LAPORAN UAS PEMROGRAMAN BERORIENTASI OBJEK: GAME ESCAPE LINE

Dosen Pengampu:

I Gde Agung Sri Sidhimantra, S.Kom., M.Kom.

Binti Kholifah, S.Kom., M.Tr.Kom.

Dimas Novian Aditia Syahputra, S.Tr.T., M.Tr.T.

Moch Deny Pratama, S.Tr.Kom., M.Kom.

Disusun Oleh:

1. Savinka Krizanantha J.A (23091397088)
2. Mohammad Azwaril Hasibi (23091397090)
3. Baharuddin Ziyat Agustian (23091397094)

PROGRAM STUDI MANAJEMEN INFORMATIKA

FAKULTAS VOKASI

UNIVERSITAS NEGERI SURABAYA 2023/2024

GAME ESCAPE LINE

Escape Line adalah game arcade 2D berbasis Python yang menguji kecepatan reaksi dan strategi pemain. Pemain harus memandu karakter untuk menghindari musuh dan mencapai garis aman, dengan tingkat kesulitan yang meningkat di setiap level.

Code:

```python
# Mengakses Library Pygame
import pygame

# Mengatur ukuran layar
SCREEN_WIDTH = 800
SCREEN_HEIGHT = 800

# Memberikan Judul Game
SCREEN_TITLE = 'Escape Line'

#mengatur warna background (RGB Code)
WHITE_COLOR = (255, 255, 255)
BLACK_COLOR = (0, 0, 0)

# Attribute clock digunakan untuk update game event dan frame
clock = pygame.time.Clock()
pygame.font.init()
font = pygame.font.SysFont('ComicSans', 75)

class Game:

    # Frame rate yg digunakan 60 frame per detik
    TICK_RATE = 60

    # Inisialisasi Class Game untuk set up width, height, dan Title
    def __init__(self, image_path, title, width, height):
        self.title = title
        self.width= width
        self.height = height

        # Menampilkan screen dengan ukuran spesifik
        self.game_screen = pygame.display.set_mode((width, height))
        # Memberi warna putih pada game screen
        self.game_screen.fill(WHITE_COLOR)
        pygame.display.set_caption(title)

        # Load and Set background image untuk layar game
        background_image = pygame.image.load(image_path)
        self.image = pygame.transform.scale(background_image, (width, height))
```

```python

    def run_game_loop(self, level_speed):
        is_game_over = False
        did_win = False
        direction = 0

        player_character = PlayerCharacter('asset/pemain.png', 375, 700, 50, 50)
        enemy_0 = EnemyCharacter('asset/musuh1.png', 20, 600, 50, 50)
        # Kecepatan naik ketika telah mencapai Goal
        enemy_0.SPEED *= level_speed

        # Membuat musuh baru
        enemy_1 = EnemyCharacter('asset/musuh2.png', self.width - 40, 400, 50, 50)
        enemy_1.SPEED *= level_speed

        # Membuat musuh baru
        enemy_2 = EnemyCharacter('asset/musuh3.png', 20, 50, 50, 50)
        enemy_2.SPEED *= level_speed

        OnePiece = GameObject('asset/onepiece.png', 375, 50, 50, 50)

        # Main game loop, digunakan untuk update semua gameplay seperti movement, checks, dan graphic
        # Berjalan sampai is_game_over = True
        while not is_game_over:

            for event in pygame.event.get():
                # ketika kita menekan tombol keluar (esc), maka akan keluar dari game loop
                if event.type == pygame.QUIT:
                    is_game_over = True
                # Terdeteksi ketika menekan panah turun
                elif event.type == pygame.KEYDOWN:
                    # Player bergerak ke atas
                    if event.key == pygame.K_UP:
                        direction = 1
                    # Ketika panah/arah dilepaskan
                    elif event.key == pygame.K_DOWN:
                        direction = -1
                # ketika menekan panah naik
                elif event.type == pygame.KEYUP:
                    # Gerakan player berhenti
                    if event.key == pygame.K_UP or event.key == pygame.K_DOWN:
                        direction = 0
                print(event)

            self.game_screen.fill(WHITE_COLOR)
            self.game_screen.blit(self.image, (0, 0))

            OnePiece.draw(self.game_screen)
            player_character.move(direction, self.height)
            player_character.draw(self.game_screen)

            enemy_0.move(self.width)
            enemy_0.draw(self.game_screen)

            # Move and draw more enemies when we reach higher levels of difficulty
            if level_speed > 2:
                enemy_1.move(self.width)
                enemy_1.draw(self.game_screen)
            if level_speed > 4:
               enemy_2.move(self.width)
               enemy_2.draw(self.game_screen)

            if player_character.detection_collision(enemy_0) or\
                (level_speed > 2 and player_character.detection_collision(enemy_1)) or \
                (level_speed > 4 and player_character.detection_collision(enemy_2)):
                is_game_over = True
                did_win = False
                text = font.render('You Lose! :(', True, BLACK_COLOR)
                self.game_screen.blit(text, (300, 350))
                pygame.display.update()
                clock.tick(1)
                break
            elif player_character.detection_collision(OnePiece):
                is_game_over = True
                did_win = True
                text = font.render('You Win! :)', True, BLACK_COLOR)
                self.game_screen.blit(text, (300, 350))
                pygame.display.update()
                clock.tick(1)
                break

            pygame.display.update()
            clock.tick(self.TICK_RATE)

        if did_win:
            self.run_game_loop(level_speed + 0.5)
        else:
            return

```

```python
# Generic game object class to be subclassed by other objects in the game
class GameObject:
    def __init__(self, image_path, x, y, width, height):
        self.x_pos = x
        self.y_pos = y

        self.width = width
        self.height = height

        object_image = pygame.image.load(image_path)
        self.image = pygame.transform.scale(object_image, (width, height))

    def draw(self, background):
        background.blit(self.image, (self.x_pos, self.y_pos))


# Class to represent the character contolled by the player
class PlayerCharacter(GameObject):

    SPEED = 10

    def __init__(self, image_path, x, y, width, height):
        super().__init__(image_path, x, y, width, height)

    def move(self, direction, max_height):
        if direction > 0:
            self.y_pos -= self.SPEED
        elif direction < 0:
            self.y_pos += self.SPEED

        if self.y_pos >= max_height - 50:
            self.y_pos = max_height - 50

    # Return False (no collision) jika posisi player baik di sumbu x dan y,
    # tidak bertabrakan dengan tubuh enemy
    def detection_collision(self, other_body):
        if self.y_pos > other_body.y_pos + other_body.height:
            return False
        elif self.y_pos + self.height < other_body.y_pos:
            return False
        if self.x_pos > other_body.x_pos + other_body.width:
            return False
        elif self.x_pos + self.width < other_body.x_pos:
            return False

        return True

# Class to represent the enemies moving left to right and right to left
class EnemyCharacter(GameObject):

    SPEED = 10

    def __init__(self, image_path, x, y, width, height):
        super().__init__(image_path, x, y, width, height)

    def move(self, max_width):
        if self.x_pos <= 20:
            self.SPEED = abs(self.SPEED)
        elif self.x_pos >= max_width - 40:
            self.SPEED = -abs(self.SPEED)
        self.x_pos += self.SPEED

pygame.init()
new_game = Game('asset/background.png', SCREEN_TITLE, SCREEN_WIDTH, SCREEN_HEIGHT)
new_game.run_game_loop(1)

#keluar dari program
pygame.quit()
quit()

#pygame.draw.rect(game_screen, BLACK_COLOR, [350, 350, 100, 100])
#pygame.draw.circle(game_screen, BLACK_COLOR, (400, 300), 50)

```

Tampilan Game:

1. **Konsep OOP yang diterapkan**

   a. **Class dan Object**

   - Class adalah blueprint atau template untuk membuat objek. Dalam kode ini:
     - Game, GameObject, PlayerCharacter, dan EnemyCharacter adalah class.
     - Setiap class memiliki atribut dan metode untuk mendefinisikan perilaku dan data mereka.

   - Object adalah instansiasi dari sebuah class:
     - Contohnya: player_character = PlayerCharacter(...) menciptakan objek pemain.

   b. **Encapsulation (Enkapsulasi)**

   - Enkapsulasi adalah mekanisme untuk menyembunyikan data internal sebuah class dan hanya menyediakan akses melalui metode atau atribut tertentu.

   - Dalam code:
     - Atribut seperti self.x_pos, self.y_pos, self.width, self.height hanya dapat diakses secara langsung di dalam class.
     - Interaksi dilakukan melalui metode seperti move dan detection_collision.

   c. **Inheritance (Pewarisan)**

   - Inheritance memungkinkan class untuk mewarisi atribut dan metode dari class lain.

   - Dalam kode ini:
     - PlayerCharacter dan EnemyCharacter mewarisi dari class GameObject.
     - Mereka mendapatkan atribut dasar seperti x_pos, y_pos, width, dan height, serta metode draw.
     - PlayerCharacter dan EnemyCharacter menambahkan logika tambahan pada metode mereka masing-masing.

   d. **Polymorphism (Polimorfisme)**

   - Polimorfisme memungkinkan penggunaan nama metode yang sama dengan perilaku berbeda di class yang berbeda.

   - Contohnya:
     - Metode move diimplementasikan secara unik dalam class PlayerCharacter dan EnemyCharacter.

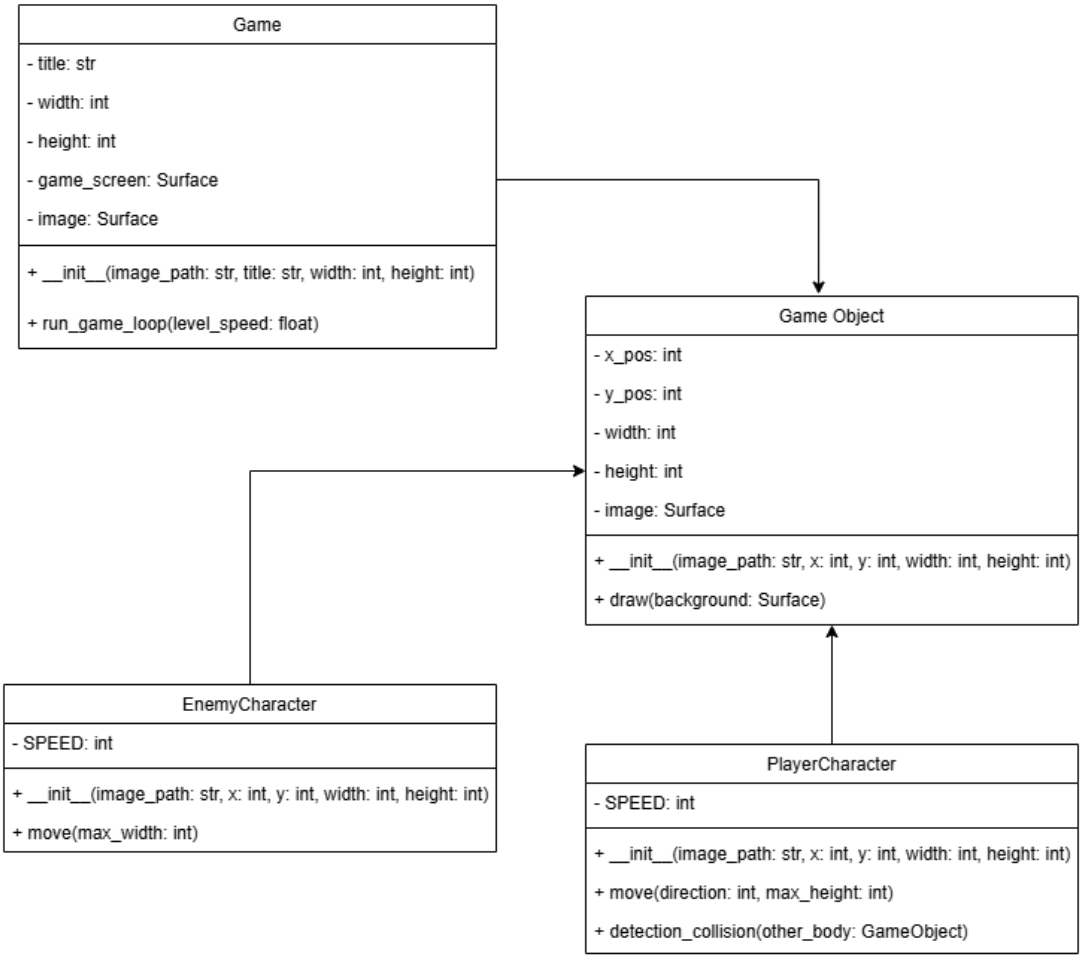
- Pada PlayerCharacter, metode move digunakan untuk pergerakan vertikal.
- Pada EnemyCharacter, metode move digunakan untuk pergerakan horizontal.

e. Abstraction (Abstraksi)

- Abstraksi adalah konsep menyembunyikan detail implementasi dan hanya menampilkan antarmuka yang diperlukan.

- Dalam kode ini:
  - GameObject bertindak sebagai class abstraksi yang menyediakan atribut dasar (x_pos, y_pos, dll.) dan fungsi draw.
  - Class PlayerCharacter dan EnemyCharacter menggunakan GameObject sebagai basis untuk memperluas fungsionalitasnya.

2. Diagram Class



Penjelasan Diagram Class :

- **Penjelasan kelas:**

  a. Kelas Game

  Kelas Game mengelola layar utama, latar belakang, dan loop utama yang menangani logika permainan seperti pergerakan karakter dan deteksi tabrakan.

  - **Atribut:**
    - **title**: Menyimpan judul dari game.
    - **width**: Menentukan lebar layar game.
    - **height**: Menentukan tinggi layar game.
    - **game_screen**: Merupakan objek Surface untuk menggambar elemen permainan di layar.

- **image**: Latar belakang permainan, berupa gambar yang dimuat ke dalam objek Surface.

- **Metode:**
  - **__init__(image_path: str, title: str, width: int, height: int)**:
    - Konstruktor untuk inisialisasi game.
    - Mengatur judul, dimensi layar, serta memuat gambar latar belakang.

b. Kelas GameObject (Superclass)

Kelas dasar untuk semua objek dalam permainan. Semua objek seperti pemain dan musuh akan mewarisi atribut dan metode dari kelas ini.

- **Atribut:**
  - **x_pos**: Posisi horizontal (X) dari objek di layar.
  - **y_pos**: Posisi vertikal (Y) dari objek di layar.
  - **width**: Lebar dari objek.
  - **height**: Tinggi dari objek.
  - **image**: Gambar yang mewakili tampilan objek.

- **Metode:**
  - **__init__(image_path: str, x: int, y: int, width: int, height: int)**:
    - Konstruktor untuk menginisialisasi posisi, ukuran, dan gambar objek.
  - **draw(background: Surface)**:
    - Menggambar objek pada layar permainan di posisi yang telah ditentukan.

c. Kelas PlayerCharacter (Subclass GameObject)

Kelas untuk karakter pemain. Memiliki kemampuan untuk bergerak secara vertikal dan mendeteksi tabrakan dengan objek lain.

- **Atribut:**
  - **SPEED**: Kecepatan karakter pemain dalam satuan piksel per frame.

- **Metode:**
  - **__init__(image_path: str, x: int, y: int, width: int, height: int)**:
    - Menginisialisasi atribut dari GameObject untuk karakter pemain.
  - **move(direction: int, max_height: int)**:
    - Menggerakkan pemain ke atas atau ke bawah berdasarkan nilai direction.
    - Memastikan pemain tetap dalam batas layar (antara 0 hingga max_height).

- **detection_collision(other_body: GameObject)**:
    - Mendeteksi apakah karakter pemain bertabrakan dengan objek lain.
    - Mengembalikan nilai True jika terjadi tabrakan, False jika tidak.

d. Kelas EnemyCharacter (Subclass GameObject)

Kelas untuk karakter musuh. Musuh bergerak secara horizontal (dari kiri ke kanan dan sebaliknya) dengan kecepatan tertentu.

- **Atribut:**
    - **SPEED**: Kecepatan pergerakan musuh dalam satuan piksel per frame.

- **Metode:**
    1. **__init__(image_path: str, x: int, y: int, width: int, height: int)**:
        - Menginisialisasi atribut dari GameObject untuk karakter musuh.
    2. **move(max_width: int)**:
        - Menggerakkan musuh secara horizontal.
        - Membalik arah pergerakan jika mencapai batas layar (0 atau max_width).

- **Relasi Antar-Class:**

a. Relasi Game ke GameObject (Has-a Relationship)

- Relasi:
  Komposisi (Has-a). Class Game memiliki objek-objek dari GameObject (dan turunannya seperti PlayerCharacter dan EnemyCharacter).

- Penjelasan:
    - Class Game menciptakan instance dari GameObject dan subclass-nya untuk digunakan dalam permainan.
    - Class Game menggunakan metode draw, move, dan detection_collision dari objek-objek ini untuk mengelola logika gameplay.

- Makna Relasi:
    - Class Game adalah pengelola utama objek dari class GameObject dan subclass-nya.
    - Ketergantungan ini satu arah: Game menggunakan GameObject, bukan sebaliknya.

b. Relasi GameObject ke PlayerCharacter dan EnemyCharacter (Inheritance/Is-a Relationship)

- Relasi:

Inheritance(Pewarisan).PlayerCharacter dan EnemyCharacter adalah subclass dari GameObject.

- Penjelasan:
  - PlayerCharacter dan EnemyCharacter mewarisi atribut dan metode dari GameObject, seperti posisi (x_pos, y_pos), ukuran (width, height), dan metode draw.
  - Subclass ini memperluas fungsionalitas GameObject dengan menambahkan logika spesifik:
    - PlayerCharacter: Memiliki metode move untuk pergerakan vertikal dan detection_collision untuk mendeteksi tabrakan.
    - EnemyCharacter: Memiliki metode move untuk pergerakan horizontal bolak-balik.

- Makna Relasi:
  - Pewarisan ini menunjukkan bahwa PlayerCharacter dan EnemyCharacter adalah turunan dari GameObject (Is-a).
  - Kedua subclass berbagi atribut dan metode dasar dari GameObject, tetapi menambahkan perilaku unik.

c. Relasi Game ke PlayerCharacter dan EnemyCharacter (Has-a Relationship)

- Relasi:
  Association (Has-a). Game menggunakan objek dari subclass GameObject (seperti PlayerCharacter dan EnemyCharacter) untuk menjalankan permainan.

- Penjelasan:
  - Dalam metode run_game_loop, Game menciptakan dan mengelola instance dari PlayerCharacter untuk mewakili pemain, dan beberapa instance EnemyCharacter untuk musuh.
  - Game memanggil metode move dan draw pada objek-objek ini untuk mengatur pergerakan dan tampilan di layar.

- Makna Relasi:
  - Hubungan ini menunjukkan bahwa Game bergantung pada objek-objek ini untuk menjalankan gameplay.

3. Alur Aplikasi

1. Inisialisasi Game:
   - Kelas Game diinstansiasi dengan parameter untuk lebar, tinggi, dan judul layar.
   - Layar utama disiapkan dengan gambar latar.

2. Gameplay Loop:
   - Game dimulai menggunakan run_game_loop(), yang memuat loop utama.
   - Pemain (PlayerCharacter) dapat bergerak ke atas atau ke bawah.
   - Musuh (EnemyCharacter) bergerak horizontal, berubah arah saat mencapai batas layar.

3. Deteksi Kemenangan atau Kekalahan:
   - Jika pemain menabrak musuh, game berakhir dengan kekalahan.
   - Jika pemain mencapai objek tujuan (OnePiece), pemain menang dan level meningkat.

4. Restart atau Exit:
   - Pemain dapat memulai kembali dari level berikutnya atau keluar.

4. Tantangan yang Dihadapi Selama Pengembangan

Dalam mengembangkan game ini kami mengalami beberapa tantangan, berikut adalah tantangan yang kami hadapi selama pengembangan game Escape Line:

1. Pengelolaan Objek:
   - Menangani beberapa musuh yang memiliki logika gerakan dan deteksi tabrakan yang berbeda.

2. Optimasi Frame Rate:
   - Menjaga frame rate tetap stabil di 60 FPS meskipun jumlah objek di layar meningkat.

3. Kesalahan Deteksi Tabrakan:
   - Mungkin ada bug pada logika detection_collision jika posisi musuh atau pemain sangat cepat berubah.

4. Kompatibilitas Resolusi:
   - Game saat ini dirancang untuk layar berukuran 800x800. Akan memerlukan penyesuaian untuk resolusi lain.