#### ****Turtle Pong Game Documentation****

****Overview:****

This is a simple Pong game implemented using the turtle module in Python. The game consists of two paddles and a ball. The players control the paddles to prevent the ball from hitting their side of the screen. Each player scores a point if the opponent misses the ball.

****Game Components:****

****Screen Setup**:**

1. The game window is set to 800x600 pixels.
2. The background color is black, and the game title is "Pong Game".

**# Set up the screen**

**wn = turtle.Screen()**

**wn.title("Pong Game")**

**wn.bgcolor("black")**

**wn.setup(width=800, height=600)**

**wn.tracer(0)**

****Paddles:****

1. There are two paddles: paddle\_a and paddle\_b.
2. They are represented as white squares stretched into rectangles (6x1).
3. paddle\_a is positioned on the left (-350, 0) and paddle\_b on the right (350, 0).

**paddle\_a = turtle.Turtle()**

**paddle\_a.speed(0)**

**paddle\_a.shape("square")**

**paddle\_a.color("white")**

**paddle\_a.shapesize(stretch\_wid=6, stretch\_len=1)**

**paddle\_a.penup()**

**paddle\_a.goto(-350, 0)**

**# Paddle B**

**paddle\_b = turtle.Turtle()**

**paddle\_b.speed(0)**

**paddle\_b.shape("square")**

**paddle\_b.color("white")**

**paddle\_b.shapesize(stretch\_wid=6, stretch\_len=1)**

**paddle\_b.penup()**

**paddle\_b.goto(350, 0)**

****Ball:****

1.The ball is a small white square that starts at the center of the screen (0, 0).

2. It moves in both the x and y directions using dx and dy values, which control its speed and direction.

**# Ball**

**ball = turtle.Turtle()**

**ball.speed(40)**

**ball.shape("square")**

**ball.color("white")**

**ball.penup()**

**ball.goto(0, 0)**

**ball.dx = 0.2**

**ball.dy = -0.2**

****Score Display:****

1. The scores for both players are initialized to 0.
2. A turtle named pen is used to display the scores at the top of the screen.

# **Score**

**score\_a = 0**

**score\_b = 0**

**# Pen**

**pen = turtle.Turtle()**

**pen.speed(0)**

**pen.color("white")**

**pen.penup()**

**pen.hideturtle()**

**pen.goto(0, 260)**

**pen.write("Player A: 0 Player B: 0", align="center", font=("Courier", 24, "normal"))**

****Player Controls:****

The paddles are controlled by the following keys:

**Player A (left)**:

Move up: w key

Move down: s key

**Player B (right)**:

Move up: Up arrow key (↑)

Move down: Down arrow key (↓)

Example: for paddle \_a\_up direction, Use the below code reference to paddle\_a and paddle\_b

**def paddle\_a\_up():**

**y = paddle\_a.ycor()**

**if y < 250:**

**y += 20**

**paddle\_a.sety(y)**

**# Keyboard bindings**

**wn.listen()**

**wn.onkeypress(paddle\_a\_up, "w")**

**wn.onkeypress(paddle\_a\_down, "s")**

**wn.onkeypress(paddle\_b\_up, "Up")**

**wn.onkeypress(paddle\_b\_down, "Down")**

****Ball Movement and Collisions:****

1. **Ball Movement**:

The ball's position is updated on each frame by adding dx to the x-coordinate and dy to the y-coordinate.

**while True:**

**wn.update()**

**# Move the ball**

**ball.setx(ball.xcor() + ball.dx)**

**ball.sety(ball.ycor() + ball.dy)**

1. **Boundary Checking**:
2. If the ball touches the top or bottom of the screen, it bounces by reversing its y-direction (dy).

**# Border checking**

**if ball.ycor() > 290:**

**ball.sety(290)**

**ball.dy \*= -1**

**if ball.ycor() < -290:**

**ball.sety(-290)**

**ball.dy \*= -1**

1. If the ball crosses the left or right side of the screen, a point is scored, and the ball resets to the center.

Example for paddle a given use the logic for paddle b

**if ball.xcor() > 390:**

**ball.goto(0, 0)**

**ball.dx \*= -1**

**score\_a += 1**

**pen.clear()**

**pen.write(f"Player A: {score\_a} Player B: {score\_b}", align="center", font=("Courier", 24, "normal"))**

**3. Paddle Collisions**:

1. If the ball hits a paddle, it bounces by reversing its x-direction (dx).

Example for paddle a given use the logic for paddle b

**if (ball.xcor() < -340 and ball.xcor() > -350) and (ball.ycor() < paddle\_a.ycor() + 50 and ball.ycor() > paddle\_a.ycor() - 50):**

**ball.setx(-340)**

**ball.dx \*= -1**