POND GAME

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By

Registration No.	Name	Roll No.
12005680	Ankit Jain	RK20RGB72
120063117	Ashirvad Kumar	RK20RGB47

Under the guidance of

Ankita Wadhawan

School of Computer Science and Engineering

Lovely Professional University

Phagwara, Punjab (India)

NOV 2021

ACKNOWLEDGMENT

Place : Lovely Professional University

We would like to thank Mrs. Ankita Wadhawan for assigning us with this project. Through the project we are able to grasp more technical and have a hands-on practical experience with python projects. Through it we are able to learn how a project is created and how necessary and crucial technical knowledge is. We are really grateful to the faculty that has provided us with the necessary guidelines.

Name	**Reg. Number**
Ankit Jain	12005680
Ashirvad Kumar	120063117

DECLARATION STATEMENT

Place : Lovely Professional University

This is to declare that this report has been written by us. No part of the report is copied from the other sources. All information included from other sources has been duly acknowledged. We aver that if any part of the report is found to be copied, we will take full responsibility for it.

Name : Ankit Jain

Reg. No. : 12005680

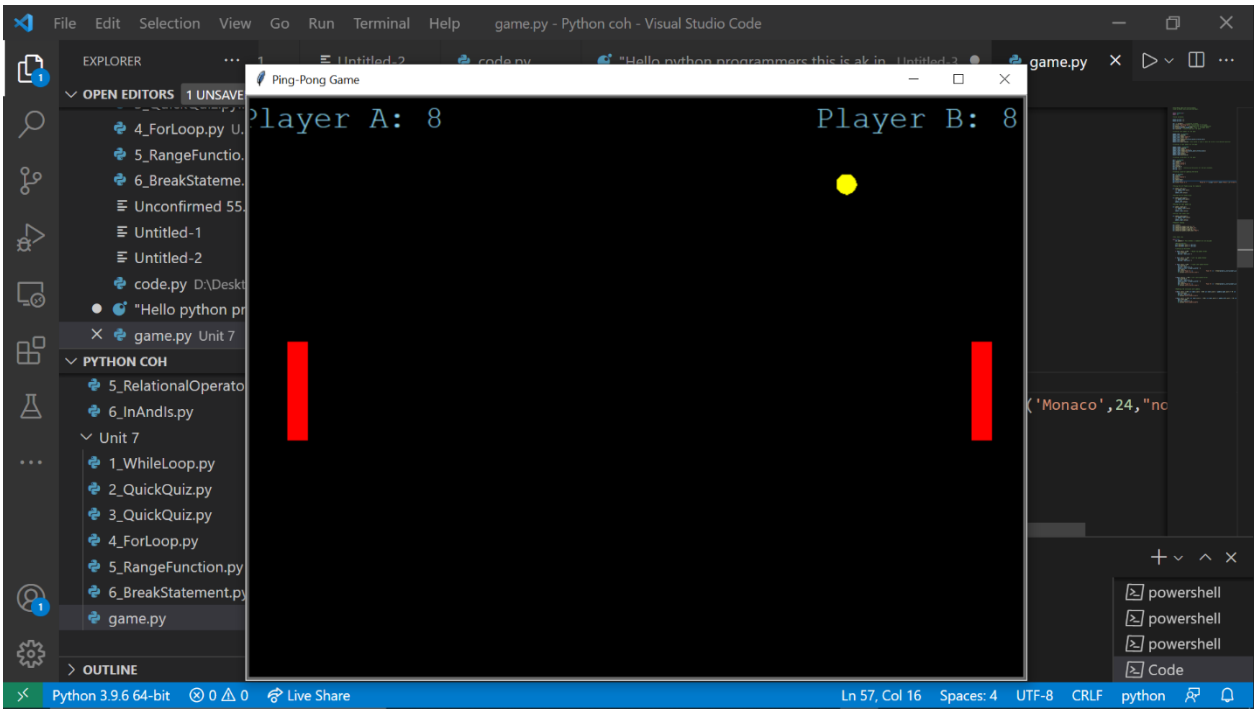
Name : Ashirvad Kumar

Reg. No. : 120063117

CHAPTER - 1
INTRODUCTION

Introduction

Pong is one of the most famous arcade games, simulating table tennis. Each player controls a paddle in the game by dragging it vertically across the screen's left or right side. Players use their paddles to strike back and forth on the ball.
Turtle is an inbuilt graphic module in Python. It uses a panel and pen to depict illustration.



Design

To create the game, we need to simulate the following:

The system:

The game consists of a single ball, and a paddle (or bat) for each player to control. The ball and paddles are limited to an area on the screen, which I'll call the playing field. The ball constantly bounces around the playing field, and the players move their paddles up and down, to deflect the ball and keep it from going past them horizontally. When the ball 'hits' a paddle, or the top or bottom of the playing field, it must 'bounce off'. If the ball goes past a paddle horizontally to collide with the left or right side of the playing field, a point is given to the other player, and the game continues with a new ball.

Motion graphics:

There is one ball that moves constantly, two paddles that move when the correct keys are

pressed, and the score which changes when a point is scored.
Control: Player 1 has a designated keyboard key that moves their paddle up, and another that moves their paddle down. Player 2 also has two keys for moving their own paddle up and down.

Approach

When using turtle graphics to implement this game, it's easiest to use a number of different turtles for the different elements:

A dedicated turtle to draw the outline of the playing field. As the playing field doesn't move around, this drawing would never change.
Another turtle to display the score. A variable would be used to keep track of each player's score. A turtle has a function for drawing text on the screen, and it would be used to draw the score (the values of the score variables). When a point is scored, the previous score drawing is cleared from the screen, before the new score is drawn, much like a very slow animation.
Another turtle represents the ball, and therefore should look like one so that the user can understand the system. The ball starts at the middle of the playing field, and gets animated as moving across the screen in some direction, just like we did with the square in the previous post. When it reaches the top or bottom of the playing field, or the front of a paddle, it 'bounces off'. This is simulated by changing the direction in which the ball is moving (or more precisely, the direction that the animation makes the ball appear to move in). If the change of direction is done at the correct time (exactly when the ball appears to collide with something), and in such a way that it resembles how a ball might bounce off of a flat surface in the real world (obeying the law of reflection), it creates the illusion that the ball has been deflected:

Bouncing off the top or bottom of the playing field is simulated by simply flipping the vertical direction of the ball's motion, so that if it was busy moving left and up for example, it would then continue moving left and down.
Bouncing off the front of a paddle can be simulated by flipping the horizontal direction of the ball's motion, so that if it was busy moving left and up for example, it would then continue moving right and up.
If the ball reaches the left or right side of the playing field, the opposing player's score increases by one point, and the ball instantly continues its movement at the middle of the playing field so that it looks like the game has continued with a new ball.
Each paddle is also represented by a turtle with a rectangular shape. Each is positioned horizontally at either the far right (for player 1) or the far left (for player 2) of the playing field. Player 1 can use the up ↑ and down ↓ arrow keys on the keyboard, and player 2 the w and s keys, to move their respective paddles up and down. If the paddle reaches the top of the playing field, it shouldn't be able to move up further, and the same goes for moving down at the bottom of the playing field: this can be done by simply ignoring that player's key-press at the appropriate time.

Algorithm

As the ball must appear to be constantly moving, we repeatedly need to display the screen as a frame many times a second, with the ball having moved a small amount between each frame or time-step — in other words, we're simply animating the movement of the ball, as explained in the previous post. We can keep track of the direction and speed of the ball's movement by giving that movement a horizontal and vertical component, which we keep track of with a variable or two. At each time-step, if the ball doesn't appear to collide with anything, we simply add the horizontal component to the ball's horizontal position, and add the vertical component to the ball's vertical position to move the ball to its new position.

If a player pushes a designated key, we also move their paddle a small amount in the up or down direction as appropriate — that is, unless it's not allowed to move further up or down, and then we simply ignore that key-press by not doing anything about it so that the paddle stays put at the bottom or top of the playing field.

And lastly, to implement all the other logic for the game, at each time-step we simply have to check if the position of the ball makes it appear to be colliding with something, and take appropriate action if it does. At each time step, does the ball appear to be colliding with:

The top or bottom of the playing field? Then we flip the vertical direction of the ball's motion, so that in the next frame, the ball will 'move' in the opposite vertical direction;
A paddle? Then we flip the horizontal direction of the ball's motion, so that in the next frame, the ball will 'move' in the opposite horizontal direction;
The left or right side of the playing field? Then we add one to the other player's score, write the new score to the screen, and move the ball to the middle of the playing field again on the next frame, to simulate a new ball.

CHAPTER 2
METHODOLOGY

Requirements

First step is gathering our requirements of the project. The functions and the modules and the necessary for the development of the project.

Functionality

In this step we will discuss the different function used in the program and how they are working.

The different types of function we used are…

Method	Parameter	Description
Turtle()	None	Creates and returns a new turtle object
forward()	amount	Moves the turtle forward by the specified amount
backward()	amount	Moves the turtle backward by the specified amount
right()	angle	Turns the turtle clockwise
left()	angle	Turns the turtle counterclockwise
penup()	None	Picks up the turtle's Pen
pendown()	None	Puts down the turtle's Pen
up()	None	Picks up the turtle's Pen
down()	None	Puts down the turtle's Pen
color()	Color name	Changes the color of the turtle's pen
fillcolor()	Color name	Changes the color of the turtle will use to fill a polygon
heading()	None	Returns the current heading

Method	Parameter	Description
position()	None	Returns the current position
goto()	x, y	Move the turtle to position x,y
begin_fill()	None	Remember the starting point for a filled polygon
end_fill()	None	Close the polygon and fill with the current fill color
dot()	None	Leave the dot at the current position
stamp()	None	Leaves an impression of a turtle shape at the current location
shape()	shapename	Should be ‘arrow’, ‘classic’, ‘turtle’ or ‘circle’

Implementation

This step is to implementing the project by coding. This is most important part of the program. A particular care should be done in order to get error free program.

Turtle module

We’ll be making use of Turtle graphics, so let’s begin by importing some things from the turtle module. We’ll obviously need a screen to begin with, so we must import the Screen class. We’ll also be using individual turtles, so we need to import the Turtle class. We’d also like to define a custom shape for the two turtles that will represent our paddles, so we need to import the Shape class as well.

Screen

As we need a screen before we can display anything, it makes sense for it to be the first thing we create

Playing field

Instead of using the entire screen as the area where the game is played, let’s designate a smaller area as the playing field so as to leave some space for displaying the score. Whether we use the entire screen or not, it makes sense to create variables that describe the top, bottom, left and right sides of the playing field, as we can then use these later to determine when the ball should bounce off the top and bottom of the area, and when it reaches the left or right. As the origin of the default coordinate system is in the middle of the screen, it divides the screen into halves both horizontally and vertically, with positive x

values on the right, negative x values on the left, positive y values in the top half, and negative y values in the bottom half. Let's make our playing field 200 units less than the screen height (100 less at top, 100 at bottom), and 100 units less than the screen width (50 less on left, 50 on the right)

Ball

We need a turtle to represent the ball. Let's change its shape to a circle so that it looks like a ball. We'll simply move it around the screen, so let's lift up its pen so that it doesn't draw. The built-in turtle shapes have a 10 unit radius which seems too big for our small playing field, so let's stretch it to half that size. Saving the ball's new radius should prove useful for animating collisions later, as we need to work out exactly when to make it look like it has collided with something:

Paddles

Now we'll create two turtles to represent the two paddles. Let's decide to use "L" in variable names that relate to the player on the left, and likewise "R" for things that relate to the player on the right. Turtles will simply look like paddles and move around, so they mustn't draw:

We'd like our paddles to look like black rectangles, so we need to change the L and R turtle's shapes. We must create the shape by specifying the points, then register the shape with the screen under a name we've made up, like "paddle". We can then change the shape of the turtles to the "paddle" shape. Let's make the paddles 10 units wide and 40 units tall.

The position of a turtle on the screen is actually the position of the origin of that turtle's local coordinate system. When we add a new shape, it therefore makes sense to have that shape centred on the origin, so that the position of the turtle is the exact middle of the shape at point (0,0), otherwise there will be a mismatch between the visual shape and the position that the turtle reports. The points that define the shape will consequently have both negative and positive x and y values

Score

Finally we'll need one more turtle for displaying the score, and variables for keeping track of each player's score. Let's create a function which we can call whenever we want our score turtle to update the score's display. A turtle has a function for drawing text on the screen, and it can be used to display the values of the score variables. The function should clear the previous score drawing, before drawing the new scores:

Debugging

This step is to check the project for any errors in the code. And linking of the module take place after checking individual modules and functions.

Execution

This step is to executing the project and simple testing.

Working

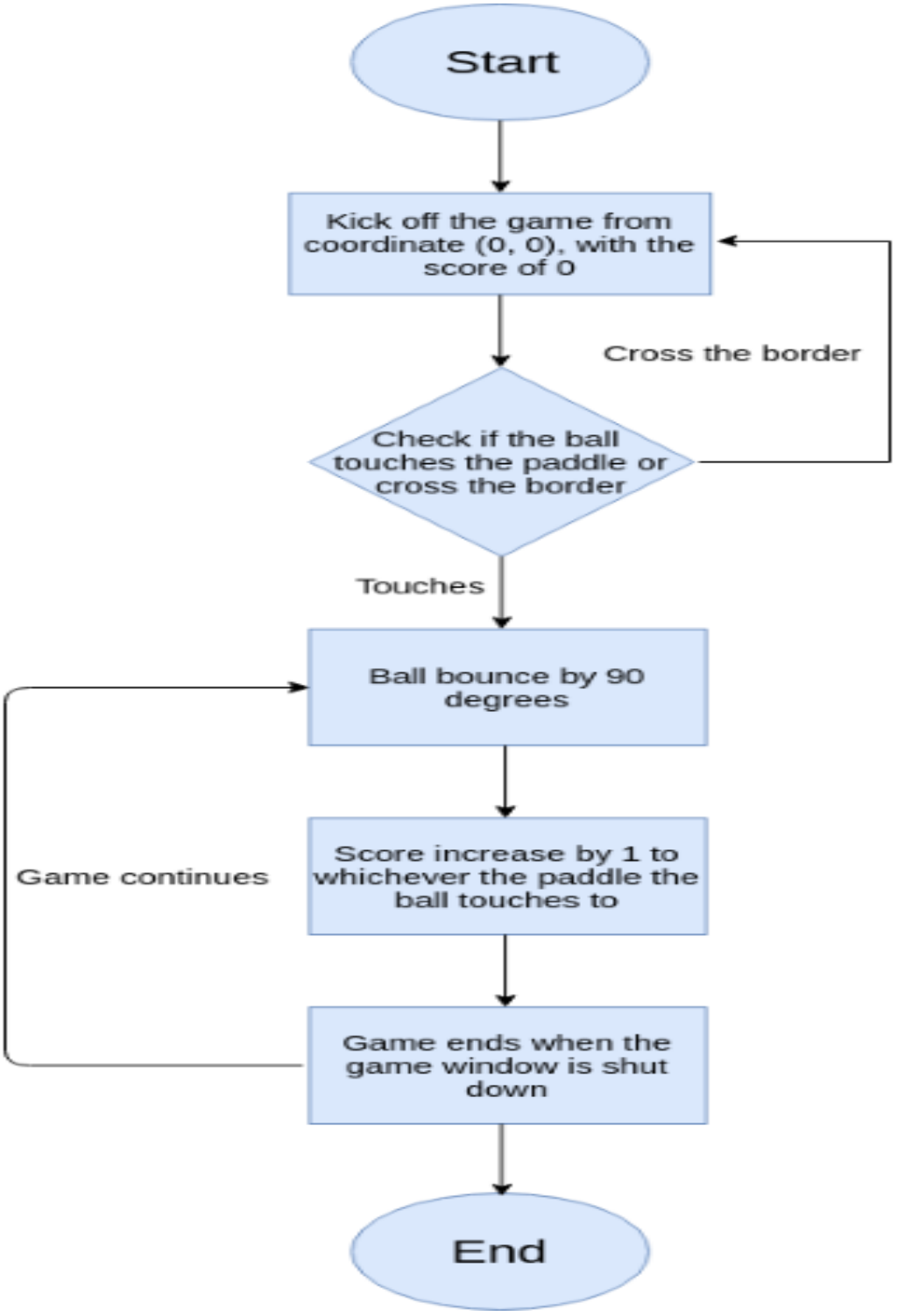
We have used turtle library to develop GUI for the project. We import sys library to know the current version we are using.

We import os for linking a program with another program.

Chapter - 3

Code And Flow Chart

Flow Chart



CODE : -

```
# Ping-Pong game with turtle module.
# Done by Ankit Jain and Ashirvad Kumar.


import turtle as t
import os

# Score varibales

player_a_score = 0
player_b_score = 0

win = t.Screen()    # creating a window
win.title("Ping-Pong Game") # Giving name to the game.
win.bgcolor('black')    # providing color to the HomeScreen
win.setup(width=800,height=600) # Size of the game panel
win.tracer(0)   # which speed up's the game.

# Creating left paddle for the game

paddle_left = t.Turtle()
paddle_left.speed(0)
paddle_left.shape('square')
paddle_left.color('red')
paddle_left.shapesize(stretch_wid=5,stretch_len=1)
paddle_left.penup()
paddle_left.goto(-350,0) # This method is used to move the turtle to
an absolute position

# Creating a right paddle for the game

paddle_right = t.Turtle()
paddle_right.speed(0)
paddle_right.shape('square')
paddle_right.shapesize(stretch_wid=5,stretch_len=1)
paddle_right.color('red')
paddle_right.penup()
paddle_right.goto(350,0)
```

```
# Creating a pong ball for the game

ball = t.Turtle()
ball.speed(0)
ball.shape('circle')
ball.color('yellow')
ball.penup()
ball.goto(0,0)
ball_dx = 1.5   # Setting up the pixels for the ball movement.
ball_dy = 1.5

# Creating a pen for updating the Score

pen = t.Turtle()
pen.speed(0)
pen.color('skyblue')
pen.penup()
pen.hideturtle()
pen.goto(0,260)
pen.write("Player A: 0                    Player B: 0
",align="center",font=('Monaco',24,"normal"))


# Moving the left Paddle using the keyboard

def paddle_left_up():
    y = paddle_left.ycor()
    y = y + 15
    paddle_left.sety(y)

# Moving the left paddle down

def paddle_left_down():
    y = paddle_left.ycor()
    y = y - 15
    paddle_left.sety(y)

# Moving the right paddle up

def paddle_right_up():
    y = paddle_right.ycor()
    y = y + 15
```

```
    paddle_right.sety(y)

# Moving right paddle down

def paddle_right_down():
    y = paddle_right.ycor()
    y = y - 15
    paddle_right.sety(y)

# Keyboard binding

win.listen()
win.onkeypress(paddle_left_up,"w")
win.onkeypress(paddle_left_down,"e")
win.onkeypress(paddle_right_up,"Up")
win.onkeypress(paddle_right_down,"Down")




# Main Game Loop

while True:
    win.update() # This methods is mandatory to run any game

    # Moving the ball
    ball.setx(ball.xcor() + ball_dx)
    ball.sety(ball.ycor() + ball_dy)

    # setting up the border

    if ball.ycor() > 290:   # Right top paddle Border
        ball.sety(290)
        ball_dy = ball_dy * -1


    if ball.ycor() < -290:  # Left top paddle Border
        ball.sety(-290)
        ball_dy = ball_dy * -1


    if ball.xcor() > 390:   # right width paddle Border
        ball.goto(0,0)
```

```
        ball_dx = ball_dx * -1
        player_a_score = player_a_score + 1
        pen.clear()
        pen.write("Player A: {}                    Player B: {}
".format(player_a_score,player_b_score),align="center",font=('Monaco
',24,"normal"))
        os.system("afplay wallhit.wav&")



    if(ball.xcor()) < -390: # Left width paddle Border
        ball.goto(0,0)
        ball_dx = ball_dx * -1
        player_b_score = player_b_score + 1
        pen.clear()
        pen.write("Player A: {}                    Player B: {}
".format(player_a_score,player_b_score),align="center",font=('Monaco
',24,"normal"))
        os.system("afplay wallhit.wav&")


    # Handling the collisions with paddles.

    if(ball.xcor() > 340) and (ball.xcor() < 350) and (ball.ycor() <
paddle_right.ycor() + 40 and ball.ycor() > paddle_right.ycor() -
40):
        ball.setx(340)
        ball_dx = ball_dx * -1
        os.system("afplay paddle.wav&")

    if(ball.xcor() < -340) and (ball.xcor() > -350) and (ball.ycor()
< paddle_left.ycor() + 40 and ball.ycor() > paddle_left.ycor() -
40):
        ball.setx(-340)
        ball_dx = ball_dx * -1
        os.system("afplay paddle.wav&")
```

CHAPTER - 4
TECHNOLOGY

Python 3.7

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together.
Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse.

GUI

Short for Graphical User Interface, a GUI (pronounced as either G-U-I or gooey) allows the use of icons or other visual indicators to interact with electronic devices, rather than using only text via the command line.
A GUI uses icons, and menus to carry out commands, such as opening, deleting, and moving files.

IDE

Integrated development environment (IDE) used is PyCharm. It provides code analysis, a graphical debugger, an integrated unit tester etc.

Turtle

"Turtle" is a Python feature like a drawing board, which lets us command a turtle to draw all over it! We can use functions like turtle.forward(…) and turtle.right(…) which can move the turtle around.

Option with values

Widgets classes contains option with their values to change the interface of the widgets classes means appearance of the window. Some of the option are width, height, fg, bg, side, row, column, text, onvalue, offvalue, variable, expand, fill, file, image, command and so on.

CHAPTER - 5
REFERENCES

We went through different website and learn those concept that will used in making this project. Some of the website are…

- https://www.python.org
- https://www.w3schools.com
- https://www.google.com/
- We followed the text book named “Introduction to Programming Using python” by Y. Daniel Liang.