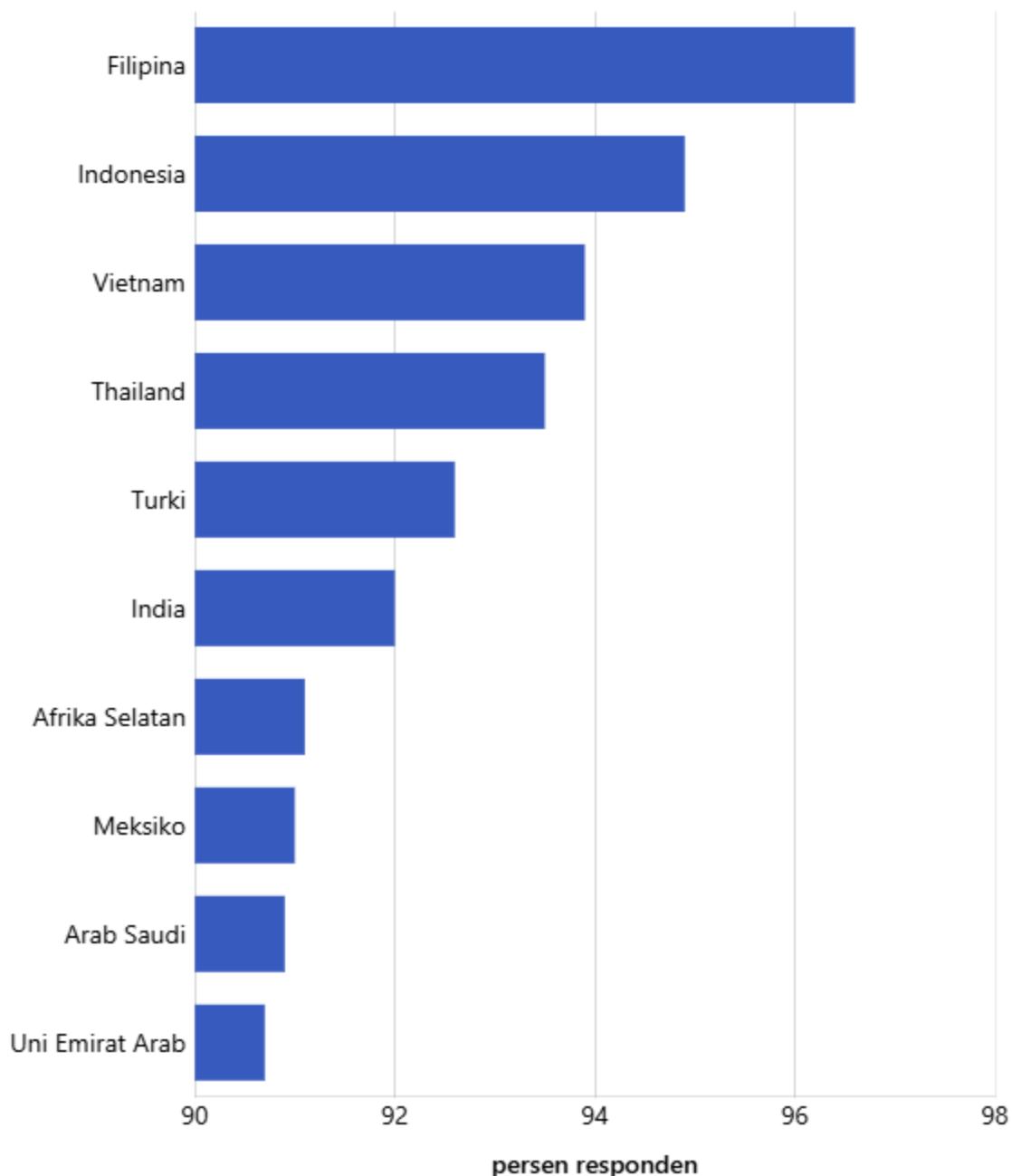
Notes: Data was converted from local currencies using average exchange rates of the respective year.

Most recent update: Mar 2024

Gambar 1. Pertumbuhan dan Proyeksi Pendapatan Pasar Video Game

Selain itu, Indonesia menempati posisi ketiga sebagai pasar game mobile terbesar di dunia. Pada tahun 2022, jumlah unduhan game mobile di Indonesia mencapai 3,45 miliar, meningkat sekitar 320 juta dari tahun sebelumnya (Katadata, 2023).

10 Negara dengan Persentase Pengguna Internet yang Main Video Game Terbanyak (Kuartal I 2022)



Gambar 2. Persentase Pengguna Internet yang Bermain Video Game

Pertumbuhan ini tidak hanya berdampak pada aspek hiburan, tetapi juga membuka peluang untuk pengembangan game dengan muatan edukatif. Penelitian menunjukkan bahwa penggunaan game edukasi dapat meningkatkan motivasi dan hasil belajar siswa. Misalnya, sebuah studi menemukan bahwa penggunaan game edukasi dalam pembelajaran fisika berpengaruh signifikan terhadap kemampuan representasi multiple siswa (TSB eJournal, 2023).

Dengan latar belakang tersebut, pengembangan game *SriRama: Dungeon Adventure* bertujuan untuk memanfaatkan momentum pertumbuhan industri game di Indonesia sekaligus memberikan kontribusi positif dalam bidang edukasi melalui media interaktif yang menarik.

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, rumusan masalah yang ingin dijawab melalui proyek ini adalah:

1. Bagaimana merancang sebuah game *top-down survival* sederhana yang mengedepankan kerja sama dua pemain untuk memberikan pengalaman bermain yang menyenangkan?
2. Bagaimana memastikan gameplay yang sederhana namun tetap menantang dan mampu menciptakan pengalaman nostalgia bagi pemain?
3. Bagaimana merancang sistem permainan dengan mekanik dasar yang stabil, seperti pengumpulan koin dan serangan musuh, agar dapat mendukung pengalaman bermain tanpa kompleksitas berlebih?

1.3 Tujuan Proyek

Tujuan dari pengembangan game *SriRama: Dungeon Adventure* adalah:

1. Mengembangkan game 2D berbasis Python dan Pygame menggunakan pendekatan Object-Oriented Programming (OOP).
2. Merancang gameplay yang menarik dengan fokus pada kolaborasi pemain menghadapi tantangan.
3. Menyediakan platform permainan yang menggabungkan hiburan dan elemen edukasi interaktif.
4. Menonjolkan elemen budaya lokal sebagai daya tarik utama game.
5. Membuktikan keandalan modul Pygame untuk pengembangan game sederhana namun berdaya saing.

1.4 Manfaat Proyek

Manfaat yang diharapkan dari proyek *SriRama: Dungeon Adventure* meliputi:

1. Pengembangan Teknologi: Memberikan wawasan praktis tentang pemrograman berbasis OOP dan pemanfaatan Pygame.
2. Industri Game Lokal: Mendorong pengembang game Indonesia untuk menciptakan konten berbasis budaya lokal.
3. Aspek Edukasi: Memfasilitasi pembelajaran melalui media permainan yang menyenangkan dan interaktif.
4. Peningkatan Kreativitas: Menginspirasi komunitas pengembang game untuk mengintegrasikan budaya tradisional dalam karya mereka.
5. Komunitas Gamer: Menawarkan pengalaman bermain kolaboratif yang dapat meningkatkan kemampuan kerja sama pemain.

1.5 Ruang Lingkup Proyek

Proyek *SriRama: Dungeon Adventure* memiliki ruang lingkup sebagai berikut:

1. Platform dan Teknologi:
 - Game dikembangkan menggunakan bahasa pemrograman Python dan pustaka Pygame.
 - Platform utama yang didukung adalah komputer personal (PC) dengan sistem operasi Windows, Linux, dan macOS.
2. Fokus Gameplay:
 - Genre: *Top-down arena game* dengan kolaborasi dua pemain.
 - Fitur utama: Skill karakter, interaksi pemain dengan lingkungan, dan mekanik permainan berbasis kolaborasi.
 - Level: Satu arena sederhana dengan musuh yang menantang dan mekanik permainan dasar.
3. Pendekatan Pengembangan:
 - Memanfaatkan konsep Object-Oriented Programming (OOP) untuk struktur kode yang modular dan terorganisasi.
 - Desain visual sementara menggunakan placeholder sederhana dengan rencana pengembangan lebih lanjut.
4. Keterbatasan Proyek:
 - Pengembangan hanya mencakup fitur inti tanpa implementasi efek suara atau visual tingkat lanjut.
 - Penilaian efektivitas permainan dilakukan melalui simulasi dan pengujian terbatas, bukan uji pasar skala besar

BAB II

Perancangan Sistem

2.1 Analisis Kebutuhan

Analisis kebutuhan dilakukan untuk menentukan kebutuhan sistem, baik perangkat keras, perangkat lunak, maupun fitur yang harus ada dalam game *SriRama: Dungeon Survivor*. Berikut adalah rincian kebutuhan:

1. Kebutuhan Perangkat Keras

Komponen	Spesifikasi Minimum	Spesifikasi Direkomendasikan
Processor	Intel Core i3 atau setara	Intel Core i5 atau setara
RAM	4 GB	8 GB
Penyimpanan	500 MB ruang kosong	1 GB ruang kosong
GPU	Integrated Graphics	Dedicated GPU seperti NVIDIA GTX 960 atau lebih baik
Resolusi Layar	1280 x 720 (HD)	1920 x 1080 (Full HD)

2. Kebutuhan Perangkat Lunak

Komponen	Detail
Sistem Operasi	Windows, Linux, atau macOS
Bahasa Pemrograman	Python 3.8 atau lebih baru
Library	Pygame versi terbaru (2.1.x)
Editor Kode	Visual Studio Code, PyCharm, atau editor lain

➤ Kebutuhan Fungsional

Kebutuhan Fungsional

Fitur Utama	Detail
Gameplay 2 Pemain	Pemain dapat bermain bersama dengan mekanik kerja sama untuk menghindari musuh dan mengumpulkan koin.
Arena Dinamis	Arena berbentuk grid sederhana dengan musuh yang muncul secara bertahap.
Penghitungan Skor	Koin yang dikumpulkan dihitung sebagai skor akhir.
Skill Unik Karakter	Sri dan Rama memiliki kemampuan khusus yang berbeda, seperti gerakan cepat atau serangan jarak dekat.

➤ Kebutuhan Non-Fungsional

Kebutuhan Non-Fungsional

Aspek	Detail
Efisiensi Kode	Kode harus modular, mudah dibaca, dan mengikuti prinsip OOP.
Kinerja	Game harus berjalan dengan frame rate stabil (minimal 30 FPS) di perangkat minimum.
Interaksi UI	Antar muka sederhana dengan fokus pada gameplay.

2.2 Konsep Game *SriRama: Dungeon Adventure*

1. Deskripsi Umum

SriRama: Dungeon Survivor adalah game survival 2D dengan perspektif *top-down* yang dapat dimainkan oleh dua orang secara bersamaan. Konsep utama adalah bertahan hidup selama mungkin di arena yang dipenuhi musuh, sambil mengumpulkan koin untuk mendapatkan skor tertinggi. Pemain harus bekerja sama untuk menghindari serangan musuh, memanfaatkan skill masing-masing karakter, dan memastikan kelangsungan hidup keduanya.

2. Karakter Utama

- **Sri:** Gadis muda dengan kemampuan gerakan cepat dan serangan jarak jauh.
- **Rama:** Pemuda tangguh dengan kekuatan melindungi dan menjauhkan monster.

3. Mekanik Gameplay

Elemen Gameplay	Deskripsi
Kerja Sama Dua Pemain	Pemain harus saling mendukung satu sama lain. Ketika salah satu karakter kalah, permainan berakhir.
Pengumpulan Koin	Pemain harus mengumpulkan koin yang muncul secara acak di arena sebagai skor utama.
Musuh Dinamis	Musuh muncul dalam gelombang yang semakin sulit seiring waktu.
Skill Karakter	Setiap karakter memiliki skill unik untuk bertahan dari serangan musuh.

4. Catchphrase Game

"Hanya dengan kerja sama seirama, SriRama mampu bertahan!"

5. Visual dan Tema

- Palet warna utama: Biru (Sri) dan Merah (Rama).
- Desain arena sederhana dengan nuansa gelap yang merepresentasikan tantangan bawah tanah (*dungeon*).
- Placeholder visual untuk musuh dan lingkungan, yang akan dikembangkan lebih lanjut pada tahap akhir.

➤ Deskripsi Gameplay

1. Konsep Utama

SriRama: Dungeon Survivor adalah game survival berbasis *top-down* yang dirancang untuk dimainkan oleh dua pemain. Fokus utama game ini adalah pada kerja sama antar pemain untuk bertahan hidup selama mungkin di dalam arena sambil mengumpulkan koin sebagai indikator skor. Game ini mengutamakan gameplay sederhana dengan elemen nostalgia dari permainan klasik yang menonjolkan interaksi sosial dan tantangan kolaboratif.

2. Tujuan Permainan

Pemain harus bertahan menghadapi gelombang musuh yang semakin sulit sambil mengumpulkan sebanyak mungkin koin untuk meningkatkan skor akhir. Ketika salah satu karakter kalah, permainan berakhir. Oleh karena itu, kolaborasi antara dua pemain menjadi kunci utama keberhasilan.

3. Mekanik Permainan

Mekanik	Penjelasan
Kerja sama dua pemain	Pemain harus mengontrol karakter Sri dan Rama secara bersamaan, dengan fokus pada koordinasi untuk menghindari musuh dan mengumpulkan koin.
Karakter unik	- Sri : Karakter dengan kecepatan tinggi dan kemampuan menghindari serangan lawan. - Rama : Karakter dengan kemampuan <i>shield</i> dan <i>healing</i> .
Arena Survival	Arena berbentuk grid sederhana dengan musuh yang muncul secara bertahap di lokasi acak.
Pengumpulan Koin	Koin muncul secara acak di arena dan harus dikumpulkan sebagai indikator skor pemain
Gelombang Musuh	Musuh muncul dalam <i>wave</i> yang semakin menantang, dengan variasi kecepatan dan pola gerakan.

4. Komponen Gameplay

- **Karakter dan Kendali**

Setiap pemain mengendalikan satu karakter:

- ❖ **Sri** dikendalikan menggunakan tombol panah untuk gerakan dan tombol tambahan untuk serangan jarak jauh.
- ❖ **Rama** menggunakan tombol WASD untuk gerakan dan tombol khusus untuk serangan jarak dekat.

- **Arena dan Lingkungan**

Arena permainan adalah lingkungan terbatas yang dirancang secara minimalis dengan nuansa bawah tanah (*dungeon*). Arena ini memiliki batasan yang mencegah pemain keluar, sehingga pemain harus bergerak strategis untuk menghindari musuh.

- **Sistem Skor**

Skor dihitung berdasarkan jumlah koin yang berhasil dikumpulkan oleh kedua pemain. Kerja sama yang baik diperlukan untuk memastikan koin terkumpul maksimal tanpa mengorbankan kelangsungan hidup.

5. Tantangan dan Kesulitan

Aspek Tantangan	Penjelasan
Musuh	Musuh akan semakin sulit di setiap gelombang, baik dari segi jumlah, kecepatan, maupun pola serangan
Kolaborasi	Pemain harus saling mendukung, karena kekalahan satu karakter berarti permainan berakhir.
Arena Terbatas	Ruang gerak yang terbatas membuat pemain harus lebih strategis dalam mengatur posisi.

6. Unik dan Menarik

Yang membedakan *SriRama: Dungeon Survivor* dari game serupa adalah:

1. **Kolaborasi yang Unik:** Pemain harus “seirama” dalam bermain. Ketidakharmonisan antara pemain dapat mengakibatkan kekalahan.
2. **Kesederhanaan Nostalgia:** Dengan desain sederhana dan fokus pada kesenangan, game ini menghidupkan kembali elemen klasik dari game *co-op*.
3. **Nuansa Lokal:** Nama karakter dan tema permainan terinspirasi dari budaya lokal, menciptakan identitas yang unik.

➤ Tema dan Estetika

1. Tema Utama

Tema utama *SriRama: Dungeon Survivor* adalah *kerja sama yang seirama*, baik dari segi gameplay maupun elemen cerita. Game ini berusaha menciptakan pengalaman kolaboratif yang sederhana namun menantang, dengan latar dunia fiksi yang menyerupai nuansa tradisional bercampur fantasi.

Aspek Tema	Penjelasan
Kolaborasi Harmonis	Pemain diwajibkan untuk saling mendukung melalui peran unik masing-masing karakter (Sri dan Rama).
Insiprasi Lokal	Nama karakter dan konsep kerja sama terinspirasi dari filosofi budaya yang menghargai keseimbangan dan harmoni.
Fantasi Dungeon	Arena permainan berbentuk bawah tanah dengan nuansa magis namun sederhana, memberikan tantangan unik.

2. Palet Warna dan Nuansa Visual

Palet warna dan estetika visual dirancang untuk menciptakan suasana yang menarik sekaligus membedakan setiap elemen penting dalam permainan.

Elemen	Deskripsi Visual	Palet Warna
Sri (Karakter Biru)	Karakter Sri memiliki desain yang ramping dengan aksen biru muda, melambangkan kecepatan dan kelincahan.	Biru Muda, Putih, dan Aksen Perak
Rama (Karakter Merah)	Rama memiliki desain yang lebih kokoh dengan warna merah tua, melambangkan kekuatan dan ketangguhan.	Merah Tua, Hitam, dan Aksen Emas
Arena Dungeon	Lingkungan permainan bergaya <i>dark fantasy</i> dengan dominasi warna gelap seperti abu-abu dan coklat tua, serta elemen berbahaya seperti lilin atau obor.	Abu-abu, Coklat Tua, Oranye Glow
Koin	Koin berwarna emas cerah dengan animasi kecil untuk menonjolkan daya tariknya sebagai elemen penting.	Emas dan Kuning Cerah

3. Desain Karakter

Karakter	Detail Visual
Sri	Figur ramping dengan kostum bergaya tradisional sederhana yang dipadukan dengan elemen modern.
Rama	Figur kokoh dengan pakaian yang lebih berat, menonjolkan peran sebagai pelindung.

4. Estetika Antarmuka (UI/UX)

Desain UI/UX dirancang untuk memberikan pengalaman bermain yang sederhana namun intuitif, dengan mempertimbangkan fokus pemain pada gameplay.

Elemen UI	Deskripsi
HUD (Head-Up Display)	Menampilkan informasi penting seperti bar HP, jumlah koin yang dikumpulkan, dan waktu bertahan.
HP Bar	Bar HP untuk Sri dan Rama ditempatkan di bagian atas layar, dengan warna biru (Sri) dan merah (Rama).
Navigasi Sederhana	Menu navigasi minimalis dengan pilihan seperti “Start Game”, “Options”, dan “Exit”.
Tip dan Indikator	Elemen visual seperti tanda panah untuk menyorot koin atau musuh yang berada di luar layar.

5. Audio dan Musik

Aspek	Deskripsi
Musik Latar	Musik latar bergaya tradisional dengan alat musik seperti gamelan dipadukan dengan nada-nada <i>ambient</i> .

6. Keselarasan Tema dan Estetika

Desain *SriRama: Dungeon Survivor* tidak hanya berfokus pada elemen gameplay, tetapi juga pengalaman emosional pemain. Dengan menciptakan tema dan estetika yang harmonis antara gameplay, visual, dan audio, game ini bertujuan untuk memberikan pengalaman yang memuaskan baik secara hiburan maupun nostalgia.

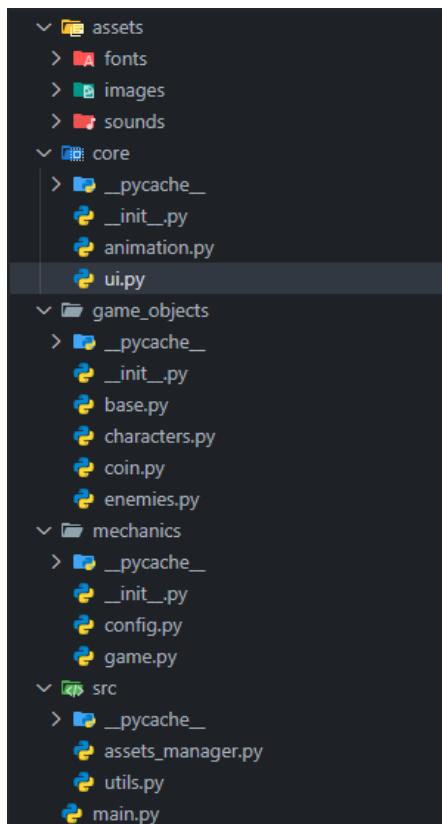
2.3 Struktur Folder dan Modularisasi

Pengembangan game *SriRama: Dungeon Survivor* dilakukan dengan pendekatan modular menggunakan Python dan Pygame. Pendekatan ini memastikan bahwa kode tetap terorganisir, mudah dibaca, dan dapat dikelola dengan baik. Modularisasi juga memudahkan proses debugging, penambahan fitur, dan pemeliharaan proyek di masa depan.

2.3.1 Desain Struktur Folder

Struktur folder dirancang untuk memisahkan elemen-elemen inti game, seperti asset visual, audio, dan kode program, guna menjaga keteraturan dan memudahkan navigasi. Berikut adalah struktur folder yang digunakan:

SriRama-Dungeon-Survivor/



Penjelasan Struktur:

1. **Folder asset - assets/**: Menyimpan semua aset visual, audio, dan font yang digunakan dalam game, dipisahkan ke dalam subfolder berdasarkan tipe aset.
2. **Folder kode - src/, mechanics/, game_objects/, core/**: Berisi kode sumber game dengan modularisasi yang dirancang untuk mempermudah pengelolaan logika game. Setiap komponen logika game memiliki folder khusus (misalnya, entities/ untuk karakter, musuh, dan item).

2.3.2 Modul dan Fungsionalitasnya

Berikut adalah penjelasan tentang modul-modul utama dan fungsionalitasnya:

Modul/Fungsi	Deskripsi	Lokasi
main.py	File utama yang digunakan untuk menjalankan game dan menginisialisasi semua komponen.	./ (Main Folder)
game.py	Modul inti untuk mengatur jalannya game, termasuk pengaturan <i>loop</i> game dan transisi antar adegan.	mechanics/
config.py	Menyimpan pengaturan global seperti resolusi layar, kecepatan karakter, dan pengaturan suara.	mechanics/
utils.py	Berisi fungsi-fungsi pendukung seperti penghitungan waktu, pembacaan file aset, atau konversi data.	src/
assets_manager.py	Menyimpan data source asset dan mengelola inisiasi dan pemanggilan asset.	src/
characters.py	Mengatur logika karakter pemain (Sri dan Rama), termasuk gerakan, serangan, dan interaksi.	game_objects/
enemies.py	Mengelola perilaku musuh, termasuk pola gerakan dan <i>spawn</i> .	game_objects/
ui.py	Modul untuk logika menu utama, seperti pilihan “Start Game” dan “Options”.	core/
animation.py	Mengatur inisiasi dan animasi sprite character.	core/

Keunggulan Modularisasi:

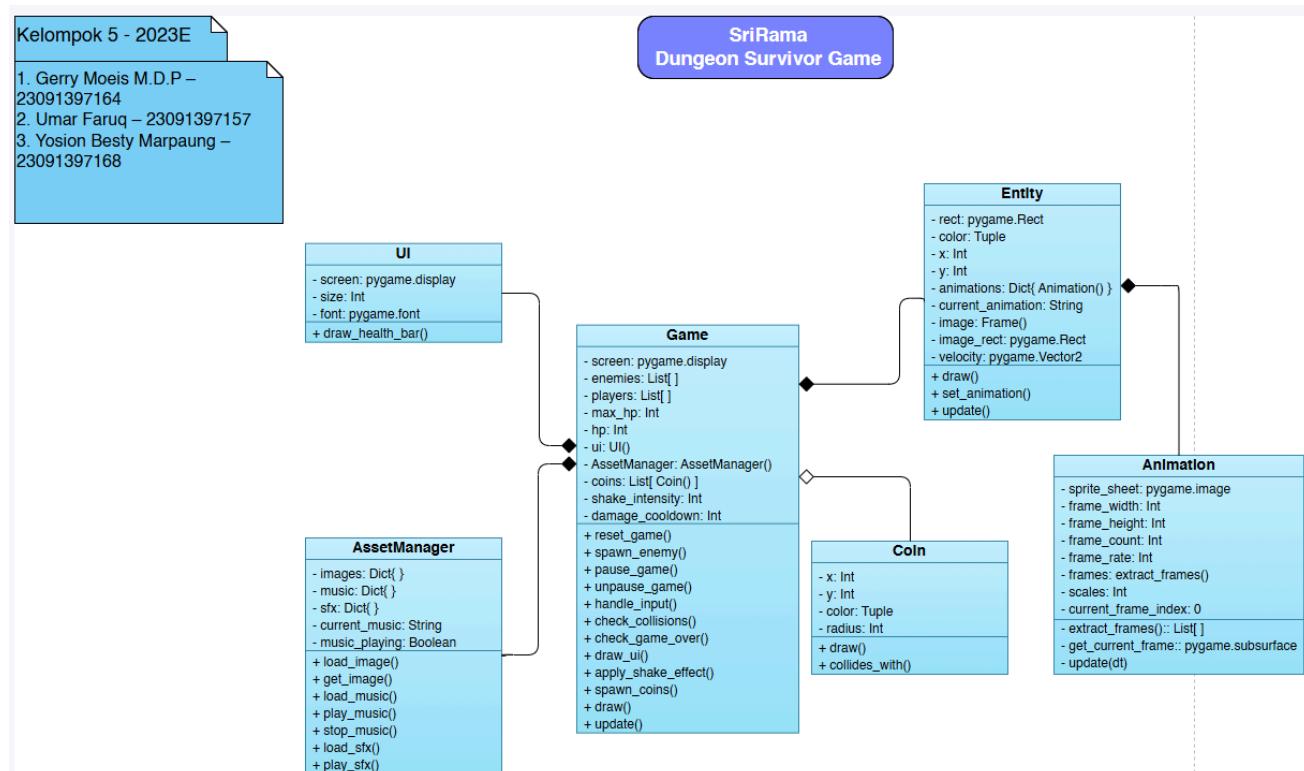
- Keterpisahan Tugas:** Setiap modul bertanggung jawab atas satu aspek spesifik, misalnya logika musuh atau tampilan UI.
- Kemudahan Perubahan:** Dengan modularisasi, perubahan pada satu bagian (misalnya, desain musuh) tidak akan memengaruhi bagian lain.
- Pemeliharaan dan Skalabilitas:** Struktur ini memungkinkan pengembang untuk menambahkan fitur baru atau memperbaiki bug dengan lebih mudah.

2.4 Perancangan Objek dan Class Diagram

Dalam pengembangan *SriRama: Dungeon Survivor*, pendekatan *Object-Oriented Programming (OOP)* digunakan untuk menciptakan struktur kode yang modular, fleksibel, dan mudah dikelola. Perancangan objek dalam game ini mencakup entitas utama seperti pemain, musuh, koin, dan adegan. Selain itu, relasi antar kelas dirancang agar setiap komponen memiliki tanggung jawab yang jelas.

2.4.1 Diagram Class Utama

Berikut adalah diagram kelas utama yang digunakan dalam game:



1. Game

Class ini berfungsi sebagai pengontrol utama (*controller*) untuk menjalankan keseluruhan alur permainan. *Game* mengatur logika permainan, memanggil *class* lain, dan bertanggung jawab untuk memastikan semua komponen berjalan sinkron.

2. UI

Class ini bertanggung jawab untuk menangani elemen antarmuka pengguna, seperti menampilkan *health bar*, jumlah koin yang terkumpul, dan waktu permainan. *Game* memanfaatkan *UI* untuk berkomunikasi dengan pemain secara **visual**.

3. AssetManager

Class ini digunakan untuk mengelola aset game, seperti gambar, suara, dan animasi. Dengan menggunakan *AssetManager*, *Game* dapat memuat aset secara efisien dan menghindari duplikasi kode.

4. Coin

Class ini merepresentasikan objek koin yang dapat dikumpulkan oleh pemain. *Coin* berinteraksi langsung dengan *Entity* untuk mendeteksi *collision* dan melaporkan koin yang berhasil dikumpulkan ke *Game*.

5. Entity

Class dasar yang menjadi induk dari semua karakter dan objek yang memiliki properti fisik dalam game, termasuk pemain dan musuh. *Entity* dilengkapi dengan fitur seperti posisi, kecepatan, dan logika *collision*.

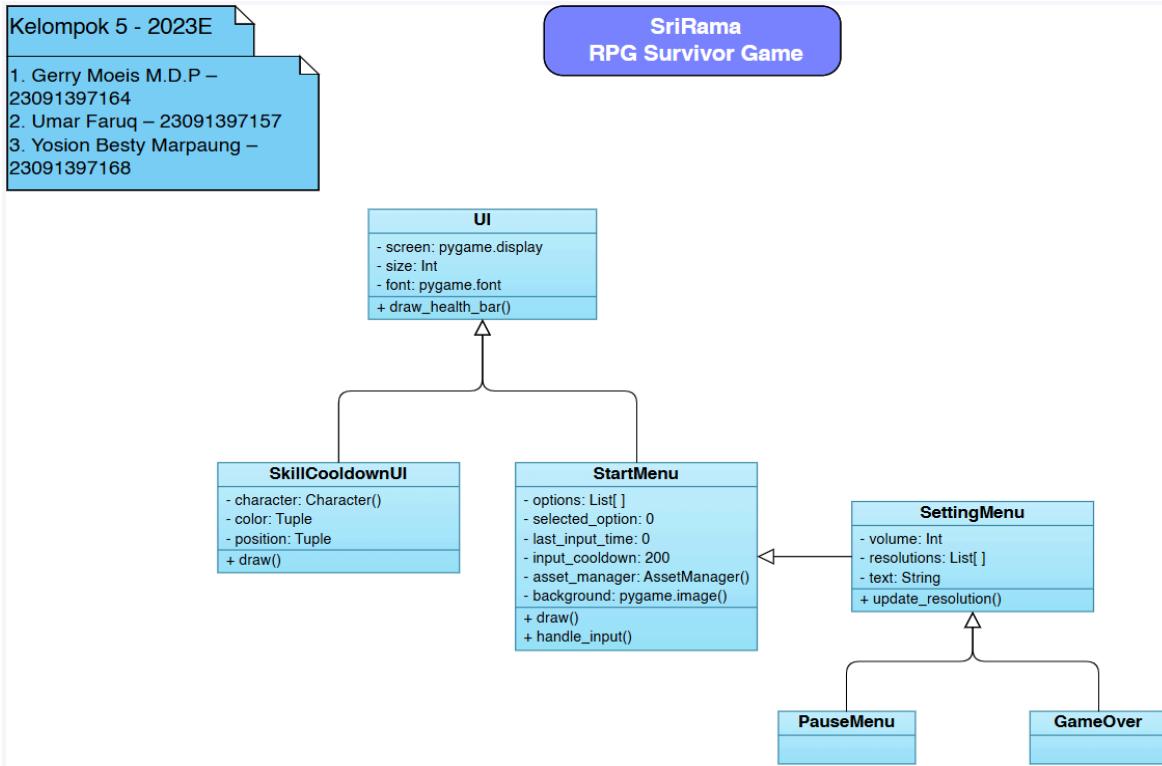
6. Animation

Class ini mengatur animasi untuk objek yang membutuhkan efek visual dinamis, seperti gerakan karakter atau musuh. *Animation* bekerja bersama *Entity* untuk memastikan setiap objek memiliki transisi visual yang mulus dan menarik.

2.4.2 Pendalaman Diagram Class

1. UI Class

UI Class dirancang untuk menangani semua elemen antarmuka pengguna dalam game *SriRama: Dungeon Survivor*. Struktur *class* ini memanfaatkan konsep generalisasi untuk mengelompokkan elemen-elemen UI yang memiliki kesamaan, sekaligus memastikan fleksibilitas dan kemudahan dalam pengembangan. Berikut penjelasan rinci:



➤ UI (Induk Class)

Class ini berperan sebagai kerangka dasar untuk semua elemen antarmuka pengguna. *UI* menyediakan atribut dan metode umum, seperti pengaturan posisi elemen, tampilan teks, atau elemen interaktif yang digunakan dalam berbagai menu dan tampilan.

➤ SkillCooldownUI (Generalization UI)

Subclass dari *UI* yang dirancang untuk menampilkan waktu *cooldown* skill karakter. Elemen ini memberikan informasi visual kepada pemain mengenai status skill, memastikan pemain dapat menggunakan skill dengan strategi yang tepat.

➤ Start Menu (Generalization)

Subclass dari *UI* yang bertugas menangani menu awal permainan. *Start Menu* mencakup tombol untuk memulai permainan, melihat pengaturan, atau keluar dari game. Komponen ini menjadi pintu masuk utama interaksi pemain dengan game.

➤ **Setting Menu** (*Generalization Start Menu*)

Subclass ini menangani pengaturan yang dapat diubah oleh pemain, seperti pengaturan suara, kontrol, atau kualitas grafis. *Setting Menu* memberikan fleksibilitas kepada pemain untuk menyesuaikan pengalaman bermain sesuai preferensi mereka.

➤ **Pause Menu** (*Generalization Setting Menu*)

Subclass dari *UI* yang muncul saat pemain menjeda permainan. Menu ini menampilkan opsi untuk melanjutkan, memulai ulang, atau keluar dari permainan tanpa mengganggu progres yang sedang berlangsung.

➤ **GameOver** (*Generalization Setting Menu*)

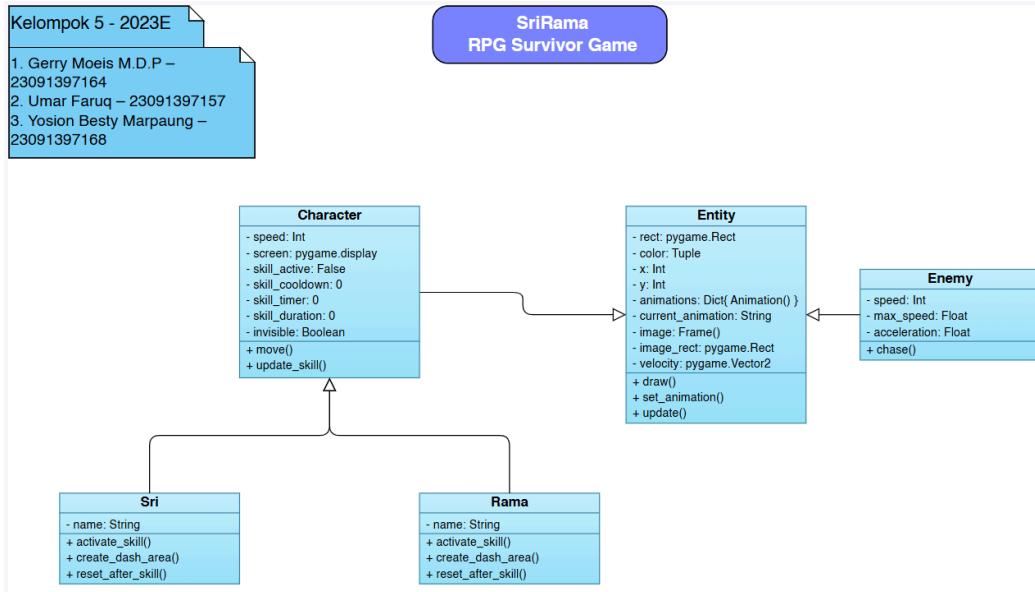
Subclass terakhir dari *UI* yang menangani layar akhir permainan. Elemen ini memberikan informasi mengenai skor akhir, jumlah koin yang dikumpulkan, dan opsi untuk bermain kembali atau keluar dari game.

Hubungan Antar Class

- *UI* menjadi induk yang mengatur metode dan atribut umum yang diwariskan ke semua subclass.
- *SkillCooldownUI* berinteraksi langsung dengan sistem *gameplay* untuk menampilkan status waktu *cooldown*.
- *Start Menu* dan *Pause Menu* berhubungan langsung dengan *Game Controller* untuk menangani navigasi logika permainan.
- *GameOver* mendapatkan data dari *Game Controller* untuk menampilkan hasil permainan.

2. Entity Class

Entity Class adalah inti dari sistem game yang menangani semua objek dengan properti fisik dan perilaku dalam dunia permainan. Dengan memanfaatkan generalisasi, diagram ini memisahkan tanggung jawab berdasarkan jenis entitas, seperti karakter pemain atau musuh. Berikut penjelasannya:



➤ Entity (Induk Class)

Class ini menjadi dasar untuk semua objek dalam game yang memiliki atribut fisik, seperti posisi, kecepatan, dan dimensi. *Entity* juga menangani mekanisme umum seperti *collision detection* dan pembaruan posisi setiap frame.

➤ Character (Generalization of Entity)

Subclass dari *Entity* yang berfokus pada karakter pemain. *Character* memperluas *Entity* dengan menambahkan atribut khusus seperti *health points (HP)*, *damage output*, dan kemampuan untuk menggunakan skill. Semua perilaku dasar karakter, seperti bergerak dan menyerang, dikelola di sini.

➤ Enemy (Generalization of Entity)

Subclass lain dari *Entity* yang merepresentasikan musuh dalam permainan. *Enemy* memiliki logika perilaku unik, seperti mengejar pemain atau menyerang secara otomatis. Atribut tambahan meliputi pola gerakan, tingkat kesulitan, dan *damage* yang dihasilkan.

➤ Sri dan Rama (Generalization of Character)

Subclass dari *Character* yang merepresentasikan dua karakter utama game, yaitu Sri dan Rama.

➤ Sri: Memiliki keahlian khusus untuk bergerak dengan cepat dan menghindari serangan musuh.

- **Rama:** Keahlian lebih kepada pertahanan yang dapat mencegah monster mendekat dan sekaligus membuat lingkaran *healing*.

Kedua karakter ini dirancang untuk saling melengkapi, sehingga pemain harus bekerja sama agar dapat bertahan dalam permainan.

Hubungan Antar Class

- *Entity* adalah induk dari semua objek dalam dunia game yang memiliki atribut fisik dasar dan logika *collision*.
- *Character* dan *Enemy* mewarisi metode dan atribut umum dari *Entity* tetapi mengimplementasikan perilaku spesifik sesuai peran mereka.
- Sri dan Rama, sebagai subclass dari *Character*, memiliki kemampuan unik yang membedakan satu sama lain, memungkinkan variasi gaya bermain.
- *Enemy* berinteraksi langsung dengan *Character* (Sri dan Rama) melalui sistem serangan dan *collision*, menciptakan tantangan gameplay.

Keunggulan Desain Class

1. **Encapsulation**
 - Atribut dan metode setiap kelas bersifat modular, sehingga memudahkan pengembangan dan perbaikan tanpa memengaruhi bagian lain.
2. **Inheritance**
 - Penggunaan kelas induk **Player** untuk Sri dan Rama meminimalkan redundansi kode.
3. **Polymorphism**
 - Metode yang sama seperti *move()* dan *attack()* dapat berfungsi berbeda sesuai dengan kelas turunan.
4. **Relasi yang Jelas**
 - Hubungan antar kelas mendukung interaksi yang logis dan memudahkan implementasi game.

BAB III

Implementasi dan Penjelasan Kode

3.1 Penggunaan Pygame dan Alasan Pemilihan

Pygame adalah pustaka Python yang dirancang untuk pengembangan game 2D. Dalam pengembangan *SriRama: Dungeon Survivor*, Pygame dipilih karena alasan berikut:

1. Kemudahan Penggunaan: Pygame menyediakan API yang intuitif untuk pengelolaan grafis, suara, dan kontrol input, sehingga mempercepat pengembangan.
2. Komunitas yang Aktif: Tersedia banyak dokumentasi, forum, dan sumber daya dari komunitas yang mendukung pembelajaran dan pemecahan masalah.
3. Kompatibilitas dengan Python: Sebagai pustaka yang berbasis Python, Pygame mempermudah integrasi dengan kode lain dan memungkinkan implementasi OOP secara penuh.
4. Fleksibilitas: Pygame mendukung berbagai format file (gambar, suara, dan font), yang membantu dalam pengelolaan aset game.

Pygame memberikan fondasi untuk menangani elemen-elemen seperti rendering grafis, pengelolaan *loop* game, deteksi tabrakan, dan pengelolaan waktu, yang semuanya penting untuk pengembangan game seperti *SriRama: Dungeon Survivor*.

3.2 Implementasi Modul Utama

Modul utama pada game ini dirancang untuk menangani berbagai aspek permainan, mulai dari pengaturan logika inti hingga pengaturan parameter. Berikut adalah implementasi masing-masing modul:

3.2.1 Modul `game.py`

Modul `game.py` bertanggung jawab untuk menangani logika inti permainan. Modul ini mencakup implementasi *game loop*, pengaturan elemen visual di layar, dan pembaruan status permainan secara real-time.

Kode:

```
import pygame
import random

from mechanics.config import SCREEN_WIDTH, SCREEN_HEIGHT, RED, BLUE, GREEN

from game_objects.characters import Sri, Rama
from game_objects.enemies import EnemyRandom
from game_objects.coin import Coin

from core.animation import Animation

from core.ui import UI, PauseMenu, GameOver, SkillCooldownUI

class Game:
    def __init__(self, screen, asset_manager):
        self.screen = screen

        self.enemies = []
        self.max_hp = 100
        self.hp = self.max_hp
        self.all_sprites = []

        self.ui = UI(screen)
        self.pause_menu = PauseMenu(screen)
        self.paused = False
        self.game_over = GameOver(screen)

        self.shake_intensity = 0
        self.damage_cooldown = 0

        self.enemy_spawn_time = 2000 # Spawn setiap 2 detik
        self.last_spawn_time = pygame.time.get_ticks()

        self.players = [
            Sri(100, 100, 40, 40, 5, self.screen),
            Rama(200, 100, 40, 40, 5, self.screen)
        ]
```

```
        self.sri_ui = SkillCooldownUI(self.screen, self.players[0], BLUE,
(SCREEN_WIDTH - 300, 50))
        self.rama_ui = SkillCooldownUI(self.screen, self.players[1], RED,
(SCREEN_WIDTH - 300, 100))

        self.font = pygame.font.Font(None, 36)
        self.start_time = 0 # Waktu mulai stopwatch
        self.elapsed_time = 0 # Waktu bertahan pemain

        self.coin_timer = 0
        self.coins = [] # Daftar koin
        self.score = 0 # Skor pemain
        self.coin_spawn_interval = 3000 # Interval spawn koin dalam
milidetik (3 detik)

        self.enemies_animations = [
            {"idle":
Animation("assets/images/monsters/normal/16x16/bat.png", 16, 16, 3, 5,
1.25) ,
            {"idle":
Animation("assets/images/monsters/normal/16x16/skeleton.png", 16, 16, 3,
5, 1.25) ,
            {"idle":
Animation("assets/images/monsters/normal/16x16/rat.png", 16, 16, 2, 5,
1.25) ,
            {"idle":
Animation("assets/images/monsters/normal/16x16/spider.png", 16, 16, 2, 5,
1.25) ,
            {"idle":
Animation("assets/images/monsters/normal/16x16/ghost.png", 16, 16, 3, 5,
1.25) ,
            }
        ]

        self.asset_manager = asset_manager
        self.background = self.asset_manager.get_image("gameplay_bg")

        self.reset_game()

    def reset_game(self):
```

```
"""Reset state game dengan randomisasi untuk posisi, jumlah enemy,  
dan atribut lainnya."""  
  
    self.players = [  
        Sri(100, 100, 40, 40, 5, self.screen),  
        Rama(200, 100, 40, 40, 5, self.screen)  
    ]  
  
    self.sri_ui = SkillCooldownUI(self.screen, self.players[0], BLUE,  
(SCREEN_WIDTH - 300, 50))  
    self.rama_ui = SkillCooldownUI(self.screen, self.players[1], RED,  
(SCREEN_WIDTH - 300, 100))  
  
    # Randomisasi jumlah musuh antara 3-6  
    num_enemies = random.randint(3, 6)  
    self.enemies = [  
        EnemyRandom(  
            random.randint(0, SCREEN_WIDTH - 50),  
            random.randint(-50, -25),  
            40, 40,  
            (0, random.randint(100, 255), 0), # Hijau acak untuk  
warna  
            random.randint(1, 2), # Kecepatan acak  
            random.randint(50, 150), # Jarak Patrol  
            self.enemies_animations[random.randint(0, 4)]  
        ) for _ in range(num_enemies)  
    ]  
  
    self.hp = self.max_hp  
    self.all_sprites = self.enemies  
    self.shake_intensity = 0  
    self.damage_cooldown = 0  
  
    self.start_time = pygame.time.get_ticks()  
    self.elapsed_time = 0  
  
    self.coin_timer = 0  
    self.coins = []  
    self.score = 0  
    self.coin_spawn_interval = 3000 # Reset interval spawn koin
```

```
    self.clock = pygame.time.Clock()

    self.unpause_game()

def spawn_enemy(self):
    """Spawn enemy secara random dari sisi-sisi batas layar."""
    current_time = pygame.time.get_ticks()
    if current_time - self.last_spawn_time >= self.enemy_spawn_time:
        spawn_side = random.choice(['top', 'bottom', 'left', 'right'])
        if spawn_side == 'top':
            x = random.randint(0, SCREEN_WIDTH)
            y = -50 # Spawn di luar layar bagian atas
        elif spawn_side == 'bottom':
            x = random.randint(0, SCREEN_WIDTH)
            y = SCREEN_HEIGHT + 50 # Spawn di luar layar bagian bawah
        elif spawn_side == 'left':
            x = -50 # Spawn di luar layar bagian kiri
            y = random.randint(0, SCREEN_HEIGHT)
        elif spawn_side == 'right':
            x = SCREEN_WIDTH + 50 # Spawn di luar layar bagian kanan
            y = random.randint(0, SCREEN_HEIGHT)

        num_enemies = random.randint(1, 3)
        for _ in range(num_enemies):
            new_enemy = EnemyRandom(
                x, y, 40, 40,
                (0, random.randint(100, 255), 0), # Hijau acak untuk
                warna
                random.randint(1, 2), # Kecepatan acak
                random.randint(50, 150), # Jarak Patrol
                self.enemies_animations[random.randint(0, 4)])
            self.enemies.append(new_enemy)
            self.all_sprites.append(new_enemy)
            self.last_spawn_time = current_time

def pause_game(self):
    self.paused = True

def unpause_game(self):
```

```

        self.paused = False

    def handle_input(self):
        keys = pygame.key.get_pressed()
        controls = [
            (pygame.K_w, pygame.K_s, pygame.K_a, pygame.K_d),
            (pygame.K_UP, pygame.K_DOWN, pygame.K_LEFT, pygame.K_RIGHT)
        ]
        for player, control in zip(self.players, controls):
            player.move(keys, *control)

    def check_collisions(self):
        if self.damage_cooldown > 0:
            self.damage_cooldown -= 1

        for player in self.players:
            if player.image_rect.left < 0:
                player.image_rect.left = 0
            if player.image_rect.right > SCREEN_WIDTH:
                player.image_rect.right = SCREEN_WIDTH
            if player.image_rect.top < 0:
                player.image_rect.top = 0
            if player.image_rect.bottom > SCREEN_HEIGHT:
                player.image_rect.bottom = SCREEN_HEIGHT

        for enemy in self.enemies:
            if player.image_rect.colliderect(enemy.image_rect) and
self.damage_cooldown == 0 and not player.invisible:
                self.hp -= 5
                player.color = (0, 0, 0)
                self.shake_intensity = 20
                self.damage_cooldown = 10

            self.draw()

    def check_game_over(self):
        if self.hp <= 0:
            return "LOSE"
        if not self.enemies:
            return "WIN"

```

```
    return None

def draw_ui(self):
    self.ui.draw_health_bar(10, 10, 500, 20, self.hp, self.max_hp)

    [sri, rama] = self.players
    self.sri_ui.draw("Sri", sri.skill_cooldown)
    self.rama_ui.draw("Rama", rama.skill_cooldown)

    # Tampilkan stopwatch
    stopwatch_text = self.font.render(f"Time Survived: {self.elapsed_time}s", True, (255, 255, 255))
    self.screen.blit(stopwatch_text, (SCREEN_WIDTH // 2 - stopwatch_text.get_width() // 2, 25))

    # Tampilkan skor
    score_text = self.font.render(f"Score: {self.score}", True, (255, 255, 0))
    self.screen.blit(score_text, (SCREEN_WIDTH // 2 - score_text.get_width() // 2, 75))

def apply_shake_effect(self):
    if self.shake_intensity > 0:
        offset_x = random.randint(-self.shake_intensity, self.shake_intensity)
        offset_y = random.randint(-self.shake_intensity, self.shake_intensity)
        self.shake_intensity = max(self.shake_intensity - 1, 0)
    return offset_x, offset_y
    return 0, 0

def spawn_coins(self):
    """Spawn koin secara random."""
    num_coins = random.randint(1, 3) # Jumlah koin dalam satu cycle
    for _ in range(num_coins):
        x = random.randint(50, SCREEN_WIDTH - 50)
        y = random.randint(50, SCREEN_HEIGHT - 50)
        self.coins.append(Coin(x, y))

def update(self):
```

```
dt = self.clock.tick(60) # Delta time (ms)
current_time = pygame.time.get_ticks()

if self.paused:
    last_input_time = 0

    if current_time - last_input_time > 50:
        self.pause_menu.draw()

    self.last_input_time = current_time # Reset cooldown
else:
    game_state = self.check_game_over()
    if game_state == "LOSE":
        last_input_time = 0

        if current_time - last_input_time > 50:
            self.game_over.draw(self.elapsed_time, self.score)
            self.last_input_time = current_time # Reset cooldown
        else:
            self.screen.blit(self.background, (SCREEN_WIDTH // 2 -
self.background.get_width() // 2, SCREEN_HEIGHT // 2 -
self.background.get_height() // 2))

        self.elapsed_time = (current_time - self.start_time) //
1000 # Dalam detik

    self.handle_input()
    self.check_collisions()

    self.spawn_enemy()

    for enemy in self.enemies:
        enemy.chase(self.players)
        enemy.draw(self.screen)

    for player in self.players:
        player.draw(self.screen)
        player.update(dt)

    player.color = player.default_color
```

```
        player.update_skill(current_time)

        # Handle input untuk skill
        [sri, rama] = self.players
        keys = pygame.key.get_pressed()
        if keys[pygame.K_f]: # Aktivasi skill untuk Sri
            sri.activate_skill()
        if keys[pygame.K_RSHIFT]: # Aktivasi skill untuk Rama
            rama.activate_skill()

        if sri.skill_active:
            sri.create_dash_area() # Tampilkan area dash selama
skill aktif
        if rama.skill_active:
            rama.create_healing_area(self.enemies) # Tampilkan
area healing selama skill aktif
            self.hp += 0.25

        if self.hp >= self.max_hp:
            self.hp = self.max_hp

        # Input untuk Sri
        if keys[pygame.K_w] or keys[pygame.K_a] or
keys[pygame.K_s] or keys[pygame.K_d]:
            sri.set_animation("walk")
        else:
            sri.set_animation("idle")

        # Input untuk Rama
        if keys[pygame.K_UP] or keys[pygame.K_LEFT] or
keys[pygame.K_DOWN] or keys[pygame.K_RIGHT]:
            rama.set_animation("walk")
        else:
            rama.set_animation("idle")

        # Spawn koin
        if current_time - self.coin_timer >
self.coin_spawn_interval:
            self.coin_timer = current_time
            self.spawn_coins()
```

```

        # Update koin
        for coin in self.coins[:]:
            if coin.collides_with(sri.image_rect) or
coin.collides_with(rama.image_rect):
                self.coins.remove(coin)
                self.score += 1 # Tambahkan skor
                self.coin_spawn_interval = max(500,
self.coin_spawn_interval - 50) # Akselerasi spawn

        for coin in self.coins:
            coin.draw(self.screen)

        self.draw_ui()

    def draw(self):
        offset_x, offset_y = self.apply_shake_effect()
        for sprite in self.all_sprites:
            sprite.draw(self.screen, offset=(offset_x, offset_y))
            # sprite.update(dt, self.screen)

        for player in self.players:
            player.draw(self.screen, offset=(offset_x, offset_y))

        self.draw_ui()

```

Fungsi Utama dalam game.py:

1. Inisialisasi Game

- Konstruktor `__init__()` menetapkan atribut awal permainan, seperti pemain (*players*), musuh (*enemies*), sistem antarmuka (*UI*), dan elemen tambahan seperti *coins* dan animasi.
- Komponen seperti *SkillCooldownUI* untuk masing-masing pemain dan sistem *background* diatur agar sesuai dengan kebutuhan visual permainan.

2. Reset Permainan

- Fungsi `reset_game()` mengatur ulang semua elemen permainan, seperti posisi karakter, jumlah musuh, dan kondisi permainan lainnya.
- Fungsi ini memanfaatkan randomisasi untuk membuat setiap permainan terasa unik.

3. Pengaturan Musuh

- `spawn_enemy()` menambahkan musuh secara acak di berbagai sisi layar dengan properti berbeda seperti kecepatan, warna, dan pola gerakan.
- Sistem animasi musuh didukung oleh kelas `Animation`, memastikan setiap jenis musuh memiliki tampilan yang berbeda.

4. Penanganan Input

- Fungsi `handle_input()` mengatur pergerakan pemain menggunakan kontrol tertentu.
- Input tambahan memungkinkan pemain untuk mengaktifkan *skill* seperti *dash* dan *healing area*.

5. Pembaruan Status Permainan

- `update()` bertanggung jawab untuk memperbarui status permainan setiap frame, termasuk pergerakan karakter, *collision detection*, spawn musuh, dan spawn koin.
- Fungsi ini juga mencatat waktu bermain dan skor pemain, serta menampilkan elemen *UI* seperti *health bar* dan stopwatch.

6. Efek Visual

- `apply_shake_effect()` memberikan efek goyangan (*shake*) pada layar saat pemain menerima serangan, meningkatkan pengalaman visual.

7. Pengecekan Kondisi Permainan

- Fungsi `check_game_over()` memeriksa apakah permainan berakhir dengan kondisi menang atau kalah.
- Jika *HP* mencapai 0 atau semua musuh berhasil dikalahkan, kondisi permainan akan diperiksa.

8. Sistem Koin

- `spawn_coins()` menambahkan elemen interaktif berupa koin di lokasi acak, yang dapat dikumpulkan pemain untuk meningkatkan skor.
- Pengumpulan koin memengaruhi tingkat kesulitan permainan dengan mempercepat interval spawn koin berikutnya.

9. Pause dan Game Over

- Fungsi `pause_game()` dan `unpause_game()` memungkinkan pemain untuk menghentikan sementara permainan.
- Antarmuka *PauseMenu* dan *GameOver* diatur untuk memberikan visualisasi yang jelas dalam kondisi tersebut.

10. Rendering dan Animasi

- Fungsi `draw()` dan `draw_ui()` menangani semua elemen visual di layar, termasuk tampilan karakter, musuh, dan elemen *UI*.
- Animasi pemain dan musuh diperbarui berdasarkan status atau masukan kontrol

3.2.2 Modul *main.py*

Kode:

```
import pygame
from mechanics.config import SCREEN_WIDTH, SCREEN_HEIGHT, FPS, WHITE
from mechanics.game import Game
from core.ui import StartMenu, SettingMenu

from src.assets_manager import AssetManager
from src.utils import AudioManager

class MainGame:
    def __init__(self, current_state="start_menu"):
        self.resolutions = [(800, 600), (1024, 768), (1280, 720), (1920, 1080)]
        self.resolution = (SCREEN_WIDTH, SCREEN_HEIGHT)

        pygame.init()
        self.screen = pygame.display.set_mode(self.resolution,
pygame.RESIZABLE)
        pygame.display.set_caption("SriRama: Dungeon Survivor")

        self.clock = pygame.time.Clock()
        self.last_input_time = 0
        self.input_cooldown = 200

        self.asset_manager = AssetManager()
        self.audio_manager = AudioManager()
        self.load()

        self.start_menu = StartMenu(self.screen, self.asset_manager)
        self.game = Game(self.screen, self.asset_manager)
        self.settings_menu = SettingMenu(self.screen)
        self.current_state = current_state
        self.running = True

    def load(self):
        self.asset_manager.load_image("start_menu_bg",
"assets/images/backgrounds/UI-Menu Background.jpg")
```

```

        self.asset_manager.load_image("gameplay_bg",
"assets/images/backgrounds/gameplay_bg.png")
        self.asset_manager.load_music("start_menu_bgm",
"assets/sounds/bgm/In Pursuit of Freedom Loop.mp3")
        self.asset_manager.load_music("gameplay_bgm",
"assets/sounds/bgm/In Freedom to Critical Clash 2.mp3")

def run(self):
    while self.running:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                self.running = False

    current_time = pygame.time.get_ticks()

    if self.current_state == "start_menu":
        self.asset_manager.play_music("start_menu_bgm")

        self.start_menu.draw()
        if current_time - self.last_input_time >
self.input_cooldown:
            selected_option = self.start_menu.handle_input()
            if selected_option == "Start Game":
                self.current_state = "gameplay"
            elif selected_option == "Settings":
                self.current_state = "settings"
            elif selected_option == "Exit Game":
                self.running = False
            if selected_option:
                self.last_input_time = current_time # Reset
cooldown

    elif self.current_state == "settings":
        self.settings_menu.draw()

        if current_time - self.last_input_time >
self.input_cooldown:
            selected_option = self.settings_menu.handle_input()
            self.resolution =
self.resolutions[self.settings_menu.current_resolution_index]

```

```

        if selected_option == "Start Menu":
            self.current_state = "start_menu"
        if selected_option:
            self.last_input_time = current_time # Reset
    cooldown

    elif self.current_state == "gameplay":
        self.asset_manager.play_music("gameplay_bgm")

        self.screen.fill(WHITE)
        self.game.update()

        if event.type == pygame.KEYDOWN and event.key ==
pygame.K_ESCAPE:
            self.game.pause_game()

        elif current_time - self.last_input_time >
self.input_cooldown and self.game.paused:
            selected_option = self.game.pause_menu.handle_input()
            self.resolution =
self.resolutions[self.game.pause_menu.current_resolution_index]

            if selected_option == "Back":
                self.game.unpause_game()
            elif selected_option == "Start Menu":
                self.current_state = "start_menu"
                self.game.reset_game()
            if selected_option:
                self.last_input_time = current_time # Reset
    cooldown

    elif self.game.check_game_over() == "LOSE":
        selected_option = self.game.game_over.handle_input()

        if selected_option == "Retry":
            self.game.reset_game()
        elif selected_option == "Back to Start Menu":
            self.current_state = "start_menu"
            self.game.reset_game()

```

```

        if selected_option:
            self.last_input_time = current_time # Reset
            cooldown

        pygame.display.flip()
        self.clock.tick(FPS)

        self.asset_manager.stop_music()
        pygame.quit()

if __name__ == "__main__":
    game = MainGame()
    game.run()

```

Fungsi Utama dalam `main.py`:

1. Import dan Konfigurasi

- Mengimpor modul `pygame`, konfigurasi layar, FPS, warna, serta komponen `Game`, `StartMenu`, dan `SettingMenu`.
- Mengimpor `AssetManager` untuk manajemen gambar-suara dan `AudioManager` untuk pengaturan audio.

2. Kelas `MainGame`

- `__init__`: Menginisialisasi resolusi layar, objek utama (game, menu, pengelola aset), dan state awal "`start_menu`".
- `load`: Memuat gambar dan musik untuk *start menu* dan gameplay.

3. *Game Loop* (`run`)

- Melakukan *game loop* yang terdiri dari event handling, pembaruan state, dan rendering frame baru.
- Menggunakan *state machine* untuk berpindah antar state (`start_menu`, `settings`, `gameplay`).

4. Logika State

- **start_menu**: Menampilkan menu utama, memutar musik, dan menangani input (mulai game, pengaturan, keluar).
- **settings**: Mengatur resolusi layar dan kembali ke menu utama.
- **gameplay**: Menjalankan logika utama game, termasuk *pause*, transisi ke *game over*, dan pembaruan layar.

5. Penanganan Keluar

- Menghentikan musik dan menutup aplikasi dengan `pygame.quit()`.

6. Pemanggilan Kelas

- Menjalankan game dengan memanggil `MainGame.run()` pada eksekusi langsung file.

3.3 Implementasi OOP pada Game

3.3.1 Encapsulation pada Class

Encapsulation diterapkan dengan memisahkan atribut dan metode dalam kelas secara modular serta menggunakan pengaturan aksesibilitas seperti *public* dan *protected*. Sebagai contoh, kelas `AssetManager` memiliki metode khusus untuk memuat dan mengakses asset seperti gambar dan musik. Pengguna hanya berinteraksi dengan metode publik tanpa mengetahui detail implementasi internalnya, sehingga mempermudah pemeliharaan kode. Pendekatan ini memastikan setiap objek hanya dapat diakses melalui metode yang telah ditentukan, mencegah modifikasi yang tidak diinginkan.

3.3.2 Inheritance untuk Reusabilitas

Inheritance digunakan untuk meningkatkan reusabilitas kode dengan mengadopsi sifat kelas induk ke kelas turunan. Misalnya, `Enemy` mewarisi sifat dari kelas `Entity`, sehingga semua logika dasar seperti posisi, pergerakan, dan deteksi tabrakan dapat digunakan kembali. Hal ini memudahkan pengembangan musuh baru dengan menambahkan metode atau atribut khusus tanpa harus mengulang implementasi dasar. Dengan begitu, kode menjadi lebih terstruktur dan mudah dikembangkan.

3.3.3 Polymorphism untuk Mekanik Karakter

Polymorphism memungkinkan metode yang sama memiliki perilaku berbeda di kelas turunan. Sebagai contoh, metode `attack()` pada kelas induk `Entity` diimplementasikan secara berbeda pada kelas turunan seperti `Player` dan `Enemy`. Pemain dapat menyerang menggunakan senjata dengan efek visual khusus, sedangkan musuh memiliki pola serangan unik. Pendekatan ini memberikan fleksibilitas dalam pengembangan mekanik karakter tanpa mengubah logika utama di kelas induk, sehingga mudah diadaptasi untuk kebutuhan gameplay yang lebih kompleks.

3.3.4 Polymorphism untuk Mekanik Karakter

Abstraction diterapkan melalui penggunaan kelas atau metode yang dirancang untuk menyembunyikan detail implementasi, memberikan antarmuka sederhana untuk digunakan. Contohnya adalah kelas abstrak `Entity`, yang mendefinisikan metode dasar seperti `update()` dan `draw()` tanpa memberikan detail implementasi spesifik. Kelas turunan seperti `Player` dan `Enemy` diwajibkan untuk mengimplementasikan metode ini sesuai dengan kebutuhan mereka. Pendekatan ini memudahkan ekspansi game, memungkinkan pengembang untuk menambahkan entitas baru hanya dengan mengikuti antarmuka yang sudah ada tanpa harus memodifikasi struktur inti dari game. Hal ini juga membantu menjaga konsistensi dan mempermudah debugging.

3.4 Penjelasan Mekanik Utama

3.4.1 Mekanik Skill Karakter

Karakter	Skill	Deskripsi	Input Button
Sri	Kecepatan Tinggi dan Tidak Terlihat	Sri bergerak sangat cepat dan menjadi tidak terlihat selama beberapa detik, memungkinkan untuk menghindari musuh dan mengatur ulang posisinya.	WASD untuk bergerak, F untuk skill.
Rama	Tameng Pelindung dan Healing Area	Rama menciptakan tameng pelindung yang mendorong musuh menjauh dan menghasilkan area penyembuhan untuk memulihkan HP karakter di sekitarnya.	Arrow Keys untuk bergerak, Shift Right untuk skill.

3.4.2 Collision Detection

Sistem collision detection digunakan untuk mendeteksi interaksi antara objek seperti karakter, musuh, peluru, dan elemen arena. Mekanik ini berbasis bounding box menggunakan `rect` pada Pygame, yang memungkinkan:

- **Damage:** Musuh menerima damage ketika terkena peluru atau serangan skill.
- **HP Loss:** Karakter kehilangan HP saat menyentuh musuh.
- **Boundary Control:** Membatasi karakter dan musuh agar tetap berada dalam arena.

Panduan input ini memastikan responsivitas optimal dalam mengontrol interaksi berbasis tabrakan di game.

3.4.3 Implementasi Interaksi UI (HP Bar dan Coin)

Sistem HP Bar memberikan informasi visual tentang kondisi kesehatan karakter. Bar ini diperbarui secara dinamis menggunakan proporsi HP terhadap total HP.

- HP Reduction: Terjadi saat karakter terkena serangan musuh.
- HP Recovery: Terjadi saat Rama menggunakan area penyembuhannya.

Sementara itu, Coin digunakan sebagai indikator pencapaian, yang bertambah setiap kali karakter mengambil koin di arena. Panduan inputnya sederhana: mendekati koin untuk otomatis mengambilnya.

- Visualisasi Koin: Ditampilkan di UI menggunakan font yang diperbarui secara real-time, memberikan insentif untuk eksplorasi.

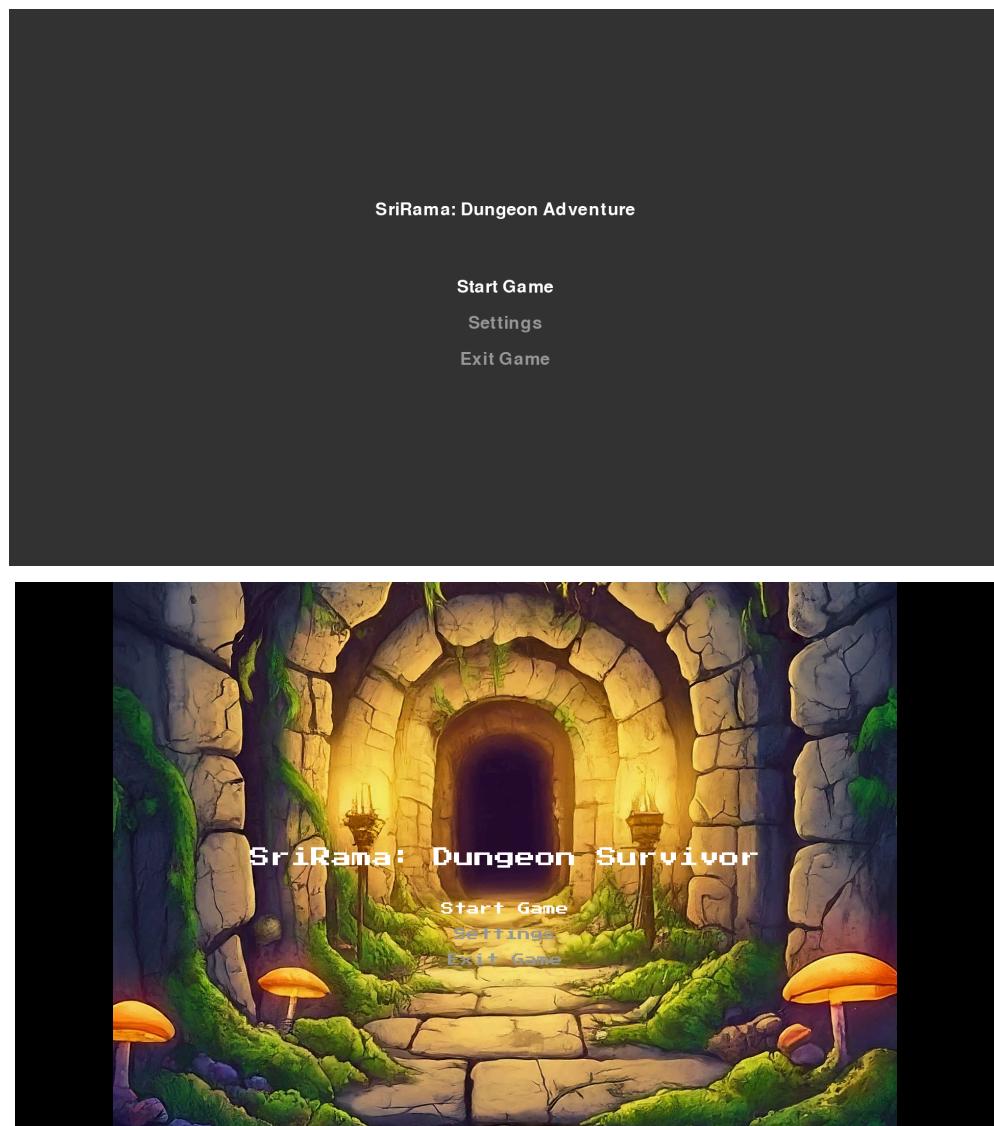
BAB IV

Hasil dan Pembahasan

4.1 Hasil Pengembangan Game

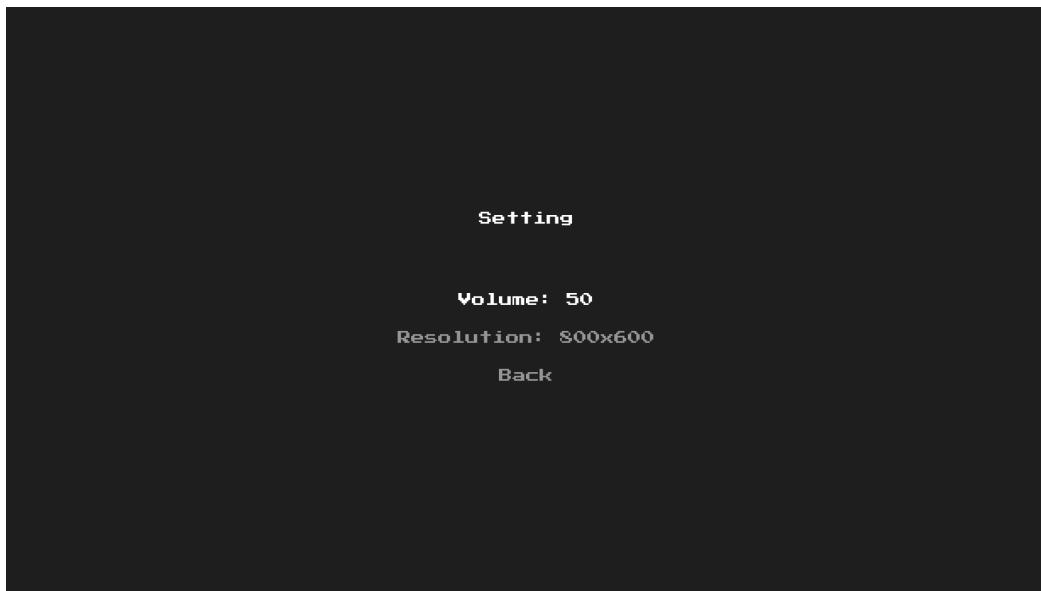
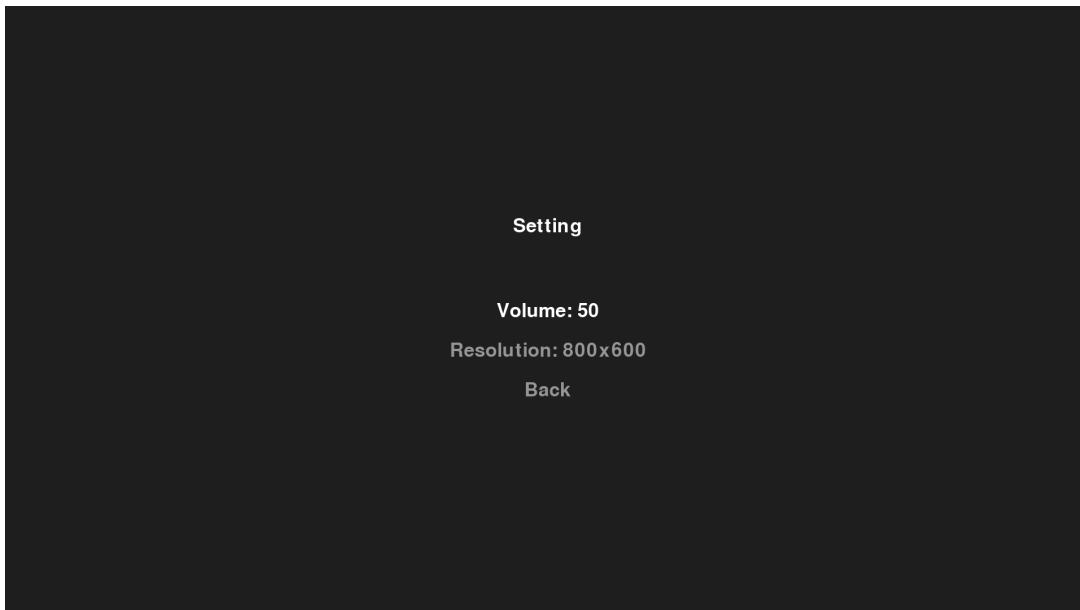
4.1.1 Screenshot dan Demonstrasi Fitur

➤ Start Menu



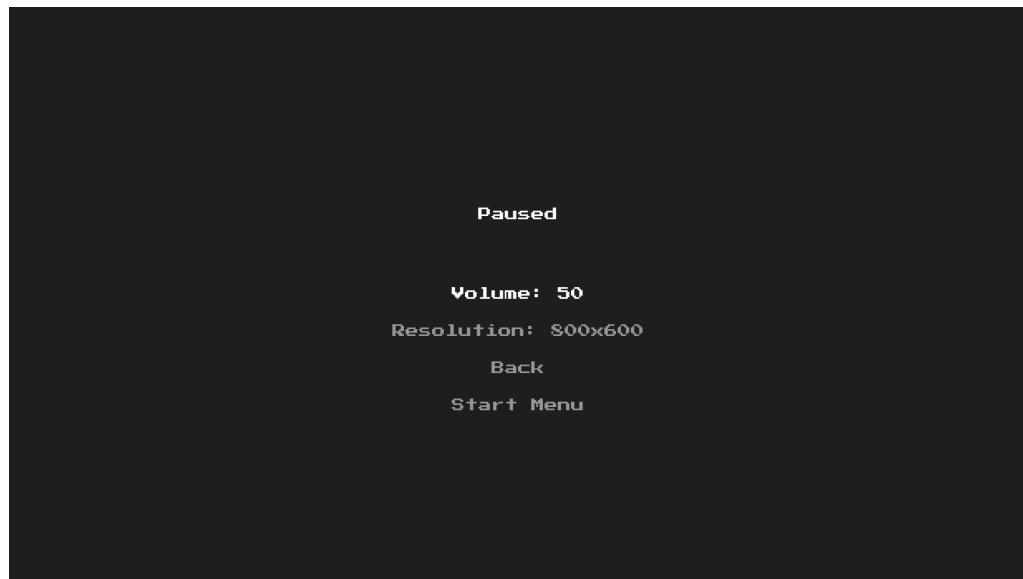
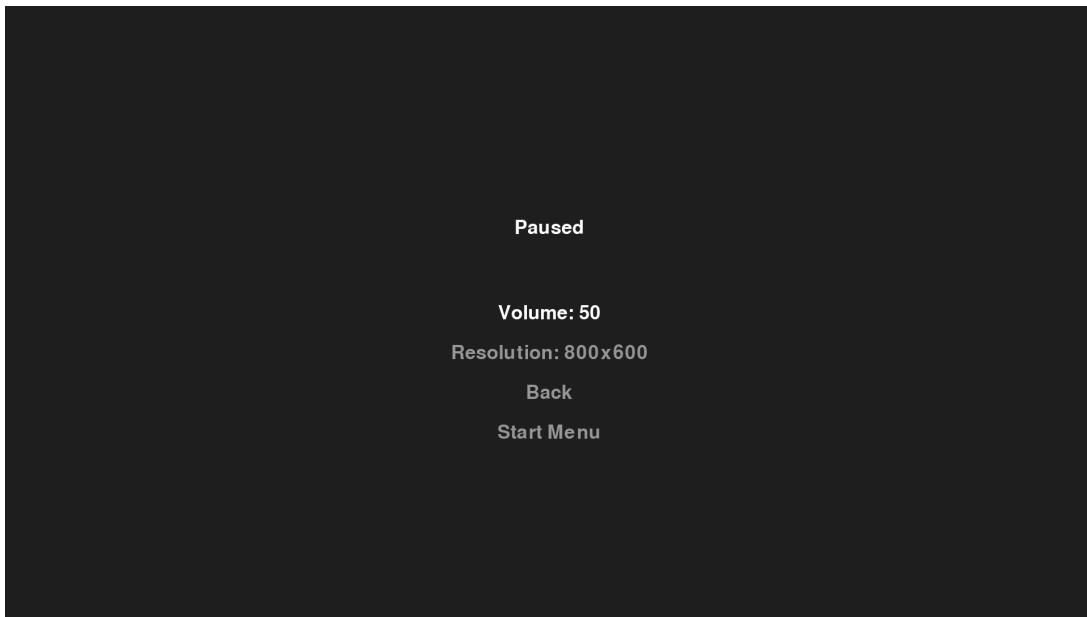
Start Menu adalah pintu masuk utama ke dalam permainan. Pada menu ini, pemain dapat memilih untuk memulai permainan baru, mengakses pengaturan, atau keluar dari aplikasi. Tampilan Start Menu dirancang sederhana dengan navigasi yang intuitif, memanfaatkan tombol yang mudah dikenali. Selain itu, animasi halus ditampilkan untuk menambah daya tarik visual, seperti efek hover pada tombol. Musik latar yang diputar di menu memberikan nuansa awal petualangan yang mengesankan.

➤ **Setting Menu**



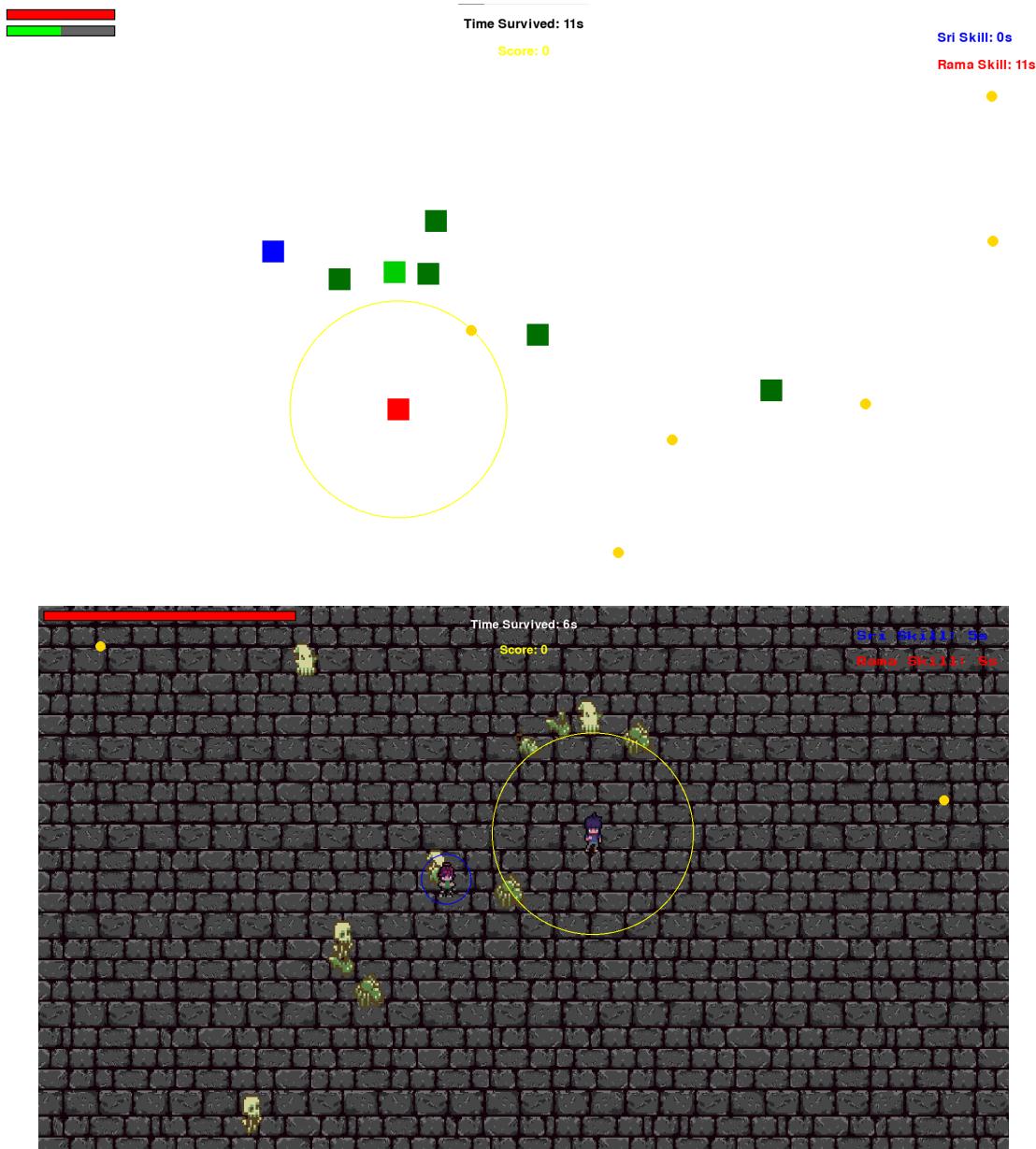
Setting Menu memungkinkan pemain untuk menyesuaikan pengalaman bermain sesuai preferensi mereka. Opsi yang tersedia mencakup pengaturan suara (volume musik dan efek suara), resolusi layar, dan konfigurasi kontrol. Setiap perubahan diterapkan secara real-time untuk memberikan pengalaman yang responsif. Antarmuka Setting Menu dibuat dengan fokus pada kemudahan penggunaan, lengkap dengan slider, dropdown, dan label yang jelas. Fitur "Reset to Default" juga disediakan untuk mengembalikan semua pengaturan ke nilai awal.

➤ Pause Menu



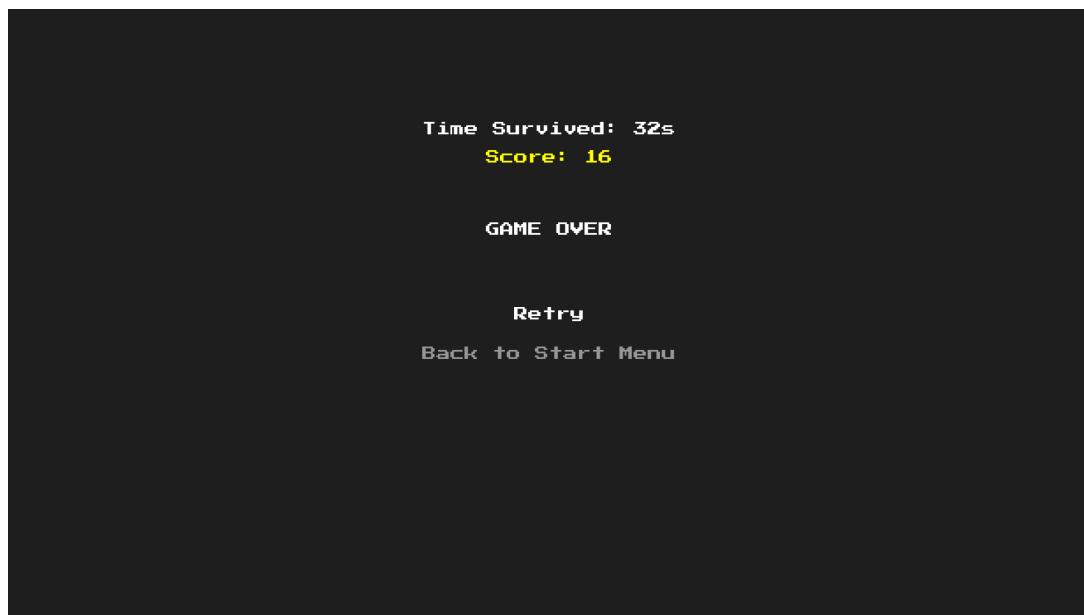
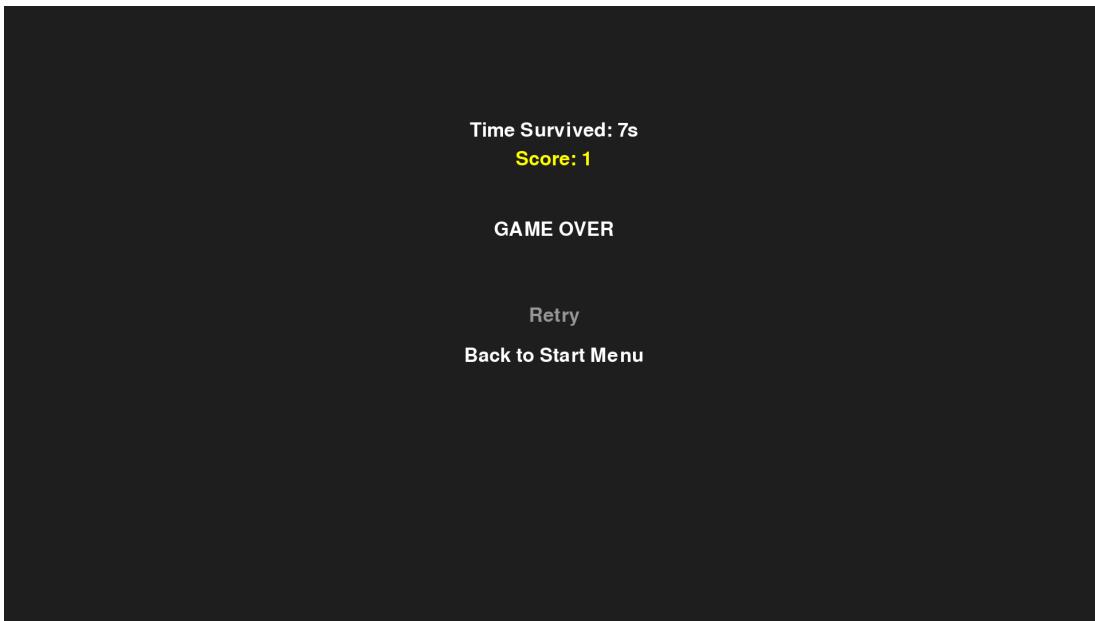
Pause Menu dirancang untuk memberikan kontrol penuh kepada pemain selama permainan berlangsung. Dengan menekan tombol tertentu, pemain dapat menjeda permainan kapan saja. Dalam menu ini, tersedia opsi untuk melanjutkan permainan, kembali ke menu utama, atau mengakses pengaturan tanpa kehilangan progres permainan. Visual latar belakang semi-transparan memberikan fokus pada elemen menu sekaligus menjaga konteks permainan yang sedang dijeda.

➤ Gameplay Scene



Gameplay Scene adalah inti dari pengalaman bermain. Pemain mengendalikan dua karakter utama, Sri dan Rama, dalam sebuah arena penuh tantangan. Elemen-elemen seperti musuh yang muncul secara dinamis, koin yang dapat dikumpulkan, dan indikator UI (seperti HP Bar dan Skill Cooldown) dirancang untuk menjaga keterlibatan pemain. Latar belakang yang menyesuaikan tema petualangan dungeon menambah kedalaman atmosfer permainan. Efek visual, seperti gempa layar saat menerima damage, memperkuat imersi dan memberikan feedback langsung terhadap tindakan pemain.

➤ Game Over Scene



Game Over Scene muncul ketika pemain kalah atau berhasil menyelesaikan permainan. Tampilan ini menyajikan skor akhir, waktu bertahan, dan status pencapaian, memberikan evaluasi performa pemain selama sesi bermain. Dengan opsi untuk mencoba kembali atau kembali ke menu utama, pemain didorong untuk terus meningkatkan kemampuan mereka. Desain visual yang dramatis, dilengkapi dengan musik yang menyesuaikan hasil permainan, membantu menciptakan dampak emosional yang sesuai.

4.1.2 Fitur-Fitur yang Berjalan

No	Fitur	Deskripsi	Status
1	Start Menu	Pemain dapat memulai permainan baru, mengakses pengaturan, atau keluar dari aplikasi.	<input checked="" type="checkbox"/> Berfungsi penuh
2	Setting Menu	Opsi untuk mengatur volume suara, resolusi layar, dan konfigurasi kontrol.	<input checked="" type="checkbox"/> Berfungsi penuh
3	Pause Menu	Pemain dapat menjeda permainan, melanjutkan, mengakses pengaturan, atau kembali ke menu utama.	<input checked="" type="checkbox"/> Berfungsi penuh
4	Gameplay Scene	Arena interaktif dengan elemen seperti musuh dinamis, koin, dan mekanisme skill.	<input checked="" type="checkbox"/> Berfungsi penuh
5	Game Over Scene	Menampilkan hasil akhir permainan (skor dan waktu), dengan opsi untuk bermain lagi.	<input checked="" type="checkbox"/> Berfungsi penuh
6	Enemy Spawning System	Musuh muncul secara acak di arena dengan pola dan interval spawn yang dinamis.	<input checked="" type="checkbox"/> Berfungsi penuh
7	Coin Spawning and Collection	Koin muncul di arena secara acak dan dapat dikumpulkan oleh pemain untuk menambah skor.	<input checked="" type="checkbox"/> Berfungsi penuh
8	Skill Activation	Pemain dapat menggunakan skill unik karakter (dash untuk Sri dan healing untuk Rama).	<input checked="" type="checkbox"/> Berfungsi penuh
9	HP Bar and Skill Cooldown UI	Menampilkan informasi kesehatan dan status cooldown skill masing-masing karakter.	<input checked="" type="checkbox"/> Berfungsi penuh
10	Score Tracking and Display	Skor pemain dihitung berdasarkan koin yang dikumpulkan dan ditampilkan secara real-time.	<input checked="" type="checkbox"/> Berfungsi penuh
11	Timer untuk Survival Time	Menampilkan durasi bertahan hidup pemain selama permainan berlangsung.	<input checked="" type="checkbox"/> Berfungsi penuh
12	Reset Game	Permainan dapat diatur ulang dengan kondisi awal tanpa perlu keluar dari aplikasi.	<input checked="" type="checkbox"/> Berfungsi penuh

4.2 Evaluasi Gameplay

4.2.1 Pengalaman Pemain

Pengalaman bermain *SriRama: Dungeon Adventure* menghadirkan kombinasi yang menyenangkan antara tantangan dan kerjasama. Pemain melaporkan bahwa mekanisme skill yang unik untuk setiap karakter memberikan dimensi strategis yang menarik, mendorong interaksi dan koordinasi antara pemain. **Sri**, dengan kemampuan *dash*-nya, efektif untuk menghindari serangan dan menciptakan peluang menyerang, sementara **Rama** yang memiliki kemampuan *healing* menjadi kunci untuk mempertahankan kelangsungan permainan.

Desain UI sederhana namun fungsional membantu pemain memahami status permainan, seperti HP dan cooldown skill, dengan mudah. Efek visual seperti gempa layar saat menerima damage memberikan rasa imersi yang lebih dalam. Namun, beberapa pemain merasa bahwa kesulitan permainan meningkat terlalu cepat, terutama ketika jumlah musuh menjadi lebih banyak. Meskipun demikian, secara keseluruhan, pengalaman gameplay dinilai memuaskan dengan potensi untuk penyempurnaan lebih lanjut.

4.2.2 Analisis Kelebihan dan Kekurangan

Kelebihan:

- **Mekanisme Gameplay:** Gameplay yang mendukung kerja sama tim menciptakan pengalaman bermain yang unik dan menyenangkan.
- **Visual Placeholder:** Meskipun sederhana, elemen visual dan animasi berjalan dengan lancar dan mendukung fungsionalitas permainan.
- **Fitur Dinamis:** Sistem spawn musuh dan koin yang adaptif memastikan tantangan tetap menarik bagi pemain dari berbagai tingkat keahlian.
- **Efek Imersif:** Elemen seperti gempa layar dan animasi yang responsif meningkatkan kualitas estetika dan rasa keterlibatan dalam permainan.
- **Sistem UI yang Jelas:** Informasi kesehatan dan cooldown skill disampaikan dengan baik melalui antarmuka sederhana yang intuitif.

Kekurangan:

- **Kurva Kesulitan:** Peningkatan tingkat kesulitan terasa terlalu tajam, membuat pemain pemula kesulitan untuk beradaptasi.
- **Variasi Konten:** Musuh dan arena masih memiliki variasi yang terbatas, yang dapat membuat permainan terasa repetitif dalam jangka panjang.
- **Balancing Gameplay:** Beberapa pemain merasa bahwa karakter **Rama** lebih berperan pasif dibandingkan **Sri**, sehingga perlu ada penguatan mekanisme skill-nya.
- **Pengaturan Resolusi:** Meskipun berfungsi, pengaturan resolusi layar belum sepenuhnya optimal untuk semua perangkat.

4.3 Pembahasan Teknis

4.3.1 Efisiensi Struktur Kode

Pengembangan *SriRama: Dungeon Adventure* menggunakan pendekatan Object-Oriented Programming (OOP) yang bertujuan untuk menjaga struktur kode modular dan efisien. Setiap entitas dalam game, seperti pemain, musuh, dan elemen UI, dikapsulasi dalam kelas tersendiri, memungkinkan pemisahan tanggung jawab yang jelas. Folder yang terorganisir dengan baik, seperti `game_objects` untuk entitas, dan modul seperti `config.py` untuk parameter global, mendukung pengelolaan kode yang rapi dan mudah dikembangkan.

Fitur utama seperti sistem collision, efek visual, dan pengaturan UI dirancang untuk memanfaatkan logika modular, yang tidak hanya mengurangi pengulangan kode tetapi juga mempermudah pengujian dan debugging. Penerapan *game loop* dengan metode terpisah untuk `draw()` dan `update()` memastikan siklus gameplay yang responsif, sekaligus menjaga kinerja pada frame rate yang stabil.

Namun, masih ada ruang untuk peningkatan, seperti penggunaan pustaka animasi yang lebih terstruktur untuk sprite atau penyempurnaan fungsi event handler yang lebih adaptif terhadap perubahan input pemain.

4.3.2 Tantangan Selama Pengembangan

Pengembangan game ini menghadapi beberapa tantangan teknis yang relevan dengan realita proses pengembangan game, yaitu:

- 1. Mengatur FPS untuk Draw dan Update Game Objek:**

Tantangan ini muncul dalam memastikan bahwa elemen gameplay, seperti pergerakan karakter, animasi, dan spawn musuh, tetap berjalan dengan mulus tanpa gangguan. Masalah sinkronisasi antara kecepatan update logika game dan rendering frame menjadi salah satu kendala utama. Best practice yang diterapkan adalah penggunaan metode `pygame.time.Clock()` untuk mengontrol FPS, serta memastikan bahwa logika game tidak bergantung langsung pada kecepatan rendering.

- 2. Mengatur Integrasi Resolusi Layar dan Volume:**

Resolusi layar yang dapat disesuaikan merupakan fitur yang memerlukan perencanaan matang, terutama untuk menjaga skala antarmuka pengguna dan elemen grafis tetap proporsional pada berbagai ukuran layar. Tantangan ini memerlukan penyesuaian menggunakan transformasi *scale* dan pengujian menyeluruh pada berbagai konfigurasi. Integrasi volume audio juga menghadapi kendala dalam memastikan bahwa slider pengaturan memberikan perubahan yang linear dan tidak terlalu sensitif, yang akhirnya diselesaikan dengan metode interpolasi nilai.

3. Konfigurasi Animasi Sprite Asset yang Lebih Customizable:

Dalam pengembangan, membuat animasi sprite yang fleksibel dan modular menjadi tantangan, terutama untuk memungkinkan variasi gerakan tanpa menambah kompleksitas kode. Solusi sementara menggunakan pendekatan berbasis daftar untuk menyimpan frame animasi dan fungsi untuk mengatur transisi frame. Namun, dalam praktik terbaik, penggunaan pustaka tambahan seperti *Pyganim* atau sistem animasi internal yang lebih generik dapat menjadi langkah optimal untuk proyek di masa depan.

Dengan menyelesaikan tantangan-tantangan ini, pengembangan game tidak hanya memberikan hasil yang sesuai dengan ekspektasi tetapi juga memberikan wawasan berharga tentang area teknis yang dapat disempurnakan pada iterasi berikutnya.



BAB V

Penutup

5.1 Kesimpulan

Pengembangan *SriRama: Dungeon Survivor* adalah perjalanan yang mencakup berbagai aspek desain, implementasi, dan pengujian sebuah game berbasis Python menggunakan pustaka Pygame. Proyek ini berhasil mengintegrasikan mekanik gameplay sederhana namun menarik, dengan konsep dua pemain yang harus bekerja sama untuk bertahan hidup. Berikut adalah poin-poin kesimpulan utama:

1. Penerapan Konsep Game

- ❖ *SriRama: Dungeon Survivor* dirancang untuk memberikan pengalaman bermain yang menyenangkan dengan gameplay kooperatif, di mana pemain harus memanfaatkan kekuatan unik dari kedua karakter, Sri dan Rama, untuk mencapai tujuan. Konsep kerja sama menjadi inti permainan dan memberikan tantangan serta kepuasan saat berhasil dicapai.

2. Implementasi Teknikal

- ❖ Implementasi menggunakan prinsip OOP memungkinkan pengembangan yang modular, terstruktur, dan mudah untuk diperluas. Penerapan *encapsulation*, *inheritance*, dan *polymorphism* memastikan bahwa kode tidak hanya efisien tetapi juga dapat digunakan kembali.
- ❖ Pygame sebagai kerangka kerja memberikan dukungan untuk rendering grafis, pengelolaan input, deteksi tabrakan, dan manajemen waktu secara efektif. Hal ini memungkinkan pencapaian performa optimal pada permainan sederhana seperti ini.

3. Mekanik dan Interaksi

- ❖ Mekanik utama, seperti kemampuan unik masing-masing karakter, sistem deteksi tabrakan, dan visualisasi UI seperti HP Bar, berhasil diimplementasikan dengan baik. Fitur-fitur ini memberikan pengalaman bermain yang imersif dan mendorong kolaborasi antar pemain.

4. Tujuan Hiburan

- ❖ Game ini tidak hanya menawarkan nostalgia kepada pemain dengan gaya permainan sederhana seperti *Fireboy and Watergirl*, tetapi juga menargetkan kepuasan bermain bersama secara lokal, yang jarang ditemukan dalam game modern saat ini. *SriRama: Dungeon Survivor* menjadi media hiburan yang menekankan kerja sama dan interaksi sosial.

5.2 Evaluasi Keseluruhan Proyek

Proyek *SriRama: Dungeon Survivor* memberikan gambaran tentang bagaimana pengembangan game sederhana dapat dilakukan secara terstruktur dengan pendekatan berbasis OOP dan menggunakan pustaka Pygame. Namun, dalam perjalanan pengembangannya, terdapat beberapa aspek yang perlu dievaluasi untuk memberikan wawasan mengenai pencapaian, tantangan, dan peluang perbaikan.

Keberhasilan Proyek

1. Pencapaian Teknis

- ❖ Penerapan prinsip OOP memungkinkan game ini memiliki modularitas yang tinggi, membuat setiap bagian kode dapat dikembangkan dan dipelihara dengan mudah.
- ❖ Modularisasi melalui pembagian kode ke dalam beberapa *module* seperti `game.py`, `main.py`, dan `config.py` memberikan efisiensi dalam pengelolaan logika permainan.
- ❖ Penggunaan mekanik berbasis deteksi tabrakan, kemampuan unik masing-masing karakter, dan desain interaksi UI (seperti HP bar) berhasil meningkatkan pengalaman bermain.

2. Pencapaian Desain Gameplay

- ❖ Konsep kerja sama antara dua karakter yang saling melengkapi (Sri dan Rama) memberikan nilai tambah unik.
- ❖ Gameplay yang sederhana namun menarik berhasil menghadirkan elemen hiburan yang ringan, cocok untuk pemain yang menginginkan pengalaman bermain santai dengan teman.

3. Kesesuaian dengan Tujuan Awal

- ❖ Game ini berhasil mengimplementasikan tujuan utamanya, yaitu menciptakan game berbasis hiburan dengan mekanik dua pemain lokal yang menekankan kolaborasi.

Kelemahan yang Ditemukan

1. Keterbatasan Visual dan Audio

- Aset grafis dan audio yang digunakan masih bersifat minimal, sehingga kurang memberikan daya tarik visual dan atmosfer yang lebih imersif.
- Animasi karakter dan musuh dapat lebih ditingkatkan untuk memberikan pengalaman yang lebih hidup.

2. Fitur yang Terbatas

- Tidak adanya sistem peningkatan level atau variasi musuh membuat gameplay menjadi repetitif dalam jangka panjang.
- Kurangnya elemen kompetitif atau tantangan tambahan, seperti pencapaian (achievements) atau papan skor, dapat mengurangi daya tarik bagi pemain yang mencari replayability.

3. Kurangnya Optimalisasi Kinerja

- Pada perangkat dengan spesifikasi rendah, beberapa elemen game seperti rendering banyak objek atau pengelolaan *collision detection* dapat menyebabkan penurunan performa.
- Pygame, meskipun efektif untuk game sederhana, memiliki keterbatasan dalam pengelolaan grafis yang lebih kompleks.

Tantangan yang Dihadapi

1. Manajemen Waktu

- Pembagian waktu antara perancangan, pengembangan, dan pengujian menjadi tantangan besar, terutama dalam memastikan kualitas fitur yang diimplementasikan.
- Penambahan mekanik baru seperti interaksi UI dan deteksi tabrakan membutuhkan waktu iterasi yang cukup panjang.

2. Penyesuaian Konsep Gameplay

- Selama pengembangan, perubahan konsep dari fokus edukasi menjadi hiburan membutuhkan perombakan desain awal, yang menambah kompleksitas dalam pelaksanaan proyek.

3. Pengujian Gameplay Dua Pemain

- Pengujian untuk mode dua pemain lokal memerlukan penyesuaian khusus, baik dalam hal kontrol maupun dinamika gameplay, yang membutuhkan iterasi desain dan pengembangan.

Peluang Perbaikan

- 1. Pengembangan Konten Tambahan**
 - Penambahan elemen seperti level, jenis musuh, dan mekanik progresi dapat membuat game lebih menarik bagi pemain jangka panjang.
 - Sistem hadiah seperti koin untuk membuka karakter atau elemen kosmetik dapat menambah variasi gameplay.
- 2. Peningkatan Estetika**
 - Aset grafis yang lebih kaya dan efek suara yang lebih beragam dapat memberikan pengalaman bermain yang lebih memuaskan.
 - Implementasi transisi halus (seperti animasi *loading screen* atau efek partikel) dapat meningkatkan kualitas keseluruhan.
- 3. Optimisasi Kinerja**
 - Penggunaan algoritma deteksi tabrakan yang lebih efisien dan manajemen *loop* game dapat meningkatkan performa, terutama pada perangkat dengan spesifikasi rendah.
- 4. Ekspansi Platform**
 - Pengembangan versi game untuk perangkat seluler atau integrasi mode online dapat menjangkau audiens yang lebih luas.

Kesimpulan Evaluasi

Proyek *SriRama: Dungeon Survivor* adalah contoh sukses pengembangan game sederhana dengan pendekatan teknis yang terstruktur dan fokus pada gameplay yang kooperatif. Meskipun memiliki beberapa keterbatasan, seperti minimnya variasi fitur dan estetika, proyek ini tetap memenuhi tujuan utamanya dan memberikan dasar yang kuat untuk pengembangan lebih lanjut. Evaluasi ini menjadi langkah penting untuk memetakan area yang perlu diperbaiki dan peluang yang dapat dioptimalkan di masa depan.

5.3 Saran dan Rencana Pengembangan Selanjutnya

Proyek *SriRama: Dungeon Survivor* memiliki potensi besar untuk terus dikembangkan menjadi permainan yang lebih kompleks dan menarik. Berdasarkan evaluasi keseluruhan, berikut adalah saran dan rencana pengembangan selanjutnya yang dapat meningkatkan kualitas, daya tarik, dan cakupan permainan ini:

Saran untuk Pengembangan Selanjutnya

1. Peningkatan Variasi Gameplay

- **Penambahan Level dan Lingkungan Baru:**

Menambahkan variasi arena dengan tema unik, seperti hutan, gua, atau benteng, untuk meningkatkan eksplorasi pemain.

- **Jenis Musuh yang Lebih Beragam:**

Menambahkan variasi musuh dengan pola serangan dan atribut yang berbeda, seperti musuh yang menyerang dari jarak jauh atau yang bergerak cepat.

- **Mekanisme Tantangan:**

Penerapan tantangan seperti *time-limited objectives* atau misi spesifik dapat memberikan dinamika tambahan dalam permainan.

2. Sistem Progresi dan Pencapaian

- **Sistem Pengumpulan dan Penggunaan Koin:**

Koin yang dikumpulkan dapat digunakan untuk membuka karakter, arena baru, atau kemampuan khusus.

- **Papan Skor dan Leaderboard:**

Menambahkan fitur *leaderboard* untuk meningkatkan elemen kompetisi antara pemain.

- **Pencapaian (Achievements):**

Memberikan penghargaan atas pencapaian tertentu, seperti jumlah musuh yang dikalahkan atau waktu bertahan terlama.

3. Pengayaan Visual dan Audio

- **Animasi dan Efek Visual:**

Menambahkan animasi yang lebih dinamis, seperti efek serangan, ledakan, atau partikel saat karakter bergerak.

- **Soundtrack dan Efek Suara:**

Peningkatan kualitas audio, seperti tema musik untuk setiap level atau efek suara yang lebih kaya, dapat menciptakan pengalaman yang lebih imersif.

4. Peningkatan Interaksi Pemain

- **Mode Permainan Alternatif:**

Menambahkan mode permainan tambahan, seperti *Survival Endless Mode* atau *Time Attack Mode*.

- **Integrasi Online Multiplayer:**
Meningkatkan aksesibilitas dengan menambahkan fitur bermain bersama secara daring (online co-op).
- **Panduan dan Tutorial:**
Menyediakan tutorial yang interaktif untuk membantu pemain baru memahami kontrol dan mekanik permainan.

5. Optimisasi dan Ekspansi Platform

- **Optimisasi Kinerja:**
Memastikan efisiensi algoritma, terutama pada perangkat dengan spesifikasi rendah, seperti penggunaan algoritma *spatial partitioning* untuk deteksi tabrakan.
- **Portabilitas ke Platform Lain:**
Mengembangkan game untuk platform seperti Android, iOS, atau konsol kecil (seperti Raspberry Pi) untuk menjangkau audiens yang lebih luas.

Rencana Pengembangan Selanjutnya

1. Tahap Perencanaan dan Penelitian

- Melakukan penelitian lebih lanjut untuk memahami kebutuhan pemain melalui survei atau uji coba (*playtesting*).
- Menetapkan prioritas pengembangan berdasarkan evaluasi dan umpan balik.

2. Pengembangan dan Iterasi

- **Sistem Progresi:**
Memulai pengembangan sistem progresi koin dan leaderboard sebagai langkah pertama peningkatan replayability.
- **Variasi Konten:**
Mendesain level baru dan jenis musuh sebagai iterasi awal untuk memperkaya gameplay.
- **Audio-Visual:**
Menambahkan aset grafis dan audio yang lebih menarik sesuai tema cerita dan estetika game.

3. Pengujian dan Validasi

- Melakukan pengujian intensif pada setiap iterasi pengembangan untuk memastikan pengalaman bermain yang stabil dan optimal.
- Menggunakan analitik permainan untuk memahami pola perilaku pemain dan area yang memerlukan peningkatan.

4. Distribusi dan Promosi

- Mendistribusikan game di platform populer seperti *Steam*, *Google Play Store*, atau *itch.io*.
- Menggunakan media sosial dan forum komunitas game untuk mempromosikan permainan dan mendapatkan umpan balik lebih lanjut.

Kesimpulan Rencana Pengembangan

Dengan mengikuti saran dan rencana pengembangan ini, *SriRama: Dungeon Survivor* dapat berkembang menjadi game yang lebih menarik, kompetitif, dan dinikmati oleh khalayak yang lebih luas. Penambahan fitur dan optimisasi pengalaman bermain akan memperkuat posisi game ini sebagai hiburan ringan yang dapat dinikmati bersama, tetap mempertahankan esensi kerjasama seirama antara dua pemain.

Daftar Pustaka

- [1] Statista. (n.d.). *Video Games - Indonesia | Statista Market Forecast.* <https://www.statista.com/outlook/dmo/digital-media/video-games/indonesia>.
- [2] Lavinda. (2023, April 19). Pasar Game Mobile Indonesia Terbesar Ketiga di Dunia. *Katadata.* <https://katadata.co.id/digital/teknologi/643f7867520fd/pasar-game-mobile-indonesia-terbesar-ketiga-di-dunia>.
- [3] Hermansyah, H., Nurhairunnisah, N., Ardianti, N. S., & Sulindra, N. I. G. M. (2023). Pengaruh Penggunaan Game Edukasi terhadap Kemampuan Kognitif Fisika Dilihat dari Gender Siswa. *JURNAL PENDIDIKAN MIPA*, 13(3), 833–838. <https://doi.org/10.37630/jpm.v13i3.1184>.
