**Chapter 1: Introduction**

Like most other sports, table tennis had humble beginnings as a “parlor game,” open to anyone with access to a table, paddle, and ball. The game began in the 1880s, when lawn tennis players adapted their game to play indoors during the winter.

Ping-Pong is a trademark name for table tennis and associated equipment. The name “Ping-Pong” was invented by the English firm J. Jaques and Son at the end of the 1800s and later trademarked in the United States by Parker Brothers, the board game company.

Table tennis/Pong has a simple set of rules where each player often as a single or double form a team and play around a rectangle table. Each are given a set of small rackets much similar to long tennis ones but these are made of wood. The game starts with each side using the racket to hit the small plastic ball outside of the opposing team’s court i.e. the opposing team fails to hit the ball back.

Today the pong game is digitalized in our systems as a game of interest played by millions of player world while. The pong game has been converted in 2-D and 3-D formats as well as played on the VR’s and T.V. video games with remotes.

**Chapter 2: Requirements**

**2.1 Hardware Requirements:**

1. Minimum Requirments: Windows 7, 2 Gb Ram ,500 MB Rom

**2.2 Software Requirements:**

1. Python Software version: 3.7 (64 bit/32 bit)

2. Pygame Installed in Python

**Chapter 3: Proposed Methodology & Status of Work**

**3.1 Flowcharts:-**

**🡪colors.py(): 🡪directions.py.py():**

Import enum

Create class directions(enum)

{

UP\_LEFT = 7

UP\_RIGHT = 9

DOWN\_LEFT = 1

DOWN\_RIGHT = 3

LEFT = 4

}

}

RIGHT = 6

Set Colors For Various Different Objects in game.

BLACK = (12, 45, 97)

WHITE = (25, 25, 24)

RED = (255, 122, 0)

BLUE = (0, 45, 255)

GRAY = (64, 64, 64)

**🡪Flowchart for player.py():**

@Score.Settler

Def score-increment score

@Property

Def Score-Return SCore

Player class:

Create constructor to initialize side, points and name.

**🡪Flowchart for ball.py():**

Set functions for four diagonal corners of the rectangle so that it knows where ball goes out of bounds.

Update function so that when ball hit edges, it knows and updates the directions where the ball can go.

import pygame

import random

import colors

import directions

Toogle –Direction function so that the ball can go in any of the directions specified in update function.

**Class Ball**

Def init:

**1.** Super.\_\_init\_\_()

**2.** Get width, height, direction as a random choice , screen size.

**3.** Fill color of ball, background color.

**4.** Set staring point, hit ad speedup of ball.

Def rectangle: Draw Rectangle

Def hit: When hit is 1, speedup=(1+hit/10)

@property

Set original positon of ball as the center of rectangle.

@position.setter

When ball goes out of bounds, increase corresponding score and set ball to original positon.

**🡪Flowchart for racket.py():**

import pygame

import colors

import directions

**Class Racket**

Def init:

**1.** Super.\_\_init\_\_()

**2.**  Set height and width of the rackets.

**3.** Fill color of racket

**4.** Place rackets on the corresponding sides of the rectangle as per the size of rectangle.

@property

Set original positon of rackets as the center of rectangle.

@position.setter

Set functions for when the rackets move up and down so that we change the position.

**🡪Flowchart for main.py():**

Set keyboard keys for the rackets to move up and down.

Also call the ball function to move the ball.

import pygame

from colors import \*

from ball import Ball

from racket import Racket

from directions import Directions

from player import Player

Printing of scores between game and at end screen.

Create clock using Pygame.time.clock()

At end game screen close clock,Pygame and quit Pygame.

Set Window height and width, display clock and screen and set max score

call other modules as well as set right and left players and the ball.

Def Gameover():

{

code for who won the game and display the won game screen

}

**3.2 Algorithm:**

1. Import Pygame, colours , ball , racket , directions ,player

2. Create clock using Pygame.

3. Set screen size and fps.

4. Get directions for left and right player and racket.

5. Get position for ball.

6. Set a game over screen when a player reaches max score.

7. Until not a game over screen do:

A . Fill screen, draw ball racket, scoreboard and fill them too.

B. Set keyboard keys to map to the racket movements.

C. Update score at every point until no one wins.

8.End

**Directions.py**

Set directions for up left, upright, down left, downright and right and left

**Colors.py**

Set colours for various objects on screen.

**Ball.py**

1. Import Pygame , random , colours and direction.

2. Set ball width, initial position colour and speedup and hits.

3. Draw ball on screen and speedup on hit methods.

4. Random function to decide next side the ball moves as per the direction of hit with racket or boundary.

5. Function for resetting position of ball when a point is scored.

**Racket.py**

1. Import Pygame, colours and directions.

2. Set racket height, width, initial position, offset from border, speedup.

3. Functions to move racket up or down by specified positions when keyboard button pressed for same.

**Player.py**

1. Set sides, scores =0 initially.

2. Set a score returning function and a function to calculate scores.

**3.3 Program:**

Program:-

**#Main.py()**

import pygame

from colors import \*

from ball import Ball

from racket import Racket

from directions import Directions

from player import Player

clock = pygame.time.Clock()

WIN\_WIDTH = 800

WIN\_HEIGHT = 640

MAX\_SCORE = 5

DISPLAY = (WIN\_WIDTH, WIN\_HEIGHT)

pygame.init()

clock = pygame.time.Clock()

screen = pygame.display.set\_mode(DISPLAY, 0, 32)

DONE = False

FPS = 30

left\_player = Player(Directions.LEFT, 'Left')

right\_player = Player(Directions.RIGHT, 'Right')

curr\_ball = Ball(screen, WIN\_WIDTH, WIN\_HEIGHT)

left\_racket = Racket(screen, WIN\_WIDTH, WIN\_HEIGHT, Directions.LEFT)

right\_racket = Racket(screen, WIN\_WIDTH, WIN\_HEIGHT, Directions.RIGHT)

rackets = pygame.sprite.Group()

rackets.add(left\_racket)

rackets.add(right\_racket)

stuff\_to\_draw = pygame.sprite.Group()

stuff\_to\_draw.add(left\_racket)

stuff\_to\_draw.add(right\_racket)

def game\_over(screen, winner, left\_paper, right\_player):

gray\_overlay = pygame.Surface((WIN\_WIDTH, WIN\_HEIGHT))

gray\_overlay.fill(GRAY)

gray\_overlay.set\_colorkey(GRAY)

pygame.draw.rect(gray\_overlay, BLACK, [0, 0, WIN\_WIDTH, WIN\_HEIGHT])

gray\_overlay.set\_alpha(99)

screen.blit(gray\_overlay, (0, 0))

font = pygame.font.SysFont(None, 100)

game\_over = font.render('{} Player WINS!'.format(winner.name), True, WHITE)

screen.blit(game\_over, (WIN\_WIDTH / 2 - 300, WIN\_HEIGHT / 2 - 100))

scoreline = font.render(

'{} - {}'.format(left\_paper.score, right\_player.score), True, WHITE)

screen.blit(scoreline, (WIN\_WIDTH / 2 - 50, WIN\_HEIGHT / 2 + 100))

pygame.display.update()

pygame.time.delay(2000)

while not DONE:

screen.fill(BLACK)

for event in pygame.event.get():

if event.type == pygame.QUIT:

DONE = True

pygame.event.pump()

keys = pygame.key.get\_pressed()

if keys[pygame.K\_q]:

DONE = True

if keys[pygame.K\_UP]:

right\_racket.move\_up()

if keys[pygame.K\_DOWN]:

right\_racket.move\_down()

if keys[pygame.K\_w]:

left\_racket.move\_up()

if keys[pygame.K\_s]:

left\_racket.move\_down()

stuff\_to\_draw.update()

curr\_ball.update()

col\_left, col\_right = curr\_ball.rect.colliderect(left\_racket.rect), curr\_ball.rect.colliderect(right\_racket.rect)

if col\_right == 1 or col\_left == 1:

curr\_ball.toggle\_direction()

curr\_ball.hit()

if curr\_ball.get\_x\_val() <= 0: # left border

right\_player.score = 1

curr\_ball = Ball(screen, WIN\_WIDTH, WIN\_HEIGHT)

elif curr\_ball.get\_x\_val() >= WIN\_WIDTH: # right border

left\_player.score = 1

curr\_ball = Ball(screen, WIN\_WIDTH, WIN\_HEIGHT)

# Print scores

font = pygame.font.SysFont('Helvetica', 25)

left\_player\_score = font.render(

'{}'.format(left\_player.score), True, (255, 255, 255))

right\_player\_score = font.render(

'{}'.format(right\_player.score), True, (255, 255, 255))

goal\_text = font.render(

'{}'.format(MAX\_SCORE), True, (255, 255, 0))

screen.blit(left\_player\_score, (WIN\_WIDTH / 2 - 100, 10))

screen.blit(right\_player\_score, (WIN\_WIDTH / 2 + 100, 10))

screen.blit(goal\_text, (WIN\_WIDTH / 2, 0))

stuff\_to\_draw.draw(screen)

curr\_ball.draw(screen)

if left\_player.score >= MAX\_SCORE:

game\_over(screen, left\_player, left\_player, right\_player)

elif right\_player.score >= MAX\_SCORE:

game\_over(screen, right\_player, left\_player, right\_player)

if left\_player.score >= MAX\_SCORE or right\_player.score >= MAX\_SCORE:

DONE = True

pygame.display.set\_caption('Ping Pong '+ str(clock.get\_fps()))

pygame.display.flip()

clock.tick(FPS)

pygame.quit()

**#Directions.py**

from enum import Enum

class Directions(Enum):

UP\_LEFT = 7

UP\_RIGHT = 9

DOWN\_LEFT = 1

DOWN\_RIGHT = 3

LEFT = 4

RIGHT = 6

**#Colors.py**

BLACK = (12, 45, 97)

WHITE = (25, 25, 24)

RED = (255, 122, 0)

BLUE = (0, 45, 255)

GRAY = (64, 64, 64)

**Ball.py**

import pygame

import random

from colors import \*

from directions import \*

class Ball(pygame.sprite.Sprite):

def \_\_init\_\_(self, screen, width, height):

super().\_\_init\_\_()

self.width, self.height = width, height

self.direction = random.choice([Directions.DOWN\_LEFT, Directions.DOWN\_RIGHT, Directions.UP\_LEFT, Directions.UP\_RIGHT])

self.screen = screen

self.image = pygame.Surface([10, 10])

self.image.fill(WHITE)

pygame.draw.rect(self.image, WHITE, [0, 0, 10, 10])

self.rect = self.image.get\_rect()

self.position = (width / 2 + 2, height / + 2)

self.hits = 0

self.speed\_up = 1.0

def draw(self, screen):

screen.blit(self.image, self.rect)

def hit(self):

self.hits += 1

self.speed\_up = 1.0+self.hits/10

@property

def position(self):

return (self.rect.x, self.rect.y)

@position.setter

def position(self, pos):

try:

pos\_x, pos\_y = pos

except ValueError:

raise ValueError("Pass an iterable with two items")

else:

self.rect.x, self.rect.y = pos\_x, pos\_y

def up\_left(self):

self.position = (self.position[0] - 10\*self.speed\_up, self.position[1] - 10\*self.speed\_up)

def up\_right(self):

self.position = (self.position[0] + 10\*self.speed\_up, self.position[1] - 10\*self.speed\_up)

def down\_left(self):

self.position = (self.position[0] - 10\*self.speed\_up, self.position[1] + 10\*self.speed\_up)

def down\_right(self):

self.position = (self.position[0] + 10\*self.speed\_up, self.position[1] + 10\*self.speed\_up)

def update(self):

if self.position[1] <= 10: # upper border

self.direction = random.choice(

[Directions.DOWN\_LEFT, Directions.DOWN\_RIGHT])

if self.position[1] >= self.height - 10: # bottom border

self.direction = random.choice(

[Directions.UP\_LEFT, Directions.UP\_RIGHT])

options = {Directions.UP\_LEFT: self.up\_left,

Directions.UP\_RIGHT: self.up\_right,

Directions.DOWN\_LEFT: self.down\_left,

Directions.DOWN\_RIGHT: self.down\_right,

}

options[self.direction]()

def toggle\_direction(self):

if self.direction == Directions.DOWN\_LEFT:

new\_direction = Directions.DOWN\_RIGHT

if self.direction == Directions.DOWN\_RIGHT:

new\_direction = Directions.DOWN\_LEFT

if self.direction == Directions.UP\_RIGHT:

new\_direction = Directions.UP\_LEFT

if self.direction == Directions.UP\_LEFT:

new\_direction = Directions.UP\_RIGHT

try:

self.direction = new\_direction

except NameError:

pass

def get\_x\_val(self):

return self.rect.x

**#Player.py:**

class Player():

def \_\_init\_\_(self, side, name):

self.side = side

self.points = 0

self.name = name

@property

def score(self):

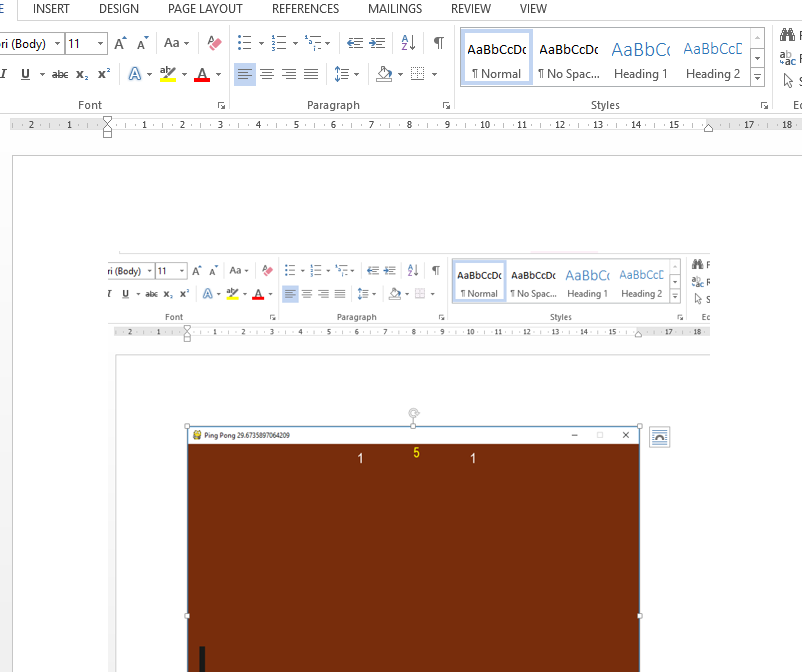
return self.points

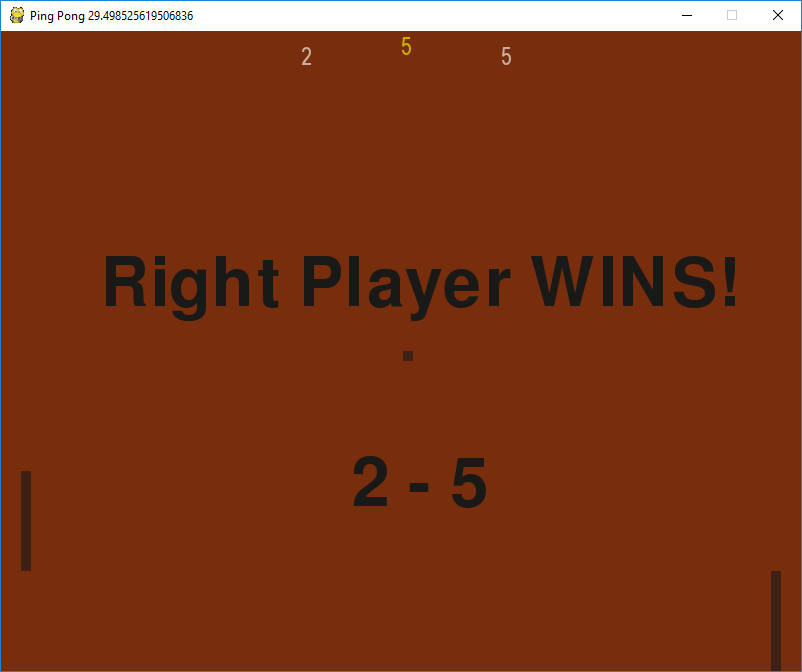
@score.setter

def score(self, val):

self.points += val

**3.4 Output:**

****

****

**`**

**3.5 Conclusion:**

Thus using the Pygame module in python we have successfully created the famous “Pong Game” as the mini project.

**3.5 Future Scope:**

These days Gaming in itself is a separate industry due to the demand; the future prospects are very bright for the deserving candidates. To become a Game developer, knowledge is an important aspect along with which one requires to have good creative skills and innovation. Encourage your student to play a fun game or two of pong or classically known as table tennis for a fun and friendly competition as well as show them the hard work behind creating a successful game and the amount of dedication it needs to be a game developer as well as have fun seeing the smiling faces of the people who play your game.