5 Building Classic Pong : Part II

Let us begin with a recap of where we got to last week; we had built the court and one of the paddles and I left you with the job of building the second paddle and the ball. *How did you do?* I will be assuming that everybody managed to complete these tasks, if you didn't then don't worry, the solution is at the back of last weeks notes.

5.1 Moving our paddles

The next stage is to make our paddles move, we will do this by implementing some simple functions, we will then call these functions when a certain button is pressed on the keyboard. Lets first look at how to get the left paddle to go up and down.

```
#Moving the Left Paddle

def paddle_a_move_up():
    '''
    Function moves paddle a up by 20 pix
    in the y direction.

    Returns
    -------
    None.

    '''
    y = paddle_a.ycor()
    y = y + 20
    paddle_a.sety(y)
```

Above I have defined a function that when called (and we will explain how to do that soon) will move the left paddle up by 20 pixels. (side notes: 20 pixels is a little arbitrary, why not try changing it in your code?).

How does the function move the paddle?

Firstly, we find out the current y position of the paddle. We then take that value (*e.g. y =0 if it is in the center of the screen*) and we add 20 to that value. We then set the position of the paddle to this new y position.

We can implement something very similar to make the paddle go down but instead of adding 20 we subtract 20.

```
def paddle_a_move_down():
    '''
    Function moves paddle a down by 20 pix
    in the y direction.

    Returns
    -------
    None.

    '''
    y = paddle_a.ycor()
    y = y - 20
    paddle_a.sety(y)
```

Exercise: Implement two more functions to move the right paddle in the same way.

So now we would like to call these functions by pressing certain keyboard buttons, we can do that by using the .listen method on our window and then specifying what function should be called if a certain button is pressed. Lets see how to implement this with the left paddle.

```
#Button Mapper

window.listen()
window.onkeypress(paddle_a_move_up, 's')
window.onkeypress(paddle_a_move_down, 'd')
```

The above code maps the button press of **s** to moving the left paddle up and **d** to moving the paddle down.

Exercise: Now write some code that causes the right paddle to move up when the 'Up' arrow key is pressed and down when the 'Down' arrow key is pressed.

5.2 Moving the ball

We will now look at how we can get the ball to move on the screen. The first thing we will do is add two lines of code to define the speed our ball will travel at. In my experience 3 is a good number to use here but why not play around with it. We add the speed by specifying the change in x and y (in pixels) that will happen after every update.

```
#Draw Ball
ball = turtle.Turtle()
ball.speed(0) #Speed of animation(this is the maxiumum)
ball.shape('circle')
ball.color('white')
ball.penup()#Stop drawing lines
ball.goto(0,0)

#Added the balls x and y pixel change per cycle (Week 5)
ball.dx = 2
ball.dy = 2
```

This can be seen in the final two lines of code. We will now animate the ball in the main game loop. To do this we get the current x and y position, add our change in x and y (dx and dy) and then set the ball to that new location. The code to achieve this is below.

```
#Main game loop
while True:
    window.update()


    #Ball Movement
    ball.setx(ball.xcor() + ball.dx)
    ball.sety(ball.ycor() + ball.dy)
```

Now run the code, see what you have made!

5.3 Collision Control

We are getting very close to having a working prototype of our game. We now need to implement some logic to decide what happens when the ball comes close to the edge or the paddle. Inside the main game loop we will implement the following code.

```
#Collison Check

    #1) Collison with top wall
    if ball.ycor() > 290:
        ball.sety(290)
        ball.dy *= -1
```

Here we are saying if the balls y position is above 290 (accounting for the balls size) then make the ball return to position 290 and send it in the opposite y direction (elastic scattering.)

Now run the code!

Exercise: Now write some code to do the same for the bottom wall, you can check it works by temporally changing ball.dy to -2.

Our final task for this week will be to deal with what happens when the ball collides with a paddle. We are aiming to make this also an elastic collision and so all we want to do is reverse the ball.dx direction. We do that using the following code in the main game loop.

```
#Paddle and Ball collisions
   if ball.xcor() < -340 and ball.ycor() < paddle_a.ycor() + 50
                                    and ball.ycor() > paddle_a.ycor() - 50:
       ball.dx *= -1
```

Here we are dealing with the left paddle. In the if statement we first make sure that the balls x coordinate is less than -340,in other words, it is in the vicinity of the paddle and then we need to check if the ball is with $\pm$ 50 of the paddles y position; if this is all true (the ball did collide with the paddle), then we reverse the balls horizontal direction.

Exercise: Now write some code to do the same for paddle b

That is the end of this weeks lesson, next week we will add the final game logic and features.

Week 5: Code Appendix

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Classic Pong - Inspired by the work of @TokyoEdTech.
File Version #: V1.2
Date: 10th September 2020

Pong is a 2D game that simulates table tennis. In this
implementation the human player will be playing against
the computer. We make use of the turtle module to draw
to the screen.

The following code is useful for WEEK 5 of the introduction
to Python course. Link avaible here:
https://futureskillsprogramme.github.io/Intro_to_python/

Course and code written by Kyle Fogarty.

"""


import turtle

window = turtle.Screen()
window.title('Pong by Kyle')
window.bgcolor('black')
window.setup(width = 800 , height = 600)
window.tracer(0)


#Dividing Line
line = turtle.Turtle()
line.hideturtle()
line.color('white')
line.penup()
line.goto(0,300)
line.pendown()
line.goto(0,-300)

#Draw Left Paddle
paddle_a = turtle.Turtle()
paddle_a.speed(0)
paddle_a.shape('square')
paddle_a.shapesize(stretch_wid = 5, stretch_len=1)
paddle_a.color('white')
```

```
paddle_a.penup()
paddle_a.goto(-350,0)

#Draw Right Paddle
paddle_b = turtle.Turtle()
paddle_b.speed(0)
paddle_b.shape('square')
paddle_b.shapesize(stretch_wid = 5, stretch_len=1)
paddle_b.color('white')
paddle_b.penup()
paddle_b.goto(350,0)


#Draw Ball
ball = turtle.Turtle()
ball.speed(0) #Speed of animation(this is the maxiumum)
ball.shape('circle')
ball.color('white')
ball.penup()#Stop drawing lines
ball.goto(0,0)

ball.dx = 3
ball.dy = 3


def paddle_a_move_up():
    '''
    Function moves paddle a up by 20 pix
    in the y direction.

    Returns
    -------
    None.

    '''
    y = paddle_a.ycor()
    y = y + 20
    paddle_a.sety(y)


def paddle_a_move_down():
    '''
    Function moves paddle a up by 20 pix
    in the y direction.

    Returns
    -------
    None.
```

```
    '''
    y = paddle_a.ycor()
    y = y - 20
    paddle_a.sety(y)


def paddle_b_move_up():
    '''
    Function moves paddle a up by 20 pix
    in the y direction.

    Returns
    -------
    None.

    '''
    y = paddle_b.ycor()
    y = y + 20
    paddle_b.sety(y)


def paddle_b_move_down():
    '''
    Function moves paddle a up by 20 pix
    in the y direction.

    Returns
    -------
    None.

    '''
    y = paddle_b.ycor()
    y = y - 20
    paddle_b.sety(y)



#Button Mapper
window.listen()
window.onkeypress(paddle_a_move_up, 's')
window.onkeypress(paddle_a_move_down, 'a')
window.onkeypress(paddle_b_move_up, 'Up')
window.onkeypress(paddle_b_move_down, 'Down')
```

```
#Main game loop
while True:
    window.update()


    #Ball Movement
    ball.setx(ball.xcor() + ball.dx)
    ball.sety(ball.ycor() + ball.dy)

    #Collison Check

    #1) Collison with top wall
    if ball.ycor() > 290:
        ball.sety(290)
        ball.dy *= -1

    if ball.ycor() < -290:
        ball.sety(-290)
        ball.dy *= -1


     #Paddle and Ball collisions
    if ball.xcor() < -340 and ball.ycor() < paddle_a.ycor() + 50 and ball.ycor() > paddle_a.yc
        ball.dx *= -1


    if ball.xcor() > 340 and ball.ycor() < paddle_b.ycor() + 50 and ball.ycor() > paddle_b.yco
        ball.dx *= -1
```