|  |  |  |
| --- | --- | --- |
| **Game Dev** | *GUI Game Programming* | Exercise 1 |

**PROJECT Console Programming with Python - Tic Tac Toe Game**

**Objective** To use Python to demonstrate a game - based Console Application.

***PROJECT DESCRIPTION***

Run a pygame program to detect collision coordinates which are crucial in

gaming theory, logic and design. Code to work with and test for collision points will follow.

***Information About This Project***

You will create a simple example shown in Figure 1 below, of a Pygame script that demonstrates collision detection between two objects (player1 and player2), shows their movements on the screen, and prints out a message with the collision coordinates when a collision occurs. The movements are controlled by the W, A, S, D keys for player1 and the arrow keys for player2.

***Steps to Complete This Project***

**STEP 1**  **Open VSC**

Open the folder to work your code and create a file called **coordinate\_collider.py**.

**STEP 2**  **Testing the code**

Copy in the code shown on the next page into your .py file!

Ensure you do a Make sure you have Pygame installed to run this script. You can install it via pip if you haven’t already in your directory via the following terminal command:

pip install pygame

**STEP 3**  **Play the Game**

Run your program with multiple executions and observe the game in action!

import pygame, sys

# Initialize Pygame

pygame.init()

# Constants

WIDTH, HEIGHT = 800, 600

WHITE = (255, 255, 255)

RED = (255, 0, 0)

BLUE = (0, 0, 255)

BLACK = (0, 0, 0)

PLAYER\_SIZE = 50

MOVEMENT\_SPEED = 5

FONT\_SIZE = 24

# Setup the display

screen = pygame.display.set\_mode((WIDTH, HEIGHT))

pygame.display.set\_caption("Collision Detection")

# Define the player objects

player1 = pygame.Rect(100, 100, PLAYER\_SIZE, PLAYER\_SIZE)

player2 = pygame.Rect(300, 300, PLAYER\_SIZE, PLAYER\_SIZE)

# Clock for controlling the frame rate

clock = pygame.time.Clock()

# Setup font

font = pygame.font.Font(None, FONT\_SIZE)

# Function to draw text on the screen

def draw\_text(surface, text, pos, color=BLACK):

text\_surface = font.render(text, True, color)

surface.blit(text\_surface, pos)

collision\_message = ""

while True:

for event in pygame.event.get():

if event.type == pygame.QUIT:

pygame.quit()

sys.exit()

# Movement for player1 (W, A, S, D)

keys = pygame.key.get\_pressed()

if keys[pygame.K\_w]:

player1.y -= MOVEMENT\_SPEED

if keys[pygame.K\_s]:

player1.y += MOVEMENT\_SPEED

if keys[pygame.K\_a]:

player1.x -= MOVEMENT\_SPEED

if keys[pygame.K\_d]:

player1.x += MOVEMENT\_SPEED

# Movement for player2 (Arrow keys)

if keys[pygame.K\_UP]:

player2.y -= MOVEMENT\_SPEED

if keys[pygame.K\_DOWN]:

player2.y += MOVEMENT\_SPEED

if keys[pygame.K\_LEFT]:

player2.x -= MOVEMENT\_SPEED

if keys[pygame.K\_RIGHT]:

player2.x += MOVEMENT\_SPEED

# Check for collision

if player1.colliderect(player2):

collision\_message = f"Collision detected at ({player1.centerx}, {player1.centery})"

else:

collision\_message = ""

# Clear the screen

screen.fill(WHITE)

# Draw the players

pygame.draw.rect(screen, RED, player1)

pygame.draw.rect(screen, BLUE, player2)

# Draw the coordinates and collision message

draw\_text(screen, f"Player 1: ({player1.x}, {player1.y})", (10, 10))

draw\_text(screen, f"Player 2: ({player2.x}, {player2.y})", (10, 40))

if collision\_message:

draw\_text(screen, collision\_message, (10, 70), RED)

# Update the display

pygame.display.flip()

# Cap the frame rate

clock.tick(30)

Summary of code

1. **Initialize Pygame**: We start by initializing Pygame and setting up the display window.
2. **Constants and Setup**: Define the window size, colors, and player size. Create pygame.Rect objects for both players which represent their positions and sizes.
3. **Game Loop**:
   * **Event Handling**: Check for quitting events.
   * **Movement**: Update the position of player1 using the W, A, S, D keys and player2 using the arrow keys.
   * **Collision Detection**: Use colliderect to check if the two rectangles intersect. If they do, print the collision coordinates.
   * **Drawing**: Clear the screen and draw the updated positions of the players.
   * **Update Display**: Refresh the screen and limit the frame rate.