LAPORAN PROYEK AKHIR SEMESTER
PEMROGRAMAN BERORIENTASI OBJEK
GAME ULAR TANGGA BERBASIS OOP

Mata Kuliah
Pemrograman Berorientasi Objek

Oleh:
Faizul Kamil
24091397045
2024B

PROGRAM STUDI D4 MANAJEMEN INFORMATIKA
FAKULTAS VOKASI
UNIVERSITAS NEGERI SURABAYA
2025

DAFTAR ISI

BAB I
PENDAHULUAN

1.1 Latar Belakang

Game Ular Tangga adalah permainan papan klasik yang dimainkan oleh 2 pemain atau lebih. Permainan ini melibatkan elemen keberuntungan melalui lemparan dadu dan strategi dalam mencapai kotak nomor 100. Proyek ini mengimplementasikan game Ular Tangga menggunakan paradigma Object-Oriented Programming (OOP) dengan framework Pygame untuk visualisasi dan interaksi pengguna.

1.2 Tujuan Proyek

- Mengimplementasikan prinsip-prinsip OOP yaitu Encapsulation, Inheritance, dan Polymorphism
- Membuat aplikasi game interaktif dengan GUI yang menarik
- Menerapkan struktur kode yang modular dan mudah dipelihara
- Menggunakan struktur data yang efisien untuk mengelola alur permainan

1.3 Spesifikasi Teknis

Aplikasi game ini dikembangkan menggunakan bahasa pemrograman Python versi 3.13.7 dengan memanfaatkan library Pygame sebagai pendukung utama dalam pengolahan grafis, input pengguna, serta manajemen waktu dan event. Game dirancang untuk berjalan pada resolusi layar 1080 x 720 piksel, sehingga mampu menampilkan tampilan visual yang jelas dan proporsional pada layar desktop.

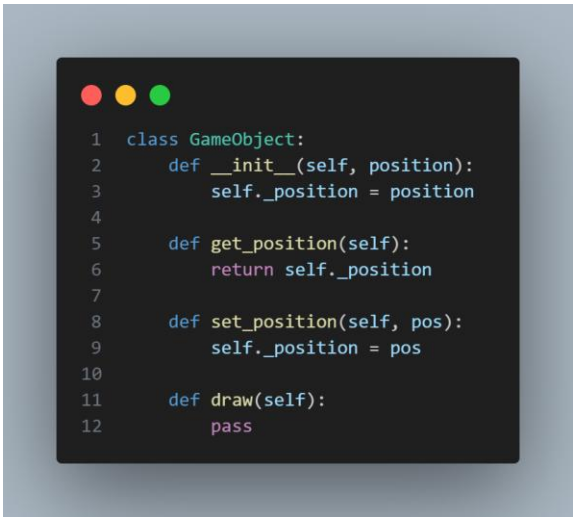
Untuk menjaga kelancaran animasi dan responsivitas permainan, game dijalankan dengan kecepatan 60 frame per second (FPS). Sistem permainan mendukung dua pemain, yang memungkinkan interaksi antar pemain dalam satu perangkat secara bersamaan. Spesifikasi ini disusun untuk memastikan performa game tetap stabil, responsif, dan nyaman digunakan oleh pengguna.

BAB II
KONSEP OOP YANG DITERAPKAN

2.1 Encapsulation (Enkapsulasi)

Encapsulation diterapkan dengan menyembunyikan detail implementasi internal dan hanya menyediakan interface publik untuk berinteraksi dengan objek.

Implementasi dalam Kode:



```python
class GameObject:
    def __init__(self, position):
        self._position = position

    def get_position(self):
        return self._position

    def set_position(self, pos):
        self._position = pos

    def draw(self):
        pass
```

Penjelasan:

- Atribut _position menggunakan underscore tunggal sebagai konvensi untuk menandakan atribut protected
- Akses ke posisi dilakukan melalui method get_position() dan set_position() (getter dan setter)
- Data internal dilindungi dari akses langsung, memastikan integritas data

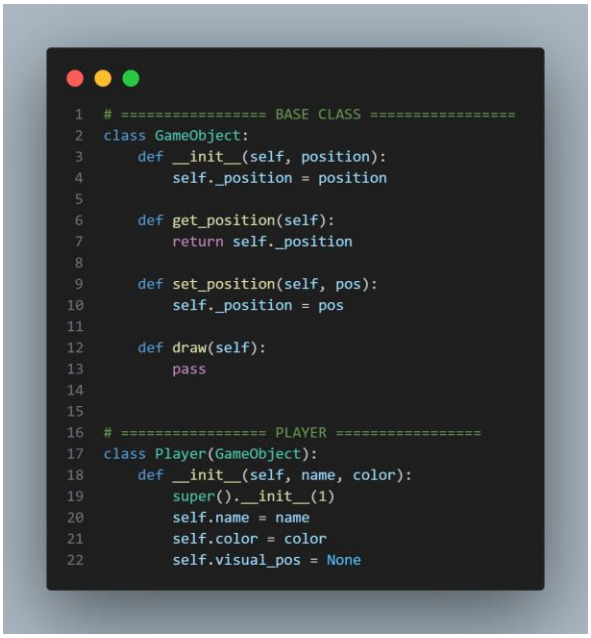
Contoh Lain:

```python
class Player(GameObject):
    def __init__(self, name, color):
        super().__init__(1)
        self.name = name
        self.color = color
        self.visual_pos = None
```

2.2 Inheritance (Pewarisan)

Inheritance memungkinkan kelas turunan mewarisi properti dan method dari kelas induk, mempromosikan code reusability.

Implementasi dalam Kode:



```python
# ================ BASE CLASS ================
class GameObject:
    def __init__(self, position):
        self._position = position

    def get_position(self):
        return self._position

    def set_position(self, pos):
        self._position = pos

    def draw(self):
        pass


# ================ PLAYER ================
class Player(GameObject):
    def __init__(self, name, color):
        super().__init__(1)
        self.name = name
        self.color = color
        self.visual_pos = None
```

Penjelasan:

- Kelas GameObject bertindak sebagai base class (parent class)
- Kelas Player mewarisi semua atribut dan method dari GameObject
- Method super().__init__(1) memanggil constructor parent class
- Player dapat menggunakan method get_position() dan set_position() dari parent class

Manfaat:

- Mengurangi duplikasi kode
- Memudahkan maintenance dan extensibility
- Menciptakan hierarki kelas yang logis

2.3 Polymorphism (Polimorfisme)

Polymorphism memungkinkan method yang sama memiliki implementasi berbeda di kelas yang berbeda.

Implementasi dalam Kode:

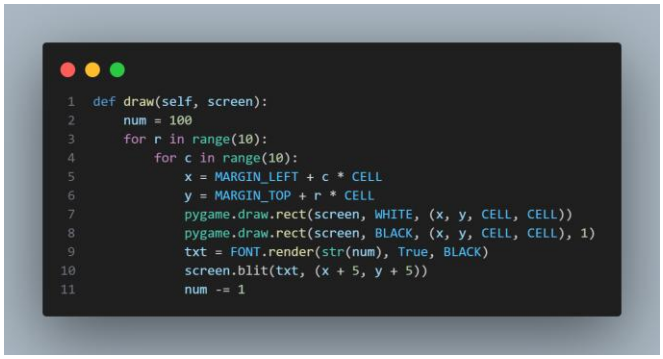
```python
class GameObject:
    def __init__(self, position):
        self._position = position

    def get_position(self):
        return self._position

    def set_position(self, pos):
        self._position = pos

    def draw(self, screen):
        pass
```

```python
def draw(self, screen, board, offset):
    x, y = board.get_coordinates(self._position)
    ox, oy = offset
    pygame.draw.circle(
        screen, self.color, (x + ox, y + oy), 14
    )
```

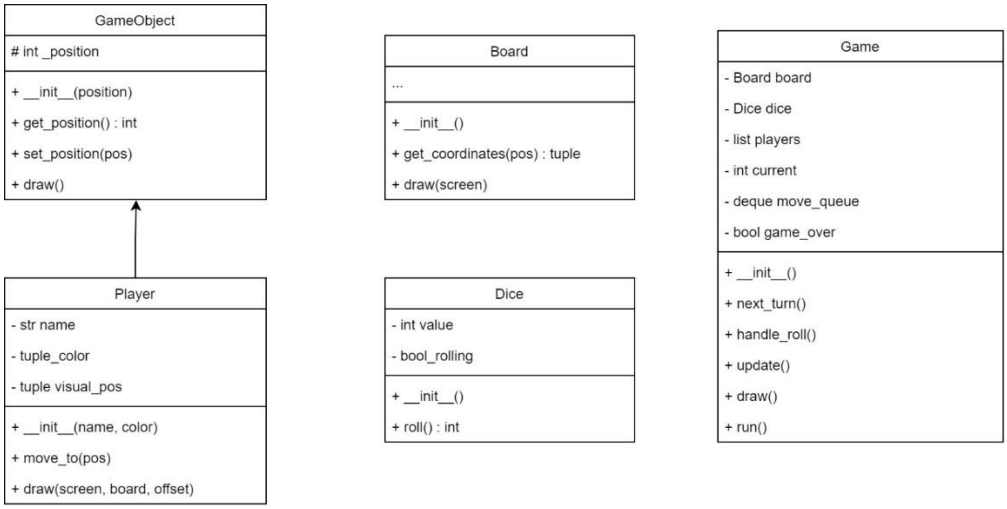


```python
def draw(self, screen):
    num = 100
    for r in range(10):
        for c in range(10):
            x = MARGIN_LEFT + c * CELL
            y = MARGIN_TOP + r * CELL
            pygame.draw.rect(screen, WHITE, (x, y, CELL, CELL))
            pygame.draw.rect(screen, BLACK, (x, y, CELL, CELL), 1)
            txt = FONT.render(str(num), True, BLACK)
            screen.blit(txt, (x + 5, y + 5))
            num -= 1
```

Penjelasan:

- Method draw() didefinisikan di base class GameObject sebagai interface
- Setiap subclass mengimplementasikan draw() sesuai kebutuhannya
- Player.draw() menggambar lingkaran pemain
- Board.draw() menggambar papan permainan
- Kedua implementasi berbeda namun menggunakan nama method yang sama

BAB III
CLASS DIAGRAM



Relasi Antar Kelas:

- Inheritance: Player → GameObject
- Composition: Game memiliki Board, Dice, dan List[Player]
- Association: Player menggunakan Board untuk koordinat

BAB IV
PENJELASAN KELAS DAN STRUKTUR APLIKASI

4.1 Class GameObject (Base Class)

Base class untuk semua objek game yang memiliki posisi

Atribut:

- _position (protected): Posisi objek di papan (1-100)

Method:

- get_position(): Mengembalikan posisi saat ini
- set_position(pos): Mengatur posisi baru
- draw(): Method abstrak untuk polymorphism

4.2 Class Player (Derived Class)

Merepresentasikan pemain dalam game

Atribut:

- name: Nama pemain (Player 1, Player 2)
- color: Warna representasi pemain (BLUE, RED)
- visual_pos: Posisi visual untuk animasi

Method:

- move_to(pos): Memindahkan pemain ke posisi tertentu
- draw(screen, board, offset): Menggambar pemain di layar dengan offset

4.3 Class Board

Mengelola papan permainan dan visualisasinya

Method:

- get_coordinates(pos): Mengkonversi posisi papan (1-100) ke koordinat pixel (x, y)
- draw(screen): Menggambar papan, angka, tangga (hijau), dan ular (merah)

Detail Implementasi:

- Board berukuran 10x10 (100 kotak)

- Numbering dimulai dari 100 (atas kiri) hingga 1 (bawah kiri)
- Baris genap: kiri ke kanan; Baris ganjil: kanan ke kiri (snake pattern)

4.4 Class Dice

Mengelola dadu dan lemparan

Atribut:

- value: Nilai dadu saat ini (1-6)
- rolling: Status apakah dadu sedang dilempar

Method:

- roll(): Menghasilkan angka random 1-6

4.5 Class Game (Main Controller)

Mengontrol alur permainan dan koordinasi antar komponen

Atribut:

- board: Instance dari Board
- dice: Instance dari Dice
- players: List berisi 2 Player
- current: Index pemain yang sedang bermain
- move_queue: Queue untuk animasi pergerakan bertahap
- game_over: Status akhir permainan

Method:

- next_turn(): Berganti ke pemain berikutnya
- handle_roll(): Menangani lemparan dadu dan mengisi move queue
- update(): Update state game setiap frame (animasi pergerakan)
- draw(): Render semua elemen visual
- run(): Game loop utama

Struktur Data yang Digunakan:

- deque (Double-ended Queue): Untuk animasi pergerakan yang smooth

BAB V
FITUR DAN CARA PENGGUNAAN

5.1 Fitur Utama

A. Papan Permainan Interaktif

- Papan 10x10 dengan 100 kotak
- Numbering yang mengikuti pola ular (snake pattern)
- Visual tangga berwarna hijau
- Visual ular berwarna merah

B. Sistem Pemain

- 2 pemain dengan warna berbeda (Biru dan Merah)
- Offset posisi agar tidak overlap
- Tracking posisi real-time

C. Mekanik Dadu

- Random number generator (1-6)
- Terintegrasi dengan sistem pergerakan

D. Animasi Pergerakan

- Pergerakan bertahap dari kotak ke kotak
- Menggunakan queue system untuk smooth animation
- Visual feedback yang jelas

E. Logika Ular dan Tangga

- 8 Tangga: Membawa pemain naik
    - 3→22, 5→8, 11→26, 20→29, 27→56, 36→44, 51→67, 71→92
- 10 Ular: Membawa pemain turun
    - 17→4, 19→7, 21→9, 43→34, 50→30, 62→18, 74→53, 89→68, 95→24, 99→78

F. Kondisi Menang

- Pemain pertama yang mencapai kotak 100 menang
- Notifikasi "MENANG!" di layar

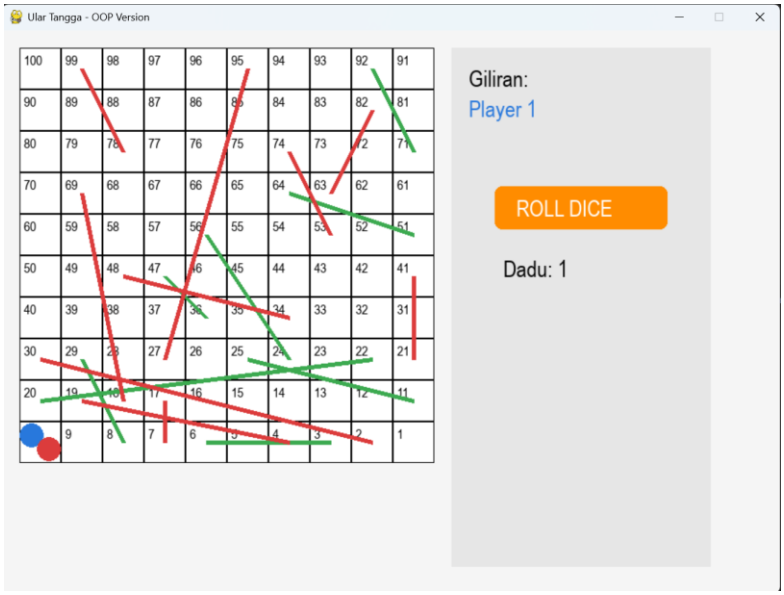
5.2 Cara Penggunaan

Langkah-langkah:

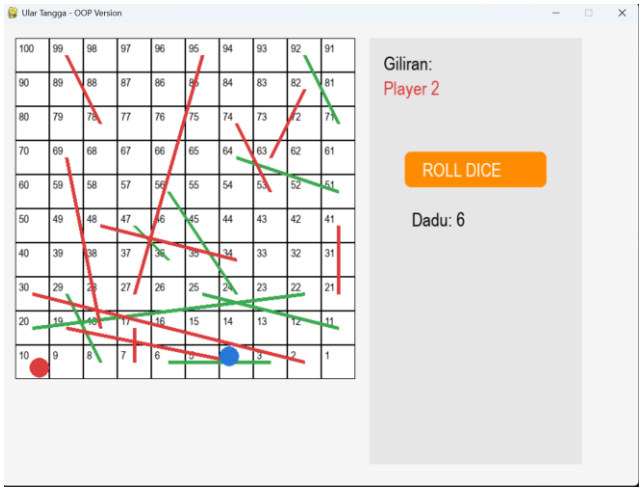
1. **Menjalankan Aplikasi**
2. **Memulai Permainan**
    - Aplikasi akan menampilkan papan permainan
    - Player 1 (Biru) memulai giliran pertama
3. **Melempar Dadu**
    - Tekan **SPASI** untuk melempar dadu
    - Pemain akan bergerak sesuai hasil dadu
4. **Pergerakan**
    - Pemain bergerak secara otomatis bertahap
    - Jika mendarat di tangga, naik otomatis
    - Jika mendarat di ular, turun otomatis
5. **Bergantian**
    - Setelah pergerakan selesai, giliran berpindah otomatis ke pemain lain
6. **Menang**
    - Pemain pertama yang mencapai kotak 100 menang
    - Tampilan "MENANG!" akan muncul
7. **Keluar**
    - Tutup window atau tekan tombol close

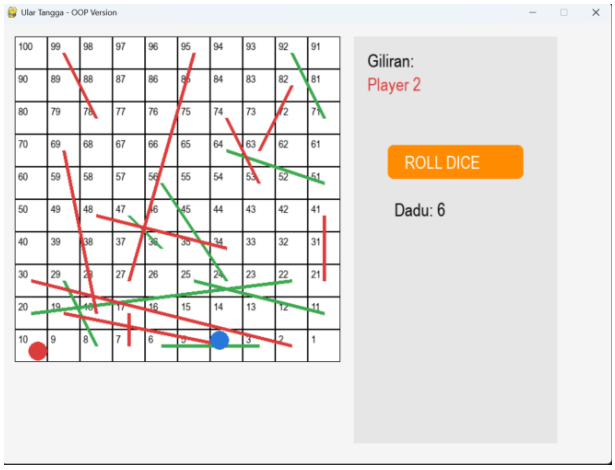
BAB VI
SCREENSHOT APLIKASI

6.1 Screenshot Tampilan Awal

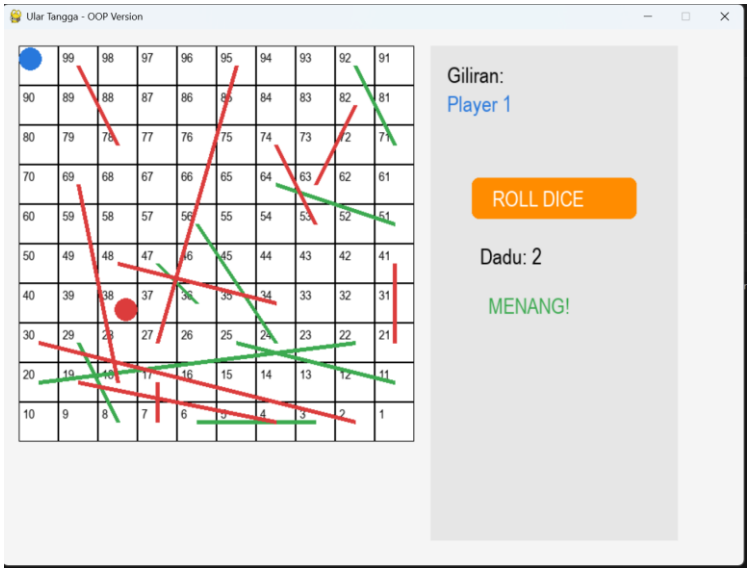


6.2 Screenshot Saat Permainan





6.3 Screenshot Kondisi Menang



BAB VII
TANTANGAN DAN SOLUSI

7.1 Tantangan Koordinat Papan

Papan permainan ular tangga memiliki pola numbering unik (snake pattern) yang berbeda dari grid normal. Kotak dimulai dari 100 di kiri atas, bergerak ke kanan hingga 91, lalu baris berikutnya bergerak dari kanan ke kiri (90 ke 81), dan seterusnya.

Solusinya adalah implementasi method get_coordinates() di class Board yang mengkonversi posisi logis (1-100) ke koordinat pixel (x, y):

```
67  class Board:
68      def get_coordinates(self, pos):
69          idx = pos - 1
70          row = idx // 10
71          col = idx % 10
72          draw_row = 9 - row
73          draw_col = col if row % 2 == 0 else 9 - col
74          x = MARGIN_LEFT + draw_col * CELL + CELL // 2
75          y = MARGIN_TOP + draw_row * CELL + CELL // 2
76          return x, y
```

7.2 Tantangan Animasi Pergerakan

Pergerakan pemain perlu terlihat smooth dan bertahap dari satu kotak ke kotak lain, bukan langsung teleport ke posisi akhir.

Solusinya adalah menggunakan deque (double-ended queue) untuk menyimpan semua posisi intermediate:

```
135     def handle_roll(self):
136         if self.game_over or self.move_queue:
137             return
138
139         value = self.dice.roll()
140         player = self.players[self.current]
141         start = player.get_position()
142         target = min(100, start + value)
143
144         for p in range(start + 1, target + 1):
145             self.move_queue.append(p)
```

Kemudian di update(), dequeue satu posisi per frame untuk animasi smooth.

7.3 Tantangan Overlap Pemain

Ketika 2 pemain berada di kotak yang sama, visualisasi mereka akan overlap dan sulit dibedakan.

Solusinya implementasi sistem offset saat drawing:

```
168         offsets = [(-10, -8), (10, 8)]
169         for i, p in enumerate(self.players):
170             p.draw(SCREEN, self.board, offsets[i])
```

7.4 Tantangan Timing Ladder/Snake

Masalahnya kapan harus mengaplikasikan efek tangga atau ular? Jika dilakukan saat masih bergerak, animasi akan terlihat aneh.

Solusinya adalah mengecek ladder/snake hanya setelah move queue kosong:

```
152             if not self.move_queue:
153                 pos = player.get_position()
154                 if pos in LADDERS:
155                     player.set_position(LADDERS[pos])
156                 elif pos in SNAKES:
157                     player.set_position(SNAKES[pos])
```

BAB VIII
PENUTUP

8.1 Pencapaian Proyek

Proyek Game Ular Tangga OOP telah berhasil diimplementasikan dengan menerapkan prinsip-prinsip OOP secara komprehensif:

1. **Encapsulation**
   Berhasil diterapkan melalui penggunaan atribut protected (_position) dan method getter/setter untuk mengontrol akses data
2. **Inheritance**
   Diimplementasikan dengan baik melalui hierarki GameObject → Player, memungkinkan code reusability dan struktur yang terorganisir
3. **Polymorphism**
   Method draw() berhasil di-override di berbagai kelas untuk behaviour yang berbeda sesuai konteks
4. **Struktur Data**
   Penggunaan deque untuk queue management memberikan performa optimal untuk animasi pergerakan

LINK GITHUB: https://github.com/Justizz10/snake-ladder-oop

8.2 Kesimpulan

Berdasarkan hasil perancangan dan implementasi yang telah dilakukan, dapat disimpulkan bahwa proyek Game Ular Tangga berbasis Object-Oriented Programming (OOP) berhasil dikembangkan dengan baik dan sesuai dengan tujuan awal. Penerapan prinsip encapsulation, inheritance, dan polymorphism terbukti mampu meningkatkan kualitas struktur kode, menjadikan program lebih terorganisir, mudah dipahami, serta memiliki tingkat maintainability yang tinggi. Pembagian tanggung jawab ke dalam kelas-kelas yang spesifik juga membantu mengurangi kompleksitas kode dan mempermudah proses pengembangan serta debugging.

Selain itu, fitur-fitur utama permainan seperti sistem dua pemain, mekanik dadu acak, pergerakan berbasis giliran, animasi yang halus, serta implementasi ular dan tangga telah berjalan secara fungsional dan stabil. Penggunaan struktur data yang tepat, khususnya deque untuk mengelola antrian pergerakan pemain, memberikan performa yang efisien dalam menangani animasi langkah demi langkah. Secara keseluruhan, proyek ini membuktikan bahwa paradigma OOP sangat efektif untuk membangun aplikasi game interaktif yang bersifat scalable, modular, dan siap dikembangkan lebih lanjut di masa depan.

8.3 Saran

Untuk pengembangan selanjutnya, aplikasi Game Ular Tangga ini masih memiliki banyak potensi yang dapat dieksplorasi guna meningkatkan kualitas dan pengalaman pengguna. Pengembangan sistem menu, penambahan jumlah pemain, serta integrasi AI sebagai lawan komputer dapat memperkaya gameplay dan membuat permainan lebih variatif. Selain itu, penambahan efek suara dan animasi lanjutan seperti animasi naik tangga dan turun ular dapat meningkatkan aspek visual dan interaktivitas permainan.

Dari sisi teknis, pengembangan fitur penyimpanan data seperti save dan load game serta pencatatan statistik permainan dapat menjadi nilai tambah, terutama untuk analisis permainan dan penggunaan jangka panjang. Disarankan pula untuk menerapkan prinsip desain perangkat lunak lanjutan seperti design pattern agar struktur kode semakin kuat dan profesional. Dengan pengembangan tersebut, game ini tidak hanya berfungsi sebagai media hiburan, tetapi juga sebagai sarana pembelajaran pemrograman berorientasi objek yang lebih komprehensif.