INTRODUCTION TO PROGRAMMING

FINAL PROJECT REPORT

GAME OF PONG

Davin Pratama Chandra

2301891090

FACULTY OF COMPUTER SCIENCE

BINUS INTERNATIONAL UNIVERSITY

A. PROJECT SPECIFICATION

1. Project Description

This project is a two-player pong game with the concept of a table tennis game. The purpose of this project is to entertain the users and they can use this project to spend time with others while practicing their gameplay strategy skills.

When the user launches the game, it will display two paddles (one for each player) and a ball. The players have to move their paddles up and down to hit the ball and prevent the ball from hitting the left side edge of the screen (for player 1) and the right side edge of the screen (for player 2). The players start the game with 0 point and will gain 1 point every time the ball hits their opponent's edge of the screen. The player who reaches 5 points first wins the game.

2. Project Features

This project is made using Python Programming Language (python 3.7) in Spyder Integrated Development Environment (IDE). In this project, I used several modules, like pygame and random.

The program will receive keypresses and convert them to movements of the paddles in the game. The program can detect collisions of the ball with the paddle and the edges of the screen. The program will stop when a player reaches 5 points and will display a winning statement. (Example: 'Player X wins!'). Players can also pause the game by pressing the space button and press it again to resume. Players can exit the game anytime by pressing the 'X' button.

B. SOLUTION DESIGN

The main code of the project is stored in *pong.py*, which also stores the codes for the game display.

1. Preparation

Before I start to make the code, I prepared the things I need first, like the colors and the fonts.

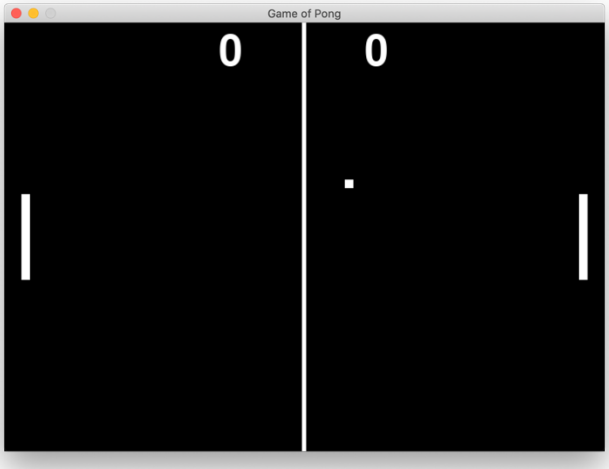
```
# Define some colours
BLACK = (0,0,0)
WHITE = (255,255,255)
```

The game's design is like an arcade game. So, I used black and white to make it look simple and arcade-like.

```
font = pygame.font.Font(None, 74)
```

In this project, I used the default font from pygame. The font will be used for the scores and winning statements.

2. Game Display




```
# Open a new window
size = (700, 500)
screen = pygame.display.set_mode(size)
pygame.display.set_caption("Game of Pong")
```

First, I need to open a new window with a certain size and caption.

```
paddle1 = Paddle(WHITE, 10, 100)
paddle1.rect.x = 20
paddle1.rect.y = 200

paddle2 = Paddle(WHITE, 10, 100)
paddle2.rect.x = 670
paddle2.rect.y = 200

ball = Ball(WHITE,10,10)
ball.rect.x = 345
ball.rect.y = 195

#This will be a list that will contain all the sprites we intend to use in our game.
all_sprites_list = pygame.sprite.Group()

# Add the paddles and the ball to the list of sprites
all_sprites_list.add(paddle1)
all_sprites_list.add(paddle2)
all_sprites_list.add(ball)
```

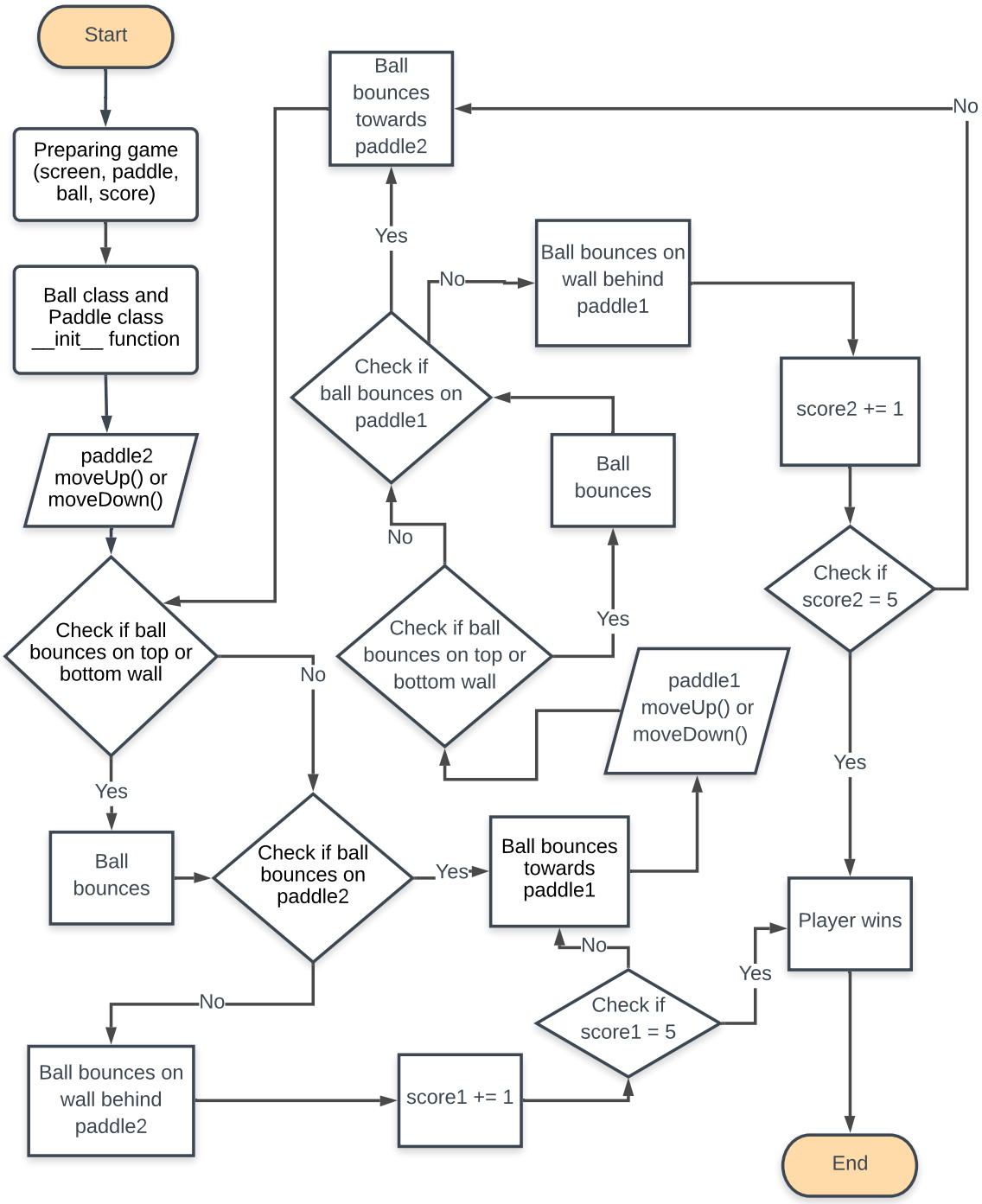
Prepare the paddles and the ball. Then, put them into a list to make it easier to draw them on the screen. (Ball and Paddle class are on a separate file.)

```
#Clear the screen to black.
screen.fill(BLACK)
#Draw the net
pygame.draw.line(screen, WHITE, [349, 0], [349, 500], 5)

#Draw all the sprites in one go.
all_sprites_list.draw(screen)

#Display scores:
font = pygame.font.Font(None, 74)
text = font.render(str(score1), 1, WHITE)
screen.blit(text, (250,10))
text = font.render(str(score2), 1, WHITE)
screen.blit(text, (420,10))
```

Fill the screen with black color and draw the net in the middle of the screen. Then, I draw the sprites (paddles and ball) on the screen and display the scores on the screen with the font I chose.

C. DISCUSSION (How the program works?)

1. Flowchart



2. Explanation of the code

Paddle.py

- __init__(self, color, width, height):

    - Creating the paddle with certain color, width and height.
- moveUp(self, pixels):
    - Moving the paddle up on the screen by the number of pixels set.
    - Check if the paddle moves too far (off the screen)
- moveDown(self, pixels):
    - Moving the paddle down on the screen by the number of pixels set.
    - Check if the paddle moves too far (off the screen)

Ball.py

- From *random* import *randint*
    - *Randint* is used to generate random integers for the *self.velocity*.
- __init__(self, color, width, height):
    - Creating the ball with certain color, width and height.
    - Moving the ball to a random direction with *randint()* when the game starts.
- update(self):
    - Moving the ball (by changing the (x,y) coordinates of the ball with the velocity vector).
- bounce(self):
    - Causing the ball to bounce in a random direction with *randint()* every time it's called.
    - Called every time the ball hits a paddle.

Pong.py (main code)

- Import paddle and ball class
- Game Design
    - Define colors (black and white) to make it easier to use them on the code.
    - Opening a new window.
    - I create the ball and paddles by using the format from the imported class from *ball.py* and *paddle.py*. Then, I put them inside a list of sprites so that I can draw them to the screen by calling them in one line of code.
- Initializing the player's scores (starting with 0 points).
- Game controls

```
keys = pygame.key.get_pressed()
if keys[pygame.K_w]:
    paddle1.moveUp(7)
if keys[pygame.K_s]:
    paddle1.moveDown(7)
if keys[pygame.K_UP]:
    paddle2.moveUp(7)
if keys[pygame.K_DOWN]:
    paddle2.moveDown(7)
```

- Several events happening inside the game is controlled by the players, like the paddle movements. Here are the controls to move the paddles:
  - 'W' or 'up arrow' key to move the paddle 7 pixels up. (*moveUp()*)
  - 'S' or 'down arrow' key to move the paddle 7 pixels down. (*moveDown()*)

```
for event in pygame.event.get(): # User did something
    if event.type == pygame.QUIT: # If user clicked close
        carryOn = False # Exit this loop
    elif event.type == pygame.KEYDOWN and event.key == pygame.K_SPACE: # The pause button
        while True: # Infinite loop that will be broken when the user press the space bar again
            event = pygame.event.wait()
            if event.type == pygame.KEYDOWN and event.key == pygame.K_SPACE:
                break # Exit infinite loop
    elif event.type == pygame.KEYDOWN and event.key == pygame.K_x:
        # Pressing the X key will quit the game
        carryOn=False
```

- There are also buttons to pause and exit the game.
  - Press space bar to pause (*pygame.event.wait()*). Press space bar again to resume (breaks the loop).
  - Press 'x' to exit the game (makes *carryOn = False*, which will exit the main loop).

- Collision check and bouncing the ball

```
# Check if the ball is bouncing against any of the 4 walls:
if ball.rect.x>=690:
    score1+=1
    ball.velocity[0] = -ball.velocity[0]
if ball.rect.x<=0:
    score2+=1
    ball.velocity[0] = -ball.velocity[0]
if ball.rect.y>490:
    ball.velocity[1] = -ball.velocity[1]
if ball.rect.y<0:
    ball.velocity[1] = -ball.velocity[1]

# Detect collisions between the ball and the paddles
if pygame.sprite.collide_mask(ball, paddle1) or pygame.sprite.collide_mask(ball, paddle2):
  ball.bounce()
```

- The game needs to check if the ball hits any of the 4 walls. If it does, it will bounce to the opposite way (*-ball.velocity*). If it hits the left or right side wall, it will add the scores of the player too.
- Then, the game needs to check if the ball hits the other sprites (the paddles). If it does, it will call the function *ball.bounce()* from the Ball class, which will bounce the ball.

- Drawing code

```
# Display scores:
font = pygame.font.Font(None, 74)
text = font.render(str(score1), 1, WHITE)
screen.blit(text, (250,10))
text = font.render(str(score2), 1, WHITE)
screen.blit(text, (420,10))
```

  - First, I clear the screen to black. Then, I draw the net in the middle of the screen.
  - Then, I draw all of the sprites in one code line (*all_sprites_list.draw(screen)*).
  - Next, I display the scores. Each score is displayed on each sides of the screen separated by the net.

- Winning condition
  - If player 1's score reaches 5 points first, a statement "Player 1 wins!" will show up on the screen. The same thing will happen to player 2 if the score reaches 5 points first.

  - I put the statements inside a variable called '*text*'. So, if *score1* reaches 5 points, *text = 'Player 1 wins!'*. If score2 reaches 5 points, *text = 'Player 2 wins!'*.

```
# Winning condition
# If a player wins, it will display a statement 'Player X wins!'
if score1 == 5:
    text = font.render('Player 1 Wins !', True, BLACK, WHITE)
elif score2 == 5:
    text = font.render('Player 2 Wins !', True, BLACK, WHITE)

if score1 == 5 or score2 == 5:
    textRect = text.get_rect()
    textRect.center = (700 // 2, 500 // 2)
    screen.blit(text, textRect)
    carryOn = False
```

3. Source Code

pong.py

```
# Import the pygame library and initialise the game engine
import pygame
```

```
# Import Paddle class and Ball class
from paddle import Paddle
from ball import Ball

pygame.init()

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

# Open a new window
size = (700, 500)
screen = pygame.display.set_mode(size)
pygame.display.set_caption("Game of Pong")

# Create the paddles and the ball
paddle1 = Paddle(WHITE, 10, 100)
paddle1.rect.x = 20
paddle1.rect.y = 200

paddle2 = Paddle(WHITE, 10, 100)
paddle2.rect.x = 670
paddle2.rect.y = 200

ball = Ball(WHITE, 10, 10)
ball.rect.x = 345
ball.rect.y = 195

# This will be a list that will contain all the sprites I intend
to use in the game.
all_sprites_list = pygame.sprite.Group()

# Add the paddles and the ball to the list of sprites
all_sprites_list.add(paddle1)
all_sprites_list.add(paddle2)
all_sprites_list.add(ball)

# The loop will carry on until the user exit the game
carryOn = True

# The clock will be used to control how fast the screen updates
clock = pygame.time.Clock()

# Initialise player scores
score1 = 0
score2 = 0

# -------- Main Program Loop -----------
while carryOn:
    # --- Main event loop
    for event in pygame.event.get(): # User did something
        if event.type == pygame.QUIT: # If user clicked close
            carryOn = False # Exit this loop
        elif event.type == pygame.KEYDOWN and event.key ==
pygame.K_SPACE: # The pause button
```

```
            while True: # Infinite loop that will be broken when
the user presses the space bar again
                event = pygame.event.wait()
                if event.type == pygame.KEYDOWN and event.key ==
pygame.K_SPACE:
                    break # Exit infinite loop
        elif event.type == pygame.KEYDOWN and event.key ==
pygame.K_x:
            # Pressing the X key will quit the game
            carryOn = False

    # Moving the paddles when the user uses "W/S" keys (player 1)
or the arrow keys (player 2)
    keys = pygame.key.get_pressed()
    if keys[pygame.K_w]:
        paddle1.moveUp(7)
    if keys[pygame.K_s]:
        paddle1.moveDown(7)
    if keys[pygame.K_UP]:
        paddle2.moveUp(7)
    if keys[pygame.K_DOWN]:
        paddle2.moveDown(7)

    all_sprites_list.update()

    # Check if the ball is bouncing against any of the 4 walls:
    if ball.rect.x>=690:
        score1+=1
        ball.velocity[0] = -ball.velocity[0]
    if ball.rect.x<=0:
        score2+=1
        ball.velocity[0] = -ball.velocity[0]
    if ball.rect.y>490:
        ball.velocity[1] = -ball.velocity[1]
    if ball.rect.y<0:
        ball.velocity[1] = -ball.velocity[1]

    # Detect collisions between the ball and the paddles
    if pygame.sprite.collide_mask(ball, paddle1) or
pygame.sprite.collide_mask(ball, paddle2):
      ball.bounce()

    # --- Drawing code
    # Clear the screen to black.
    screen.fill(BLACK)
    # Draw the net
    pygame.draw.line(screen, WHITE, [349, 0], [349, 500], 5)

    # Draw all the sprites in one go.
    all_sprites_list.draw(screen)

    # Display scores:
    font = pygame.font.Font(None, 74)
    text = font.render(str(score1), 1, WHITE)
    screen.blit(text, (250,10))
```

```
    text = font.render(str(score2), 1, WHITE)
    screen.blit(text, (420,10))

    # Winning condition
    # If a player wins, it will display a statement 'Player X
wins!'
    if score1 == 5:
        text = font.render('Player 1 Wins !', True, BLACK, WHITE)
    elif score2 == 5:
        text = font.render('Player 2 Wins !', True, BLACK, WHITE)

    if score1 == 5 or score2 == 5:
        textRect = text.get_rect()
        textRect.center = (700 // 2, 500 // 2)
        screen.blit(text, textRect)
        carryOn = False

    # --- Update the screen
    pygame.display.flip()

    # --- Limit to 60 frames per second
    clock.tick(60)

# Once we have exited the main program loop we can stop the game
engine:
pygame.quit()
```

paddle.py

```
import pygame

BLACK = (0,0,0)

class Paddle(pygame.sprite.Sprite):
    # This class represents the Paddle. It derives from the
"Sprite" class in Pygame.

    def __init__(self, color, width, height):
        # Call the parent class (Sprite)
        super().__init__()

        # Pass in the color of the paddle, and its x and y
position, width and height.
        # Set the background color and set it to be transparent
        self.image = pygame.Surface([width, height])
        self.image.fill(BLACK)
        self.image.set_colorkey(BLACK)

        # Draw the paddle
        pygame.draw.rect(self.image, color, [0, 0, width,
height])
```

```
        # Fetch the rectangle object that has the dimensions of
the image.
        self.rect = self.image.get_rect()

    def moveUp(self, pixels):
        self.rect.y -= pixels
         #Check that the paddle is not going too far (off the
screen)
        if self.rect.y < 0:
          self.rect.y = 0

    def moveDown(self, pixels):
        self.rect.y += pixels
        #Check that the paddle is not going too far (off the
screen)
        if self.rect.y > 400:
          self.rect.y = 400
```

ball.py

```
import pygame
from random import randint

BLACK = (0,0,0)

class Ball(pygame.sprite.Sprite):
    # This class represents the ball. It derives from the
"Sprite" class in Pygame.

    def __init__(self, color, width, height):
        # Call the parent class (Sprite)
        super().__init__()

        # Pass in the color of the ball, and its x and y
position, width and height.
        # Set the background color and set it to be transparent
        self.image = pygame.Surface([width, height])
        self.image.fill(BLACK)
        self.image.set_colorkey(BLACK)

        # Draw the ball
        pygame.draw.rect(self.image, color, [0, 0, width,
height])

        self.velocity = [randint(9,13),randint(-3,13)]

        # Fetch the rectangle object that has the dimensions of
the image.
        self.rect = self.image.get_rect()

    def update(self):
        # Moving the ball
        self.rect.x += self.velocity[0]
```

```
        self.rect.y += self.velocity[1]

    def bounce(self):
        self.velocity[0] = -self.velocity[0]
        self.velocity[1] = randint(-3,13)
```

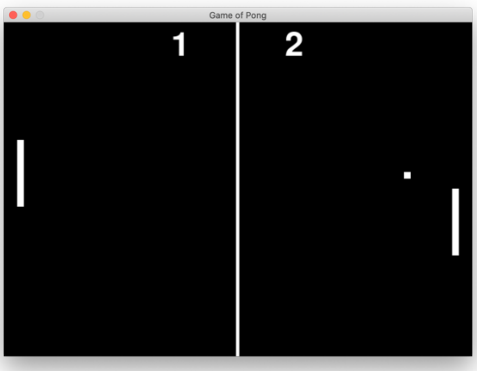
D. CONCLUSION

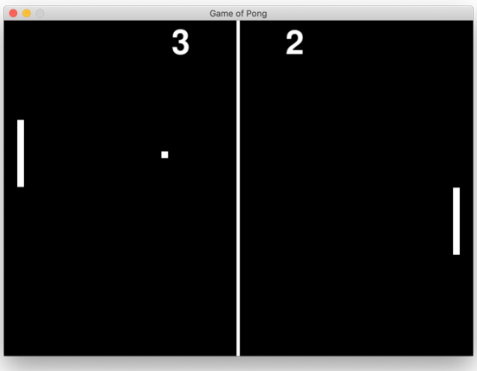
The program I create is a 'Pong' game. This game requires two players to play the game. Player 1 will use the left paddle and Player 2 will use the right paddle. The objective of the game is to reach 5 points first by hitting the ball to the opponent's wall.

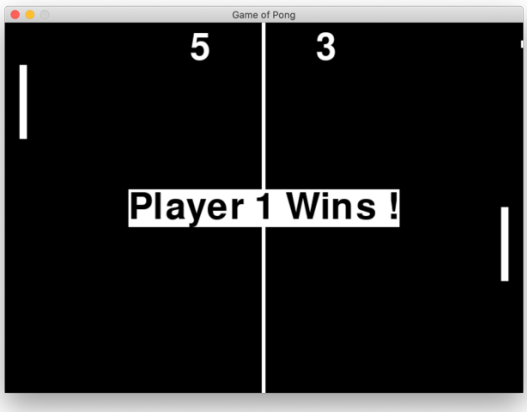
I choose to create this program because it is pretty simple to create and it meets all the requirements needed. The codes in this program is quite complicated but easy to understand.

E. EVIDENCE OF WORKING PROGRAM

Here are more screenshots of the program.







F. References

- Source code: https://www.101computing.net/pong-tutorial-using-pygame-getting-started/

- To help me fix the errors: https://stackoverflow.com
- Pygame documentations and libraries: https://www.pygame.org/docs/
- Creating flowchart: https://lucidchart.com