LAPORAN TUGAS AKHIR
PEMROGRAMAN BERORIENTASI OBJEK
PING PONG GAME

Dosen Pengampu:

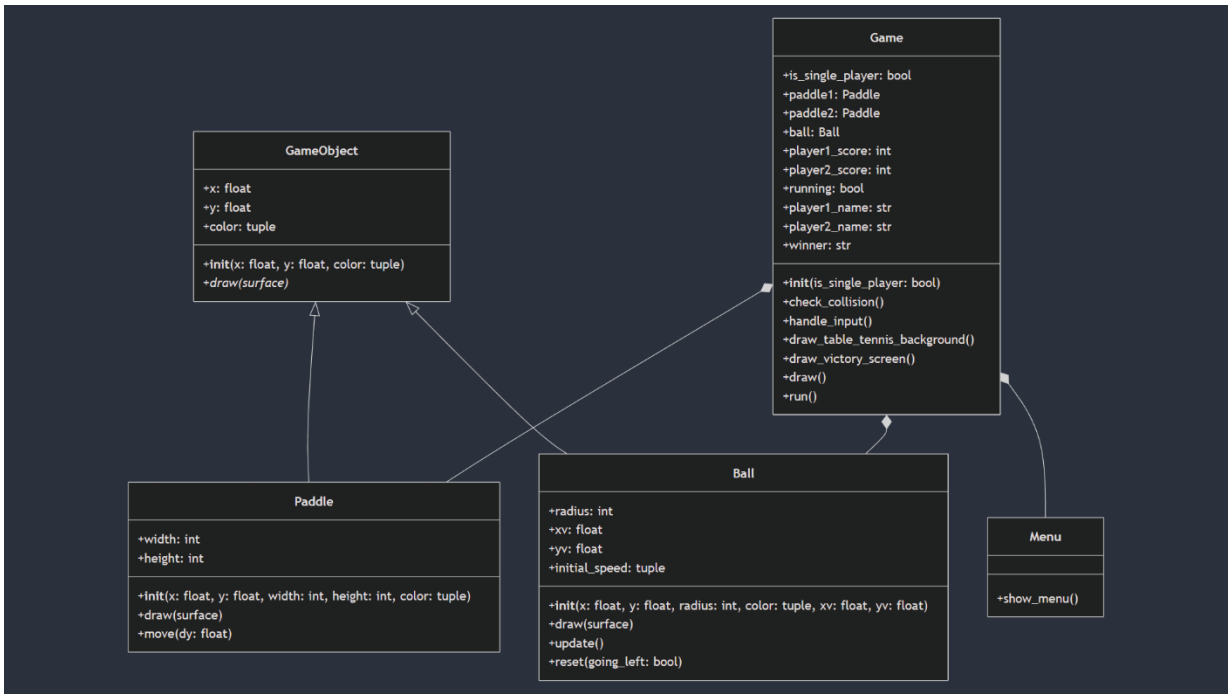
Binti Kholifah, S.Kom., M.Tr.Kom.
I Gde Agung Sri Sidhimantra, S.Kom., M.Kom.
Dimas Novian Aditia Syahputra, S.Tr.T., M.Tr.T.
Moch Deny Pratama, S.Tr.Kom., M.Kom.

Disusun Oleh:

Lusida Cynthia Winayu (23091397075)
Muhammad Dafa Alvin Zuhdi (23091397083)
Bahauddin Rahman Hakim (23091397093)

MATA KULIAH PEMROGRAMAN BERORIENTASI OBJEK
PROGRAM STUDI D4 MANAJEMEN INFORMATIKA
FAKULTAS VOKASI
UNIVERSITAS NEGERI SURABAYA

Class Diagram



1. **Kelas Menu**
   Adalah class yang menangani antarmuka menu awal permainan.

   Metode:
   - Show_menu():
     Metode ini mengembalikan nilai boolean, digunakan untuk menunjukkan apakah menu permainan sedang ditampilkan atau tidak. Menu ini menjadi titik awal permainan.

2. **Kelas Game**
   adalah class utama yang mengatur logika permainan, termasuk pengelolaan pemain, bola, skor, dan kondisi permainan.

   Atribut:
   - Paddle1 dan paddle2:
     Masing-masing adalah objek dari kelas Paddle yang mewakili dua paddle (raket) dalam permainan.
   - Ball
     Objek dari kelas Ball yang mewakili bola dalam permainan.

- Player1_score dan player2_score
  Skor masing-masing pemain.
- is_single_player
  Boolean yang menunjukkan apakah permainan dijalankan dalam mode pemain tunggal (single player) atau dua pemain.
- running
  Boolean yang menunjukkan apakah permainan sedang berjalan atau tidak.
- player1_name
  Nama pemain 1
- player2_name
  Nama pemain 2 atau CPU
- winner
  Nama pemenang dari game pingpong

Metode:

- __init__(is_single_player: bool)
  Konstruktor untuk menginisialisasi permainan, termasuk objek paddle, bola, skor, dan nama pemain.
- check_collision()
  Metode untuk memeriksa apakah terjadi tabrakan, seperti tabrakan bola dengan paddle atau batas permainan.
- handle_input()
  Mengelola input pemain, seperti gerakan paddle menggunakan press keyboard.
- draw()
  Metode untuk menggambar elemen-elemen permainan (seperti paddle dan bola) ke layar.
- draw_table_tennis_background()
  Menggambar latar belakang permainan.
- draw_victory_screen()
  Menampilkan layar kemenangan.
- run()
  Menjalankan loop utama permainan, yang mengatur alur permainan secara keseluruhan.

3. **Kelas GameObject**
   adalah superclass yang mendefinisikan atribut dan perilaku dasar untuk objek dalam game, seperti posisi dan warna.

Atribut:

- x, y
  Koordinat posisi objek di layar.
- Color
  Warna objek yang disimpan dalam bentuk tuple (misalnya, (255, 0, 0) untuk merah).

Metode:

- __init__(x: float, y: float, color: tuple)
  Konstruktor untuk menginisialisasi posisi dan warna objek.
- draw(surface)
  Metode untuk menggambar objek di atas permukaan layar (misalnya menggunakan Pygame surface).

4. **Kelas Paddle**
   adalah subclass yang mewakili pemukul dalam game. Mewarisi atribut dasar dari GameObject dan menambahkan atribut khusus pemukul.

   Atribut:
   - width dan height
     Dimensi paddle (lebar dan tinggi).

   Metode:

   - draw(surface)
     Metode untuk menggambar paddle ke permukaan layar.
   - move(dy)
     Metode untuk menggerakkan paddle pada sumbu y. Parameter dy adalah perubahan posisi paddle pada sumbu y.

5. **Kelas Ball**
   adalah subclass yang mewakili bola dalam game. Mewarisi atribut dasar dari GameObject dan menambahkan atribut serta perilaku khusus bola.

   Atribut:
   - Radius
     Jari-jari bola.
   - xv dan yv
     Kecepatan bola pada sumbu x dan y, yang menggambarkan arah dan kecepatan Gerakan bola.
   - initial_speed: tuple
     Kecepatan awal bola

Metode:

- __init__(x: float, y: float, radius: int, color: tuple, xv: float, yv: float)
  Konstruktor untuk menginisialisasi atribut dasar dan tambahan.
- draw(surface)
  Metode untuk menggambar bola pada permukaan layar.
- update()
  Metode untuk memperbarui posisi bola berdasarkan kecepatan (velocity) bola pada sumbu x dan y.
- reset(going_left: bool)
  Mengatur ulang posisi bola dan arah awalnya.

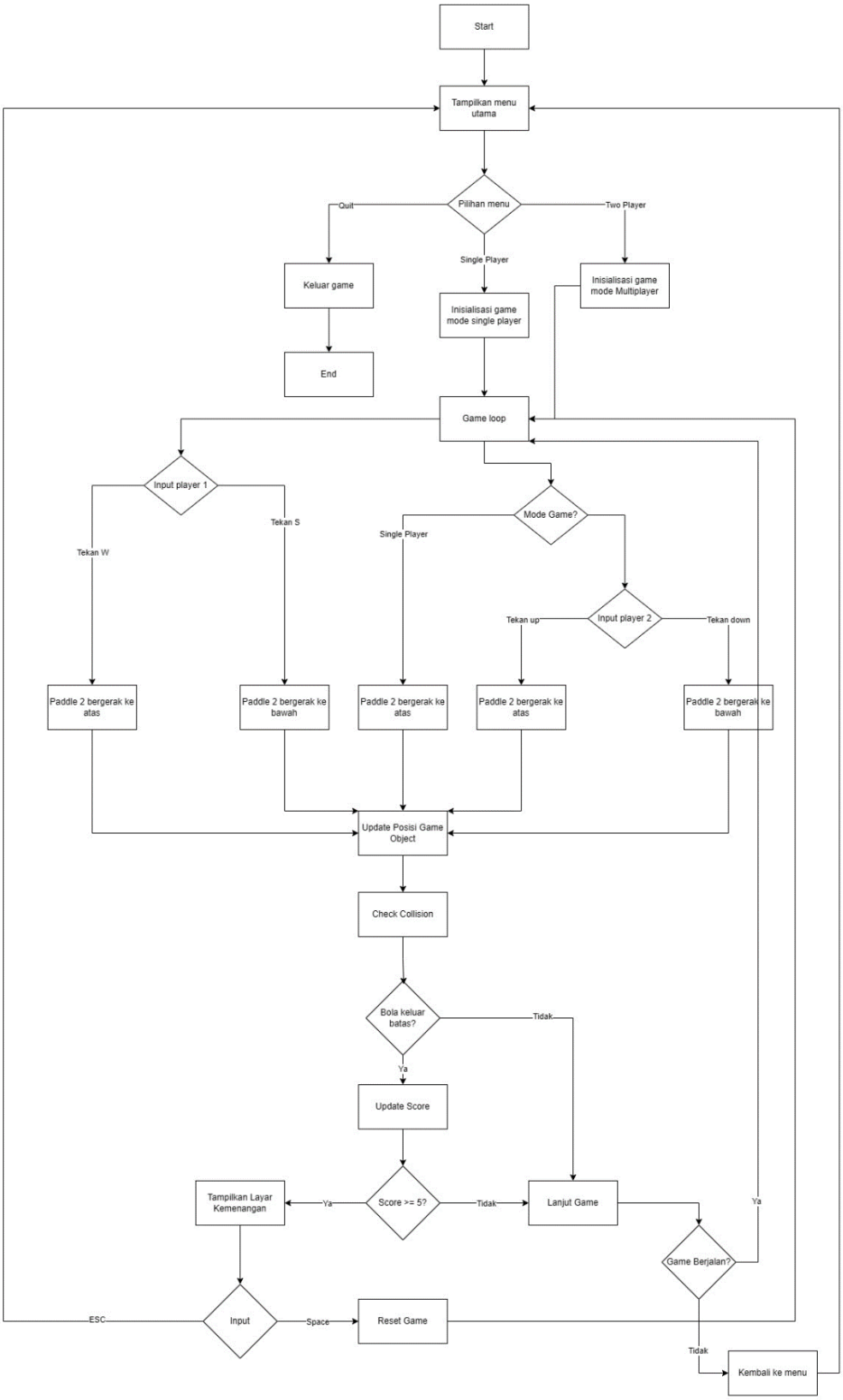
Hubungan Antar Kelas:

1) **GameObject ke Paddle dan Ball**:
   Paddle dan Ball adalah subclass dari GameObject. Keduanya mewarisi atribut dasar (x, y, color) dan metode seperti draw(surface), tetapi memiliki perilaku khusus yang berbeda.

2) **Game ke Paddle dan Ball**:
   - Game menggunakan dua objek Paddle (paddle1, paddle2) dan satu objek Ball untuk menjalankan permainan.
   - Game mengatur posisi, gerakan, serta interaksi antara bola dan pemukul.

3) **Game ke Menu**:
   Game dimulai melalui menu awal yang ditangani oleh Menu.

Rancang Alur Aplikasi Pingpong Game



Penerapan Prinsip Object Oriented Programming

1. **Abstraction**

```python
class GameObject:
    def __init__(self, x, y, color):
        self.x = x
        self.y = y
        self.color = color

    def draw(self, surface):
        raise NotImplementedError
```

GameObject adalah abstract class yang mendefinisikan properti dasar (x, y, color) dan method abstract draw() yang harus diimplementasikan oleh child class. Kode ini merupakan abstraction karena menyembunyikan implementasi detail dan hanya menampilkan fungsionalitas yang diperlukan.

2. **Inheritance**

```python
class Paddle(GameObject):
    def __init__(self, x, y, width, height, color):
        super().__init__(x, y, color)
        self.width = width
        self.height = height

class Ball(GameObject):
    def __init__(self, x, y, radius, color, xv, yv):
        super().__init__(x, y, color)
        self.radius = radius
        self.xv = xv
        self.yv = yv
        self.initial_speed = (xv, yv)
```

Paddle dan Ball mewarisi properti dan method dari GameObject. Dengan menggunakan inheritance, keduanya dapat menggunakan atribut atau metode yang didefinisikan di GameObject, seperti posisi (x, y) dan warna (color), tanpa perlu mendefinisikannya ulang. Hal ini dilakukan melalui pemanggilan super().__init__(...) di konstruktor __init__

masing-masing subclass, yang memanggil konstruktor dari superclass untuk menginisialisasi atribut dasar.

3. **Polymorphism**

```
class Paddle(GameObject):
    def draw(self, surface):
        pygame.draw.rect(surface, self.color, (self.x, self.y, self.width, self.height))

class Ball(GameObject):
    def draw(self, surface):
        pygame.draw.circle(surface, self.color, (int(self.x), int(self.y)), self.radius)
```

Method draw() diimplementasikan secara berbeda di setiap child class sesuai kebutuhan. Paddle menggambar rectangle sedangkan Ball menggambar circle.

4. **Encapsulation**

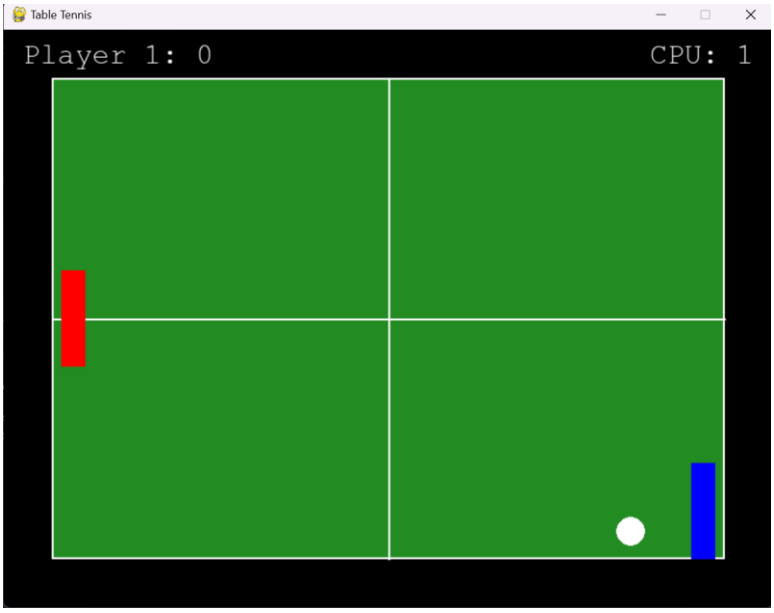
```
class Game:
    def __init__(self, is_single_player=True):
        self.is_single_player = is_single_player
        self.paddle1 = Paddle(COURT_MARGIN + 10, SCREEN_HEIGHT // 2 - 50, 25, 100, COLORS["red"])
        self.paddle2 = Paddle(SCREEN_WIDTH - COURT_MARGIN - 35, SCREEN_HEIGHT // 2 - 50, 25, 100, COLORS["blue"])
        self.ball = Ball(SCREEN_WIDTH // 2, SCREEN_HEIGHT // 2, 15, COLORS["white"], 7, 7)
        self.player1_score = 0
        self.player2_score = 0
        self.running = True
        self.player1_name = "Player 1"
        self.player2_name = "Player 2" if not is_single_player else "CPU"
        self.winner = None
```

Pada kode di atas, class Game menggunakan enkapsulasi dengan mendefinisikan atribut-atribut seperti is_single_player, paddle1, paddle2, ball, player1_score, player2_score, running, player1_name, player2_name, dan winner sebagai bagian dari objek game. Atribut ini diinisialisasi dalam konstruktor __init__ dan hanya dapat diakses atau dimodifikasi melalui objek Game. Dengan cara ini, pengelolaan status permainan terpusat di satu tempat, sehingga lebih mudah untuk memantau dan mengendalikan alur permainan.

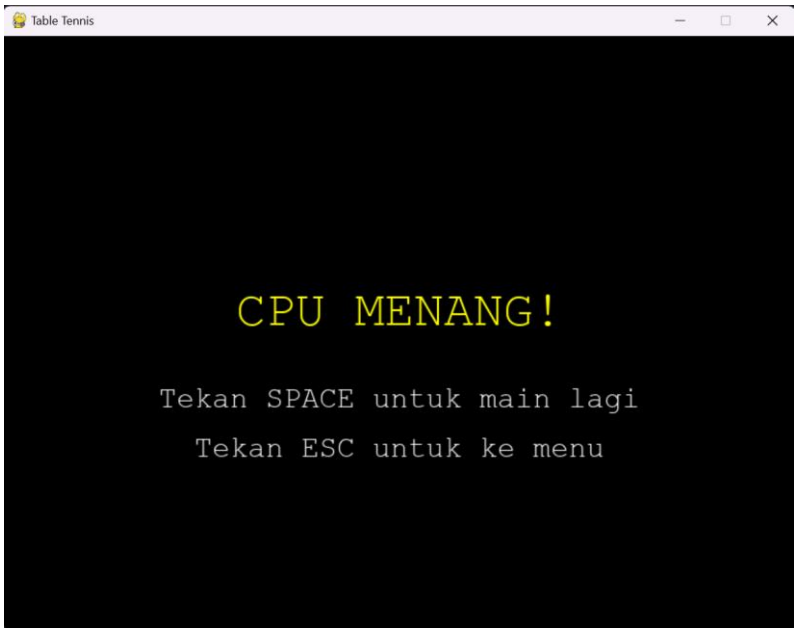
Output Tampilan Game:

Pada menu awal, player diberikan 2 pilihan yaitu single player atau two player, saat pemain memilih single player maka pemain hanya bisa menggerakkan satu paddle dengan menggunakan keyboard W dan S, lalu paddle lain digerakkan oleh computer/cpu. Sementara itu jika player memilih pilihan multiplayer bisa menggerakkan kedua paddle secara bersamaan dengan paddle kiri menggunakan keyboard W dan S untuk ke atas dan kebawah, sementara pada paddle kanan dengan menggunakan keyboard tombol panah atas, dan tombol panah bawah.

Berikut ini adalah tampilan game yang akan dilihat oleh pemain, pemain harus bisa memantulkann bola dengan menggunakan paddle, jika pemain gagall untuk memantulkan bola sebanyak 5 kali maka CPU menang, sebaliknya jika pemain berhasil sebanyak 5 kali maka player atau pemain menang



Berikut adalah tampilan saat CPU menang, pemain bisa bermain lagi dengan menekan tombol “Space” di keyboard. Pemain juga bisa Kembali ke menu awal dengan menekan tombol “Esc” jika ingin mengganti opsi multiplayer atau single player



Penjelasan Code

Source Code Lengkap: [Github Pingpong Game]

1. **Ball.py**

```
import pygame
```

Baris tersebut mengimpor pustaka pygame, yang merupakan modul untuk membuat game 2D dalam Python. Modul ini memberikan berbagai fungsi untuk memanipulasi grafik, suara, dan input pengguna, yang digunakan dalam pengembangan game.

```
from config import COURT_MARGIN, SCREEN_HEIGHT, SCREEN_WIDTH
```

Baris terserbut mengimpor tiga konstanta dari file config, yaitu COURT_MARGIN, SCREEN_HEIGHT, dan SCREEN_WIDTH. Ketiga konstanta ini digunakan untuk menentukan margin lapangan dan dimensi layar permainan, sehingga memudahkan pengelolaan ukuran layar dan batas-batasnya.

```
from game_objects import GameObject
```

Baris ini mengimpor kelas GameObject dari file game_objects. GameObject adalah kelas dasar yang digunakan untuk mewariskan properti dan metode umum ke kelas lainnya, seperti Ball. Hal ini memungkinkan penggunaan atribut dan logika yang dapat digunakan ulang.

```
class Ball(GameObject):
    def __init__(self, x, y, radius, color, xv, yv):
        super().__init__(x, y, color)
        self.radius = radius
        self.xv = xv
        self.yv = yv
        self.initial_speed = (xv, yv)
```

Baris ini mendeklarasikan kelas Ball, yang merupakan turunan dari GameObject. Kelas ini mewakili bola dalam permainan dengan fitur tambahan seperti menggambar di layar,

memperbarui posisi, dan mengatur ulang posisinya. Metode __init__ adalah konstruktor kelas Ball. Konstruktor ini dipanggil saat objek Ball dibuat. Parameter x dan y menentukan posisi awal bola, radius menentukan ukuran bola, color menentukan warna bola, sedangkan xv dan yv adalah kecepatan horizontal dan vertikal bola.
Di dalamnya, super().__init__(x, y, color) digunakan untuk memanggil konstruktor kelas induk (GameObject) untuk menginisialisasi atribut dasar. Selain itu, atribut baru seperti radius, xv, yv, dan initial_speed (kecepatan awal bola) diatur untuk digunakan dalam logika permainan.

```
  def draw(self, surface):
          pygame.draw.circle(surface, self.color, (int(self.x), int(self.y)), self.radius)
```

Metode draw bertanggung jawab untuk menggambar bola di layar permainan. Fungsi ini menggunakan pygame.draw.circle untuk menggambar lingkaran di posisi (x, y) dengan warna self.color dan ukuran self.radius. Parameter surface adalah objek layar tempat bola akan digambar, memastikan visualisasi bola selalu diperbarui sesuai dengan posisinya.

```
def update(self):
        self.x += self.xv
        self.y += self.yv

        if self.y - self.radius <= COURT_MARGIN:
            self.y = COURT_MARGIN + self.radius
            self.yv *= -1
        elif self.y + self.radius >= SCREEN_HEIGHT - COURT_MARGIN:
            self.y = SCREEN_HEIGHT - COURT_MARGIN - self.radius
            self.yv *= -1
```

Metode update memperbarui posisi bola berdasarkan kecepatan horizontal (xv) dan vertikal (yv). Logika berikutnya memeriksa apakah bola menyentuh batas atas atau bawah lapangan menggunakan COURT_MARGIN. Jika menyentuh, bola memantul dengan membalik arah kecepatan vertikalnya (yv *= -1). Posisi bola juga disesuaikan agar tidak keluar dari layer

```
def reset(self, going_left=False):
        self.x = SCREEN_WIDTH // 2
        self.y = SCREEN_HEIGHT // 2
        self.xv = -self.initial_speed[0] if going_left else self.initial_speed[0]
        self.yv = self.initial_speed[1]
```

Metode reset mengatur ulang posisi dan kecepatan bola ke nilai awal. Posisi bola diatur ke tengah layar menggunakan dimensi layar (SCREEN_WIDTH dan SCREEN_HEIGHT). Parameter going_left menentukan arah horizontal bola saat diatur ulang, dengan nilai kecepatan horizontal xv menjadi negatif jika going_left bernilai True. Kecepatan vertikal yv selalu diatur ke nilai awal.

2. **Config.py**

```
import pygame
```

Baris ini mengimpor modul pygame, sebuah library Python yang digunakan untuk membuat game. Library ini menyediakan berbagai fitur seperti manipulasi grafis, suara, dan input pengguna.

```
pygame.init()
```

Baris ini menginisialisasi semua modul yang tersedia di pygame, termasuk modul untuk grafis, suara, dan input. Ini merupakan langkah pertama yang diperlukan sebelum menggunakan fitur pygame.

```
COLORS = {
    "red": (255, 0, 0),
    "blue": (0, 0, 255),
    "yellow": (255, 255, 0),
    "orange": (255, 127, 0),
    "white": (255, 255, 255),
    "black": (0, 0, 0),
    "green": (34, 139, 34)
}
```

Kode ini mendefinisikan sebuah dictionary bernama COLORS, yang berisi nama-nama warna sebagai kunci dan nilai RGB-nya sebagai nilai. Warna-warna ini nantinya dapat digunakan untuk menggambar elemen permainan seperti bola, latar belakang, atau teks.

```
SCREEN_WIDTH = 800
SCREEN_HEIGHT = 600
COURT_MARGIN = 50
COURT_WIDTH = SCREEN_WIDTH - (2 * COURT_MARGIN)
COURT_HEIGHT = SCREEN_HEIGHT - (2 * COURT_MARGIN)
WINNING_SCORE = 5
```

Bagian ini mendefinisikan berbagai variabel konfigurasi yang berkaitan dengan dimensi layar dan lapangan permainan. SCREEN_WIDTH dan SCREEN_HEIGHT menentukan ukuran jendela game. COURT_MARGIN menentukan margin di sekitar lapangan, sedangkan COURT_WIDTH dan COURT_HEIGHT menghitung dimensi lapangan berdasarkan ukuran layar dan margin. WINNING_SCORE adalah skor yang diperlukan untuk memenangkan permainan.

```
game_screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))
pygame.display.set_caption("Table Tennis")
font = pygame.font.SysFont("monospace", 50)
small_font = pygame.font.SysFont("monospace", 30)
```

game_screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))
Baris ini membuat jendela permainan menggunakan dimensi yang telah didefinisikan (SCREEN_WIDTH dan SCREEN_HEIGHT). Objek yang dihasilkan, game_screen, digunakan untuk menggambar semua elemen permainan di layar.

pygame.display.set_caption("Table Tennis")
Kode ini mengatur judul jendela permainan menjadi "Table Tennis". Judul ini akan muncul di bagian atas jendela game.

font = pygame.font.SysFont("monospace", 50)
small_font = pygame.font.SysFont("monospace", 30)
Bagian ini mendefinisikan dua font yang digunakan untuk menampilkan teks di permainan. Font pertama (font) memiliki ukuran 50 piksel dan biasanya digunakan untuk teks besar seperti skor atau pengumuman pemenang. Font kedua (small_font) memiliki ukuran 30 piksel dan dapat digunakan untuk teks yang lebih kecil seperti petunjuk atau informasi tambahan.

3. Game.py

```
class Game:
    def __init__(self, is_single_player=True):
        self.is_single_player = is_single_player
        self.paddle1 = Paddle(COURT_MARGIN + 10, SCREEN_HEIGHT // 2 - 50, 25, 100, COLORS["red"])
        self.paddle2 = Paddle(SCREEN_WIDTH - COURT_MARGIN - 35, SCREEN_HEIGHT // 2 - 50, 25, 100, COLORS["blue"])
        self.ball = Ball(SCREEN_WIDTH // 2, SCREEN_HEIGHT // 2, 15, COLORS["white"], 7, 7)
        self.player1_score = 0
        self.player2_score = 0
        self.running = True
        self.player1_name = "Player 1"
        self.player2_name = "Player 2" if not is_single_player else "CPU"
        self.winner = None
```

Konstruktor __init__ menginisialisasi semua elemen utama permainan, termasuk paddle (bet), bola, skor pemain, dan status permainan. Parameter is_single_player menentukan apakah permainan melibatkan satu pemain melawan CPU atau dua pemain. Properti seperti paddle1 dan paddle2 adalah objek paddle, sedangkan ball adalah objek bola. Variabel player1_score dan player2_score menyimpan skor masing-masing pemain. Properti running mengontrol status utama loop permainan, dan winner menyimpan nama pemain yang memenangkan permainan.

```
  def check_collision(self):
        if self.ball.x - self.ball.radius <= self.paddle1.x + self.paddle1.width and \
           self.paddle1.y <= self.ball.y <= self.paddle1.y + self.paddle1.height:
            self.ball.x = self.paddle1.x + self.paddle1.width + self.ball.radius
            self.ball.xv *= -1.1

        if self.ball.x + self.ball.radius >= self.paddle2.x and \
           self.paddle2.y <= self.ball.y <= self.paddle2.y + self.paddle2.height:
            self.ball.x = self.paddle2.x - self.ball.radius
            self.ball.xv *= -1.1

        if self.ball.x <= COURT_MARGIN:
            self.player2_score += 1
            self.ball.reset(going_left=False)
        elif self.ball.x >= SCREEN_WIDTH - COURT_MARGIN:
            self.player1_score += 1
            self.ball.reset(going_left=True)

        if self.player1_score >= WINNING_SCORE:
            self.winner = self.player1_name
        elif self.player2_score >= WINNING_SCORE:
            self.winner = self.player2_name
```

Metode ini memeriksa dan menangani semua kondisi tabrakan dalam permainan. Deteksi tabrakan dilakukan antara bola dengan paddle atau dengan tepi lapangan. Jika bola keluar dari batas kiri atau kanan lapangan, skor pemain yang bersangkutan diperbarui, dan bola di-reset ke posisi tengah. Skor yang mencapai nilai tertentu akan mengakhiri permainan dengan menentukan pemenang.

```
def handle_input(self):
        keys = pygame.key.get_pressed()
        if keys[pygame.K_w]:
            self.paddle1.move(-8)
        if keys[pygame.K_s]:
            self.paddle1.move(8)

        if self.is_single_player:
            if self.ball.xv > 0:
                if self.ball.y < self.paddle2.y + self.paddle2.height / 2:
                    self.paddle2.move(-7)
                elif self.ball.y > self.paddle2.y + self.paddle2.height / 2:
                    self.paddle2.move(7)
        else:
            if keys[pygame.K_UP]:
                self.paddle2.move(-8)
            if keys[pygame.K_DOWN]:
                self.paddle2.move(8)
```

Metode ini menangani masukan pemain untuk menggerakkan paddle. Pemain 1 menggunakan tombol W dan S, sedangkan untuk mode dua pemain, pemain 2 menggunakan tombol panah atas dan bawah. Dalam mode satu pemain, paddle pemain 2 digerakkan secara otomatis berdasarkan posisi bola.

```
  def draw_table_tennis_background(self):
        # Gambar background hitam
        game_screen.fill(COLORS["black"])

        # Gambar lapangan hijau
        pygame.draw.rect(game_screen, COLORS["green"],
                         (COURT_MARGIN, COURT_MARGIN,
                          COURT_WIDTH, COURT_HEIGHT))
```

```
        # Gambar garis tepi lapangan
        pygame.draw.rect(game_screen, COLORS["white"],
                         (COURT_MARGIN, COURT_MARGIN,
                          COURT_WIDTH, COURT_HEIGHT), 2)

        # Garis tengah vertikal (net)
        pygame.draw.line(game_screen, COLORS["white"],
                         (SCREEN_WIDTH // 2, COURT_MARGIN),
                         (SCREEN_WIDTH // 2, SCREEN_HEIGHT - COURT_MARGIN),
                         2)

        # Garis tengah horizontal
        pygame.draw.line(game_screen, COLORS["white"],
                         (COURT_MARGIN, SCREEN_HEIGHT // 2),
                         (SCREEN_WIDTH - COURT_MARGIN, SCREEN_HEIGHT // 2),
                         2)
```

Metode ini menggambar latar belakang permainan berupa lapangan tenis meja, termasuk warna hijau untuk lapangan, garis putih di tepi lapangan, garis tengah horizontal, dan net vertikal di tengah.

```
def draw_victory_screen(self):
        game_screen.fill(COLORS["black"])
        victory_text = font.render(f"{self.winner} MENANG!", True, COLORS["yellow"])
        play_again_text = small_font.render("Tekan SPACE untuk main lagi", True, COLORS["white"])
        menu_text = small_font.render("Tekan ESC untuk ke menu", True, COLORS["white"])

        game_screen.blit(victory_text,
            (SCREEN_WIDTH // 2 - victory_text.get_width() // 2, SCREEN_HEIGHT // 2 - 50))
        game_screen.blit(play_again_text,
            (SCREEN_WIDTH // 2 - play_again_text.get_width() // 2, SCREEN_HEIGHT // 2 + 50))
        game_screen.blit(menu_text,
            (SCREEN_WIDTH // 2 - menu_text.get_width() // 2, SCREEN_HEIGHT // 2 + 100))
```

Metode ini menggambar layar kemenangan ketika salah satu pemain mencapai skor yang cukup untuk menang. Teks kemenangan, opsi untuk bermain kembali, dan kembali ke menu ditampilkan di layar.

```
def draw(self):
        if self.winner:
            self.draw_victory_screen()
        else:
            self.draw_table_tennis_background()
            self.paddle1.draw(game_screen)
            self.paddle2.draw(game_screen)
            self.ball.draw(game_screen)

            p1_text = small_font.render(f"{self.player1_name}: {self.player1_score}", True, COLORS["white"])
            p2_text = small_font.render(f"{self.player2_name}: {self.player2_score}", True, COLORS["white"])

            game_screen.blit(p1_text, (20, 10))
            game_screen.blit(p2_text, (SCREEN_WIDTH - p2_text.get_width() - 20, 10))
```

Metode ini menangani semua proses rendering dalam permainan. Jika ada pemenang, layar kemenangan akan ditampilkan. Jika tidak, metode ini menggambar lapangan, paddle, bola, dan skor pemain saat ini.

```
def run(self):
        clock = pygame.time.Clock()
        while self.running:
            for event in pygame.event.get():
                if event.type == pygame.QUIT:
                    self.running = False
                if event.type == pygame.KEYDOWN:
                    if self.winner:
                        if event.key == pygame.K_SPACE:
                            self.player1_score = 0
                            self.player2_score = 0
                            self.winner = None
                            self.ball.reset()
                        elif event.key == pygame.K_ESCAPE:
                            return "menu"

            if not self.winner:
                self.handle_input()
                self.ball.update()
                self.check_collision()

            self.draw()
            pygame.display.flip()
            clock.tick(60)
        return "quit"
```

Metode ini adalah inti dari loop permainan. Selama permainan berjalan, metode ini menangani input pemain, pembaruan posisi bola, deteksi tabrakan, dan rendering semua elemen. Setelah pemain menang, metode ini memberikan opsi untuk bermain lagi atau kembali ke menu.

4. **Game Object.py**

```
class GameObject:
    def __init__(self, x, y, color):
        self.x = x
        self.y = y
        self.color = color

    def draw(self, surface):
        raise NotImplementedError
```

Kelas GameObject adalah kelas dasar untuk semua objek dalam permainan. Konstruktor __init__ menerima parameter posisi (x, y) dan warna color, yang digunakan untuk mengatur atribut dasar dari objek. Metode draw didefinisikan sebagai metode abstrak (menggunakan raise NotImplementedError), yang berarti kelas turunan harus mengimplementasikan logika untuk menggambar objek pada layar.

```
class Paddle(GameObject):
    def __init__(self, x, y, width, height, color):
        super().__init__(x, y, color)
        self.width = width
        self.height = height

    def draw(self, surface):
        pygame.draw.rect(surface, self.color, (self.x, self.y, self.width, self.height))

    def move(self, dy):
        self.y += dy
        self.y = max(COURT_MARGIN, min(self.y, SCREEN_HEIGHT - COURT_MARGIN - self.height))
```

Kelas Paddle adalah turunan dari GameObject yang merepresentasikan paddle (bet) dalam permainan. Konstruktor __init__ memperluas atribut kelas induk dengan menambahkan atribut width dan height, yang menentukan ukuran paddle.

Metode draw dalam kelas Paddle mengimplementasikan logika menggambar paddle pada layar permainan. Paddle digambar sebagai sebuah persegi panjang menggunakan fungsi pygame.draw.rect dengan posisi (x, y), lebar width, tinggi height, dan warna color.

Metode move mengatur pergerakan paddle secara vertikal berdasarkan nilai dy. Paddle hanya dapat bergerak dalam batas lapangan yang ditentukan oleh COURT_MARGIN di bagian atas dan bawah layar. Perhitungan menggunakan fungsi max dan min memastikan paddle tidak keluar dari batas.

5. **Main.py**

```
def main():
    while True:
        is_single_player = show_menu()
        if is_single_player is None:
            break
```

Fungsi main adalah titik masuk utama dari aplikasi. Bagian awal ini menampilkan menu utama menggunakan fungsi show_menu. Nilai yang dikembalikan oleh show_menu menentukan mode permainan: apakah permainan akan dimainkan dalam mode satu pemain atau dua pemain. Jika show_menu mengembalikan None, aplikasi keluar dari loop utama dan melanjutkan ke proses penghentian.

```
            result = Game(is_single_player).run()
                    if result == "quit":
                        break
```

Setelah menentukan mode permainan, fungsi ini menciptakan instance dari kelas Game dengan parameter is_single_player yang telah ditentukan. Metode run dari objek Game dijalankan, yang memulai loop permainan utama. Jika metode run mengembalikan string "quit", fungsi main keluar dari loop, yang menandakan bahwa pengguna telah memilih untuk keluar dari permainan.

```
    pygame.quit()
        sys.exit()
```

Setelah keluar dari loop utama, fungsi ini memastikan bahwa semua modul pygame dihentikan dengan memanggil pygame.quit(). Kemudian, sys.exit() dipanggil untuk benar-benar

menghentikan program. Kedua langkah ini diperlukan untuk memastikan penghentian aplikasi dilakukan secara bersih dan menghindari kesalahan atau proses yang berjalan di latar belakang.

```
if __name__ == "__main__":
    main()
```

Blok ini memastikan bahwa fungsi main hanya dijalankan jika file ini dieksekusi langsung, bukan diimpor sebagai modul oleh file lain. Ini adalah konvensi Python untuk mengontrol bagaimana sebuah skrip dijalankan dan menjaga modularitas kode.

6. **Menu.py**

```
def show_menu():
    running = True
```

Fungsi show_menu bertanggung jawab untuk menampilkan menu utama permainan. Variabel running digunakan sebagai kontrol untuk menjaga loop menu tetap berjalan sampai pengguna memilih salah satu opsi atau keluar.

```
while running:
        game_screen.fill(COLORS["black"])

        title_text = font.render("Table Tennis", True, COLORS["white"])
        single_player_text = small_font.render("Single Player", True, COLORS["white"])
        two_player_text = small_font.render("Two Player", True, COLORS["white"])
        quit_text = small_font.render("Quit", True, COLORS["white"])
```

Loop utama dimulai dengan membersihkan layar dan mengisi latar belakang dengan warna hitam (COLORS["black"]). Teks untuk judul menu dan pilihan menu ("Single Player", "Two Player", dan "Quit") dirender menggunakan font yang telah didefinisikan, dengan warna putih (COLORS["white"]).

```
title_x = SCREEN_WIDTH // 2 - title_text.get_width() // 2
        single_player_x = SCREEN_WIDTH // 2 - single_player_text.get_width() // 2
        two_player_x = SCREEN_WIDTH // 2 - two_player_text.get_width() // 2
        quit_x = SCREEN_WIDTH // 2 - quit_text.get_width() // 2

        title_y = 100
        single_player_y = 200
        two_player_y = 250
        quit_y = 300
```

Bagian ini menghitung posisi horizontal dan vertikal untuk setiap teks. Posisi horizontal (x) dihitung agar teks berada di tengah layar berdasarkan lebar layar (SCREEN_WIDTH) dan lebar teks itu sendiri. Posisi vertikal (y) ditentukan secara manual dengan nilai-nilai tetap, memastikan teks dirender dengan jarak yang sesuai satu sama lain.

```
single_player_rect = pygame.Rect(single_player_x, single_player_y,
                                 single_player_text.get_width(), single_player_text.get_height())
        two_player_rect = pygame.Rect(two_player_x, two_player_y,
                                 two_player_text.get_width(), two_player_text.get_height())
        quit_rect = pygame.Rect(quit_x, quit_y,
                            quit_text.get_width(), quit_text.get_height())
```

Untuk setiap opsi menu, sebuah objek pygame.Rect dibuat berdasarkan posisi dan ukuran teks. Objek-objek ini digunakan untuk mendeteksi interaksi mouse ketika pengguna mengklik area yang relevan.

```
    game_screen.blit(title_text, (title_x, title_y))
    game_screen.blit(single_player_text, (single_player_x, single_player_y))
    game_screen.blit(two_player_text, (two_player_x, two_player_y))
    game_screen.blit(quit_text, (quit_x, quit_y))
```

Teks yang telah dirender ditempatkan pada layar menggunakan fungsi blit. Setiap teks ditempatkan sesuai dengan koordinat yang telah dihitung sebelumnya.

```
pygame.display.flip()
```

Layar diperbarui dengan memanggil pygame.display.flip(), yang memastikan semua perubahan (teks, latar belakang, dan elemen lainnya) ditampilkan kepada pengguna.

```
for event in pygame.event.get():
            if event.type == pygame.QUIT:
                return None
```

Bagian ini menangani event yang terjadi selama loop menu. Jika event QUIT terdeteksi (seperti menutup jendela permainan), fungsi mengembalikan None untuk keluar dari menu dan menghentikan aplikasi.

```
if event.type == pygame.MOUSEBUTTONDOWN:
                mouse_pos = pygame.mouse.get_pos()
                if single_player_rect.collidepoint(mouse_pos):
                    return True
                if two_player_rect.collidepoint(mouse_pos):
                    return False
                if quit_rect.collidepoint(mouse_pos):
                    return None
```

Jika event MOUSEBUTTONDOWN terdeteksi (klik mouse), posisi mouse diperiksa terhadap area yang ditentukan oleh Rect untuk setiap opsi menu. Jika posisi mouse berada di dalam area tombol: True dikembalikan untuk memilih mode Single Player, False dikembalikan untuk memilih mode Two Playe, None dikembalikan untuk keluar dari aplikasi.

Tantangan Pengembangan

Ketika membuat game tenis meja 2D menggunakan Pygame, terdapat berbagai tantangan yang harus dihadapi. Salah satu tantangan utama adalah merancang struktur program menggunakan teknik Object-Oriented Programming (OOP), di mana pemisahan logika ke dalam kelas-kelas seperti Ball, Paddle, dan Game membutuhkan desain yang baik agar kode tetap modular dan mudah dikelola. Selain itu, implementasi logika gerakan bola dan paddle yang realistis, termasuk menjaga kecepatan bola tetap konstan dan memastikan pantulan sesuai sudut, menjadi tantangan penting. Deteksi tabrakan antara bola dengan paddle, dinding, atau batas area permainan juga harus dilakukan dengan akurat agar gameplay terasa alami.

Di sisi lain, memastikan responsivitas kontrol paddle sesuai input pemain adalah hal yang krusial untuk memberikan pengalaman bermain yang nyaman. Kendala performa juga sering muncul, terutama jika logika dalam game loop tidak dioptimalkan, yang dapat menyebabkan game berjalan lambat pada perangkat dengan spesifikasi rendah. Penambahan fitur tambahan, seperti AI untuk lawan atau mode multiplayer, memerlukan logika yang lebih kompleks, baik dalam mengelola kontrol maupun komunikasi antar pemain. Terakhir, pengelolaan audio dan visual menjadi tantangan tersendiri, di mana efek suara dan animasi harus disinkronkan dengan aksi permainan untuk menciptakan pengalaman bermain yang imersif.