BINUS UNIVERSITY
BINUS INTERNATIONAL

Assignment Cover Letter

(Individual Work)

Student Information:		Surname	Given Names	Student ID Number
	1.	**Salim**	**Mikhael**	**2301890245**

Course Code	**: COMP6056**	**Course Name**	**: Program Design Methods and Introduction to Programming**
Class	**: L1AC**	**Name of Lecturer(s)**	**: Ida Bagus Kerthyayana Manuaba**
Major	**: CS**		
Title of Assignment (if any)	: PongZ (The Pong Game)		
Type of Assignment	**: Final Project**		
Submission Pattern			
Due Date	**: 17-01-20**	**Submission Date**	**: 13-01-20**

The assignment should meet the below requirements.

1. Assignment (hard copy) is required to be submitted on clean paper, and (soft copy) as per lecturer's instructions.
2. Soft copy assignment also requires the signed (hardcopy) submission of this form, which automatically validates the softcopy submission.
3. The above information is complete and legible.
4. Compiled pages are firmly stapled.

5. Assignment has been copied (soft copy and hard copy) for each student ahead of the submission.

Plagiarism/Cheating

BiNus International seriously regards all forms of plagiarism, cheating and collusion as academic offenses which may result in severe penalties, including loss/drop of marks, course/class discontinuity and other possible penalties executed by the university. Please refer to the related course syllabus for further information.

Declaration of Originality

By signing this assignment, I understand, accept and consent to Binus International terms and policy on plagiarism. Herewith I declare that the work contained in this assignment is my own work and has not been submitted for the use of assessment in another course or class, except where this has been notified and accepted in advance.

Signature of Student: (Name of Student)
1. Brenda Spears

"PongZ (The Pong Game)"

Name : Mikhael Angeloputra Salim

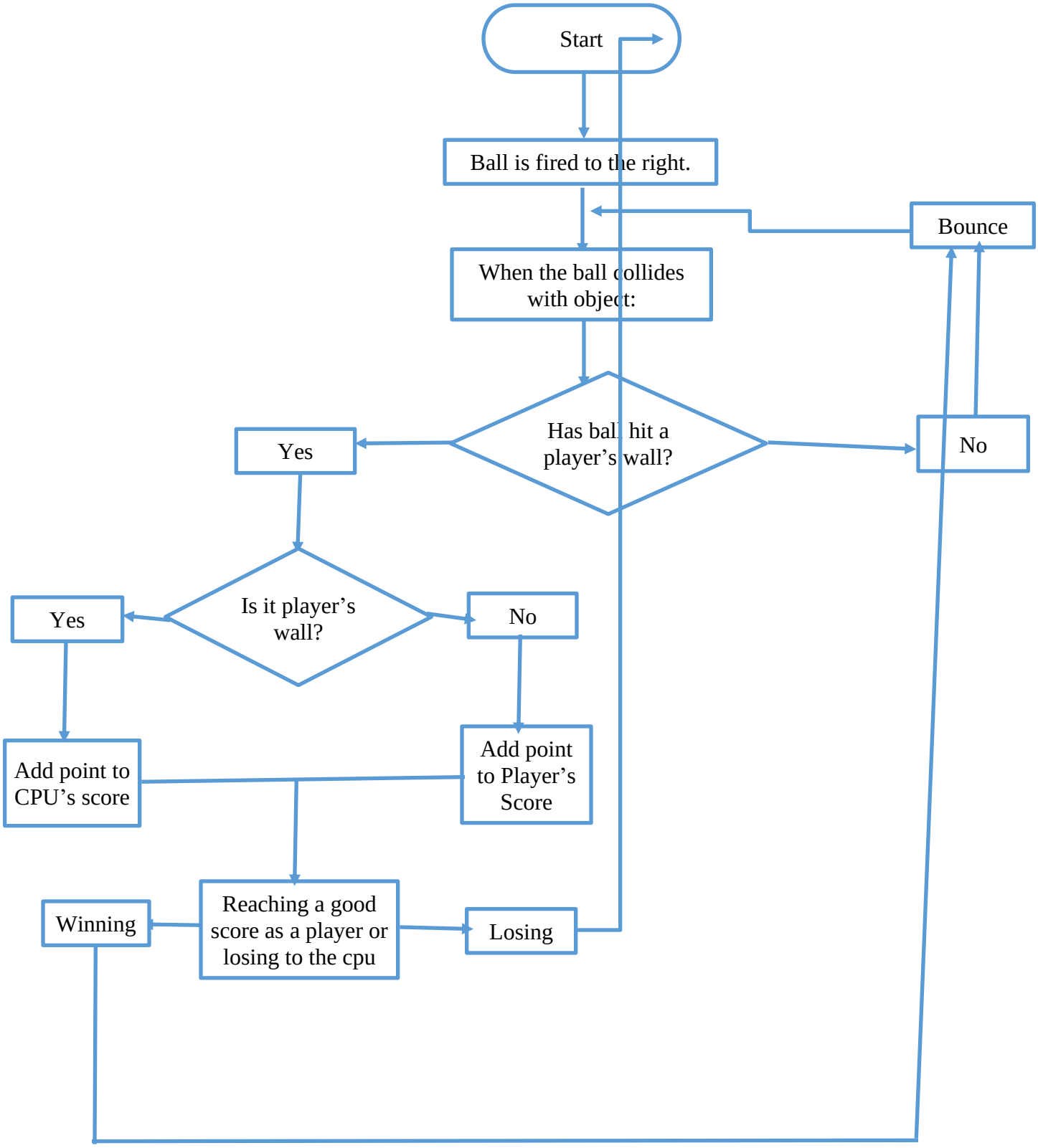
ID : 2301890245

I. Project Specifications

PongZ is a table tennis sports game featuring simple two-dimensional graphics and was made using pygame (a library), which is a Free and Open Source python programming language library for making multimedia applications like games built on top of the excellent SDL library. With some imports inside the game's code such as os, math, sys, time, and random to make the game capable of working with the addition of numerals, cpu, actual time inside the code. With this game people can have fun achieving any amount of points against the built in CPU and you can get several milestones when u reach a certain score against the CPU.

II. Solution Design

Flowchart



III. A Discussion about what is Implemented and How it works

PongZ is made using pygame which is a Free and Open Source python programming language library for making multimedia applications like games built on top of the excellent SDL library.

```
import pygame
```

Puts a limited set of constants and functions into the global namespace of your script.

```
from pygame import *
```

This will attempt to initialize all the pygame modules for you.

```
pygame.init()
```

Color variables that are used inside the game are based on the RGB table.

```
white = (255,255,255)#color variables u want inside the game
black = (0,0,0)
cyan = (0,255,255)
```

A function that displays text inside the game that is based on the coordinates, fontsize, and color that has been set inside the function. This works as the text for the milestones and scoreboard inside the game.

```
def displaytext(text,fontsize,x,y,color):
    font = pygame.font.SysFont('szerko', fontsize, True)
    text = font.render(text, 1, color)
    textpos = text.get_rect(centerx=x, centery=y)
    screen.blit(text, textpos)
```

A function that makes the cpu inside the game move with the ball and cpu parameter inside the function. The cpu moves based on the ball's movement and if the ball goes towards the cpu's paddle. It has the random movement added to make it so the cpu misses the ball every now and then.

```
def cpumove(ball,cpu):
    if ball.movement[0] > 0: #en-
sures that the cpu moves only when the ball is directed towards it
        if ball.rect.bottom > cpu.rect.bottom + cpu.rect.height/5:#add-
ing the cpu.rect.height/5 to make sure the cpu pad-
dle misses the ball every now and then or the game wouldnt be fun..
            cpu.movement[1] = 8
        elif ball.rect.top < cpu.rect.top - cpu.rect.height/5:
            cpu.movement[1] = -8
        else:
            cpu.movement[1] = 0
    else:
        cpu.movement[1] = 0
```

A class (2 classes, the ball class and the paddle class) that has the draw, checkbounds, update function and the init function to initialize some elements such as size, image, coordinates, movement, speed, points,and etc. The draw function is for making the object's image such as the paddle and the ball. The Checkbound function is for checking if the paddle/ball if they touch/reach the borderline of the window/game. The Update function is to update every state and position of the ball and the paddle made.

```
#a class for the player's paddle, and has the checkbound, draw, update func-
tion inside it.
class Paddle(pygame.sprite.Sprite):
    def __init__(self,x,y,sizex,sizey,color):#x,y for the coordi-
nates, sizes in each coordinate, color
        pygame.sprite.Sprite.__init__(self)
        self.image = pygame.Surface((sizex,sizey),SRCALPHA,32)#im-
age that we get from the pygame library
        self.rect = self.image.get_rect()#getting the image from the library
        self.image.fill(color)#fill the color inside the paddle
        self.image.convert_alpha()

        self.rect.left = x#coordinates
        self.rect.top = y



        self.color = color
        self.movement = [0,0]#initial movement
        self.speed = 8#speed of the paddle
        self.points = 0
    #A draw function which draws our paddle onto the screen
    def draw(self):
        screen.blit(self.image,self.rect)
    #An update function which updates the state and position of the paddle
    def update(self):
        self.rect = self.rect.move(self.movement)
        self.checkbounds()
    #A function which checks whether the paddle is go-
ing out of bounds and make corrections accordingly
    def checkbounds(self):
        if self.rect.top < 0:
            self.rect.top = 0
        if self.rect.bottom > height:
            self.rect.bottom = height
        if self.rect.left < 0:
            self.rect.left = 0
        if self.rect.right > width:
```

```
            self.rect.right = width
#same as the paddle class, its a class for the moving ball inside the game
class Ball(pygame.sprite.Sprite):
    def __init__(self,x,y,size,color,movement=[0,0]):#x,y is the coords,move-
ment of the ball,color,and size is the diameter of the ball
        pygame.sprite.Sprite.__init__(self)
        self.image = pygame.Surface((size,size),SRCALPHA,32)#same as paddle
        self.rect = self.image.get_rect()
        self.image.convert_alpha()
        self.rect.centerx = x
        self.rect.centery = y
        #pygame function that draws a circle with 5 parame-
ters in it (self,color,x,y,and size)
        pygame.draw.circle(self.im-
age,color,(int(self.rect.width/2),int(self.rect.height/2)),int(size/2))

        self.movement = movement
        self.maxspeed = 10 #max speed of the ball
        self.score = 0 #initial score
    #same as the one in paddle class
    def draw(self):
        screen.blit(self.image,self.rect)

    #this update function determine how the ball will move
    def update(self):
        if self.rect.top == 0 or self.rect.bottom == height:
            self.movement[1] = -1*self.movement[1]

        if self.rect.left == 0:#resets the ball's posi-
tion and notes that the point is scored by the CPU
            self.rect.centerx = width/2
            self.rect.centery = height/2
            self.movement = [random.randrange(-1,2,2)*4,random.randrange(-
1,2,2)*4]
            self.score = 1
        if self.rect.right == width:#resets the posi-
tion of the ball and notes that the point is scored by the user
            self.rect.centerx = width/2
            self.rect.centery = height/2
            self.movement = [random.randrange(-1,2,2)*4,random.randrange(-
1,2,2)*4]
            self.score = -1

        self.rect = self.rect.move(self.movement)
        self.checkbounds()
    #same as inside the paddle class
    def checkbounds(self):
```

```
        if self.rect.top < 0:
            self.rect.top = 0
        if self.rect.bottom > height:
            self.rect.bottom = height
        if self.rect.left < 0:
            self.rect.left = 0
        if self.rect.right > width:
            self.rect.right = width
```

Main Function of our program, inside the function has the sizes of the cpu, ball, paddle, as well as the main looping function inside the game. If its not game over the game will not stop playing until the players chooses to quit the game (Game over initial state is false). Inside this main function there is also milestones for the player if they reach a certain score or losing to the cpu. Theres also using pygame.sprite.collide_mask function to check for 'pixel perfect' collision between user's and cpu's paddle with the ball. The collision is based on the real world concept of perfectly elastic collisions. hence, whenever the ball strikes the paddle (placed vertically), the horizontal velocity is reversed, while the vertical velocity is equal to the relative velocity of ball w.r.t the paddle. In order to add some randomness, some error is introduced to the relative velocity of the ball and the paddle. The paddle's velocity is multiplied by a factor between 0.5 to 1 before calculating the relative velocity. This function also draws every classes and update and display text of every scoreboard etc. At the end there's also the pygame quit and quit function to quit the pygame and quit the program whenever we stop the game or close the program.

```
def main():
    game_over = False#sets the initial state of the game
    my_paddle = Pad-
dle(int(width/10),int(height/3),int(width/60),int(height*60/400),white)#creat-
ing the player's paddle(size,coord,color)
    cpu_paddle = Pad-
dle(int(width - width/10),int(height/3),int(width/60),int(height*60/400),white)#c
pu's paddle with the size,coords,and color of the paddle
    pongz_ball = Ball(width/2,height/2,width/60,cyan,[5,5])#creat-
ing the ball (size,coord,color)
    while not game_over:#running the game loop
        for event in pygame.event.get():
            if event.type == pygame.QUIT:#checks, if the user has clicked the clo
se button
                quit()#quits the game/program
            if event.type == pygame.KEYDOWN:#check whether the play-
ers presses the down key
                if event.key == pygame.K_UP:#if player pressed up key
```

```
                    my_paddle.movement[1] = -1*my_paddle.speed#paddle moves up-
wards with a certain speed
                elif event.key == pygame.K_DOWN:#if player pressed down key
                    my_paddle.movement[1] = my_paddle.speed#paddle moves down-
ward with a certain speed

            if event.type == pygame.KEYUP:#If the player press the up key

                    my_paddle.movement[1] = 0#paddle stops moving




        cpumove(pongz_ball,cpu_paddle)#moves the cpu paddle


        screen.fill(black)#fill the entire screen with black color

        #draws the player's paddle,cpu's paddle, and the ball
        my_paddle.draw()
        cpu_paddle.draw()
        pongz_ball.draw()

        #displaying the points scored by the player and cpu
        displaytext(str(my_paddle.points),20,width/8,25,(123,104,238))
        displaytext(str(cpu_paddle.points),20,width - width/8,25,(123,104,238))

        #using pygame.sprite.collide_mask function to check for 'pixel per-
fect' collision between user's and cpu's paddle with the ball

        if pygame.sprite.collide_mask(my_paddle,pongz_ball):
            pongz_ball.movement[0] = -1*pongz_ball.movement[0]
            pongz_ball.movement[1] = pongz_ball.movement[1] - int(0.1*ran-
dom.randrange(5,10)*my_paddle.movement[1])
            if pongz_ball.movement[1] > pongz_ball.maxspeed:
                pongz_ball.movement[1] = pongz_ball.maxspeed
            if pongz_ball.movement[1] < -1*pongz_ball.maxspeed:
                pongz_ball.movement[1] = -1*pongz_ball.maxspeed

        if pygame.sprite.collide_mask(cpu_paddle,pongz_ball):
            pongz_ball.movement[0] = -1*pongz_ball.movement[0]
            pongz_ball.movement[1] = pongz_ball.movement[1] - int(0.1*ran-
dom.randrange(5,10)*cpu_paddle.movement[1])
            if pongz_ball.movement[1] > pongz_ball.maxspeed:
                pongz_ball.movement[1] = pongz_ball.maxspeed
            if pongz_ball.movement[1] < -1*pongz_ball.maxspeed:
                pongz_ball.movement[1] = -1*pongz_ball.maxspeed
```

```
        #checks whether user or cpu scored a point and increments accordingly
        if pongz_ball.score == 1:
            cpu_paddle.points += 1
            pongz_ball.score = 0
        elif pongz_ball.score == -1:
            my_paddle.points += 1
            pongz_ball.score = 0

        if my_paddle.points == 15:#milestones for the game(players milestone)
            displaytext('Wow not bad!',50,width/2,height/3,(255,255,0))

        if my_paddle.points == 20:
            displaytext('Ur get-
ting good at this!',50,width/2,height/3,(255,0,255))

        if my_paddle.points == 35:
            displaytext('Wicked sick!',50,width/2,height/3,(128,0,128))

        if my_paddle.points == 100:
            displaytext('Ur actually a gamer',50,width/2,height/3,(255,192,203))

        if cpu_paddle.points == 5:#milestones for the game(cpus milestone)
            displaytext('Better try next time!',50,width/2,height/3,(255,255,0))
        if cpu_paddle.points == 10:
            displaytext('I think u should stop play-
ing.',50,width/2,height/3,(135,206,250))



        #updating the states of the player's paddle,cpu's paddle and the ball
        my_paddle.update()
        cpu_paddle.update()
        pongz_ball.update()

        #updating the entire display
        pygame.display.update()
        #adding the time delay based on the number of frames per second u set up
        clock.tick(FPS)
    #exiting the game by safely quitting pygame
    pygame.quit()
    quit()
```

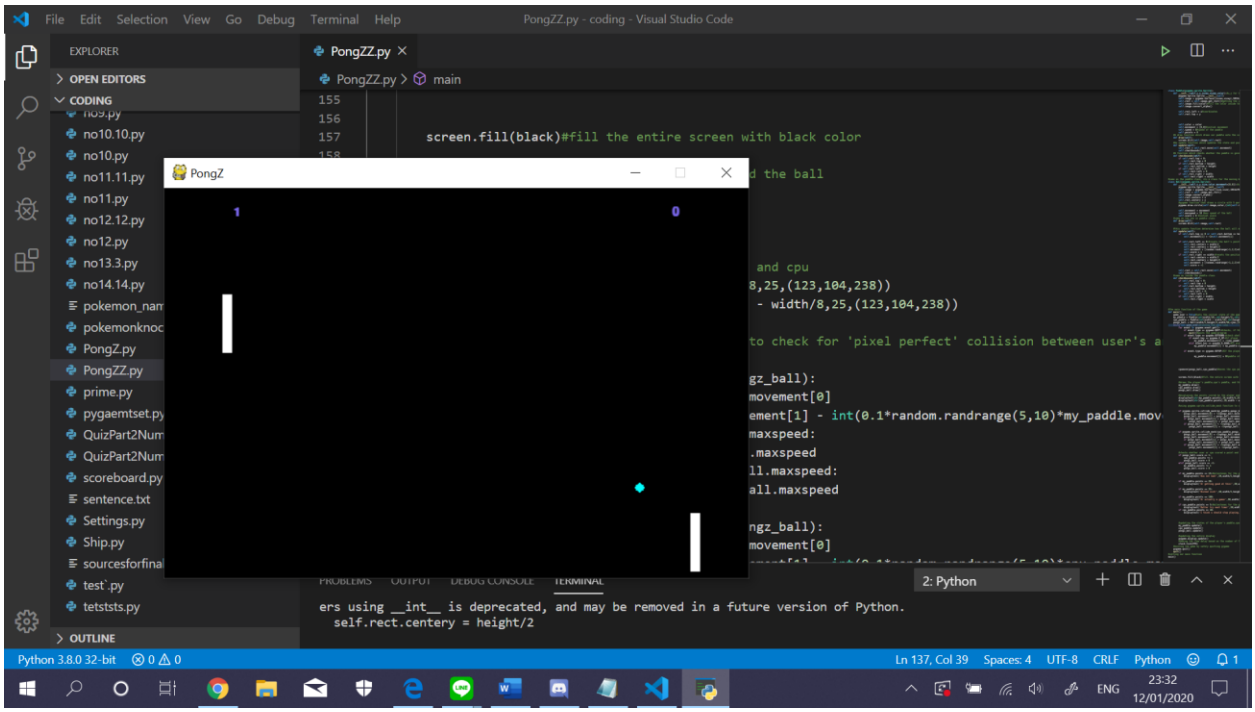
After that we call the main function to run the game.

```
main()
```

IVa. Problem that has been overcomed

When I first started my project, I didn't really know what to make or do as I was very much still a beginner when it comes to coding or programming. I browsed through the internet to see what was possible for me at my current skill level and looked up through youtube and watched some videos. Thankfully I did find something that was simple enough for me to understand and that I thought I could recreate. Whilst coding I did of course happen to run into some errors and after many trial and errors I finally finished the game with a relieved feeling. The errors that I came through is font errors, cpu not moving correctly, balls not resetting after touching cpu's wall and etc. I solved those problems by looking up through stackoverflow. I found many guides there and it was very helpful for solving errors inside my code.

IV. Evidence and Screenshots of the working app



V. Resources

https://stackoverflow.com/questions/46862739/sprite-mask-collision-problems-in-pygame(for sprite problems)
https://github.com/Udyam-Warp-Zone-Event/Pong/blob/master/Pong.py(background code with algorithms)
https://www.rapidtables.com/web/color/RGB_Color.html(for rgb colouring)
https://stackoverflow.com/questions/28517979/pygame-font-error(font error)
(Youtube video on how to make the pong game)

VI. Source Code

```
#im-
port
li-
brar-
ies
        import pygame
        import os  #so that you can access the functions and anything else defined within
        the module using the module name.
        import sys
        import time
        import math
        import random
        from pygame import *

        #initializing pygame
        pygame.init()#now use display and fonts

        #any size you want for the game(u can change the number anytime u want)
        screen_size = (width,height) = (600,400)

        #prefered fps u want for the game/ setting up the frames per second for the game
        FPS = 75
        white = (255,255,255)#color variables u want inside the game(source:
        https://www.rapidtables.com/web/color/RGB_Color.html)
        black = (0,0,0)
        cyan = (0,255,255)

        screen = pygame.display.set_mode(screen_size)#screen settings

        clock = pygame.time.Clock()#creating a clock object from pygame.time.Clock class

        pygame.display.set_caption('PongZ')#caption for the game/ name of the game shown on
        the window
        #a function to display the text on the screen with text, coordinates, fontsize and
        color parameter
        def displaytext(text,fontsize,x,y,color):
            font = pygame.font.SysFont('szerko', fontsize, True)#fontsize of the text dis-
        played
            text = font.render(text, 1, color)
            textpos = text.get_rect(centerx=x, centery=y)
            screen.blit(text, textpos)

        #a function to make the cpu paddle move
        def cpumove(ball,cpu):
```

```
    if ball.movement[0] > 0: #ensures that the cpu moves only when the ball is directed towards it
        if ball.rect.bottom > cpu.rect.bottom + cpu.rect.height/5:#adding the cpu.rect.height/5 to make sure the cpu paddle misses the ball every now and then or the game wouldnt be fun..
            cpu.movement[1] = 8
        elif ball.rect.top < cpu.rect.top - cpu.rect.height/5:
            cpu.movement[1] = -8
        else:
            cpu.movement[1] = 0
    else:
        cpu.movement[1] = 0

#a class for the player's paddle, and has the checkbound, draw, update function inside it.
class Paddle(pygame.sprite.Sprite): #to access the available sprites(as in a computer graphics term for any object on the screen that can move around.) inside the pygame library
    def __init__(self,x,y,sizex,sizey,color):#x,y for the coordinates, sizes in each coordinate, color
        pygame.sprite.Sprite.__init__(self)
        self.image = pygame.Surface((sizex,sizey),SRCALPHA,32)#image that we get from the pygame library
        self.rect = self.image.get_rect()#getting the image from the library
        self.image.fill(color)#fill the color inside the paddle
        self.image.convert_alpha()

        self.rect.left = x#coordinates
        self.rect.top = y



        self.color = color
        self.movement = [0,0]#initial movement
        self.speed = 8#speed of the paddle
        self.points = 0
    #A draw function which draws our paddle onto the screen
    def draw(self):
        screen.blit(self.image,self.rect)
    #An update function which updates the state and position of the paddle
    def update(self):
        self.rect = self.rect.move(self.movement)
        self.checkbounds()
```

```
    #A function which checks whether the paddle is going out of bounds and make
corrections accordingly(in each perspective directions(top,bottom,left,right)
    def checkbounds(self):
        if self.rect.top < 0:
            self.rect.top = 0
        if self.rect.bottom > height:
            self.rect.bottom = height
        if self.rect.left < 0:
            self.rect.left = 0
        if self.rect.right > width:
            self.rect.right = width
#same as the paddle class, its a class for the moving ball inside the game
class Ball(pygame.sprite.Sprite):
    def __init__(self,x,y,size,color,movement=[0,0]):#x,y is the coords,movement of
the ball,color,and size is the diameter of the ball
        pygame.sprite.Sprite.__init__(self)
        self.image = pygame.Surface((size,size),SRCALPHA,32)#same as paddle class
        self.rect = self.image.get_rect()
        self.image.convert_alpha()
        self.rect.centerx = x
        self.rect.centery = y
        #pygame function that draws a circle with 5 parameters in it
(self,color,x,y,and size)
        pygame.draw.circle(self.im-
age,color,(int(self.rect.width/2),int(self.rect.height/2)),int(size/2))

        self.movement = movement
        self.maxspeed = 10 #max speed of the ball
        self.score = 0 #initial score
    #same as the one in paddle class
    def draw(self):
        screen.blit(self.image,self.rect)

    #this update function determine how the ball will move
    def update(self):
        if self.rect.top == 0 or self.rect.bottom == height:
            self.movement[1] = -1*self.movement[1]

        if self.rect.left == 0:#resets the ball's position and notes that the point
is scored by the CPU
            self.rect.centerx = width/2
            self.rect.centery = height/2
            self.movement = [random.randrange(-1,2,2)*4,random.randrange(-1,2,2)*4]
            self.score = 1
```

```
        if self.rect.right == width:#resets the position of the ball and notes that
the point is scored by the user
            self.rect.centerx = width/2
            self.rect.centery = height/2
            self.movement = [random.randrange(-1,2,2)*4,random.randrange(-1,2,2)*4]
            self.score = -1

        self.rect = self.rect.move(self.movement)
        self.checkbounds()
    #same as inside the paddle class
    def checkbounds(self):
        if self.rect.top < 0:
            self.rect.top = 0
        if self.rect.bottom > height:
            self.rect.bottom = height
        if self.rect.left < 0:
            self.rect.left = 0
        if self.rect.right > width:
            self.rect.right = width



#the main function of the game
def main():
    game_over = False#sets the initial state of the game
    my_paddle = Pad-
dle(int(width/10),int(height/3),int(width/60),int(height*60/400),white)#creating
the player's paddle(size,coord,color)
    cpu_paddle = Paddle(int(width -
width/10),int(height/3),int(width/60),int(height*60/400),white)#cpu's paddle with
the size,coords,and color of the paddle
    pongz_ball = Ball(width/2,height/2,width/60,cyan,[5,5])#creating the ball
(size,coord,color)
    while not game_over:#running the game loop
        for event in pygame.event.get():
            if event.type == pygame.QUIT:#checks, if the user has clicked the close
button
                quit()#quits the game/program
            if event.type == pygame.KEYDOWN:#check whether the players presses the
down key
                if event.key == pygame.K_UP:#if player pressed up key
                    my_paddle.movement[1] = -1*my_paddle.speed#paddle moves upwards
with a certain speed
                elif event.key == pygame.K_DOWN:#if player pressed down key
```

```
                    my_paddle.movement[1] = my_paddle.speed#paddle moves downward
with a certain speed

            if event.type == pygame.KEYUP:#If the player press the up key

                    my_paddle.movement[1] = 0#paddle stops moving



        cpumove(pongz_ball,cpu_paddle)#moves the cpu paddle


        screen.fill(black)#fill the entire screen with black color

        #draws the player's paddle,cpu's paddle, and the ball
        my_paddle.draw()
        cpu_paddle.draw()
        pongz_ball.draw()

        #displaying the points scored by the player and cpu
        displaytext(str(my_paddle.points),20,width/8,25,(123,104,238))
        displaytext(str(cpu_paddle.points),20,width - width/8,25,(123,104,238))

        #using pygame.sprite.collide_mask function to check for 'pixel perfect'
collision between user's and cpu's paddle with the ball

        if pygame.sprite.collide_mask(my_paddle,pongz_ball):
            pongz_ball.movement[0] = -1*pongz_ball.movement[0]
            pongz_ball.movement[1] = pongz_ball.movement[1] - int(0.1*ran-
dom.randrange(5,10)*my_paddle.movement[1])
            if pongz_ball.movement[1] > pongz_ball.maxspeed:
                pongz_ball.movement[1] = pongz_ball.maxspeed
            if pongz_ball.movement[1] < -1*pongz_ball.maxspeed:
                pongz_ball.movement[1] = -1*pongz_ball.maxspeed

        if pygame.sprite.collide_mask(cpu_paddle,pongz_ball):
            pongz_ball.movement[0] = -1*pongz_ball.movement[0]
            pongz_ball.movement[1] = pongz_ball.movement[1] - int(0.1*ran-
dom.randrange(5,10)*cpu_paddle.movement[1])
            if pongz_ball.movement[1] > pongz_ball.maxspeed:
                pongz_ball.movement[1] = pongz_ball.maxspeed
            if pongz_ball.movement[1] < -1*pongz_ball.maxspeed:
                pongz_ball.movement[1] = -1*pongz_ball.maxspeed
```

```
        #checks whether user or cpu scored a point and increments accordingly
        if pongz_ball.score == 1:
            cpu_paddle.points += 1
            pongz_ball.score = 0
        elif pongz_ball.score == -1:
            my_paddle.points += 1
            pongz_ball.score = 0

        if my_paddle.points == 15:#milestones for the game(players milestone)
            displaytext('Wow not bad!',50,width/2,height/3,(255,255,0))

        if my_paddle.points == 20:
            displaytext('Ur getting good at this!',50,width/2,height/3,(255,0,255))

        if my_paddle.points == 35:
            displaytext('Wicked sick!',50,width/2,height/3,(128,0,128))

        if my_paddle.points == 100:
            displaytext('Ur actually a gamer',50,width/2,height/3,(255,192,203))

        if cpu_paddle.points == 5:#milestones for the game(cpus milestone)
            displaytext('Better try next time!',50,width/2,height/3,(255,255,0))
        if cpu_paddle.points == 10:
            displaytext('I think u should stop play-
ing.',50,width/2,height/3,(135,206,250))



        #updating the states of the player's paddle,cpu's paddle and the ball
        my_paddle.update()
        cpu_paddle.update()
        pongz_ball.update()

        #updating the entire display
        pygame.display.update()
        #adding the time delay based on the number of frames per second u set up
        clock.tick(FPS)
    #exiting the game by safely quitting pygame
    pygame.quit()
    quit()
#calling our main function
main()
```