

**LAPORAN PROJECT GAME ENEMY WAR
PEMROGRAMAN BERORIENTASI OBJEK**



Dosen Pengampu :

I Gde Agung Sri Sidhimantra, S.Kom., M.Kom.
Moch Deny Pratama, S. Tr.Kom., M.Kom.
Dimas Novian Aditia Syahputra, S.Tr.T., M.Tr.T.
Binti Kholifah, S.Kom., M.Tr.Kom

Disusun Oleh :

Widirga Putri Aditya Wardaningtyas/ 23091397079/ 2023 C
Fathir Rahma Dani/ 23091397080/ 2023 C
M. Mardlian Nurofiq/ 23091397105/ 2023 C

Program Studi D4 Manajemen Informatika

Fakultas Vokasi

Universitas Negeri Surabaya

Tahun 2024

1. PENJELASAN ENEMY WAR

1.1 Latar Belakang

Dalam era digital sekarang, permainan game menjadi pilihan hiburan yang populer di berbagai kalangan. Menggunakan bahasa pemrograman Python dan pustaka Pygame untuk membuat game yang sederhana memberi kesempatan bagi mahasiswa untuk lebih memahami konsep pemrograman berorientasi objek secara langsung. Dengan proyek ini, diharapkan dapat menerapkan konsep Class, inheritance, dan polimorfism saat mengembangkan aplikasi permainan. Enemy War adalah game aksi bertema peperangan di mana pemain harus bertahan melawan serangan musuh yang terus bertambah. Pemain mengontrol karakter utama yang dilengkapi dengan senjata untuk menghancurkan musuh. Tantangan utama dalam game ini adalah bertahan selama mungkin sambil mengumpulkan skor dengan mengalahkan musuh.

1.2 Tujuan :

Adapun tujuan dari pembuatan proyek game ini :

1. Menerapkan prinsip pemrograman berorientasi objek/*OOP* ketika menciptakan permainan.
2. Memanfaatkan Pygame sebagai pustaka untuk mendesain permainan yang mudah.
3. Meningkatkan daya kreasi dan kemampuan untuk memecahkan masalah dalam proses pembuatan permainan.

1.3 Ruang Lingkup

Game yang kami buat adalah game 2d sederhana yang melibatkan :

1. Pemain (Player) yang dapat digerakkan.
2. Musuh (Enemy) yang bergerak secara otomatis.
3. Sistem level
4. Tampilan grafis menggunakan Pygame

2. DESAIN GAME

2.1 Deskripsi Game

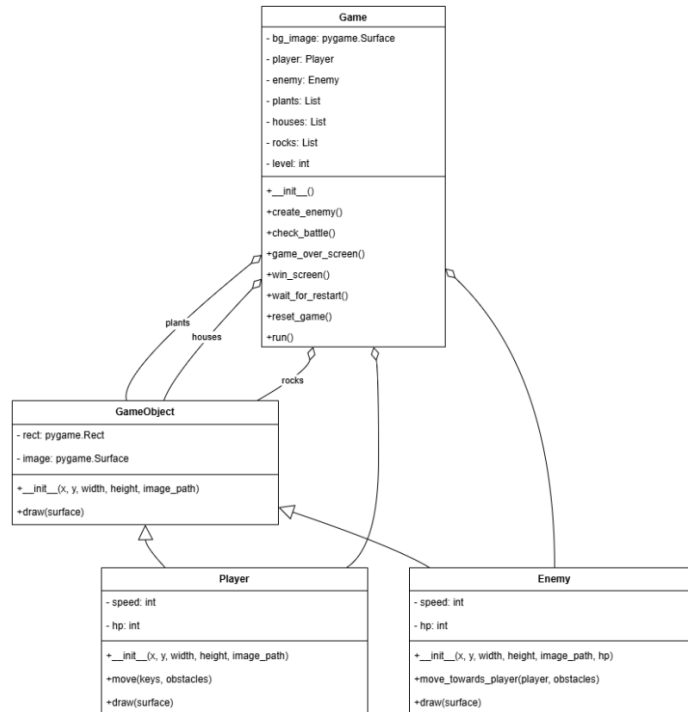
Enemy War RPG merupakan sebuah permainan dua dimensi di mana pemain mengambil peran sebagai pahlawan yang berusaha melawan serangan dari lawan. Pemain dapat menyelidiki area pertempuran, menyerang lawan, serta menghindari hantaman untuk tetap hidup. Setiap lawan memiliki tingkat kesulitan yang bervariasi, dan pemain dapat meningkatkan keterampilan dengan mengumpulkan poin serta barang. Permainan ini memadukan unsur aksi, taktik, dan pengembangan karakter.

2.2 Fitur Utama

1. Kontrol Pemain: Pemain dapat bergerak bebas di medan pertempuran dan menyerang musuh.
2. Musuh Beragam: Musuh memiliki jenis dan tingkat kesulitan yang berbeda-beda.

3. Sistem Level dan Pengalaman: Pemain dapat meningkatkan level dan kekuatan serangan.
4. Grafis 2D Interaktif: Tampilan grafis sederhana namun menarik menggunakan Pygame.
5. Sistem Level: Level pemain meningkat berdasarkan jumlah musuh yang dikalahkan.
6. Game Over dan Restart: Permainan berakhir jika pemain kalah, tetapi dapat di-restart.

2.3 Class Diagram



Berikut penjelasan untuk setiap bagian dalam class diagram tersebut :

1. Class Game (Kelas utama yang mengatur seluruh mekanisme permainan, termasuk pemain, musuh, dan berbagai objek dalam permainan.)

Atribut:

- **bg_image: pygame.Surface:** Gambar latar belakang permainan.
- **player: Player:** Objek pemain dari class Player.
- **enemy: Enemy:** Objek musuh dari class Enemy.
- **plants: List:** Daftar objek tanaman di dalam permainan.
- **houses: List:** Daftar objek rumah di dalam permainan.
- **rocks: List:** Daftar objek batu di dalam permainan.
- **level: int:** Level permainan saat ini.

Method:

- **__init__()**: Konstruktor untuk menginisialisasi atribut permainan.
- **create_enemy()**: Membuat objek musuh di dalam permainan.
- **check_battle()**: Mengecek kondisi pertempuran antara pemain dan musuh.
- **game_over_screen()**: Menampilkan layar akhir jika pemain kalah.
- **win_screen()**: Menampilkan layar kemenangan.
- **wait_for_restart()**: Menunggu input untuk mereset permainan.
- **reset_game()**: Mengatur ulang permainan ke kondisi awal.
- **run()**: Metode utama untuk menjalankan loop permainan.

2. Class GameObject (Kelas induk yang mendefinisikan atribut umum untuk objek dalam permainan seperti pemain, musuh, tanaman, rumah, dan batu.)

Atribut:

- **rect: pygame.Rect**: Objek persegi untuk menentukan posisi dan ukuran objek.
- **image: pygame.Surface**: Gambar yang digunakan untuk objek tersebut.

Method:

- **__init__(x, y, width, height, image_path)**: Konstruktor untuk menginisialisasi posisi, ukuran, dan gambar objek.
- **draw(surface)**: Menggambar objek di layar permainan.

3. Class Player (*Turunan dari GameObject*) (Merepresentasikan pemain dalam permainan)

Atribut:

- **speed: int**: Kecepatan pergerakan pemain.
- **hp: int**: Health point atau nyawa pemain.

Method:

- **__init__(x, y, width, height, image_path)**: Konstruktor untuk menginisialisasi posisi, ukuran, dan gambar pemain.
- **move(keys, obstacles)**: Mengatur pergerakan pemain berdasarkan input keyboard dan menghindari rintangan.
- **draw(surface)**: Menggambar pemain di layar permainan.

4. Class Enemy (*Turunan dari GameObject*) (Merepresentasikan musuh dalam permainan.)

Atribut:

- **speed: int**: Kecepatan pergerakan musuh.
- **hp: int**: Health point atau nyawa musuh.

Method:

- **__init__(x, y, width, height, image_path, hp):** Konstruktor untuk menginisialisasi posisi, ukuran, gambar, dan nyawa musuh.
- **move_towards_player(player, obstacles):** Mengatur pergerakan musuh agar mendekati pemain sambil menghindari rintangan.
- **draw(surface):** Menggambar musuh di layar permainan.

Hubungan Antar Kelas

Relasi	Simbol	Class yang terlibat	Penjelasan
Pewarisan	Panah segitiga kosong	GameObject → Player GameObject → Enemy	Player dan Enemy mewarisi atribut dan metode dari GameObject.
Komposisi	Panah berlian penuh	Game → Player Game → Enemy	Player dan Enemy adalah bagian integral dari Game.
Agregasi	Panah berlian kosong	Game → plants Game → houses Game → rocks	plants, houses, dan rocks adalah elemen permainan tetapi dapat ada secara independen.
Asosiasi	Garis dengan panah biasa	Enemy → Player Player → obstacles	Enemy menggunakan Player untuk menentukan gerakan; Player berinteraksi dengan rintangan.
One-to-Many	Garis biasa (dengan daftar)	Game → List[GameObject]	Satu Game memiliki banyak elemen seperti tanaman (plants) atau batu (rocks).

Konsep OOP

1. Encapsulation (Enkapsulasi)

- **Definisi:** Menyembunyikan detail implementasi objek dan hanya menyediakan antarmuka publik untuk berinteraksi dengan objek tersebut.
- **Implementasi dalam kode:**
 - 1) Atribut seperti speed, hp, dan rect dalam kelas Player dan Enemy tidak langsung diakses dari luar kelas. Sebaliknya, mereka digunakan melalui metode seperti move() atau move_towards_player().
 - 2) Dengan encapsulation, logika pergerakan, tabrakan, dan rendering dikelola di dalam metode masing-masing objek.
 - 3) Source Code:

```

if self.rect.x < player.rect.x:
    self.rect.x += self.speed
elif self.rect.x > player.rect.x:
    self.rect.x -= self.speed

```

2. Inheritance (Pewarisan)

- **Definisi:** Kelas dapat mewarisi atribut dan metode dari kelas lain.
- **Implementasi dalam kode:**
 - 1) Player dan Enemy adalah turunan dari kelas GameObject. Mereka mewarisi atribut seperti rect dan image serta metode draw().
 - 2) Pada konstruktor Player, metode super().__init__() digunakan untuk memanggil konstruktor kelas induk (GameObject) agar atribut dasar seperti posisi, ukuran, dan gambar dapat diinisialisasi
 - 3) Player dan Enemy juga memiliki atribut atau metode tambahan seperti move() (pada Player) dan move_towards_player() (pada Enemy) untuk memperluas fungsionalitas kelas induk.
 - 4) Source Code:

```

43 class Player(GameObject):
44
45     def __init__(self, x, y, width, height, image_path):
46         super().__init__(x, y, width, height, image_path)

```

3. Polymorphism (Polimorfisme)

- **Definisi:** Kemampuan objek untuk menggunakan metode yang sama, tetapi dengan implementasi yang berbeda.
- **Implementasi dalam kode:**
 - 1) Metode draw() pada Player dan Enemy di-*override* dari kelas induk GameObject. Walaupun nama metodenya sama, implementasinya berbeda (misalnya, Enemy menampilkan HP di atas objeknya).
 - 2) Source Code:

```

def draw(self, surface):
    super().draw(surface)
    draw_text(f"Enemy: {self.hp} HP", font, (255, 0, 0), surface, self.rect.x, self.rect.y - 20)

```

4. Composition (Komposisi)

- **Definisi:** Objek dibuat dari kumpulan objek lain.
- **Implementasi dalam kode:**
 - 1) Kelas Game mengelola seluruh objek dalam game seperti pemain (Player), musuh (Enemy), dan berbagai obstacle (GameObject seperti plants, houses, dan rocks).
 - 2) Objek-objek ini disimpan sebagai atribut dalam kelas Game dan digunakan dalam metode seperti run() dan check_battle().
 - 3) Source Code :

```

123 class Game:
124
125     def __init__(self):
126         # Latar belakang
127         self.bg_image = pygame.image.load("assets/map2.png")
128         self.bg_image = pygame.transform.scale(self.bg_image, (SCREEN_WIDTH, SCREEN_HEIGHT))
129
130         # Pemain
131         self.player = Player(100, 100, 60, 60, "assets/player.png")
132
133         # Musuh
134         self.level = 1 # Level awal permainan
135         self.enemy = self.create_enemy()
136
137         # Obstacle
138         self.plants = [
139             GameObject(random.randint(0, SCREEN_WIDTH - 32), random.randint(0, SCREEN_HEIGHT - 32), 60, 70, "assets/pohon.png")
140             for _ in range(10)
141         ]
142         self.houses = [
143             GameObject(200, 150, 140, 100, "assets/rumah.png"),
144             GameObject(500, 300, 140, 100, "assets/rumah.png")
145         ]
146         self.rocks = [
147             GameObject(random.randint(0, SCREEN_WIDTH - 40), random.randint(0, SCREEN_HEIGHT - 40), 40, 40, "assets/batu.png")
148             for _ in range(10)
149         ]

```

5. Abstraction (Abstraksi)

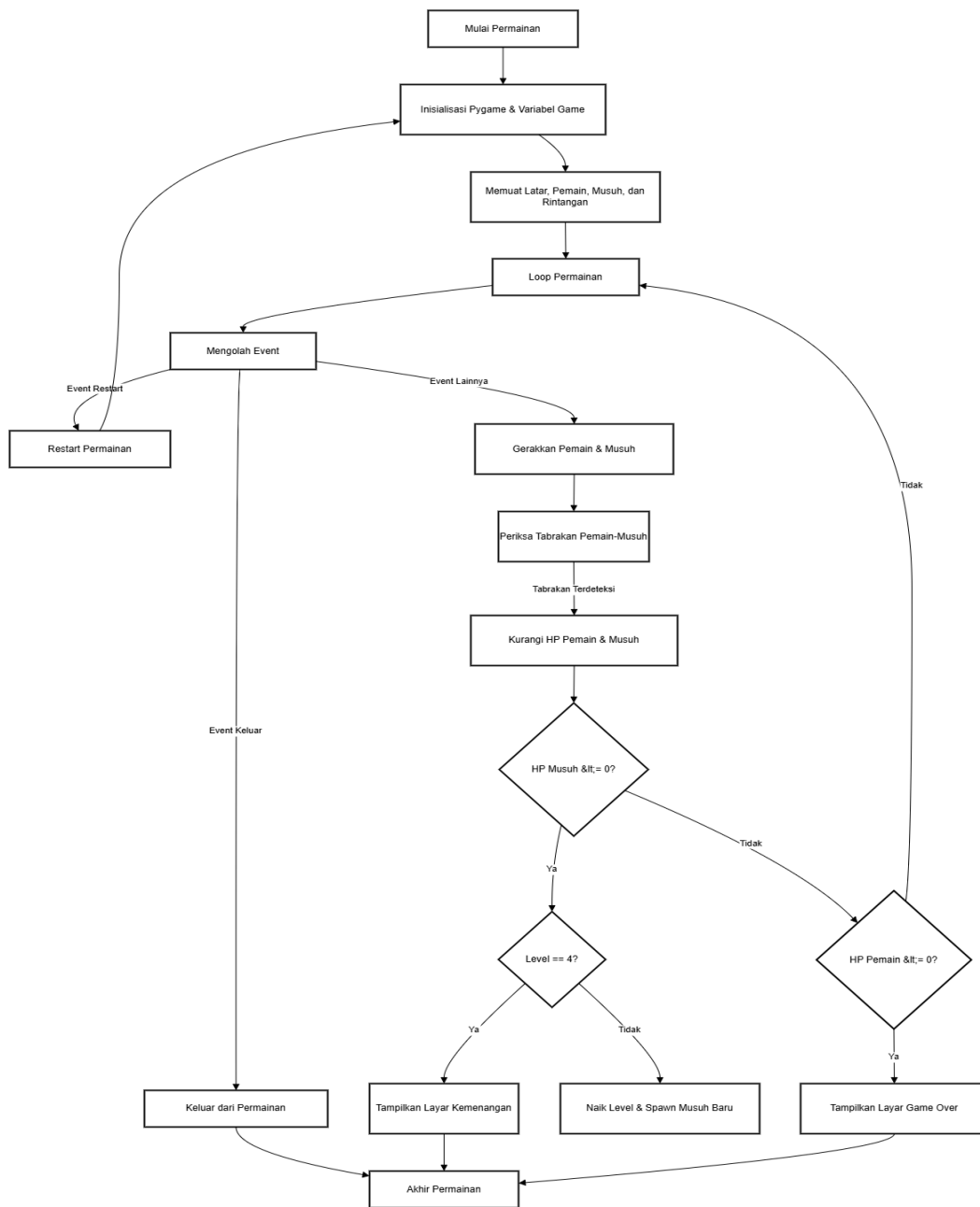
- **Definisi:** Menyembunyikan detail kompleks dan hanya menampilkan fungsi penting kepada pengguna.
- **Implementasi dalam kode:**
 - 1) Detail seperti bagaimana pemain bergerak (move()), bagaimana musuh mendekati pemain (move_towards_player()), atau bagaimana tabrakan ditangani (check_battle()) di-*abstract* ke dalam metode-metode yang spesifik.
 - 2) Pengembang tidak perlu tahu detail implementasi, cukup memanggil metode seperti player.move() atau enemy.move_towards_player().

6. Reusability (Ketergunaan Ulang)

- **Definisi:** Memanfaatkan kembali kode dengan membuat kelas yang modular dan generik.
- **Implementasi dalam kode:**

- 1) Kelas GameObject bersifat generik dan dapat digunakan untuk berbagai jenis objek seperti plants, houses, rocks, Player, dan Enemy.
- 2) Dengan inheritance dan composition, kode dapat diperluas tanpa harus menduplikasi logika.

Flowchart game Enemy War:



3. Implementasi

3.1 Teknologi Yang Digunakan :

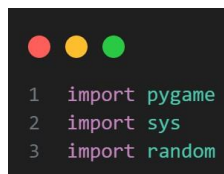
1. Bahasa Pemrograman : Python
2. Library : Pygame
3. IDE : Visual Studio Code

3.2 Struktur Proyek

Project UAS PBO/

```
-- enemy war.py  
-- assets/  
    |-- player.png  
    |-- enemy.png  
    |-- background.png
```

3.3 Penjelasan Source Code :



```
1 import pygame  
2 import sys  
3 import random
```

Penjelasan :

1. **import pygame**

Mengimpor pustaka **Pygame**, yang digunakan untuk membuat permainan 2D dengan fitur seperti grafis, suara, dan kontrol input.

2. **import sys**

Mengimpor pustaka **sys**, yang berguna untuk menangani fungsi sistem, seperti keluar dari program menggunakan `sys.exit()`.

3. **import random**

Mengimpor pustaka **random**, yang digunakan untuk menghasilkan angka acak. Ini berguna untuk menciptakan elemen permainan yang dinamis, seperti posisi acak musuh atau item di dalam game.

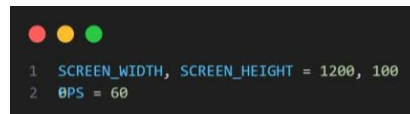
Inisialisasi Pygame



```
1 pygame.init()
```

`pygame.init()` adalah fungsi dari pustaka Pygame yang digunakan untuk menginisialisasi semua modul yang diperlukan agar program Pygame dapat berjalan dengan baik.

Konstanta



```
1 SCREEN_WIDTH, SCREEN_HEIGHT = 1200, 1000
2 FPS = 60
```

Penjelasan :

1. Ukuran Layar (Resolusi)

- **SCREEN_WIDTH**: Lebar jendela permainan (dalam piksel), di sini diatur menjadi **1200 piksel**.
- **SCREEN_HEIGHT**: Tinggi jendela permainan (dalam piksel), di sini diatur menjadi **1000 piksel**.

2. Frame Rate (FPS)

- **FPS (Frames Per Second)**: Menentukan jumlah frame yang ditampilkan per detik.
- Nilai **60 FPS** adalah standar umum untuk banyak game karena memberikan animasi yang mulus.

Inisialisasi Layar



```
1 screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))
2 pygame.display.set_caption("Game dengan Level dan Pemenang")
3 clock = pygame.time.Clock()
```

Penjelasan :

1. `screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))`:

- Baris ini digunakan untuk membuat jendela permainan dengan ukuran yang ditentukan oleh variabel `SCREEN_WIDTH` dan `SCREEN_HEIGHT`. Fungsi `pygame.display.set_mode()` mengembalikan objek yang digunakan untuk menggambar dan memperbarui tampilan jendela.
- `SCREEN_WIDTH` dan `SCREEN_HEIGHT` adalah nilai yang mendefinisikan lebar dan tinggi jendela permainan. Misalnya, jika `SCREEN_WIDTH = 800` dan

SCREEN_HEIGHT = 600, maka jendela permainan yang dibuat akan memiliki ukuran 800 piksel x 600 piksel.

2. `pygame.display.set_caption("Game dengan Objek Rumah dan Batu")`:

- Fungsi `pygame.display.set_caption()` digunakan untuk mengatur judul atau caption yang muncul di bagian atas jendela permainan. Dalam hal ini, judul jendela permainan adalah **"Game dengan Objek Rumah dan Batu"**.
- Judul ini akan muncul pada bar jendela permainan di sistem operasi.

3. `clock = pygame.time.Clock()`:

- Baris ini membuat objek `Clock` yang digunakan untuk mengatur kecepatan permainan, yaitu berapa kali gambar akan diperbarui dalam satu detik.
- Objek `Clock` ini berguna untuk membatasi FPS (frames per second) permainan. Misalnya, dengan menggunakan `clock.tick(60)`, Anda bisa mengatur agar permainan berjalan pada 60 FPS.

Font



```
1 font = pygame.font.Font(None, 36)
```

Penjelasan :

1. `font = pygame.font.Font(None, 36)`

- **`pygame.font.Font`**: Fungsi ini digunakan untuk membuat objek font yang akan digunakan untuk menggambar teks di layar permainan. Font ini dapat disesuaikan dengan ukuran dan jenis font yang diinginkan.
 - 1) **`None`**: Parameter pertama adalah jenis font yang akan digunakan. Jika menggunakan `None`, maka Pygame akan menggunakan font default sistem yang terpasang di komputer. Dan juga dapat memberikan path ke file font (misalnya, "path/to/font.ttf") jika ingin menggunakan font khusus.
 - 2) **`36`**: Parameter kedua adalah ukuran font dalam piksel. Di sini, ukuran font yang digunakan adalah 36 piksel. Ini berarti teks yang digambar dengan font ini akan menggunakan ukuran 36 piksel.

Fungsi Untuk Menggambar Teks



```
1 def draw_text(text, font, color, surface, x,
2 y): textobj = font.render(text, True, color)
3     textrect = textobj.get_rect()
4     textrect.topleft = (x, y)
5     surface.blit(textobj, textrect)
```

Penjelasan :

1. `def draw_text(text, font, color, surface, x, y):`

Fungsi `draw_text` ini bertujuan untuk menggambar teks di layar permainan. Berikut adalah penjelasan parameter-parameter yang digunakan dalam fungsi ini:

- **text**: Merupakan string yang berisi teks yang ingin digambar di layar.
- **font**: Objek font yang digunakan untuk menggambar teks. Objek font ini bisa dibuat menggunakan `pygame.font.Font` atau font default dari sistem.
- **color**: Warna teks yang ingin ditampilkan, biasanya dalam format RGB (merah, hijau, biru), misalnya (255, 0, 0) untuk teks berwarna merah.
- **surface**: Permukaan tempat teks akan digambar. Biasanya ini adalah objek layar (screen) atau permukaan lain dalam permainan.
- **x, y**: Koordinat posisi tempat teks akan digambar pada permukaan. `x` adalah posisi horizontal, dan `y` adalah posisi vertikal.

2. `textobj = font.render(text, True, color)`

- **font.render(text, True, color)**: Fungsi ini merender atau menggambar teks dalam objek yang bisa digambar (dalam hal ini, objek Pygame Surface).
 - 1) **text**: Teks yang ingin digambar.
 - 2) **True**: Parameter ini mengaktifkan anti-aliasing (efek halus pada tepi teks), membuat teks lebih rapi dan jelas.
 - 3) **color**: Warna teks yang digunakan untuk menggambar teks tersebut.
- Fungsi ini mengembalikan objek Surface yang berisi teks yang sudah dirender. Hasil ini disimpan dalam variabel `textobj`.

3. `textrect = textobj.get_rect()`

- **textobj.get_rect()**: Mengambil objek **Rect** (persegi panjang) yang mengelilingi objek teks yang baru saja digambar. Objek Rect ini diperlukan untuk menentukan posisi dan ukuran teks.
- Variabel `textrect` menyimpan informasi tentang ukuran dan posisi yang dibutuhkan untuk menggambar teks pada permukaan.

4. `textrect.topleft = (x, y)`

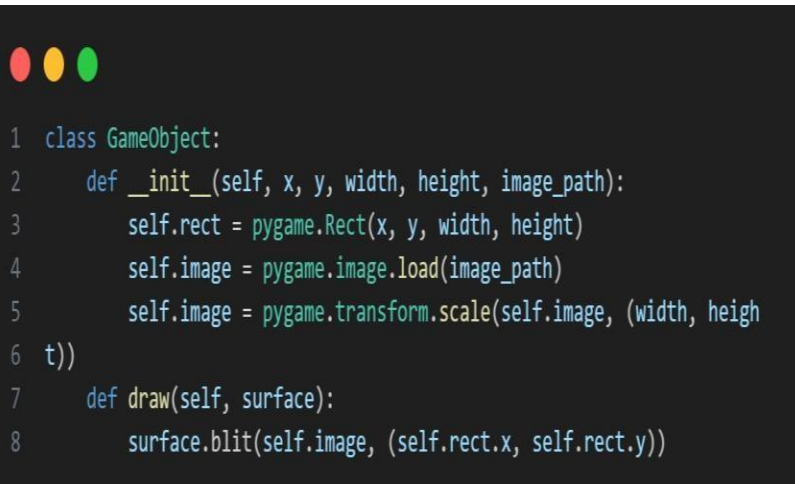
textrect.topleft = (x, y): Menetapkan posisi sudut kiri atas objek teks (dalam bentuk Rect) ke koordinat (x, y) yang diterima sebagai parameter. Ini menentukan tempat teks akan digambar pada permukaan.

5. `surface.blit(textobj, textrect)`

surface.blit(textobj, textrect): Fungsi ini digunakan untuk menggambar objek textobj (yang berisi teks yang sudah dirender) pada permukaan (surface) pada posisi yang ditentukan oleh textrect.

- **blit:** Fungsi ini digunakan untuk menyalin atau menggambar gambar (dalam hal ini, teks) ke permukaan yang lain.
- **textrect:** Posisi dan ukuran dari teks yang akan digambar pada permukaan, yang sudah disesuaikan sebelumnya.

Kelas Dasar GameObject

A screenshot of a code editor with a dark background and light-colored text. At the top left, there are three colored circles: red, yellow, and green. The code is a Python class definition for 'GameObject'. It has an '__init__' method that takes 'x', 'y', 'width', 'height', and 'image_path' as arguments. Inside this method, 'self.rect' is set to a 'pygame.Rect' object, 'self.image' is loaded from the 'image_path', and then scaled to the specified width and height. There is also a 'draw' method that takes a 'surface' as an argument and calls 'surface.blit' to draw the object's image at its current position.

```
1 class GameObject:
2     def __init__(self, x, y, width, height, image_path):
3         self.rect = pygame.Rect(x, y, width, height)
4         self.image = pygame.image.load(image_path)
5         self.image = pygame.transform.scale(self.image, (width, height))
6     def draw(self, surface):
7         surface.blit(self.image, (self.rect.x, self.rect.y))
```

Penjelasan :

1. class GameObject:

Kelas GameObject mendefinisikan objek dasar dalam permainan yang memiliki posisi, ukuran, dan gambar untuk ditampilkan di layar. Kelas ini dapat digunakan sebagai dasar untuk objek-objek lain dalam game, seperti pemain, musuh, atau benda lainnya.

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

Fungsi __init__() adalah konstruktor dari kelas GameObject. Fungsi ini dipanggil saat objek dari kelas ini dibuat dan digunakan untuk menginisialisasi atribut-atribut objek.

- **Parameter:**

- 1) **x:** Koordinat horizontal posisi objek pada layar.
- 2) **y:** Koordinat vertikal posisi objek pada layar.
- 3) **width:** Lebar objek.
- 4) **height:** Tinggi objek.
- 5) **image_path:** Lokasi atau path ke file gambar yang digunakan untuk menggambar objek.

3. **self.rect = pygame.Rect(x, y, width, height)**

- **self.rect** adalah objek Rect dari Pygame yang digunakan untuk merepresentasikan posisi dan ukuran objek di layar.
 - **pygame.Rect(x, y, width, height)**: Membuat objek persegi panjang (rectangular) yang dimulai pada posisi (x, y) dan memiliki lebar (width) dan tinggi (height).
 - Objek Rect ini sangat penting dalam permainan karena digunakan untuk mendeteksi tabrakan (collision detection) dan posisi objek di layar.

4. **self.image = pygame.image.load(image_path)**

self.image adalah objek gambar yang dimuat dari file gambar yang ditentukan oleh **image_path**.

pygame.image.load(image_path): Fungsi ini digunakan untuk memuat gambar dari file yang ditunjukkan oleh **image_path**. Gambar ini akan disimpan dalam objek Surface yang digunakan untuk menggambar di layar.

5. **self.image = pygame.transform.scale(self.image, (width, height))**

- Fungsi ini digunakan untuk mengubah ukuran gambar agar sesuai dengan dimensi objek yang ditentukan.
 - 1) **pygame.transform.scale(self.image, (width, height))**: Mengubah ukuran gambar (**self.image**) sesuai dengan lebar dan tinggi yang diberikan (**width** dan **height**). Ini memastikan bahwa gambar memiliki ukuran yang sama dengan ukuran objek yang diinginkan.
 - 2) Gambar yang sudah diubah ukurannya akan disimpan kembali ke dalam **self.image**.

6. **def draw(self, surface):**

- Fungsi **draw()** digunakan untuk menggambar objek pada layar.
- **surface** adalah permukaan tempat objek akan digambar, yang biasanya merupakan layar permainan (screen) atau permukaan lain dalam game.

7. **surface.blit(self.image, (self.rect.x, self.rect.y))**

Fungsi **blit()** digunakan untuk menggambar gambar (**self.image**) pada permukaan (**surface**). **self.rect.x** dan **self.rect.y** adalah posisi koordinat objek yang diambil dari objek Rect (**self.rect**). Ini adalah posisi sudut kiri atas objek pada layar.

Dengan menggunakan **blit()**, gambar objek akan digambar pada posisi yang sesuai di layar, berdasarkan koordinat (**self.rect.x**, **self.rect.y**).

Class Pemain

```
1 class Player(GameObject):
2     def __init__(self, x, y, width, height, image_path):
3         super().__init__(x, y, width, height, image_path)
4         self.speed = 5
5         self.hp = 500
6
7     def move(self, keys, obstacles):
8         initial_position = self.rect.topleft
9
10        if keys[pygame.K_w]: self.rect.y -= self.speed
11        if keys[pygame.K_s]: self.rect.y += self.speed
12        if keys[pygame.K_a]: self.rect.x -= self.speed
13        if keys[pygame.K_d]: self.rect.x += self.speed
14
15        for obstacle in obstacles:
16            if self.rect.colliderect(obstacle.rect):
17                self.rect.topleft = initial_position
18
19    def draw(self, surface):
20        super().draw(surface)
21        draw_text(f"You: {self.hp} HP", font, (255, 255, 255), surface, 10, 10)
```

Penjelasan :

1. class Player(GameObject):

- **Kelas Player** adalah subclass (kelas turunan) dari kelas GameObject. Ini berarti Player mewarisi semua atribut dan metode yang ada di kelas GameObject. Kelas ini digunakan untuk mendefinisikan objek pemain dalam permainan, dengan tambahan fungsi dan atribut khusus untuk pemain.

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

- **Konstruktor (__init__):** Fungsi ini digunakan untuk menginisialisasi objek Player. Ketika objek Player dibuat, konstruktor ini dipanggil untuk menginisialisasi atribut objek.
 - **super().__init__(x, y, width, height, image_path):** Fungsi ini memanggil konstruktor dari kelas induk (GameObject) untuk menginisialisasi posisi (x, y), ukuran (width, height), dan gambar objek (image_path). Ini memastikan bahwa atribut dasar seperti posisi dan gambar diwarisi dari GameObject.
 - **self.speed = 5:** Atribut baru yang ditambahkan di kelas Player adalah speed. Ini mengatur kecepatan gerak pemain dalam permainan, dan diset ke 5 (yang berarti pemain bergerak 5 piksel setiap kali tombol ditekan).
 - **self.hp = 500:** Atribut hp mewakili jumlah kesehatan pemain, yang diset ke 500 pada awalnya.

3. def move(self, keys, obstacles):

- Fungsi move() digunakan untuk memindahkan pemain berdasarkan input dari tombol keyboard dan mencegah tabrakan dengan objek penghalang (obstacles).

- **keys:** Parameter ini adalah status tombol yang sedang ditekan pada keyboard. Biasanya diperoleh dengan `pygame.key.get_pressed()`, yang mengembalikan status setiap tombol.
- **obstacles:** Parameter ini adalah daftar objek penghalang, yang bisa berupa pohon, batu, atau objek lain yang bisa menghalangi pergerakan pemain.

Logika pergerakan pemain:

- **initial_position = self.rect.topleft:** Menyimpan posisi awal pemain dalam variabel `initial_position`. Ini digunakan untuk mengembalikan posisi pemain jika terjadi tabrakan dengan objek penghalang.
- **Pergerakan berdasarkan input dari pengguna:**
 - 1) **if keys[pygame.K_w]: self.rect.y -= self.speed:** Jika tombol "W" ditekan (tombol untuk bergerak ke atas), posisi pemain diubah dengan mengurangi nilai `y` pada `self.rect` sebesar `self.speed` (5).
 - 2) **if keys[pygame.K_s]: self.rect.y += self.speed:** Jika tombol "S" ditekan (tombol untuk bergerak ke bawah), posisi pemain diubah dengan menambahkan nilai `y` pada `self.rect` sebesar `self.speed`.
 - 3) **if keys[pygame.K_a]: self.rect.x -= self.speed:** Jika tombol "A" ditekan (tombol untuk bergerak ke kiri), posisi pemain diubah dengan mengurangi nilai `x` pada `self.rect` sebesar `self.speed`.
 - 4) **if keys[pygame.K_d]: self.rect.x += self.speed:** Jika tombol "D" ditekan (tombol untuk bergerak ke kanan), posisi pemain diubah dengan menambahkan nilai `x` pada `self.rect` sebesar `self.speed`.
- **Tabrakan dengan penghalang:**
 - 1) **for obstacle in obstacles::** Fungsi ini memeriksa setiap objek penghalang dalam daftar `obstacles`.
 - 2) **if self.rect.colliderect(obstacle.rect)::** Jika persegi panjang pemain (`self.rect`) bertabrakan dengan persegi panjang penghalang (`obstacle.rect`), maka posisi pemain dikembalikan ke posisi awal (`self.rect.topleft = initial_position`).

4. def draw(self, surface):

- Fungsi `draw()` digunakan untuk menggambar objek pemain pada permukaan (misalnya layar permainan).
 - 1) **super().draw(surface):** Memanggil metode `draw()` dari kelas induk (`GameObject`) untuk menggambar gambar objek pemain pada layar. Fungsi ini menggambar gambar pemain sesuai dengan posisi yang disimpan dalam `self.rect`.

- 2) **draw_text(f"You: {self.hp} HP", font, (255, 255, 255), surface, 10, 10):** Fungsi draw_text() digunakan untuk menampilkan teks yang menunjukkan jumlah kesehatan pemain (hp). Teks ini ditampilkan dalam warna putih pada posisi (10, 10) di sudut kiri atas layar.

Class Enemy

```
1 class Enemy(GameObject):
2     def __init__(self, x, y, width, height, image_path, hp):
3         super().__init__(x, y, width, height, image_path)
4         self.speed = 2
5         self.hp = hp
6
7     def move_towards_player(self, player, obstacles):
8         initial_position = self.rect.topleft
9
10        if self.rect.x < player.rect.x:
11            self.rect.x += self.speed
12        elif self.rect.x > player.rect.x:
13            self.rect.x -= self.speed
14
15        for obstacle in obstacles:
16            if self.rect.colliderect(obstacle.rect):
17                self.rect.topleft = initial_position
18                break
19
20        initial_position = self.rect.topleft
21
22        if self.rect.y < player.rect.y:
23            self.rect.y += self.speed
24        elif self.rect.y > player.rect.y:
25            self.rect.y -= self.speed
26
27        for obstacle in obstacles:
28            if self.rect.colliderect(obstacle.rect):
29                self.rect.topleft = initial_position
30                break
31
32    def draw(self, surface):
33        super().draw(surface)
34        draw_text(f"Enemy: {self.hp} HP", font, (255, 0, 0), surface, self.rect.x, self.rect.y - 20)
```

Penjelasan :

1. class Enemy(GameObject):

- **Kelas Enemy** adalah subclass (kelas turunan) dari kelas GameObject. Artinya, Enemy mewarisi semua atribut dan metode yang ada di kelas GameObject, dan menambahkan fungsionalitas khusus untuk objek musuh dalam permainan.

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

Konstruktor (__init__): Fungsi ini digunakan untuk menginisialisasi objek Enemy (musuh).

- 1) **super().__init__(x, y, width, height, image_path):** Fungsi ini memanggil konstruktor dari kelas induk (GameObject) untuk menginisialisasi atribut posisi (x, y), ukuran (width, height), dan gambar (image_path) dari objek musuh.
- 2) **self.speed = 2:** Atribut speed mengatur kecepatan musuh dalam permainan. Kecepatan ini diset ke 2, artinya musuh bergerak 2 piksel per frame.
- 3) **self.hp = hp:** Atribut hp adalah jumlah kesehatan musuh, yang diterima sebagai parameter pada saat pembuatan objek musuh.

3. def move_towards_player(self, player, obstacles):

Fungsi move_towards_player() digunakan untuk memindahkan musuh menuju posisi pemain (player), dengan mempertimbangkan kemungkinan tabrakan dengan penghalang (obstacles).

- 1) **player:** Parameter ini adalah objek pemain yang akan diikuti oleh musuh.
- 2) **obstacles:** Parameter ini adalah daftar objek penghalang yang bisa menghalangi gerakan musuh.

Logika pergerakan musuh:

- **initial_position = self.rect.topleft:** Menyimpan posisi awal musuh dalam variabel initial_position. Ini digunakan untuk mengembalikan posisi musuh jika terjadi tabrakan dengan objek penghalang.
- **Pergerakan musuh menuju pemain:**
 - 1) **if self.rect.x < player.rect.x:** Jika posisi x musuh lebih kecil dari posisi x pemain, maka musuh bergerak ke kanan (self.rect.x += self.speed).
 - 2) **elif self.rect.x > player.rect.x:** Jika posisi x musuh lebih besar dari posisi x pemain, maka musuh bergerak ke kiri (self.rect.x -= self.speed).
- **Memeriksa tabrakan dengan penghalang di arah x:**
 - 1) **for obstacle in obstacles:** Fungsi ini memeriksa setiap objek penghalang di dalam daftar obstacles.

- 2) **if self.rect.colliderect(obstacle.rect):** Jika musuh bertabrakan dengan penghalang pada posisi horizontal (x), maka posisi musuh dipulihkan ke posisi awal (`self.rect.topleft = initial_position`).
- **Pergerakan musuh secara vertikal (arah y):**
 - 1) **if self.rect.y < player.rect.y:** Jika posisi y musuh lebih kecil dari posisi y pemain, maka musuh bergerak ke bawah (`self.rect.y += self.speed`).
 - 2) **elif self.rect.y > player.rect.y:** Jika posisi y musuh lebih besar dari posisi y pemain, maka musuh bergerak ke atas (`self.rect.y -= self.speed`).
 - **Memeriksa tabrakan dengan penghalang di arah y:**
 - 1) **for obstacle in obstacles:** Fungsi ini memeriksa setiap objek penghalang di dalam daftar obstacles.
 - 2) **if self.rect.colliderect(obstacle.rect):** Jika musuh bertabrakan dengan penghalang pada posisi vertikal (y), maka posisi musuh dipulihkan ke posisi awal (`self.rect.topleft = initial_position`).

4. **def draw(self, surface):**

- Fungsi `draw()` digunakan untuk menggambar objek musuh pada permukaan (misalnya layar permainan).
 - 1) **super().draw(surface):** Memanggil metode `draw()` dari kelas induk (`GameObject`) untuk menggambar gambar objek musuh pada layar. Fungsi ini menggambar gambar musuh sesuai dengan posisi yang disimpan dalam `self.rect`.
 - 2) **draw_text(f'Enemy: {self.hp} HP', font, (255, 0, 0), surface, self.rect.x, self.rect.y - 20):** Fungsi `draw_text()` digunakan untuk menampilkan teks yang menunjukkan jumlah kesehatan musuh (hp). Teks ini ditampilkan dalam warna merah (255, 0, 0) di posisi sedikit di atas objek musuh, yaitu pada koordinat (`self.rect.x, self.rect.y - 20`).

Class Utama Game

```
1 class Game:
2     def __init__(self):
3         # Latar belakang
4         self.bg_image = pygame.image.load("map2.png")
5         self.bg_image = pygame.transform.scale(self.bg_image, (SCREEN_WIDTH, SCREEN_HEIGHT))
6
7         # Pemain
8         self.player = Player(100, 100, 60, 60, "player.png")
9
10        # Musuh
11        self.level = 1
12        self.enemy = self.create_enemy()
13
14        # Obstacle
15        self.plants = [
16            GameObject(random.randint(0, SCREEN_WIDTH - 32), random.randint(0, SCREEN_HEIGHT - 32), 60, 70, "pohon.png")
17            for _ in range(10)
18        ]
19        self.houses = [
20            GameObject(200, 150, 140, 100, "rumah.png"),
21            GameObject(500, 300, 140, 100, "rumah.png")
22        ]
23        self.ocks = [
24            GameObject(random.randint(0, SCREEN_WIDTH - 40), random.randint(0, SCREEN_HEIGHT - 40), 40, 40, "batu.png")
25            for _ in range(10)
26        ]
27
28    def create_enemy(self):
29        enemy_hp = 50 + (self.level - 1) * 20
30        enemy_images = ["enemy_level_1.png", "enemy_level_2.png", "enemy.png", "enemy_level_5.png"]
31        enemy_image = enemy_images[min(self.level - 1, len(enemy_images) - 1)]
32        return Enemy(random.randint(0, SCREEN_WIDTH - 40), random.randint(0, SCREEN_HEIGHT - 40), 60, 60, enemy_image, enemy_h
33    p)
34
35    def check_battle(self):
36        if self.player.rect.colliderect(self.enemy.rect):
37            self.enemy.hp -= 10
38            self.player.hp -= 5
39
40            if self.enemy.hp <= 0:
41                if self.level <= 4:
42                    self.win_screen()
43                else:
44                    print("Enemy defeated! Level up to (self.level + 1)")
45                    self.level += 1
46                    self.enemy = self.create_enemy()
47
48            if self.player.hp <= 0:
49                self.game_over_screen()
50
51    def game_over_screen(self):
52        screen.fill((0, 0, 0))
53        draw_text("GAME OVER", font, (255, 0, 0), screen, SCREEN_WIDTH // 2 - 100, SCREEN_HEIGHT // 2 - 50)
54        draw_text("Press R to Restart", font, (255, 255, 255), screen, SCREEN_WIDTH // 2 - 120, SCREEN_HEIGHT // 2)
55        pygame.display.flip()
56        self.wait_for_restart()
57
58    def win_screen(self):
59        screen.fill((0, 255, 0))
60        draw_text("YOU WIN!", font, (0, 0, 255), screen, SCREEN_WIDTH // 2 - 100, SCREEN_HEIGHT // 2 - 50)
61        pygame.display.flip()
62        self.wait_for_restart()
63
64    def wait_for_restart(self):
65        waiting = True
66        while waiting:
67            for event in pygame.event.get():
68                if event.type == pygame.QUIT:
69                    pygame.quit()
70                    sys.exit()
71                if event.type == pygame.KEYDOWN and event.key == pygame.K_r:
72                    waiting = False
73                    self.reset_game()
74
75    def reset_game(self):
76        self.__init__()
77
78    def run(self):
79        obstacles = self.plants + self.houses + self.ocks
80
81        while True:
82            for event in pygame.event.get():
83                if event.type == pygame.QUIT:
84                    pygame.quit()
85                    sys.exit()
86
87            keys = pygame.key.get_pressed()
88            self.player.move(keys, obstacles)
89            self.enemy.move_towards_player(self.player, obstacles)
90            self.check_battle()
91
92            # Gambar latar belakang
93            screen.blit(self.bg_image, (0, 0))
94
95            # Gambar semua objek
96            self.player.draw(screen)
97            if self.enemy.hp > 0:
98                self.enemy.draw(screen)
99            for plant in self.plants:
100                plant.draw(screen)
101            for house in self.houses:
102                house.draw(screen)
103            for rock in self.ocks:
104                rock.draw(screen)
105
106            pygame.display.flip()
107            clock.tick(FPS)
```

Penjelasan :

1. Class Game:

Kelas ini bertanggung jawab untuk mengelola seluruh aspek permainan, seperti pengaturan latar belakang, pemain, musuh, penghalang (obstacle), level, serta kondisi permainan seperti menang atau kalah.

2. `def __init__(self):`

- **Inisialisasi objek-objek permainan:** Fungsi konstruktor ini dipanggil saat pertama kali objek Game dibuat. Di sini, berbagai elemen permainan diinisialisasi.
 - 1) **`self.bg_image = pygame.image.load("map2.png")`:** Memuat gambar latar belakang dari file `map2.png`.
 - 2) **`self.bg_image = pygame.transform.scale(self.bg_image, (SCREEN_WIDTH, SCREEN_HEIGHT))`:** Mengubah ukuran gambar latar belakang agar sesuai dengan ukuran layar permainan.
 - 3) **`self.player = Player(100, 100, 60, 60, "player.png")`:** Membuat objek pemain dengan posisi awal (100, 100), ukuran (60, 60), dan gambar `player.png`.
 - 4) **Obstacles:**
 - **`self.plants`:** Membuat daftar objek penghalang berupa pohon yang tersebar acak di layar dengan ukuran (60, 70).
 - **`self.houses`:** Membuat daftar rumah yang tetap berada di posisi (200, 150) dan (500, 300).
 - **`self.rocks`:** Membuat daftar batu yang tersebar acak di layar dengan ukuran (40, 40).

3. `def create_enemy(self):`

- Fungsi ini membuat objek musuh baru.
 - 1) **`enemy_hp = 50 + (self.level - 1) * 20`:** Menentukan jumlah kesehatan musuh. Setiap level meningkatkan kesehatan musuh sebesar 20.
 - 2) **`enemy_images = ["enemy_level_1.png", "enemy_level_2.png", "enemy.png", "enemy_level_5.png"]`:** Daftar gambar musuh berdasarkan level.
 - 3) **`enemy_image = enemy_images[min(self.level - 1, len(enemy_images) - 1)]`:** Memilih gambar musuh berdasarkan level permainan, dengan pembatas agar tidak melebihi jumlah gambar yang tersedia.
 - 4) **`return Enemy(...)`:** Mengembalikan objek musuh baru dengan posisi acak, gambar yang dipilih, dan jumlah kesehatan yang ditentukan.

4. `def check_battle(self):`

- Fungsi ini memeriksa apakah terjadi pertempuran antara pemain dan musuh.
 - 1) **`if self.player.rect.colliderect(self.enemy.rect):`:** Jika posisi pemain bertabrakan dengan posisi musuh, maka pertempuran dimulai.

- 2) **self.enemy.hp -= 10**: Pemain menyerang musuh, mengurangi kesehatan musuh sebesar 10.
- 3) **self.player.hp -= 5**: Musuh menyerang pemain, mengurangi kesehatan pemain sebesar 5.

4) **Mengecek jika musuh kalah:**

Jika musuh kehilangan seluruh kesehatan (`self.enemy.hp <= 0`), dan jika level sudah mencapai 4, maka pemain menang dan menuju layar kemenangan, jika tidak, pemain naik level dan musuh baru dibuat.

5) **Mengecek jika pemain kalah:**

Jika kesehatan pemain habis (`self.player.hp <= 0`), maka layar permainan akan menampilkan Game Over.

5. def game_over_screen(self):

- Fungsi ini menampilkan layar Game Over.
 - **screen.fill((0, 0, 0))**: Mengisi layar dengan warna hitam.
 - **draw_text("GAME OVER", ...)**: Menampilkan teks "GAME OVER" dengan warna merah di tengah layar.
 - **draw_text("Press R to Restart", ...)**: Menampilkan pesan untuk menekan tombol R untuk memulai ulang permainan.
 - **pygame.display.flip()**: Memperbarui layar untuk menampilkan perubahan.
 - **self.wait_for_restart()**: Menunggu input dari pemain untuk memulai ulang permainan.

6. def win_screen(self):

- Fungsi ini menampilkan layar kemenangan.
 - **screen.fill((0, 255, 0))**: Mengisi layar dengan warna hijau (menandakan kemenangan).
 - **draw_text("YOU WIN!", ...)**: Menampilkan pesan "YOU WIN!" dengan warna biru di tengah layar.
 - **pygame.display.flip()**: Memperbarui layar.
 - **self.wait_for_restart()**: Menunggu input pemain untuk memulai ulang permainan.

7. def wait_for_restart(self):

- Fungsi ini membuat permainan menunggu input pemain untuk memulai ulang permainan setelah Game Over atau Kemenangan.
 - **for event in pygame.event.get():** Menangani event yang terjadi, seperti menutup permainan atau menekan tombol.
 - **if event.type == pygame.QUIT:** Jika pemain menutup permainan, maka keluar dari permainan dengan `pygame.quit()` dan `sys.exit()`.
 - **if event.type == pygame.KEYDOWN and event.key == pygame.K_r:** Jika pemain menekan tombol R, maka permainan akan mereset dengan memanggil `self.reset_game()`.

8. def reset_game(self):

- Fungsi ini digunakan untuk mereset permainan dengan memanggil kembali konstruktor `__init__()` untuk menginisialisasi ulang semua objek permainan.

9. def run(self):

- Fungsi utama untuk menjalankan permainan.
 - **obstacles = self.plants + self.houses + self.rock:** Menggabungkan semua objek penghalang (tanaman, rumah, batu) menjadi satu daftar.
 - **while True:** Loop utama permainan yang terus berjalan selama permainan aktif.
 - **for event in pygame.event.get():** Menangani semua event yang terjadi, seperti menutup permainan.
 - **keys = pygame.key.get_pressed():** Mengambil status tombol yang ditekan oleh pemain.
 - **self.player.move(keys, obstacles):** Memindahkan pemain berdasarkan input tombol dan menghindari tabrakan dengan penghalang.
 - **self.enemy.move_towards_player(self.player, obstacles):** Memindahkan musuh menuju pemain dan menghindari tabrakan dengan penghalang.
 - **self.check_battle():** Memeriksa apakah terjadi pertempuran antara pemain dan musuh.
 - **Gambar objek permainan:**
 - **screen.blit(self.bg_image, (0, 0)):** Menampilkan gambar latar belakang.
 - **self.player.draw(screen):** Menampilkan pemain.
 - **self.enemy.draw(screen):** Menampilkan musuh jika musuh masih hidup.

- **Gambar penghalang:** Menggambar tanaman, rumah, dan batu di layar.
- **pygame.display.flip():** Memperbarui layar.
- **clock.tick(FPS):** Mengatur kecepatan permainan agar tetap konsisten (frame per second).

Jalankan Game



```

1  if __name__ == "__main__"
2  _": game = Game()
3      game.run()

```

Penjelasan :

1. `if __name__ == "__main__":`:

- `__name__` adalah variabel khusus di Python yang berisi nama modul yang sedang dijalankan. Ketika sebuah skrip Python dijalankan langsung, Python akan menyetel nilai dari `__name__` menjadi `"__main__"`.
- Dengan menggunakan `if __name__ == "__main__":`, kita memastikan bahwa kode yang ada di dalam blok tersebut hanya akan dijalankan jika file tersebut dijalankan secara langsung sebagai skrip utama, bukan ketika file tersebut diimpor sebagai modul dalam program lain.

Contoh:

- Jika kita menjalankan file ini dengan `python game.py`, maka `__name__` akan bernilai `"__main__"`, dan kode dalam blok ini akan dijalankan.
- Jika file ini diimpor ke file lain dengan `import game`, maka `__name__` akan bernilai `"game"`, dan kode dalam blok ini tidak akan dieksekusi.

2. `game = Game():`

Ini membuat objek `game` yang merupakan instansi dari kelas `Game`. Konstruktor `__init` dari kelas `Game` akan dipanggil dan menyiapkan semua elemen permainan seperti pemain, musuh, objek penghalang, dan lainnya.

3. `game.run():`

Ini menjalankan permainan dengan memanggil metode `run()` dari objek `game`. Fungsi `run()` berisi loop utama permainan yang bertanggung jawab untuk menangani input pemain, memperbarui status permainan, menggambar objek ke layar, dan memeriksa kondisi permainan seperti pertempuran dan kemenangan.

Tantangan Pengembangan

Ketika mengembangkan game Enemy War menggunakan Pygame, terdapat berbagai tantangan yang harus diatasi. Salah satu tantangan utama adalah merancang struktur program menggunakan teknik Object-Oriented Programming (OOP), di mana pemisahan logika game ke dalam kelas-kelas seperti Player, Enemy, Attack, dan Game harus dirancang dengan baik agar memudahkan pengelolaan dan pengembangan fitur baru. Desain yang modular memungkinkan setiap bagian game dapat diubah atau diperbaiki tanpa memengaruhi keseluruhan sistem.

Selain itu, implementasi logika pergerakan musuh dan pemain menjadi tantangan penting, terutama saat menambahkan fitur seperti serangan (attack). Responsivitas kontrol pemain dan presisi serangan harus dijaga agar pengalaman bermain terasa menyenangkan. Pergerakan musuh, baik secara acak maupun mengikuti pola tertentu, juga perlu dirancang sedemikian rupa agar gameplay tetap menantang tanpa terasa tidak adil.

Tantangan lainnya adalah menjaga performa game ketika jumlah objek, seperti musuh dan serangan, meningkat. Loop logika game yang tidak optimal dapat menyebabkan permainan terasa lambat, terutama pada perangkat dengan spesifikasi rendah. Untuk mengatasi ini, optimasi kinerja menjadi hal penting, misalnya dengan membatasi jumlah objek aktif di layar atau menggunakan algoritma yang lebih efisien.

Faktor visual dan audio juga tidak boleh diabaikan. Memberikan efek visual untuk serangan, tabrakan, dan animasi musuh yang dihancurkan dapat meningkatkan daya tarik permainan. Efek suara, seperti suara serangan dan ledakan, juga penting untuk menciptakan pengalaman bermain yang imersif.

Pada tahap yang lebih lanjut, menambahkan sistem level, power-up, dan mode permainan yang bervariasi menjadi tantangan menarik. Fitur seperti tabel skor atau leaderboard juga dapat menambah daya tarik kompetitif pada game ini. Selain itu, implementasi AI sederhana untuk musuh yang mampu mengejar atau menghindari dari pemain dapat membawa game ke tingkat kesulitan yang lebih tinggi, sehingga memberikan pengalaman bermain yang lebih memuaskan.

Dokumentasi Game

1. Karakter Utama



Karakter utama adalah seorang ksatria berbaju baja berwarna hitam dan kuning, lengkap dengan pedang. Karakter ini melambangkan kekuatan dan keberanian.

2. Demon Enemy



Musuh tipe Demon memiliki bentuk yang menakutkan dengan warna gelap dan efek api. Demon ini bergerak dengan kecepatan sedang dan memiliki nyawa yang cukup besar.

3. Goblin Enemy



Goblin adalah musuh dengan kecepatan tinggi namun memiliki nyawa lebih sedikit. Goblin biasanya digunakan untuk membuat pemain lebih waspada karena pergerakannya cepat.

4. Wizard Enemy



Musuh tipe Wizard menyerang dari jarak jauh menggunakan sihir. Wizard memiliki tampilan berjubah dengan efek cahaya di sekitar serangannya.

5. Raja Iblis Enemy



Raja Iblis adalah bos terakhir dalam permainan. Memiliki desain yang besar, kuat, dan nyawa tertinggi. Raja Iblis memiliki serangan mematikan yang memerlukan strategi khusus untuk dikalahkan.

6. Map Pertama



Map pertama menggambarkan padang rumput yang luas dengan jalan setapak. Map ini memberikan tantangan awal dengan rute yang relatif sederhana.

7. Map Kedua



Map kedua memiliki elemen kompleks yang mencakup laut, danau, padang rumput, dan tundra. Variasi medan ini memberikan tantangan tambahan bagi pemain untuk menavigasi permainan.

8. Objek Batu



Batu digunakan sebagai penghalang dalam permainan. Batu ini berfungsi untuk memblokir pergerakan karakter pemain dan musuh, sehingga menciptakan tantangan strategis.

9. Objek Pohon



Pohon digunakan sebagai elemen dekorasi dan penghalang tambahan di dalam permainan. Objek ini membuat map terlihat lebih realistis.

10. Objek Rumah



Rumah berfungsi sebagai elemen lingkungan untuk menambahkan variasi visual dalam permainan. Rumah dapat digunakan sebagai