LAPORAN PROYEK UAS PEMROGRAMAN BERORIENTASI OBJEK

Judul Aplikasi:

Harvest Arena: Isekai Knight Saga

Oleh:

M. Fikkan El Haq

24091397058

2024B/25

PROGRAM STUDI MANAJEMEN INFORMATIKA

FAKULTAS VOKASI

UNIVERSITAS NEGERI SURABAYA

2025

BAB I
PENDAHULUAN

1.1 Latar Belakang

Industri pengembangan game modern kini tidak hanya berfokus pada mekanik permainan yang statis, tetapi mulai mengadopsi kecerdasan buatan (Artificial Intelligence) untuk menciptakan pengalaman yang dinamis. Proyek akhir ini terinspirasi dari permainan klasik bergenre Farming RPG (seperti Harvest Moon), yang merupakan studi kasus ideal untuk penerapan Pemrograman Berorientasi Objek (PBO) karena banyaknya interaksi antar objek (pemain, tanaman, musuh, dan lingkungan).

Keunikan utama dari proyek "Harvest Arena: Isekai Knight Saga" ini adalah penggabungan konsep OOP yang kokoh dengan teknologi Generative AI (Google Gemini). Penulis mengimplementasikan sistem di mana misi permainan (Quest) tidak dibuat secara manual (hardcoded), melainkan dihasilkan secara otomatis oleh AI, memberikan variasi permainan yang tidak terbatas.

Tantangan teknis utama dalam pengembangan ini adalah mengintegrasikan panggilan API (API Call) ke dalam Game Loop tanpa menghentikan jalannya permainan. Oleh karena itu, penulis menerapkan teknik Multithreading untuk menangani permintaan AI di latar belakang, serta sistem serialisasi JSON untuk penyimpanan data yang persisten.

1.2 Rumusah Masalah

Berdasarkan latar belakang tersebut, rumusan masalah yang diangkat dalam proyek ini adalah:

1. Bagaimana merancang struktur kelas (Class) yang efisien untuk membedakan perilaku entitas (Hero vs Enemy) menggunakan konsep Inheritance dan Polymorphism?
2. Bagaimana menerapkan konsep Encapsulation untuk melindungi status vital pemain (HP, Gold, XP) dari manipulasi yang tidak valid?
3. Bagaimana mengintegrasikan API Google Gemini untuk fitur Dynamic Quest tanpa menyebabkan lag atau freeze pada aplikasi?
4. Bagaimana membangun sistem penyimpanan data (Save/Load) yang mampu menyimpan kondisi kompleks seperti status ratusan petak tanah (FarmTile) dan inventaris pemain?

1.3 Tujuan

Tujuan dari pengembangan aplikasi ini adalah:

1. Tujuan Akademis: Memenuhi tugas Proyek Akhir mata kuliah PBO dengan menerapkan pilar utama OOP: Encapsulation, Inheritance, dan Polymorphism.
2. Tujuan Teknis: Menghasilkan aplikasi game berbasis Python (Pygame) yang stabil dengan integrasi teknologi Cloud AI dan manajemen memori yang baik.
3. Tujuan Fungsional: Menyediakan permainan simulasi pertanian dan pertarungan yang interaktif dengan sistem progresi yang dapat disimpan (saveable).

BAB II
ANALISIS DAN PERANCANGAN SISTEM

2.1 Deskripsi Sistem

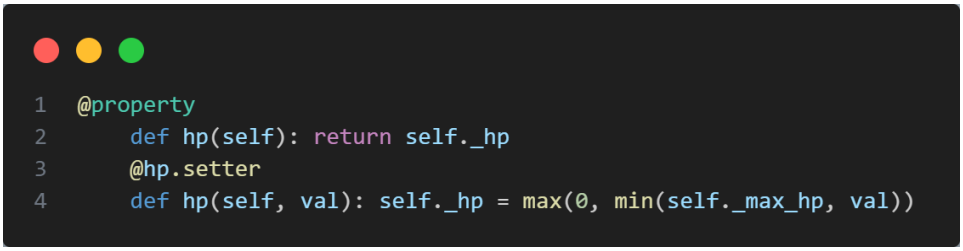
Aplikasi "Harvest Arena" adalah game RPG berbasis desktop yang menggunakan arsitektur Game Loop real-time (60 FPS). Sistem dikelola oleh Controller utama bernama MasterGame yang mengatur transisi antara beberapa State utama:

- Main Menu: Tampilan awal untuk memulai atau memuat permainan.
- Overworld: Mode eksplorasi, pertanian, dan interaksi dengan lingkungan.
- Battle Scene: Mode pertarungan turn-based melawan musuh.
- Shop & Inventory: Antarmuka manajemen sumber daya pemain.

2.2 Penerapan Konsep PBO (Object-Oriented Programming)

Aplikasi ini dibangun menggunakan paradigma PBO penuh. Berikut analisis penerapannya:

- Class dan Object
  Program mendefinisikan kelas sebagai cetak biru.
  - MasterGame: Controller utama.
  - FarmTile: Merepresentasikan setiap petak tanah.
  - Quest: Objek data untuk misi permainan.
- Inheritance (Pewarisan)
  Digunakan untuk menciptakan hierarki entitas agar kode lebih efisien (reusable).
  - Kelas Entity adalah induk (Parent) yang memiliki atribut hp, max_mana, dan metode take_damage().
  - Kelas Hero dan Enemy mewarisi Entity. Hero menambahkan logika input pemain, sedangkan Enemy menambahkan logika AI pergerakan sederhana.
- Encapsulation (Pembungkusan)
  Digunakan pada kelas PlayerStats untuk memvalidasi data.
  - Atribut _hp, _gold, dan _xp bersifat protected.
  - Akses dilakukan melalui Python Decorator @property untuk memastikan HP tidak pernah bernilai negatif.



```python
@property
    def hp(self): return self._hp
    @hp.setter
    def hp(self, val): self._hp = max(0, min(self._max_hp, val))
```

Image 1 Encapsulation

- **Polymorphism (Polimorfisme)** Diterapkan melalui *Method Overriding*. Metode update() pada kelas Hero menangani input keyboard, sedangkan update() pada kelas OverworldEnemy menangani pergerakan acak, meskipun nama metodenya sama.

2.3 Rancangan Desain (Class Diagram)

Struktur arsitektur perangkat lunak "Harvest Arena" dirancang menggunakan pendekatan berorientasi objek yang memisahkan logika permainan, data, dan visualisasi. Berdasarkan Class Diagram yang telah dibuat, berikut adalah analisis mendetail mengenai hubungan antar kelas:

1. Kelas Pengendali Utama (Main Controller) Kelas MasterGame bertindak sebagai Root Class atau pengendali utama sistem. Kelas ini memiliki hubungan Komposisi (Composition) yang kuat dengan sub-sistem vital lainnya, yaitu PlayerStats (Data), WorldSystem (Lingkungan), BattleScene (Pertarungan), dan Hotbar (Antarmuka). Artinya, objek-objek sub-sistem ini diinisialisasi di dalam konstruktor MasterGame dan siklus hidupnya bergantung sepenuhnya pada keberadaan MasterGame. Jika MasterGame dihancurkan, maka seluruh state permainan juga akan berhenti.
2. Hierarki Entitas Karakter (Inheritance) Penerapan konsep Pewarisan (Inheritance) terlihat jelas pada hierarki Entity.
    - Parent Class (Entity): Menyediakan atribut dasar yang dimiliki oleh semua makhluk hidup dalam game, seperti Health Point (hp), Mana (max_mana), posisi koordinat (rect), serta metode umum seperti take_damage() dan logika animasi dasar.
    - Child Classes (Hero & Enemy):
        - Kelas Hero memperluas fungsi Entity dengan menambahkan logika input pengguna (kontrol keyboard) dan manajemen regenerasi mana.
        - Kelas Enemy memperluas fungsi Entity dengan menambahkan logika kecerdasan buatan (AI) sederhana untuk mendeteksi pemain dan melakukan serangan otomatis.
    - Hubungan ini menerapkan prinsip Polimorfisme, di mana metode update() pada Hero dan Enemy memiliki implementasi perilaku

yang berbeda meskipun berasal dari induk yang sama.

3. Sistem Lingkungan (Aggregation) Kelas WorldSystem memiliki hubungan Agregasi dengan kelas FarmTile. Satu objek WorldSystem menampung dan mengelola koleksi (Dictionary) yang berisi ribuan objek FarmTile. Setiap FarmTile merepresentasikan satu petak tanah yang memiliki status independen (seperti biome, crop_type, dan growth_stage). Hubungan ini memungkinkan dunia permainan untuk dimanipulasi secara granular (per petak) namun tetap terkelola dalam satu wadah sistem.
4. Manajemen Data dan Misi (*Association*)
    - Kelas PlayerStats berdiri secara independen untuk menangani Encapsulation data pemain (Level, Gold, Inventory). Kelas ini memiliki Asosiasi dengan kelas Quest.
    - Objek Quest dihasilkan (baik melalui AI maupun hardcoded) dan kemudian disematkan ke dalam PlayerStats untuk dipantau progresnya (misalnya: target panen atau jumlah musuh yang dikalahkan).
5. Antarmuka dan Utilitas
    - Kelas Hotbar mengelola slot item yang dimiliki pemain dan berinteraksi langsung dengan inventaris di PlayerStats.
    - Kelas SoundSynth dan MenuParticle merupakan kelas utilitas pendukung yang dipanggil oleh berbagai kelas lain untuk memberikan umpan balik audio-visual tanpa terikat logika permainan yang ketat.

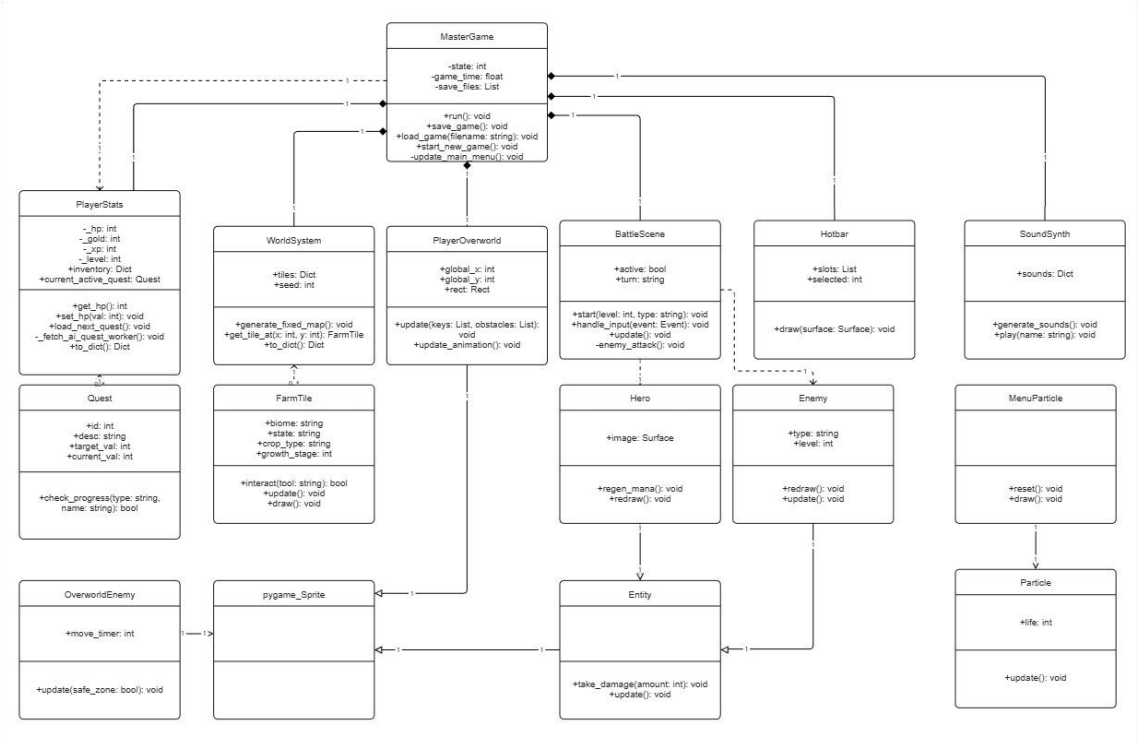


image 2 ClassDiagram

BAB III
IMPLEMENTASI SISTEM

3.1 Tampilan Menu & Prolog

Saat aplikasi dijalankan, sistem menampilkan menu utama dengan animasi partikel. Jika "New Game" dipilih, sistem menampilkan prolog cerita bergaya teks mesin ketik (*typewriter effect*) sebelum masuk ke dunia permainan.

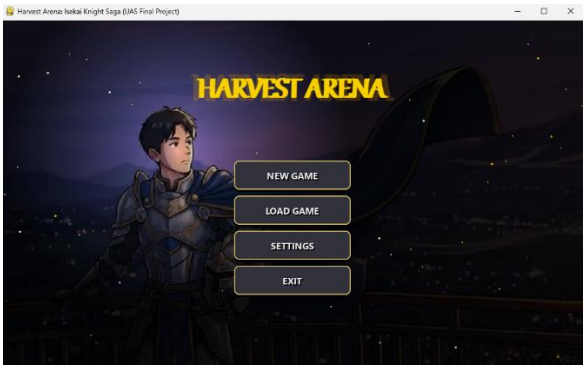


Image 3 main menu

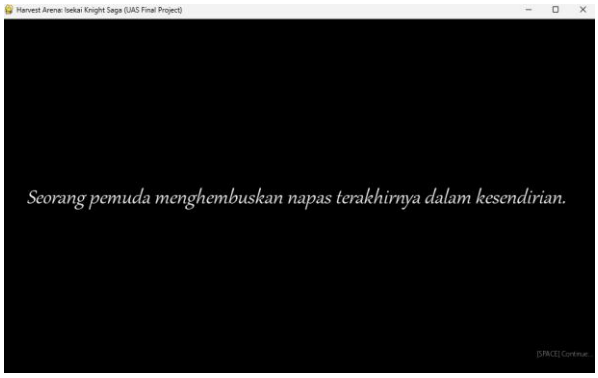


Image 4 prolog scene

3.2 Implementasi Fitur AI Quest (Multithreading)

Fitur unggulan sistem ini adalah integrasi Google Gemini. Karena permintaan internet memakan waktu (blocking), proses ini dijalankan pada *Thread* terpisah agar *game* tidak macet.

```python
def _fetch_ai_quest_worker(self):
        """Worker thread untuk memanggil Gemini API."""
        try:
            # --- PROMPT AI YANG LEBIH KETAT & BAHASA INDONESIA ---
            prompt = f"""
            You are a quest generator for a survival RPG game where the player is ALONE.
            Context: The player is a lone survivor in the wild.
            There are NO other NPCs (NO chefs, NO kings, NO villagers).

            Current Player Level: {self.level}

            Task: Generate a simple quest based on these available resources:
            - Targets: Slime, Golem, Grape, Mushroom, Potato, Carrot

            Response MUST be strictly valid JSON format like this:
            {{
                "desc": "Kumpulkan 5 Jamur untuk bertahan hidup",
                "target_val": 5,
                "type": "harvest",
                "target_name": "Mushroom"
            }}

            Constraints:
            1. 'type' must be "harvest" or "kill".
            2. 'target_name' must be exactly one of the targets listed above (Must be English).
            3. 'desc' MUST be in BAHASA INDONESIA. Use phrases like "Kumpulkan stok makanan", "Bersihkan area", "Cari sumber daya".
            4. 'desc' must NOT mention any other characters (chef, king, etc.).
            5. 'target_val' between 3 and 10.
            """

            response = AI_MODEL.generate_content(prompt)
            # Bersihkan response jika ada format markdown
            txt = response.text.strip().replace("```json", "").replace("```", "")
            data = json.loads(txt)

            new_quest = Quest(random.randint(1000,9999), data['desc'], data['target_val'], data['type'], data['target_name'])
            self.current_active_quest = new_quest
            print(f"System: AI Generated Quest: {data['desc']}")

        except Exception as e:
            print(f"System: AI Error ({e}). Switching to offline quest.")
            self._load_offline_quest()
        finally:
            self.is_generating_quest = False
```

image 5 AI Quest code

3.3 Implementasi Gameplay (Farming & Battle)

- **Farming:** Pemain dapat mencangkul, menanam benih, dan menyiram tanaman. Setiap FarmTile memiliki *timer* pertumbuhan sendiri.
- **Battle:** Saat pemain menabrak musuh, *state* berubah menjadi BattleScene. Pertarungan menggunakan sistem *Turn-Based* di mana pemain dapat memilih aksi Serang, Sihir, atau Sembuh.

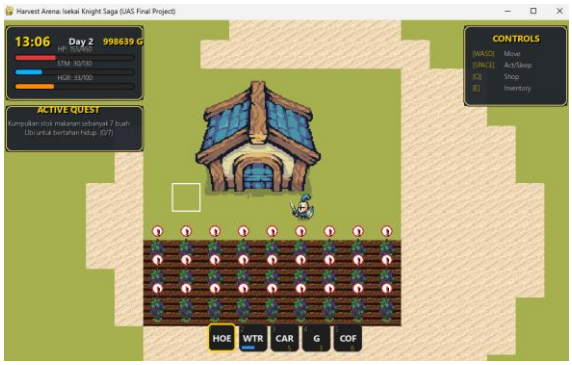


image 6 farming

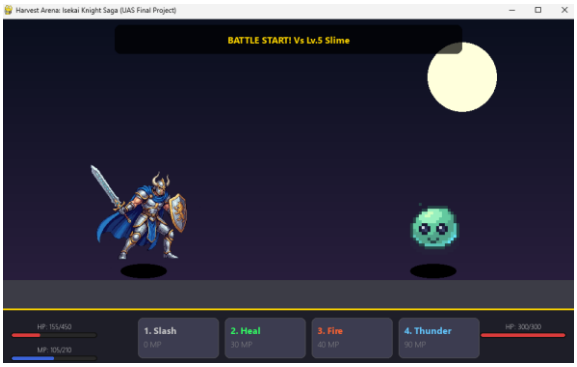


image 7 battle scene

3.4 Implementasi Penyimpanan Data

Fitur persistensi data pada aplikasi "Harvest Arena" dibangun menggunakan teknik Serialisasi Objek. Serialisasi adalah proses penerjemahan struktur data objek yang kompleks (seperti instance dari sebuah class yang hidup di memori RAM) menjadi format linear yang dapat disimpan ke dalam media penyimpanan fisik (hard drive) atau ditransmisikan melalui jaringan.

Dalam proyek ini, format serialisasi yang dipilih adalah JSON (JavaScript Object Notation) karena sifatnya yang ringan, mudah dibaca manusia (human-readable), dan didukung secara native oleh pustaka standar Python. Proses penyimpanan data berjalan melalui tahapan berikut:

1. Transformasi Objek ke Dictionary (to_dict) Setiap kelas yang datanya perlu disimpan (PlayerStats, WorldSystem, FarmTile) memiliki metode khusus bernama to_dict(). Metode ini bertugas memilah atribut-atribut vital dari objek tersebut dan membungkusnya ke dalam struktur data Dictionary Python (pasangan Key-Value).
2. Penanganan Struktur Data Kompleks Tantangan utama dalam serialisasi ini adalah menyimpan status WorldSystem yang berisi ribuan objek FarmTile. Karena JSON standar tidak mendukung Tuple sebagai Key (kunci), maka koordinat peta (x, y) dikonversi menjadi tipe data String "x,y".
3. Encoding ke Format JSON Setelah seluruh data terkumpul dalam satu Dictionary besar di MasterGame, modul json Python digunakan untuk melakukan encoding (penulisan) ke file teks .json. Proses ini mengubah struktur data Python menjadi string teks panjang.

4. Deserialisasi (Load Game) Saat memuat permainan (Load Game), proses kebalikannya terjadi (Deserialisasi). File JSON dibaca, di-decode kembali menjadi Dictionary, lalu data tersebut "disuntikkan" kembali ke dalam atribut objek menggunakan metode load_from_dict(). Hal ini mengembalikan status permainan persis seperti saat ditinggalkan.


```
{
    "game_time": 875.1999999997042,
    "hotbar": [
        "HOE",
        "WATER",
        "CROP_CARROT",
        "SEED_GRAPE",
        "COFFEE"
    ],
    "player_pos": [
        1605,
        1015
    ],
    "stats": {
        "day": 2,
        "gold": 998639,
        "hp": 68,
        "hunger": 24,
        "inventory": {
            "Carrot": 5,
            "Carrot Seed": 0,
            "Coffee": 6,
            "Dark Essence": 0,
            "Gold Nugget": 1,
            "Grape Seed": 3,
            "Hoe": 1,
            "Mushroom": 5,
            "Mushroom Seed": 0,
            "Potato Seed": 0,
            "Potion": 0,
            "Slime Gel": 1,
            "Watering Can": 1
        },
```


image 8 Struktur Output JSON

BAB IV
FITUR DAN CARA PENGGUNAAN

4.1 Kesimpulan

Berdasarkan tahapan perancangan hingga implementasi yang telah dilakukan, dapat disimpulkan:

1. Keberhasilan PBO: Aplikasi berhasil menerapkan struktur kelas yang modular. Pewarisan (Entity -> Hero) mengurangi duplikasi kode secara signifikan.
2. Integrasi AI: Penggunaan Multithreading terbukti efektif menangani integrasi Google Gemini AI, memungkinkan pembuatan misi yang dinamis tanpa mengganggu performa game (tetap 60 FPS).
3. Manajemen Data: Sistem Save/Load berbasis JSON berhasil menyimpan kondisi kompleks dunia permainan (tiles) dan inventaris secara persisten.

4.2 Saran

Untuk pengembangan selanjutnya, beberapa aspek yang dapat ditingkatkan adalah:

1. Perluasan Konten AI: AI dapat digunakan tidak hanya untuk quest, tetapi juga untuk menjurikan dialog NPC yang dinamis.
2. Optimasi Rendering: Mengimplementasikan teknik Chunking pada WorldSystem agar peta bisa dibuat jauh lebih luas tanpa membebani memori.
3. Porting Platform: Mengadaptasi kontrol permainan agar mendukung layar sentuh untuk deployment ke perangkat Android.

DAFTAR PUSTAKA

[1] Python Software Foundation. (2025). *Python 3.12 Documentation*. Diakses dari https://docs.python.org/3/

[2] Shinners, P. (2024). *Pygame Documentation*. Diakses dari https://www.pygame.org/docs/

[3] Google AI. (2025). *Gemini API Documentation: Generative AI for Developers*. Diakses dari https://ai.google.dev/docs

[4] Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Reading, MA: Addison-Wesley.

[5] ECMA International. (2017). *The JSON Data Interchange Syntax (ECMA-404)*. Geneva: ECMA International.