

**Laporan UAS Pemrograman Berorientasi Objek**

**Game Obstacle Car Rush**



**Dosen Pengampu :**

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

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

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

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

**Disusun Oleh :**

Muhammad Ulil Amri / 23091397091

M. Syafi'ul Masruri / 23091397085

Ika Cahya Agustina / 23091397096

**Program Studi D4 Manajemen Informatika**

**Fakultas Vokasi**

**Universitas Negeri Surabaya**

**Tahun 2024/2025**

## **Obstacle Car Rush**

Obstacle Car Rush, pemain akan mengendalikan mobil yang melaju di jalanan yang penuh tantangan. Tugas pemain adalah menghindari rintangan yang muncul secara acak di sepanjang jalan. Setiap detik yang [emain lewati tanpa menabrak rintangan akan menambah skor kamu! Semakin lama pemain bertahan, semakin cepat jalanan bergerak, dan semakin sulit rintangan yang muncul. Dengan setiap 100 poin yang dikumpulkan, permainan akan semakin menantang karena kecepatan akan meningkat.

Laporan UAS Pemrograman Berorientasi Objek Kelompok kami membuat permainan menggunakan pygame yang berjudul “Obstacle Car Rush”. Melalui proyek ini, kami mengimplementasikan prinsip-prinsip dasar OOP yaitu enkapsulasi, Inheritance, Polymorphism dan abstraksi ke dalam pengembangan sebuah game sederhana. Berikut adalah hasil laporan dari perencanaan hingga pengimplementasian OOP dalam pembuatan permainan Obstacle Car Rush menggunakan pygame.

### **Ketentuan**

- 1. Diagram class harus mencerminkan desain aplikasi dan mencakup semua hubungan antar kelas.**
- 2. Laporan harus mencakup penjelasan lengkap tentang konsep OOP yang diterapkan, diagram class, alur aplikasi, serta tantangan yang dihadapi selama pengembangan.**

*Berikut ini adalah keseluruhan kodenya:*

```
1 import pygame
2 import random
3 import os
4
5 # Inisialisasi Pygame
6 pygame.init()
7
8 # Konfigurasi Layar
9 WIDTH, HEIGHT = 800, 600
10 screen = pygame.display.set_mode((WIDTH, HEIGHT))
11 pygame.display.set_caption("Balapan Mobil OOP")
12
13 # Warna
14 WHITE = (255, 255, 255)
15 BLACK = (0, 0, 0)
16 RED = (255, 0, 0)
17
18 # Kecepatan permainan
19 FPS = 30
20 clock = pygame.time.Clock()
21
22 # File untuk menyimpan skor tertinggi
23 HIGH_SCORE_FILE = "high_score.txt"
24
25 # Gambar mobil untuk dipilih
26 CAR_IMAGES = ["car1.png", "car2.png", "car3.png"] # Ganti dengan path gambar mobil
27
28
29 # Fungsi untuk membaca skor tertinggi dari file
30 def read_high_score():
31     if os.path.exists(HIGH_SCORE_FILE):
32         with open(HIGH_SCORE_FILE, "r") as file:
33             return int(file.read())
34     else:
35         return 0
36
37
38 # Fungsi untuk menyimpan skor tertinggi ke file
39 def save_high_score(score):
40     with open(HIGH_SCORE_FILE, "w") as file:
41         file.write(str(score))
42
43
44 # Fungsi untuk menampilkan menu pemilihan mobil
45 def select_car_menu():
46     selected_car = 0 # Default pilihan pertama
47     menu_running = True
48     font = pygame.font.Font(None, 48)
```

```

1  while menu_running:
2      screen.fill(BLACK)
3
4      # Tampilkan teks judul
5      title_text = font.render("Select Your Car", True, WHITE)
6      screen.blit(title_text, (WIDTH // 2 - title_text.get_width() // 2, 50))
7
8      # Tampilkan gambar mobil untuk dipilih
9      for i, car_image in enumerate(CAR_IMAGES):
10         car_img = pygame.image.load(car_image).convert_alpha()
11         car_img = pygame.transform.scale(car_img, (140, 140)) # Resize gambar mobil
12         x_pos = WIDTH // 4 * (i + 1) - car_img.get_width() // 2
13         y_pos = HEIGHT // 2 - car_img.get_height() // 2
14         screen.blit(car_img, (x_pos, y_pos))
15
16         # Highlight mobil yang dipilih
17         if i == selected_car:
18             pygame.draw.rect(
19                 screen,
20                 WHITE,
21                 (
22                     x_pos - 5,
23                     y_pos - 5,
24                     car_img.get_width() + 10,
25                     car_img.get_height() + 10,
26                 ),
27                 2,
28             )
29
30     # Event handling untuk navigasi menu
31     for event in pygame.event.get():
32         if event.type == pygame.QUIT:
33             pygame.quit()
34             exit()
35         elif event.type == pygame.KEYDOWN:
36             if event.key == pygame.K_LEFT and selected_car > 0:
37                 selected_car -= 1
38             elif event.key == pygame.K_RIGHT and selected_car < len(CAR_IMAGES) - 1:
39                 selected_car += 1
40             elif event.key == pygame.K_RETURN:
41                 menu_running = False
42
43     # Update layar
44     pygame.display.flip()
45     clock.tick(FPS)
46
47     return CAR_IMAGES[selected_car]

```

```

1  # Class untuk mobil pemain
2  class Car:
3      def __init__(self, car_image):
4          self.width = 70 # Lebar mobil lebih kecil
5          self.height = 70 # Tinggi mobil lebih kecil
6          self.x = WIDTH // 2 - self.width // 2
7          self.y = HEIGHT - self.height - 10
8          self.speed = 5
9          self.image = pygame.image.load(car_image).convert_alpha()
10         self.image = pygame.transform.scale(
11             self.image, (self.width, self.height)
12         ) # Resize gambar mobil
13
14     def draw(self):
15         screen.blit(self.image, (self.x, self.y))
16
17     def move(self, keys):
18         if keys[pygame.K_LEFT] and self.x > 133:
19             self.x -= self.speed
20         if keys[pygame.K_RIGHT] and self.x < 667 - self.width:
21             self.x += self.speed
22         if keys[pygame.K_UP] and self.y > 0:
23             self.y -= self.speed
24         if keys[pygame.K_DOWN] and self.y < HEIGHT - self.height:
25             self.y += self.speed
26
27
28 # Class untuk rintangan
29 class Obstacle:
30     def __init__(self, speed):
31         self.width = 30
32         self.height = 40
33         # Menentukan area jalan di mana obstacle dapat muncul
34         road_left_bound = 133 # Batas kiri jalan
35         road_right_bound = 667 - self.width # Batas kanan jalan
36         self.x = random.randint(
37             road_left_bound, road_right_bound
38         ) # Random dalam area jalan
39         self.y = -self.height
40         self.speed = speed
41         self.color = RED
42
43     def draw(self):
44         pygame.draw.rect(screen, self.color, (self.x, self.y, self.width, self.height))
45
46     def move(self):
47         self.y += self.speed
48
49
50 class FastObstacle(Obstacle):
51     def __init__(self, speed):
52         super().__init__(speed)
53         self.speed = speed * 2 # Rintangan lebih cepat

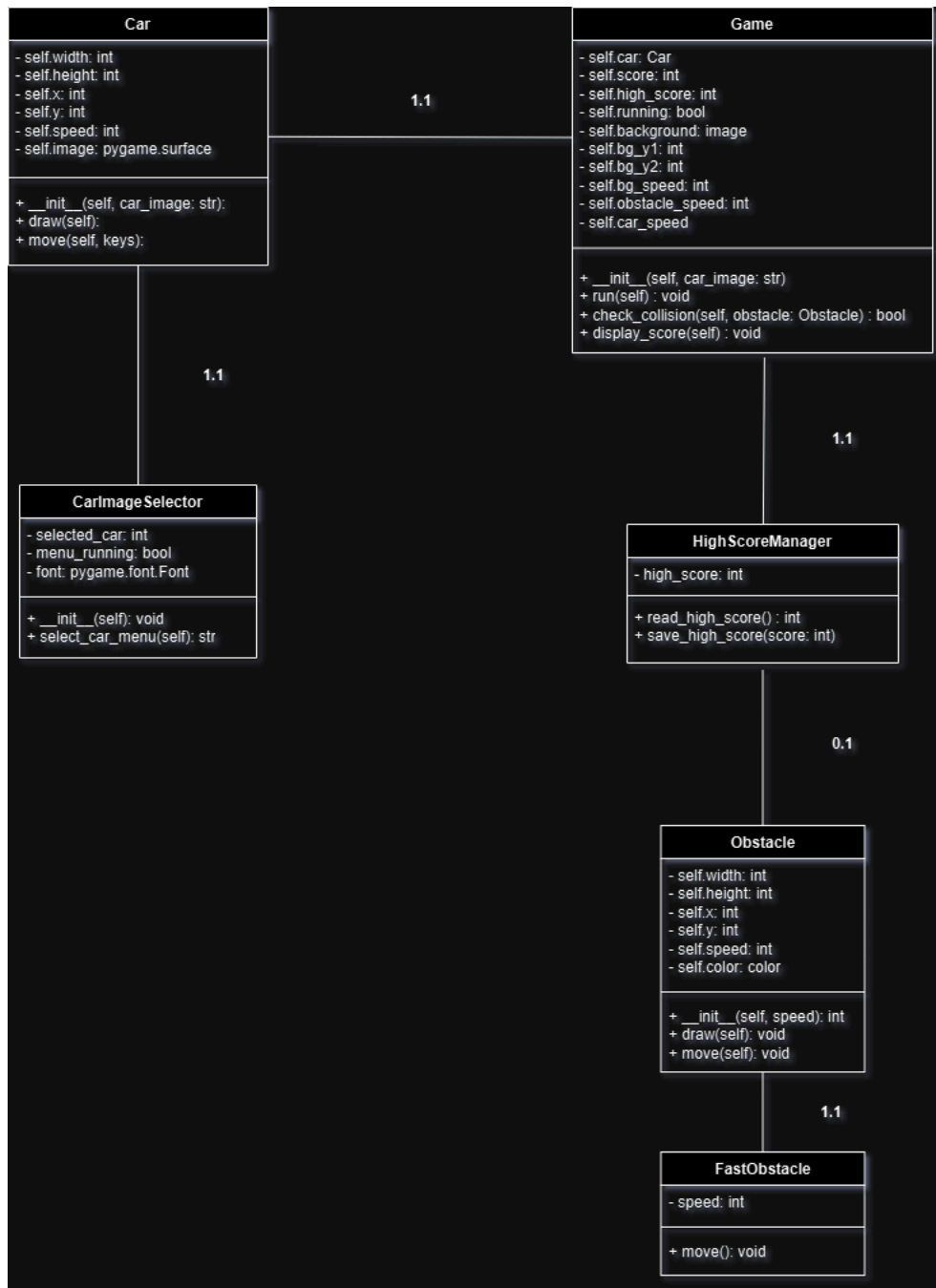
```



```
1  # Class utama untuk mengatur permainan
2  class Game:
3      def __init__(self, car_image):
4          self.car = Car(car_image)
5          self.obstacles = []
6          self.score = 0
7          self.high_score = read_high_score()
8          self.running = True
9          self.background = pygame.image.load("road.png").convert()
10         self.background = pygame.transform.scale(self.background, (WIDTH, HEIGHT))
11         self.bg_y1 = 0
12         self.bg_y2 = -HEIGHT
13         self.bg_speed = 5
14         self.obstacle_speed = 6
15         self.car_speed = 5
16
17     def run(self):
18         while self.running:
19             clock.tick(FPS)
20
21             self.score += 1
22
23             if self.score % 100 == 0:
24                 self.bg_speed += 1
25                 self.obstacle_speed += 1
26                 self.car_speed += 1
27
28             self.bg_y1 += self.bg_speed
29             self.bg_y2 += self.bg_speed
30
31             if self.bg_y1 >= HEIGHT:
32                 self.bg_y1 = -HEIGHT
33             if self.bg_y2 >= HEIGHT:
34                 self.bg_y2 = -HEIGHT
35
36             screen.blit(self.background, (0, self.bg_y1))
37             screen.blit(self.background, (0, self.bg_y2))
38
39             for event in pygame.event.get():
40                 if event.type == pygame.QUIT:
41                     self.running = False
42
43             keys = pygame.key.get_pressed()
44             self.car.move(keys)
45
46             if random.randint(1, 50) == 1:
47                 self.obstacles.append(Obstacle(self.obstacle_speed))
```

```
1         for obstacle in self.obstacles:
2             obstacle.move()
3             obstacle.draw()
4             if self.check_collision(obstacle):
5                 print(f"Game Over! Final Score: {self.score}")
6                 if self.score > self.high_score:
7                     self.high_score = self.score
8                     save_high_score(self.high_score)
9                 self.running = False
10
11         self.obstacles = [
12             obstacle for obstacle in self.obstacles if obstacle.y < HEIGHT
13         ]
14         self.car.draw()
15         self.display_score()
16         pygame.display.flip()
17
18     def check_collision(self, obstacle):
19         return (
20             self.car.x < obstacle.x + obstacle.width
21             and self.car.x + self.car.width > obstacle.x
22             and self.car.y < obstacle.y + obstacle.height
23             and self.car.y + self.car.height > obstacle.y
24         )
25
26     def display_score(self):
27         font = pygame.font.Font(None, 36)
28         score_text = font.render(f"Score: {self.score}", True, WHITE)
29         screen.blit(score_text, (10, 10))
30         high_score_text = font.render(f"High Score: {self.high_score}", True, WHITE)
31         screen.blit(high_score_text, (WIDTH - 200, 10))
32
33
34 # Menjalankan game
35 if __name__ == "__main__":
36     selected_car_image = select_car_menu()
37     game = Game(selected_car_image)
38     game.run()
39     pygame.quit()
40
```

Berikut ini adalah Class Diagram:



### Gambaran Umum Class Diagram

Class diagram ini terdiri dari lima kelas utama: Car, Obstacle, FastObstacle, Game, dan HighScore. Selain itu, terdapat class tambahan seperti CarImageSelector untuk membantu pemilihan mobil. Diagram ini menunjukkan hubungan antar-class, atribut, dan metode, serta menggambarkan bagaimana objek saling berinteraksi dalam sistem game.



## **Penjelasan Tiap Class**

### **A. Class Car. bahwa ini adalah representasi dari mobil pemain. atributnya:**

- "x dan y menentukan posisi mobil di layar, sedangkan speed menentukan kecepatan mobil."
- Metode: "draw() digunakan untuk menggambar mobil di layar, dan move() digunakan untuk menggerakkan mobil berdasarkan input pemain."

### **B. Class Obstacle. Menjelaskan bahwa ini merepresentasikan rintangan dalam game.**

- "x dan y menentukan posisi rintangan, sedangkan speed menentukan kecepatan rintangan."
- Metode: "move() membuat rintangan bergerak ke bawah, dan draw() digunakan untuk menggambar rintangan."

### **C. Class High Score. Menjelaskan bahwa ini bertanggung jawab untuk mencatat skor tertinggi.**

- "file\_path menyimpan lokasi file skor, dan score menyimpan nilai skor tertinggi."
- Metode: "read\_score() membaca skor tertinggi dari file, dan save\_score() menyimpan skor baru jika lebih tinggi."

### **D. Class Game. Bahwa ini adalah class utama yang mengatur logika permainan.**

- "car adalah objek dari class Car, obstacles adalah daftar rintangan yang muncul, dan score menyimpan skor pemain."
- Metode: "run() adalah loop utama permainan, dan check\_collision() memeriksa tabrakan antara mobil pemain dan rintangan."

### **E. Class CarImageSelector. Menjelaskan bahwa ini adalah class tambahan untuk memilih mobil sebelum permainan dimulai.**

- Metode: "select\_car\_menu() menampilkan menu pemilihan mobil dan mengembalikan gambar mobil yang dipilih pemain."

### **F. Class FastObstacle. Class ini adalah turunan dari class Obstacle, merepresentasikan rintangan dengan kecepatan lebih tinggi.**

- (Inherited) x, y, dan speed dari class Obstacle. boost\_speed: Tambahan kecepatan untuk rintangan.
- Metode: (Overridden) move(): Menggerakkan rintangan lebih cepat dengan menambahkan boost\_speed ke speed. (Inherited) draw(): Menggambar rintangan cepat di layar.

## Relasi Class Diagram

### 1. **Game → Car**

- Cardinalitas: 1..1

Setiap Game hanya memiliki satu objek Car, dan Car hanya dapat dimiliki oleh satu Game.

### 2. **Game → Obstacle**

- Cardinalitas: 1.. (one-to-many)

Setiap Game dapat memiliki banyak Obstacle, tetapi setiap Obstacle hanya dimiliki oleh satu Game.

### 3. **Game → HighScore**

- Cardinalitas: 1..1

Setiap Game terhubung dengan satu objek HighScore, dan HighScore hanya digunakan oleh satu Game.

### 4. **Game → CarImageSelector**

- Cardinalitas: 1..1

Setiap Game menggunakan satu objek CarImageSelector, dan CarImageSelector hanya digunakan oleh satu Game.

### 5. **FastObstacle → Obstacle**

- Cardinalitas: 1..1 (Inheritance)

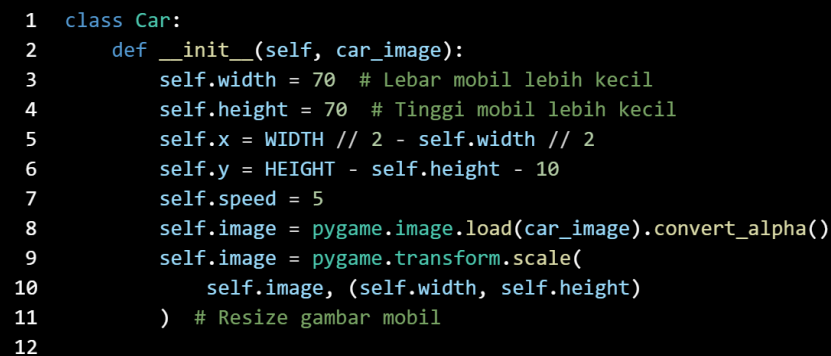
FastObstacle adalah subclass dari Obstacle, sehingga setiap FastObstacle adalah satu jenis Obstacle.

## Penjelasan Konsep OOP

### A. Enkapsulasi

Merupakan proses menyembunyikan detail implementasi suatu objek dan hanya menyediakan antarmuka tertentu untuk berinteraksi dengan objek tersebut. Setiap komponen utama dalam permainan (mobil, rintangan, permainan) diimplementasikan sebagai kelas dengan atribut (data) dan metode (fungsi) yang berhubungan.

- Class Car membuat data mobil pemain
  - Konstruktor: `__init__()` berfungsi menginisialisasi properti dari mobil pemain, seperti ukuran, posisi, kecepatan, dan gambar mobil.



```
1 class Car:
2     def __init__(self, car_image):
3         self.width = 70 # Lebar mobil lebih kecil
4         self.height = 70 # Tinggi mobil lebih kecil
5         self.x = WIDTH // 2 - self.width // 2
6         self.y = HEIGHT - self.height - 10
7         self.speed = 5
8         self.image = pygame.image.load(car_image).convert_alpha()
9         self.image = pygame.transform.scale(
10             self.image, (self.width, self.height)
11         ) # Resize gambar mobil
12
```

- Atribut:
  1. `self.x`, `self.y` Posisi mobil pada layar.
  2. `self.width`, `self.height` Ukuran mobil.
  3. `self.speed` Kecepatan gerak mobil.
  4. `image` Gambar mobil pemain.
- Metode:
  1. `draw()` menggambar mobil pemain di layar.
  2. `move()` untuk mengatur tombol panah untuk menggerakkan mobil ke empat arah: kiri, kanan, atas, dan bawah.
    - A. keys Input dari keyboard yang dicek menggunakan `pygame.key.get_pressed()`.
    - B. `pygame.K_LEFT` Jika tombol panah kiri ditekan dan posisi mobil belum melewati batas kiri jalan (`self.x > 133`), maka mobil bergerak ke kiri dengan kecepatan `self.speed`.
    - C. `pygame.K_RIGHT` Mobil bergerak ke kanan jika tidak melewati batas kanan jalan.
    - D. `pygame.K_UP` Mobil bergerak ke atas selama belum mencapai tepi atas layar.

- E. `pygame.K_DOWN` Mobil bergerak ke bawah selama belum mencapai tepi bawah layar.

```
1 def draw(self):
2     screen.blit(self.image, (self.x, self.y))
3
4 def move(self, keys):
5     if keys[pygame.K_LEFT] and self.x > 133:
6         self.x -= self.speed
7     if keys[pygame.K_RIGHT] and self.x < 667 - self.width:
8         self.x += self.speed
9     if keys[pygame.K_UP] and self.y > 0:
10        self.y -= self.speed
11    if keys[pygame.K_DOWN] and self.y < HEIGHT - self.height:
12        self.y += self.speed
```

- Class `Obstacle` membuat data rintangan
  - Konstruktor: `__init__(self, speed)` berfungsi mengatur properti awal rintangan, seperti posisi, ukuran, warna, dan kecepatan.

```
1 class Obstacle:
2     def __init__(self, speed):
3         self.width = 30
4         self.height = 40
5         # Menentukan area jalan di mana obstacle dapat muncul
6         road_left_bound = 133 # Batas kiri jalan
7         road_right_bound = 667 - self.width # Batas kanan jalan
8         self.x = random.randint(
9             road_left_bound, road_right_bound
10        ) # Random dalam area jalan
11        self.y = -self.height
12        self.speed = speed
13        self.color = RED
```

- Atribut:
  1. `self.x`, `self.y` Posisi rintangan pada layar.
  2. `self.width`, `self.height` Ukuran rintangan.
  3. `self.speed` Kecepatan jatuh rintangan.

- Metode:
  1. `draw()` Menggambar rintangan sebagai kotak berwarna RED.
  2. `move()` Menggerakkan rintangan ke bawah.

```

1 def draw(self):
2     pygame.draw.rect(screen, self.color, (self.x, self.y, self.width, self.height))
3
4 def move(self):
5     self.y += self.speed

```

## B. Inheritance

Merupakan mekanisme yang memungkinkan sebuah kelas (kelas turunan/child class) mewarisi atribut dan metode dari kelas lain (kelas induk/parent class). Menjadi lebih efisien karena kelas turunan dapat menggunakan kembali atribut dan metode dari kelas induk.

- Class Game pengatur permainan
  - Konstruktor: `__init__(self, car, image)` digunakan untuk menginisialisasi atribut yang diperlukan ketika objek Game dibuat.

```

1 class Game:
2     def __init__(self, car_image):
3         self.car = Car(car_image)
4         self.obstacles = []
5         self.score = 0
6         self.high_score = read_high_score()
7         self.running = True
8         self.background = pygame.image.load("road.png").convert()
9         self.background = pygame.transform.scale(self.background, (WIDTH, HEIGHT))
10        self.bg_y1 = 0
11        self.bg_y2 = -HEIGHT
12        self.bg_speed = 5
13        self.obstacle_speed = 6
14        self.car_speed = 5

```

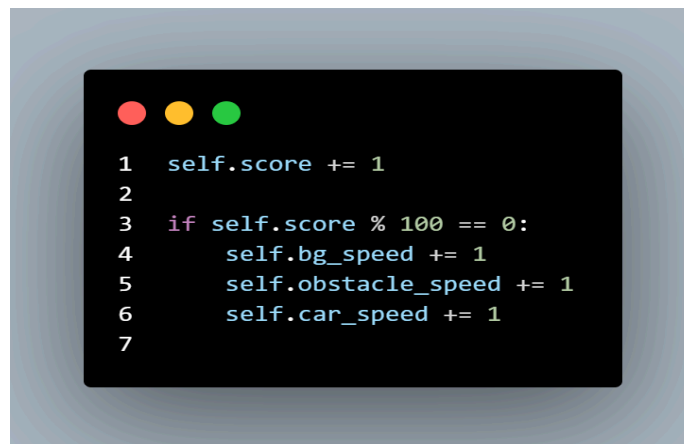
- Atribut:
  1. `self.car` Membuat objek mobil pemain menggunakan kelas
  2. `Car` dengan gambar mobil yang dipilih.
  3. `self.obstacles` Daftar kosong untuk menyimpan objek rintangan yang muncul selama permainan.
  4. `self.score` Skor pemain yang dimulai dari 0 dan bertambah seiring permainan berjalan.
  5. `self.high_score` Membaca skor tertinggi dari file menggunakan fungsi `read_high_score()`.

6. `self.running` Variabel flag untuk menjaga agar permainan terus berjalan (loop utama).
7. `self.background` Memuat gambar latar belakang jalan dan menyesuaikan ukurannya agar sesuai dengan ukuran layar.
8. `self.bg_y1`, `self.bg_y2`: Dua variabel untuk membuat efek scrolling background (jalan bergerak).
9. `self.bg_speed`, `self.obstacle_speed`, `self.car_speed` Kecepatan awal untuk pergerakan background, rintangan, dan mobil pemain.

- Metode:

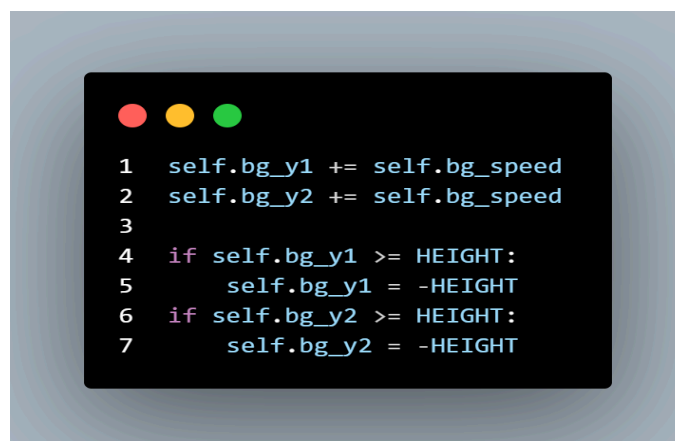
1. `run()` sebagai inti permainan yaitu:

- Skor pemain bertambah setiap iterasi loop, setiap kelipatan 100 poin, kecepatan permainan meningkat.



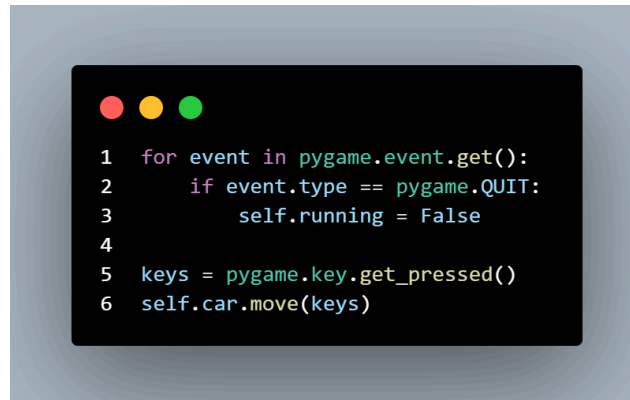
```
1 self.score += 1
2
3 if self.score % 100 == 0:
4     self.bg_speed += 1
5     self.obstacle_speed += 1
6     self.car_speed += 1
7
```

- Membuat efek scrolling background, latar belakang bergerak ke bawah dengan kecepatan `self.bg_speed`. Jika gambar latar sudah keluar layar, posisinya direset ke atas.



```
1 self.bg_y1 += self.bg_speed
2 self.bg_y2 += self.bg_speed
3
4 if self.bg_y1 >= HEIGHT:
5     self.bg_y1 = -HEIGHT
6 if self.bg_y2 >= HEIGHT:
7     self.bg_y2 = -HEIGHT
```

- Menginput perintah keyboard jika pemain menekan tombol tutup layar (QUIT) maka permainan berhenti. Jika tombol panah ditekan maka mobil bergerak `self.car.move(keys)`.



```

1  for event in pygame.event.get():
2      if event.type == pygame.QUIT:
3          self.running = False
4
5  keys = pygame.key.get_pressed()
6  self.car.move(keys)

```

- Membuat, memindahkan, dan menghapus gambar rintangan. Rintangan baru ditambahkan secara acak. Setiap rintangan dalam daftar `self.obstacles` akan digerakkan dan digambar di layar. Jika terjadi tabrakan permainan berhenti. Rintangan yang sudah keluar layar dibuang dari daftar.

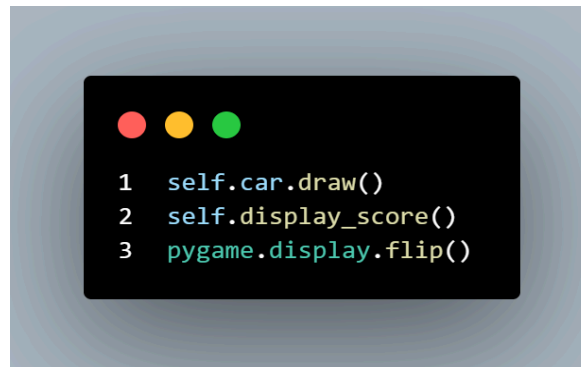


```

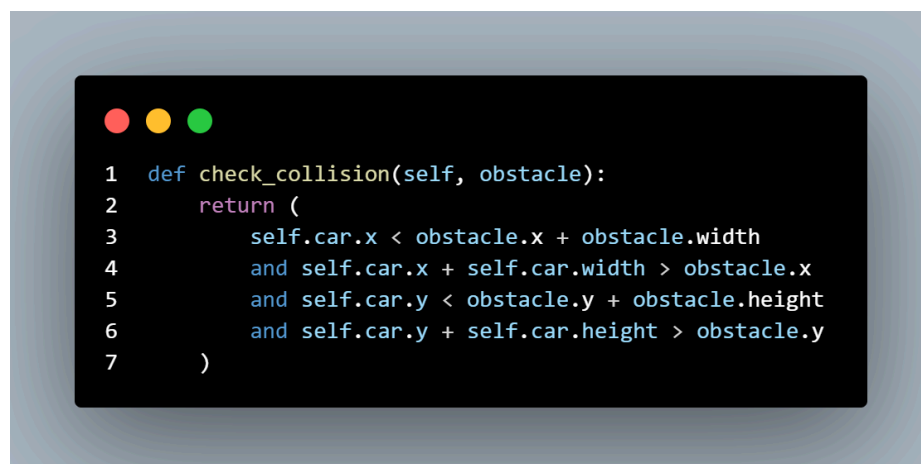
1  if random.randint(1, 50) == 1:
2      self.obstacles.append(Obstacle(self.obstacle_speed))
3
4  for obstacle in self.obstacles:
5      obstacle.move()
6      obstacle.draw()
7      if self.check_collision(obstacle):
8          print(f"Game Over! Final Score: {self.score}")
9          if self.score > self.high_score:
10             self.high_score = self.score
11             save_high_score(self.high_score)
12             self.running = False
13
14  self.obstacles = [
15      obstacle for obstacle in self.obstacles if obstacle.y < HEIGHT
16  ]

```

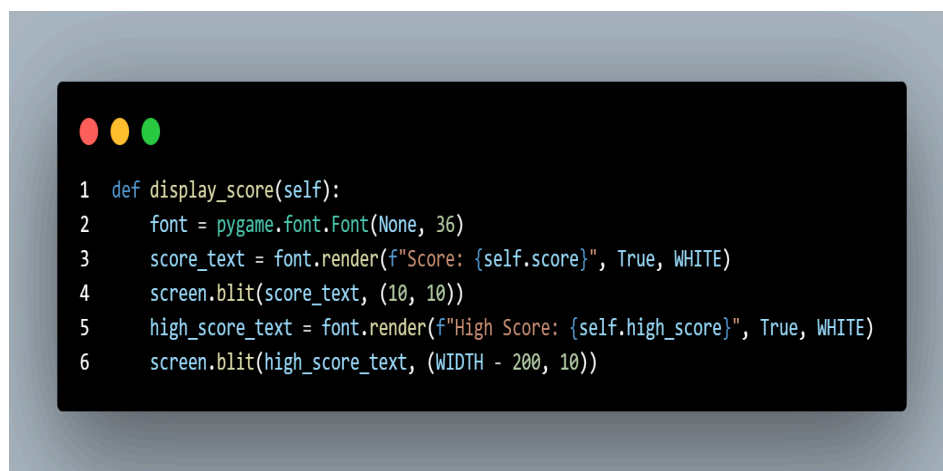
- Menggambar mobil pemain menggunakan `self.car.draw()` dan menampilkan skor di layar dengan `self.display_score()`.



2. `check_collision()` untuk mengecek logika tabrakan antara mobil pemain dan rintangan. Jika posisi mobil pemain bertemu dengan posisi rintangan (berdasarkan koordinat x dan y), terjadi tabrakan.



3. `display_score()` menampilkan skor pemain dan skor tertinggi di layar selama permainan. Prosesnya yaitu membuat teks skor dengan menggunakan font bawaan pygame, menggambar skor saat ini di pojok kiri atas, dan menggambar skor tertinggi di pojok kanan atas.



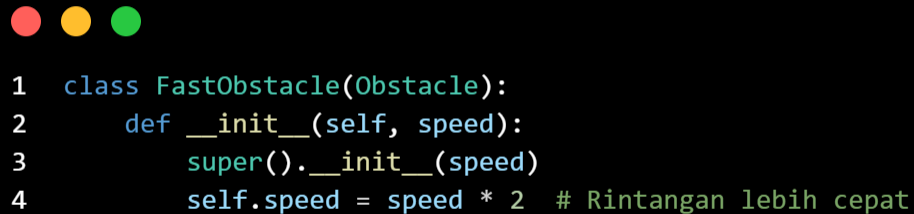


### C. Polymorphism

Merupakan metode atau objek untuk memiliki bentuk atau perilaku yang berbeda berdasarkan konteks penggunaannya. Menambahkan kelas turunan rintangan yang memiliki perilaku berbeda tetapi menggunakan metode yang berbeda. Dengan membuat 2 rintangan kecepatan yang berbeda yaitu Obstacle (kecepatan normal) dan FastObstacle (kecepatan dua kali lipat). Berikut kode beserta penjelasannya:

Pada kodingan ini, digunakan dua kelas yaitu Obstacle (kecepatan normal) dan FastObstacle (kecepatan dua kali lipat). Kelas FastObstacle merupakan turunan dari kelas Obstacle. Dengan polymorphism, meskipun kedua kelas memiliki atribut speed, perilaku dari atribut tersebut berbeda tergantung pada implementasi kelas.

Pada kelas FastObstacle, kecepatan dihitung sebagai dua kali lipat dari kecepatan normal, yang diatur melalui metode konstruktor `__init__`. Pemanggilan `super().__init__(speed)` memastikan bahwa konstruktor dari kelas induk (Obstacle) tetap dipanggil, sehingga pewarisan atribut atau metode dari kelas induk tidak terganggu. Setelah itu, atribut `self.speed` dimodifikasi untuk mencerminkan peningkatan kecepatan. Keterangan Class FastObstacle untuk menambahkan kecepatan pemain menjadi 2 kali lipat yang tertera pada metode `self.speed = speed * 2`

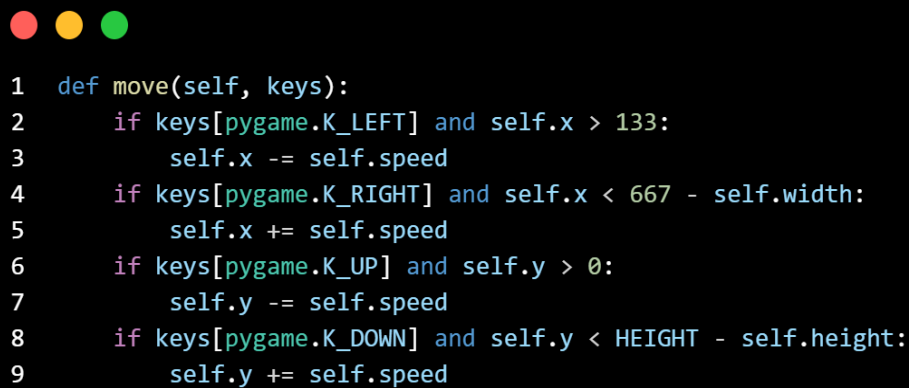
A code editor window with a dark background and three colored window control buttons (red, yellow, green) at the top left. It contains Python code for the FastObstacle class.

```
1 class FastObstacle(Obstacle):
2     def __init__(self, speed):
3         super().__init__(speed)
4         self.speed = speed * 2 # Rintangan lebih cepat
```

#### D. Abstraksi

Merupakan proses menyembunyikan detail implementasi yang kompleks dan hanya menampilkan fungsionalitas yang penting kepada pengguna. Pengguna dapat memanggil `car.move(keys)` untuk menggerakkan mobil tanpa memikirkan logika pergerakan. Logika permainan disederhanakan dalam metode `run()` pada kelas `Game`. Berikut kode beserta penjelasannya:

1. Metode yang mengatur gerakan horizontal (kiri-kanan) dan vertikal (atas-bawah) berdasarkan input dari tombol panah pada keyboard.
2. Terdapat batasan posisi agar objek tidak keluar dari area permainan:
  - Kiri:  $x > 133$
  - Kanan:  $x < 667 - \text{self.width}$
  - Atas:  $y > 0$
  - Bawah:  $y < \text{HEIGHT} - \text{self.height}$
3. `self.speed` menentukan seberapa jauh objek bergerak pada setiap input, memberikan kesan kecepatan.
4. `pygame.K_LEFT`, `pygame.K_RIGHT`, `pygame.K_UP`, `pygame.K_DOWN` adalah konstanta dari `pygame` yang mewakili tombol panah.



```
1 def move(self, keys):
2     if keys[pygame.K_LEFT] and self.x > 133:
3         self.x -= self.speed
4     if keys[pygame.K_RIGHT] and self.x < 667 - self.width:
5         self.x += self.speed
6     if keys[pygame.K_UP] and self.y > 0:
7         self.y -= self.speed
8     if keys[pygame.K_DOWN] and self.y < HEIGHT - self.height:
9         self.y += self.speed
```

## **Tantangan Pengembangan Game (Teknis & Desain )**

### **A. Tantangan Teknis**

#### **1. Optimasi Performa**

Game harus berjalan lancar di berbagai perangkat dengan spesifikasi berbeda.

#### **2. Deteksi Tabrakan (Collision Detection)**

Memastikan tabrakan antara mobil pemain dan rintangan terdeteksi dengan akurat tanpa memperlambat performa

### **B. Tantangan Desain**

#### **1. Desain Visual**

Grafik dan suara harus menarik tanpa mengganggu pengalaman bermain.

#### **2. Pemilihan Gambar Mobil (CarImageSelector)**

Pemain mungkin merasa bosan jika pilihan mobil terlalu sedikit atau tidak menarik.

#### **3. Penanganan Skor dan Progres Pemain**

Skor pemain harus disimpan dengan aman dan dapat diakses kembali.

## Output Tampilan Game “Obstacle Car Rush”

