Overview
Object-Oriented Programming serves as a fundamental framework for structuring, organizing, and developing games with Pygame. By leveraging OOP principles such as encapsulation, inheritance, and polymorphism, developers can create modular, reusable, and maintainable game codebases that are well-suited for building immersive and engaging gaming experiences.

In this practical session, you will delve into the exciting world of game development with Pygame, a popular library for creating games in Python. You will be creating a simple Ping Pong game using Pygame.

Problem Statement

Before diving into game development, ensure that Pygame is installed on your computer by following the steps given below under **simple pygame**.

In the first hour of the practical class, your task is to create a basic version of the Ping Pong game. Follow these guidelines to get started:

1. Install:
    - Install Pygame using '**`pip install pygame`**'.
    - Create a new Python file named *pingpong.py* to write your game code.

2. Importing and Initializing Pygame:
    - import the `pygame` and `random` modules.
    - Define colors (`BLACK` and `WHITE`) and game constants (`WIDTH`, `HEIGHT`, `BALL_RADIUS`, `PADDLE_WIDTH`, and `PADDLE_HEIGHT`) for the game window size, ball radius, and paddle dimensions.

```
import pygame
import random

# Define colors
BLACK = (0, 0, 0)
WHITE = (255, 255, 255)

# Define game constants
WIDTH, HEIGHT = 800, 600
BALL_RADIUS = 10
PADDLE_WIDTH = 100
PADDLE_HEIGHT = 20
```

3. The `Ball` class represents the ball object in the game. It has an `__init__` method to initialize the ball's position and speed, and a `move` method to update the ball's position based on its speed. The `check_collision` method checks if the ball has collided with the walls and reverses its speed accordingly.

```
class Ball:
    def __init__(self):
        self.x = WIDTH // 2
        self.y = HEIGHT // 2
        self.speed_x = random.choice([-3, 3])
        self.speed_y = 2

    def move(self):
        self.x += self.speed_x
        self.y += self.speed_y

    def check_collision(self):
        if self.x <= BALL_RADIUS or self.x >= WIDTH -
BALL_RADIUS:
            self.speed_x = -self.speed_x
        if self.y <= BALL_RADIUS:
            self.speed_y = -self.speed_y
```

4. The `Paddle` class represents the paddle object in the game. It has an `__init__` method to initialize the paddle's dimensions and position, and a `move` method to move the paddle left or right based on the provided direction.

```
class Paddle:
    def __init__(self):
        self.width = PADDLE_WIDTH
        self.height = PADDLE_HEIGHT
        self.x = (WIDTH - self.width) // 2
        self.y = HEIGHT - self.height - 10

    def move(self, direction):
        if direction == "left" and self.x > 0:
```

```
            self.x -= 10
        elif direction == "right" and self.x < WIDTH -
self.width:
            self.x += 10
```

5. The `Game` class is the main class that manages the game logic. It has an `__init__` method to initialize the ball, paddle, score, font, and game_over flag. The `handle_events` method handles events like quitting the game. The `update` method updates the ball's position, checks for collisions with the paddle, and updates the score. The `draw` method draws the ball, paddle, score, and game over message on the screen.

```
class Game:
    def __init__(self):
        self.ball = Ball()
        self.paddle = Paddle()
        self.score = 0
        self.font = pygame.font.Font(None, 36)
        self.game_over = False

    def handle_events(self):
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                self.game_over = True

    def update(self):
        self.ball.move()
        self.ball.check_collision()

        if self.ball.y >= HEIGHT - BALL_RADIUS - PADDLE_HEIGHT:
            if self.paddle.x <= self.ball.x <= self.paddle.x +
PADDLE_WIDTH:
                self.ball.y = HEIGHT - BALL_RADIUS -
PADDLE_HEIGHT - 1
                self.ball.speed_y = -self.ball.speed_y
                self.score += 1
            else:
```

```
                self.game_over = True

    def draw(self, screen):
        screen.fill(BLACK)
        pygame.draw.circle(screen, WHITE, (self.ball.x,
self.ball.y), BALL_RADIUS)
        pygame.draw.rect(screen, WHITE, (self.paddle.x,
self.paddle.y, self.paddle.width, self.paddle.height))
        score_text = self.font.render(f"Score: {self.score}",
True, WHITE)
        screen.blit(score_text, (10, 10))
        if self.game_over:
            game_over_text = self.font.render(f"Your score:
{self.score}", True, WHITE)
            game_over_rect = game_over_text.get_rect()
            game_over_rect.center = (WIDTH // 2, HEIGHT // 2)
            screen.blit(game_over_text, game_over_rect)
```

6. The `main` function is the entry point of the program. It initializes Pygame, creates a game window, and instantiates the `Game` class. The main game loop continuously handles events, moves the paddle based on key presses, updates the game state, and draws the game objects on the screen. The loop runs at a fixed frame rate of 60 frames per second using `clock.tick(60)`.

```
def main():
    pygame.init()
    screen = pygame.display.set_mode((WIDTH, HEIGHT))
    pygame.display.set_caption("Ping Pong")
    clock = pygame.time.Clock()
    game = Game()

    while not game.game_over:
        game.handle_events()

        keys = pygame.key.get_pressed()
        if keys[pygame.K_LEFT]:
```

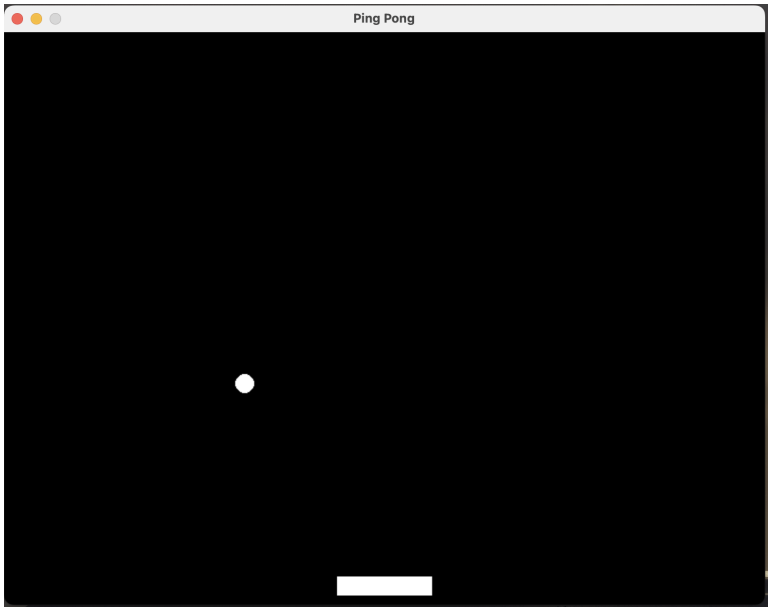
```
            game.paddle.move("left")
        if keys[pygame.K_RIGHT]:
            game.paddle.move("right")

        game.update()
        game.draw(screen)

        pygame.display.flip()
        clock.tick(60)

    pygame.quit()
main()
```

Output:



In the second hour of the practical class, experiment with adding features such as countdown timers and score tracking to make the game more engaging. Here are some ideas to consider:

1. *Countdown Timer:* Implement a countdown timer at the beginning of each round to add a sense of urgency to the game.
2. *Score Tracking:* Track the player's score as they hit the ball with the paddle and display it on the screen.

You can add other features of your choice as well.
Sample Outputs:

Start of the Game:

End of Game: