IMPLEMENTASI OBJECT ORIENTED PROGRAMMING PADA GAME “MAZE BLOCK”

Mata Kuliah

Pemrograman Berorientasi Objek

Oleh:

Reva Anindya Octavina

24091397042

2024B

PROGRAM STUDI D4 MANAJEMEN INFORMATIKA

FAKULTAS VOKASI

UNIVERSITAS NEGERI SURABAYA

2025

BAB 1

PENDAHULUAN

1.1. Latar Belakang

Pemrograman berorientasi objek telah membantu pengembang untuk memodelkan entitas dunia nyata sebagai objek yang memiliki atribut dan perilaku. Proyek ini menggunakan bahasa pemrograman python dan library pygame untuk diimplementasikan. Python digunakan untuk menunjang pengetahuan yang digunakan sebagai media pembelajaran selama perkuliahan, pygame digunakan untuk memberikan dan mengembangkan game visual 2D yang cukup sederhana dan sangat cocok untuk menerapkan bagaimana konsep OOP.

Sistem yang dikembangkan adalah permainan labirin 2D yang sederhana dengan nama “Maze Block” yang memuat tiga pilar utama dalam OOP di dalamnya yaitu Encapsulation, Inheritance, dan Polymorphism. Pemilihan permainan ini didasarkan untuk menciptakan lingkungan yang kaya objek dan juga untuk mengasah otak dalam bermain. Permainan ini menyediakan struktur yang ideal dalam menunjukkan bagaimana objek pemain (player), dinding (wall), dan koin (coin) dapat berinteraksi melalui aturan tabrakan yang berbeda yang efektif untuk memuat konsep polymorphism.

1.2. Tujuan

1. Untuk menerapkan tiga pilar utama dalam OOP (Encapsulation, Inheritance, dan Polymorphism) dalam pembuatan “Maze Block”.
2. Untuk memahami bagaimana penerapan tiga pilar utama dalam OOP ke dalam permainan “Maze Block”.
3. Untuk menciptakan permainan sederhana yang fungsional.

BAB 2

PERANCANGAN GAME

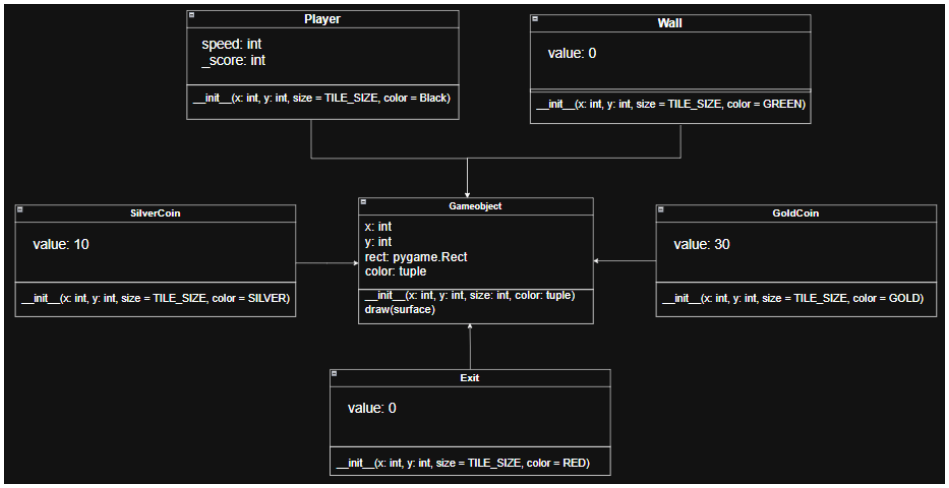
2.1. Kode Pembuatan Permainan

```python
import pygame

#Warna
TILE_SIZE = 32
BLACK = (0, 0, 0)
BROWN = (139, 69, 19)
GREEN = (124, 252, 0)
SILVER = (192, 192, 192)
GOLD = (255, 215, 0)
RED = (255, 0, 0)

class GameObject:
    def __init__(self, x, y, color, size=TILE_SIZE):
        self.x = x * size
        self.y = y * size
        self.size = size
        self.rect = pygame.Rect(self.x, self.y, size, size)
        self.color = color

    def draw(self, surface):
        pygame.draw.rect(surface, self.color, self.rect)

class Player(GameObject):
    def __init__(self, x, y, size=TILE_SIZE):
        super().__init__(x, y, BLACK, size)
        self.speed = TILE_SIZE
        self.__score = 0

    def get_score(self):
        return self.__score

    def add_score(self, amount):
        if amount > 0:
            self.__score += amount

    def move(self, dx, dy, walls):
        new_rect = self.rect.move(dx * self.speed, dy * self.speed)


        can_move = True
        for wall in walls:
            if new_rect.colliderect(wall.rect):
                can_move = False
                wall.on_collision(self)
                break

        if can_move:
            self.rect = new_rect
            self.x = self.rect.x
            self.y = self.rect.y
            return True
        return False

class Wall(GameObject):
    def __init__(self, x, y, size=TILE_SIZE):
        super().__init__(x, y, GREEN, size)

    def on_collision(self, player):
        pass

class SilverCoin(GameObject):
    def __init__(self, x, y, size=TILE_SIZE):
        super().__init__(x, y, SILVER, size)
        self.value = 10

    def on_collision(self, player):
        player.add_score(self.value)
        self.rect.width = 0
        self.rect.height = 0
        return True

class GoldCoin(GameObject):
    def __init__(self, x, y, size=TILE_SIZE):
        super().__init__(x, y, GOLD, size)
        self.value = 30

    def on_collision(self, player):
        player.add_score(self.value)
        self.rect.width = 0
        self.rect.height = 0
        return True

class Exit(GameObject):
    def __init__(self, x, y, size=TILE_SIZE):
        super().__init__(x, y, RED, size)

    def on_collision(self, player):
        return True
```

```python
# 'W': Wall, 'P': Player Start, 'C': Coin, 'E': Exit ' ': Empty
MAZE_MAP = [
    ['W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W'],
    ['W', 'P', ' ', ' ', 'W', ' ', ' ', ' ', 'S', ' ', ' ', 'W', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'W', 'W', 'W', ' ', ' ', 'W'],
    ['W', 'W', 'W', ' ', 'W', ' ', 'W', 'W', ' ', ' ', ' ', ' ', 'W', 'W', 'W', 'W', 'W', 'S', ' ', ' ', 'W', ' ', 'W', ' ', 'W'],
    ['W', 'W', 'S', ' ', ' ', ' ', 'W', ' ', ' ', 'W', 'W', 'W', 'W', 'W', ' ', ' ', ' ', 'W', ' ', 'W', ' ', ' ', ' ', 'W', 'W'],
    ['W', 'W', 'W', 'W', 'W', ' ', 'W', 'W', 'W', ' ', ' ', ' ', ' ', ' ', 'W', ' ', ' ', ' ', 'W', 'W', 'W', 'W', 'W', ' ', 'W'],
    ['W', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'W', 'S', ' ', ' ', ' ', ' ', ' ', 'W', 'W', 'W', 'W', 'W', 'W', ' ', ' ', ' ', 'W', 'W', 'G', 'W'],
    ['W', ' ', 'W', 'W', 'W', 'W', 'W', 'W', ' ', 'W', 'W', 'W', 'W', ' ', 'S', 'W', ' ', ' ', ' ', ' ', ' ', 'W', ' ', ' ', 'S', 'W', 'W'],
    ['W', 'S', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'W', 'W', ' ', ' ', ' ', 'W', ' ', ' ', ' ', 'W', 'W', 'W', 'W', 'W', 'W'],
    ['W', 'W', 'W', ' ', 'W', 'W', 'W', 'W', 'W', ' ', ' ', ' ', 'W', ' ', 'W', 'W', ' ', ' ', 'W', 'W', ' ', ' ', ' ', ' ', 'W', ' ', 'W', 'W'],
    ['W', ' ', ' ', ' ', 'W', 'S', ' ', ' ', 'W', 'W', ' ', ' ', ' ', ' ', 'W', 'W', ' ', 'W', 'W', 'W', 'W', 'W', ' ', 'W', ' ', 'W', 'W'],
    ['W', 'W', ' ', ' ', ' ', ' ', ' ', ' ', 'W', 'W', 'W', ' ', ' ', ' ', ' ', 'W', ' ', ' ', 'W', ' ', 'G', 'W', ' ', ' ', ' ', ' ', 'W'],
    ['W', 'W', ' ', 'W', 'W', ' ', 'W', 'W', ' ', 'W', 'S', 'W', 'W', ' ', ' ', ' ', 'W', ' ', 'W', ' ', 'W', 'W', 'W', 'W', 'W', 'G', 'W'],
    ['W', ' ', ' ', 'W', ' ', ' ', ' ', ' ', 'W', 'S', ' ', ' ', ' ', ' ', ' ', 'W', 'S', ' ', ' ', ' ', 'W', ' ', ' ', 'W', 'C', ' ', ' ', 'W', 'W'],
    ['W', ' ', 'W', 'W', 'S', 'W', 'W', 'W', ' ', ' ', 'W', 'W', 'W', ' ', ' ', 'W', 'W', 'W', 'W', 'W', 'W', ' ', ' ', ' ', ' ', 'W', ' ', 'W', 'W'],
    ['W', 'S', ' ', 'W', 'W', 'W', ' ', ' ', ' ', ' ', ' ', 'W', 'G', 'W', ' ', ' ', ' ', 'W', ' ', ' ', ' ', ' ', 'W', 'W', 'W', 'W', 'W', ' ', ' ', 'W'],
    ['W', 'W', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'W', 'W', ' ', ' ', 'W', 'W', ' ', ' ', ' ', ' ', 'W', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'W', ' ', ' ', 'W'],
    ['W', ' ', ' ', ' ', 'W', ' ', ' ', 'W', 'W', ' ', ' ', ' ', 'W', 'S', 'W', 'W', 'W', ' ', ' ', ' ', 'W', 'S', ' ', ' ', 'W', 'W', ' ', ' ', ' ', ' ', 'W'],
    ['W', ' ', 'W', ' ', ' ', ' ', 'W', ' ', ' ', ' ', 'W', ' ', ' ', 'W', ' ', ' ', ' ', ' ', ' ', ' ', 'W', ' ', 'W', 'W', ' ', ' ', ' ', ' ', 'W', 'S', 'W', 'W'],
    ['W', ' ', 'W', ' ', 'W', 'G', ' ', 'W', ' ', ' ', ' ', ' ', 'W', ' ', 'W', ' ', ' ', ' ', ' ', ' ', 'W', 'W', 'W', ' ', 'W', 'W', 'E', 'W'],
    ['W', ' ', ' ', ' ', ' ', 'S', 'W', 'W', 'W', ' ', ' ', 'W', ' ', ' ', 'W', ' ', 'W', 'W', 'S', 'W', 'W', 'W', 'W', ' ', ' ', ' ', 'W', ' ', ' ', 'W'],
    ['W', 'W', ' ', ' ', ' ', 'W', 'W', ' ', ' ', ' ', ' ', 'W', ' ', 'W', ' ', 'W', 'W', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'W', ' ', ' ', ' ', 'W', 'W'],
    ['W', 'W', 'W', ' ', ' ', ' ', ' ', ' ', 'W', 'S', ' ', ' ', ' ', 'W', ' ', ' ', ' ', ' ', ' ', 'W', 'W', ' ', 'W', 'S', ' ', ' ', ' ', 'W', 'W', 'W'],
    ['W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W']
]

MAP_WIDTH = len(MAZE_MAP[0])
MAP_HEIGHT = len(MAZE_MAP)

SCREEN_WIDTH = MAP_WIDTH * TILE_SIZE
SCREEN_HEIGHT = MAP_HEIGHT * TILE_SIZE
FPS = 60

pygame.init()
screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))
pygame.display.set_caption("Maze Block")
clock = pygame.time.Clock()

player = None
walls = []
collectables = []
exit_point = None

def load_maze(maze_map):
    """Membuat objek dari blueprint MAZE_MAP."""
    global player, exit_point, walls, collectables

    walls = []
    collectables = []

    for r, row in enumerate(maze_map):
        for c, tile in enumerate(row):
            if tile == 'W':
                walls.append(Wall(c, r))
            elif tile == 'P':
                player = Player(c, r)
            elif tile == 'S':
                collectables.append(SilverCoin(c, r))
            elif tile == 'G':
                collectables.append(GoldCoin(c, r))
            elif tile == 'E':
                exit_point = Exit(c, r)

def game_loop():
    running = True
    load_maze(MAZE_MAP)

    while running:
        dx, dy = 0, 0
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                running = False
            elif event.type == pygame.KEYDOWN:

                if event.key == pygame.K_UP:
                    dy = -1
                elif event.key == pygame.K_DOWN:
                    dy = 1
                elif event.key == pygame.K_LEFT:
                    dx = -1
                elif event.key == pygame.K_RIGHT:
                    dx = 1


        if player and (dx != 0 or dy != 0):
            player.move(dx, dy, walls)

        items_to_remove = []
        for item in collectables:
            if player.rect.colliderect(item.rect):

                if item.on_collision(player):
                    items_to_remove.append(item)

        for item in items_to_remove:
            collectables.remove(item)

        if exit_point and player.rect.colliderect(exit_point.rect):
            if exit_point.on_collision(player):
                print(f"YEYY MAZE BLOCK SELESAI! Skor Anda: {player.get_score()}")
                running = False # Game selesai

        screen.fill(BROWN)
```

```
194
195         for wall in walls:
196             wall.draw(screen)
197         for item in collectables:
198             item.draw(screen)
199         if exit_point:
200              exit_point.draw(screen)
201         if player:
202             player.draw(screen)
203
204
205         font = pygame.font.Font(None, 24)
206         score_text = font.render(f"Score: {player.get_score()}", True, (BLACK))
207         screen.blit(score_text, (5, 5))
208
209         pygame.display.flip()
210         clock.tick(FPS)
211
212     pygame.quit()
213
214 if __name__ == "__main__":
215     game_loop()
```

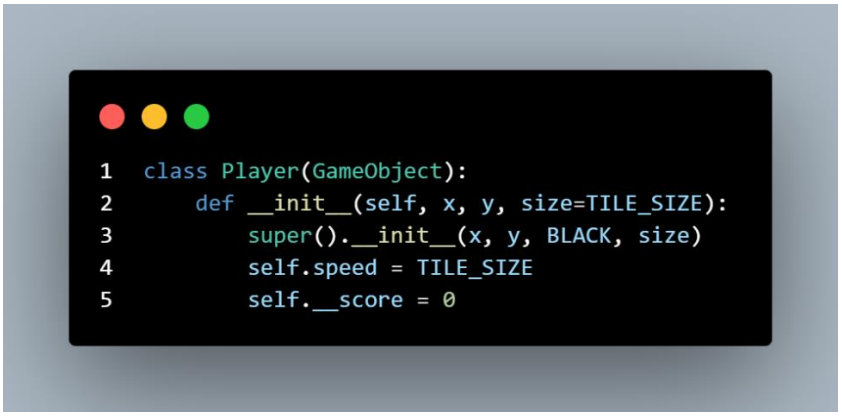
2.2. Class Diagram



1. **GameObject (Kelas Induk)**
   - Atribut:
     - X: int (objek dengan posisi horizontal)
     - Y: int (objek dengan posisi vertikal)
     - rect: pygame.Rect
     - color: tuple (warna objek)
   - Method:
     - draw(surface)
2. **Player (Kelas Turunan)**
   - Atribut:
     - Speed: int (kecepatan gerak pemain)
     - _score: int (skor pemain)
3. **Wall (Kelas Turunan)**
   - Atribut:
     - Value: 0
     - Warna: GREEN
     - Ukuran: TILE_SIZE
4. **SilverCoin (Kelas Turunan)**
   - Atribut:
     - Value: 10 (skor dengan poin 10)
     - Warna: SILVER
     - Ukuran: TILE_SIZE
5. **GoldCoin (Kelas Turunan)**
   - Atribut:
     - Value: 30 (skor dengan poin 30)
     - Warna: GOLD
     - Ukuran: TILE_SIZE
6. **Exit:**
   - Atribut:
     - Value: 0
     - Warna: RED
     - Ukuran: TILE_SIZE

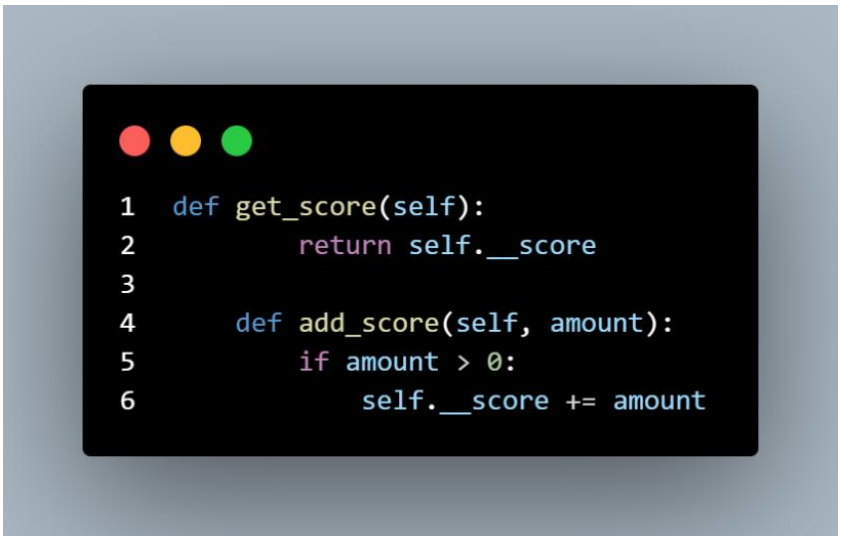
2.3. Implementasi Prinsip OOP

2.3.1. Encapsulation



```
class Player(GameObject):
    def __init__(self, x, y, size=TILE_SIZE):
        super().__init__(x, y, BLACK, size)
        self.speed = TILE_SIZE
        self.__score = 0
```

Gambar di atas merupakan atribut privat dalam encapsulation, penggunaan **double underscore** pada nama atribut **(__score)** menandai atribut ini sebagai atribut yang privat yang tidak boleh diakss atau diubah secara langsung.



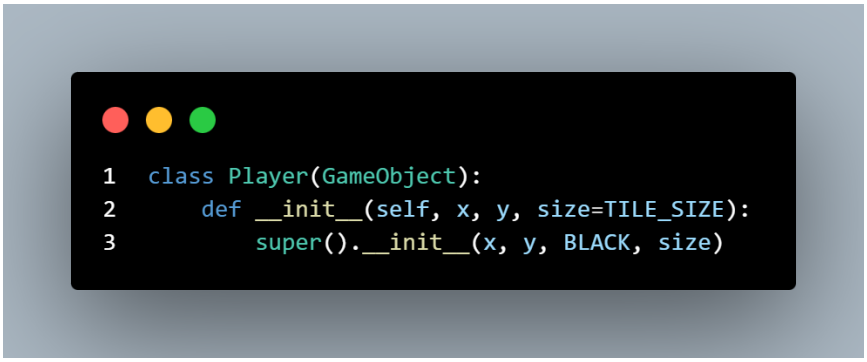
```
def get_score(self):
        return self.__score

    def add_score(self, amount):
        if amount > 0:
            self.__score += amount
```

Gambar di atas merupakan metode getter dalam encapsulation. (get_score) dan (add_score) berfungsi untuk mengontrol bagaimana score dapat diakses dan diubah.

2.3.2. Inheritance

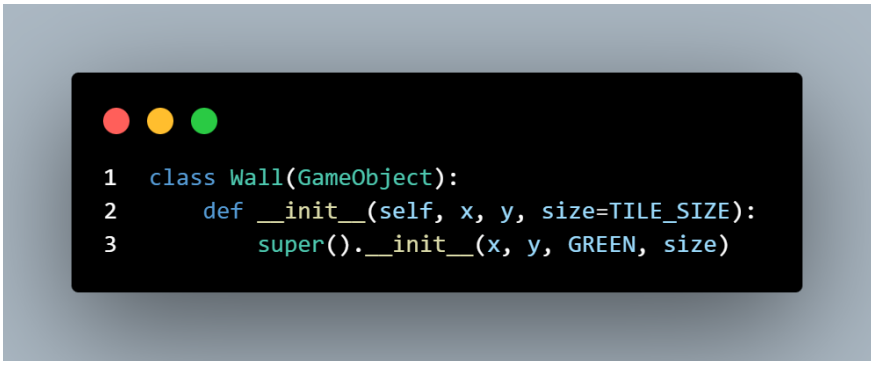
```python
class GameObject:
    def __init__(self, x, y, color, size=TILE_SIZE):
        self.x = x * size
        self.y = y * size
        self.size = size
        self.rect = pygame.Rect(self.x, self.y, size, size)
        self.color = color

    def draw(self, surface):
        pygame.draw.rect(surface, self.color, self.rect)
```

Gambar di atas merupakan class GameObject yang berfungsi sebagai kelas induk (superclass). Kelas ini akan mewarisi segala metode ke kelas turunannya.



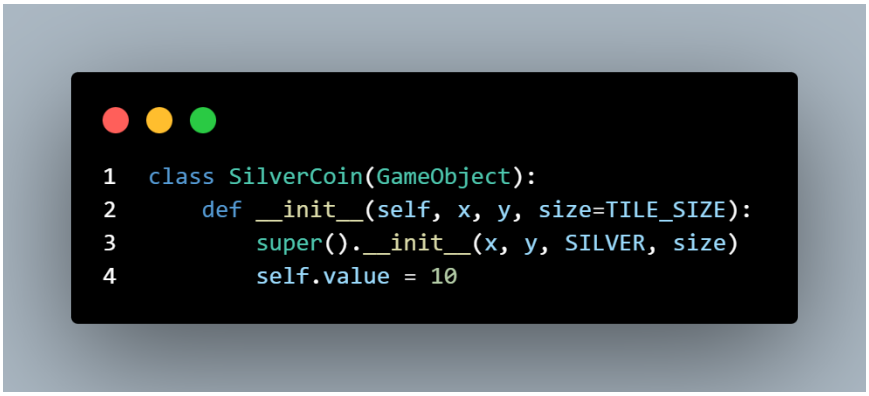
```python
class Player(GameObject):
    def __init__(self, x, y, size=TILE_SIZE):
        super().__init__(x, y, BLACK, size)
```

Gambar di atas merupakan class Player (pemain) yang merupakan kelas turunan dari class GameObject. Kelas ini diwarisi segala atribut dan metode yang kelas induk punya seperti posisi, ukuran, serta metode. “BLACK” pada kode tersebut menandakan warna yang akan muncul pada player.



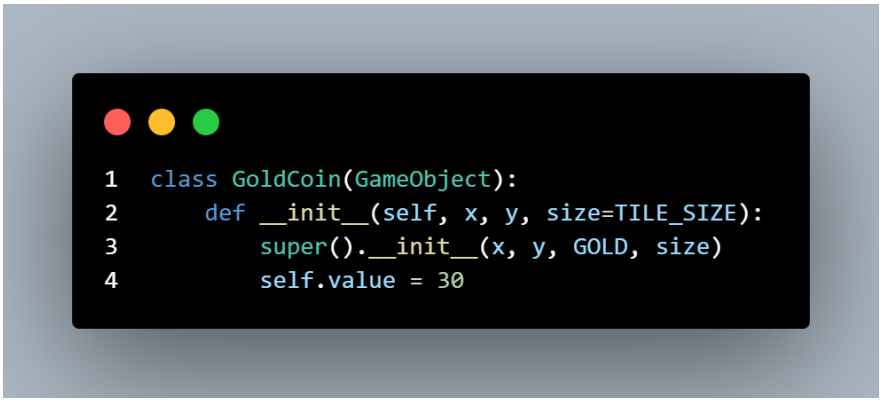
```python
class Wall(GameObject):
    def __init__(self, x, y, size=TILE_SIZE):
        super().__init__(x, y, GREEN, size)
```

Gambar di atas merupakan class Wall (dinding) yang merupakan kelas turunan dari class GameObject. Kelas ini diwarisi segala atribut dan metode yang kelas induk punya seperti posisi, ukuran, serta metode. “GREEN” pada kode tersebut menandakan warna yang akan muncul pada dinding.



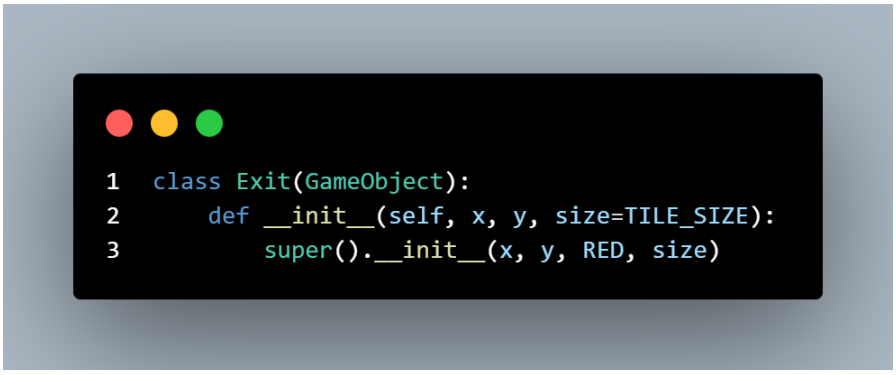
```
class SilverCoin(GameObject):
    def __init__(self, x, y, size=TILE_SIZE):
        super().__init__(x, y, SILVER, size)
        self.value = 10
```

Gambar di atas merupakan class SilverCoin (koin perak) yang merupakan kelas turunan dari class GameObject. Kelas ini diwarisi segala atribut dan metode yang kelas induk punya seperti posisi, ukuran, serta metode. “SILVER” pada kode tersebut menandakan warna yang akan muncul pada koin. “self.value = 10” menandakan poin yang didapatkan ketika player menabrak koin perak.



```
class GoldCoin(GameObject):
    def __init__(self, x, y, size=TILE_SIZE):
        super().__init__(x, y, GOLD, size)
        self.value = 30
```

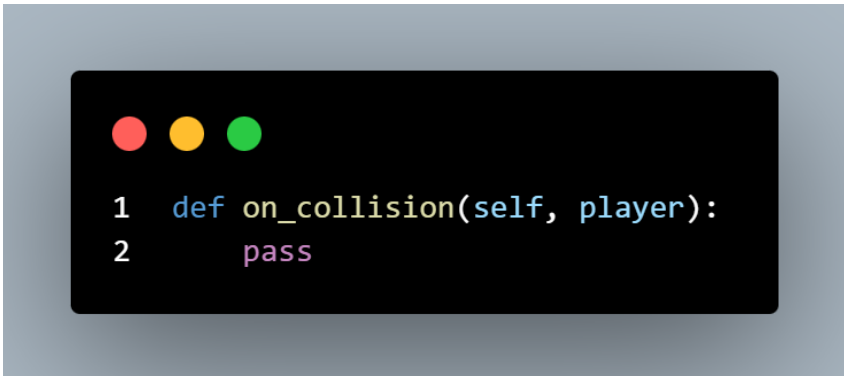
Gambar di atas merupakan class GoldCoin (koin emas) yang merupakan kelas turunan dari class GameObject. Kelas ini diwarisi segala atribut dan metode yang kelas induk punya seperti posisi, ukuran, serta metode. “GOLD” pada kode tersebut menandakan warna yang akan muncul pada koin. “self.value = 30” menandakan poin yang didapatkan ketika player menabrak koin emas.



```
class Exit(GameObject):
    def __init__(self, x, y, size=TILE_SIZE):
        super().__init__(x, y, RED, size)
```

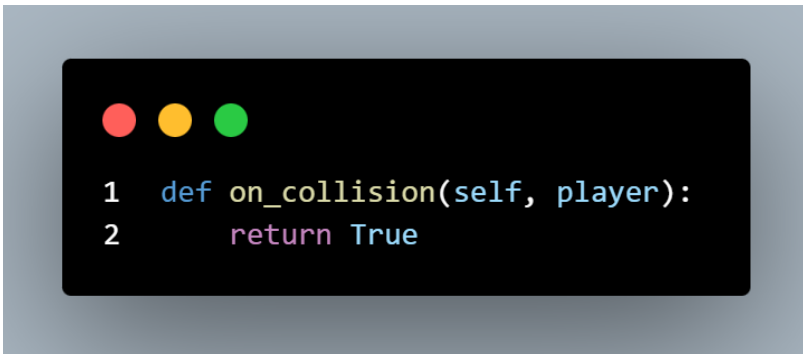
Gambar di atas merupakan class Exit (pintu keluar) yang merupakan kelas turunan dari class GameObject. Kelas ini diwarisi segala atribut dan metode yang kelas induk punya seperti posisi, ukuran, serta metode. “RED” pada kode tersebut menandakan warna yang akan muncul pada pintu keluar.

2.3.3. Polymorphism



Gambar di atas merupakan konsep polimorfisme pada class Wall yang diimplementasikan ke dalam metode overriding dari metode on_collision. “pass” pada kode ini tidak akan melakukan apapun karena memang tujuannya untuk menghentikan semua pergerakan pemain.

Gambar di atas ini merupakan kode yang ada pada class SilverCoin dan GoldCoin yang fungsinya untuk menambahkan skor setiap kali player menabrak koin, dan mengembalikan “True”.



Gambar di atas merupakan kode yang ada pada class Exit, berfungsi untuk mengembalikan “True” yang menyampaikan bahwa permainan telah selesai.

2.4. Fungsionalitas Utama

Seluruh pergerakan objek dan tabrakan pada dinding ditangani oleh metode move yang ada pada class Player. Metode ini dirancang untuk selalu memastikan bahwa pergerakan pemain (player) tidak akan menembus objek dinding saat permainan sedang dijalankan. Struktur data pada permainan ini yaitu berupa list yang di ada pada variabel MAZE_MAP yang di mana setiap elemen nya bertindak sebagai simbol. P = player (pemain), W = wall (dinding), S = silver coin (koin perak), G = gold coin (koin emas), dan E = exit (pintu keluar). Peta pada sistem ini menggunakan fungsi load_maze yang di mana kode ini melakukan looping yaitu perulangan baris dan kolom melalui MAZE_MAP yang digunakan sebagai dinding labirin.

2.5. Implementasi Fitur Tambahan

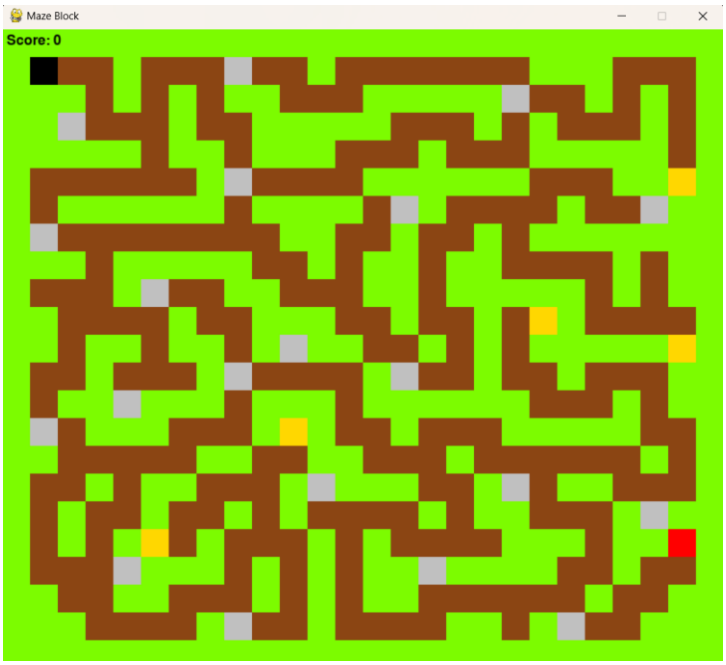
Keunikan permainan ini terdapat pada koin dan juga warna yang diterapkan. Permainan ini memiliki 2 tipe koin yang berbeda di dalamnya, terdapat koin berwarna perak (silver coin) dan koin berwarna emas (gold coin) yang di mana poin di setiap koinnya juga berbeda. Pada koin perak memiliki poin dengan total 10, sedangkan koin emas memiliki poin yang lebih tinggi dari pada koin perak, yaitu 30 poin. Kemudian untuk warna yang diterapkan pun berbeda-beda pada setiap objek. Untuk pemain (player) memiliki warna hitam, dinding labirin (wall) memiliki warna hijau, lantai dasar memiliki warna cokelat. Pemilihan warna ini didasari oleh warna rumput dan juga tanah yang memiliki warna hijau pada rumput yang digunakan sebagai dinding labirin dan warna cokelat pada tanah yang digunakan sebagai lantai dasar.

BAB 3

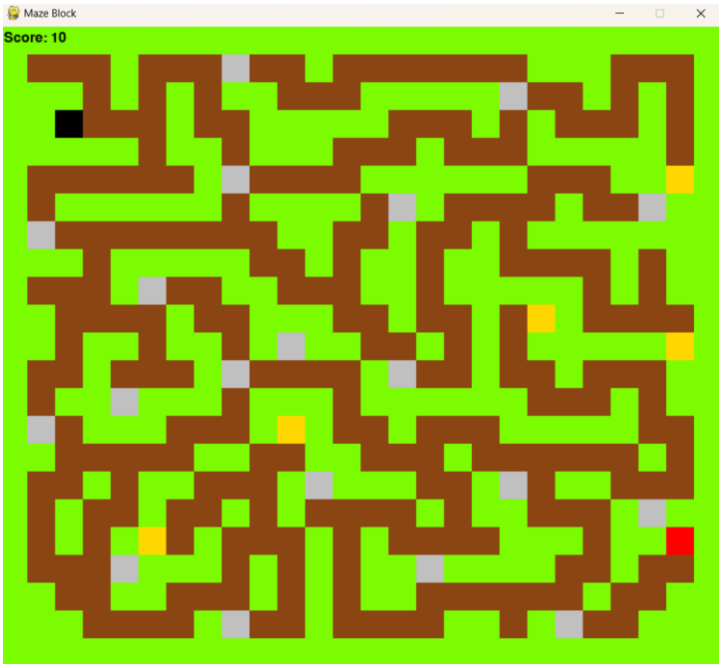
HASIL

3.1. Tampilan Permainan

3.1.1. Tampilan Awal Permainan

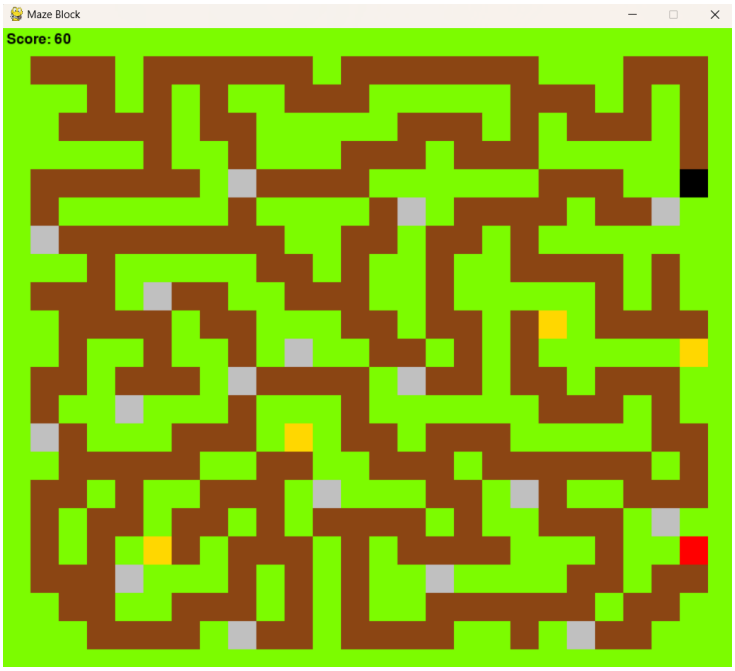


3.1.2. Tampilan Saat Menabrak Koin Perak



Gambar di atas merupakan tampilan saat pemain (player) menabrak koin perak, skor berubah bertambah 10 poin.

3.1.3. Tampilan Saat Menabrak Koin Emas



Gambar di atas merupakan tampilan saat pemain (player) menabrak koin emas, skor akan bertambah 30 poin. Skor berubah menjadi 60 poin karena pemain telah menabrak 3 koin perak yang di mana koin perak memiliki poin 10.

3.1.4. Tampilan Saat Permainan Selesai

```
pygame 2.6.1 (SDL 2.28.4, Python 3.11.0)
Hello from the pygame community. https://www.pygame.org/contribute.html
YEYY MAZE BLOCK SELESAI! Skor Anda: 350
PS D:\PBO>
```

Gambar di atas merupakan tampilan saat permainan telah selesai dimainkan, muncul kalimat bahwa permainan telah selesai dan juga menampilkan total skor yang pemain dapatkan selama bermain.

3.2. Tantangan dan Solusi

Adapun tantangan yang dihadapi selama membuat permainan Maze_Block ini diantara yaitu pemilihan warna yang cocok untuk permainan ini serta pencarian kode untuk warna-warna yang digunakan. Akhirnya untuk pemilihan warna yang digunakan diambil dari referensi warna rumput dan tanah, kemudian untuk pencarian kode warna yang digunakan ini membutuhkan waktu yang cukup untuk dicocokkan kembali dengan pemilihan warna rumput dan tanah agar lebih nyaman untuk dilihat.

Kemudian untuk tantangan yang kedua ada pada peletakkan dinding dan koin. Membutuhkan waktu yang cukup lama juga untuk memikirkan letak dinding dan koin yang menghasilkan bentuk labirin yang pas agar nyaman untuk dilihat dan tidak membuat pusing. Solusi dari tantangan ini adalah membuka referensi dari berbagai sumber tentang bentuk labirin dan peletakkan koin perak dan emas untuk menghadirkan tantangan yang seru.

BAB 4

PENUTUP

4.1. Kesimpulan

Berdasarkan implementasi yang telah selesai dilakukan, dapat disimpulkan bahwa pembuatan permainan labirin dengan nama Maze_Block ini telah berhasil mencapai tujuan utamanya dengan mengimplementasikan tiga pila utama dalam OOP yaitu encapsulation, inheritance, dan polymorphism.

1. Encapsulation ditunjukkan melalui class Player dengan melindungi data seperti (score), dan hanya data (add_score, get_score) yang merupakan data public.
2. Inheritance ditunjukkan melalui class induk GameObject yang mewarisi atribut yang dimilikinya seperti ukuran dan posisi ke semua entitas yaitu pemain (player), dinding (wall), koin perak (silver coin), koine mas (gold coin), dan pintu keluar (exit).
3. Polymorphism ditunjukkan melalui metode (on_collision) yang menghasilkan perilaku yang bermacam-macam.