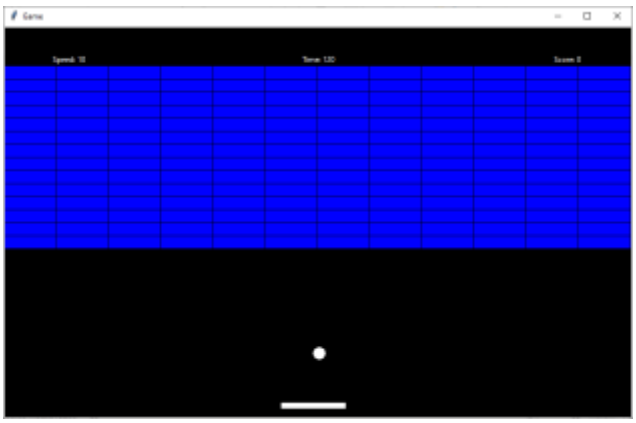
SOMAIYA SCHOOLS
The Somaiya School

The Somaiya School

Academic Year 2020-2021

A Project Report On

Brick Breaker



Submitted By:

ABHIJEET MANKANI

Grade: XI Science

Roll No.: 10

Gr. No.: 064

SOMAIYA SCHOOLS
The Somaiya School

The Somaiya School

Certificate

This is to certify that **ABHIJEET MANKANI**, student of standard **XI Science** has successfully completed the **Computer Science Project Work** in partial fulfillment of curriculum of the **CENTRAL BOARD OF SECONDARY EDUCATION** for the year **2020-2021**.

Ms. Anuja Ajay
Teacher Incharge

Ms. Parveen Shaikh
Principal

Date:

School Seal

Acknowledgement

During the process of preparation of the project, I have got immense help from various persons, without which it would not have been possible to achieve this goal.

First of all, I would like to place on record the effort that our respected Principal has made in providing us fully equipped laboratory.

Our Computer Teacher has rendered all support and guidance from time to time to see that I am thorough to the project.

We are equally thankful to our parents for giving us moral support and ideas to carve out this project.

Table Of Contents

1. Introduction

This is the Recreation of the Retro game Brick Breaker in Python.
The User Controls the paddle with his/her mouse to try and break as many bricks as possible with the ball in the given time (2 minutes).

The ball bounces off the walls, the ceiling, the paddle, and bricks.
By bouncing on each brick, the brick is destroyed, and the player gains 2 points.

If the player fails to hit the ball with his paddle, the ball falls below, and the user loses 5 points.

If the user clears ~95%(162 of 168 bricks) of the bricks, the bricks are reset, and the user gains a bonus of 50 points.

The maximum number of bricks on the screen is 168.

To add an extra dimension, the player can control the speed of the ball, to take a risk of sorts;

- By increasing the speed, he can hit more bricks in the given time, but make it more likely to lose points. (maximum speed cap is 20)
- By decreasing the speed, he can make it less likely to lose points, but he can hit lesser bricks in the given time. (minimum speed cap is 5)

Help:
Paddle is controlled by using the mouse/pointing device.
Speed is increased using ‘F’ key
Speed is decreased using ‘S’ key

2. Package/Module Used In The Project

1. math module: Gives access to various mathematical function in python functions:
   - floor – It rounds down the number to the Greatest Integer Smaller than or equal to the passed Argument
2. time module: Gives access to various time-related functions in python functions:
   - perf_counter – Returns Time (in seconds) since the program started
   - sleep – Stops the program execution for a specific number of seconds
3. Tkinter module: One of the Python Standard Libraries, used for adding GUI support to programs:
   - Tk – Creates a root for the GUI
   - Canvas – creates a canvas, to show stuff on
4. keyboard module: Allows Access to keyboard related functions
   - wait – Stops Program Execution until the specified key is presses
   - is_pressed – Checks if the specified key is pressed
5. pyautogui module: A module which can help automate your computer, and navigate through the GUI:
   - moveTo – moves the mouse cursor to the specifies coordinates

6. classes module: A user defined module that contains all the classes' structure in it

User Defined Functions:

start_game(): Starts the game

motion(): To get mouse's X Position – to be used only for binding with root.bind()

awake_ball(): Makes the ball move again, after being reset, or after the game starts – to be used only for binding with root.bind()

brickTileToIndex(): Converts Row-Column Location to Index

brickPresent(): Checks if brick's present value is 1(it is present) or not(it is removed)

resetBricks(): Resets the Brick wall

reset(): Rests the Ball, and Paddle Position, is called when ball falls down

removeAndBounceOffBrick(): Removes the brick that the ball collides with, and bounce off it

speed_increment(): You can decrease the speed of the ball, making it more likely to lose points, but possibly have a higher max score. – to be used only for binding with root.bind()

speed_decrement(): You can decrease the speed of the ball, making it less likely to lose points, but have a lower possible max score. – to be used only for binding with root.bind()

def move(): Defines the basic physical movement of ball, according to speed, bounces it off the walls and resets the ball when it goes below paddle

def call(): Place holder function, which is run every frame, to calculate current position of things

3. Coding

main.py :

```
from math import floor
from time import perf_counter, sleep
from tkinter import Canvas, Tk

import keyboard
from pyautogui import moveTo

from classes import *

FPS = 30

c_width, c_height = 963, 600

brickGap = 2
brickCol = 12
brickRow = 18
brickCounter = brickCol * 14
brickWidth, brickHeight = 80, 20

brickGrid = [None] * (brickCol * brickRow)

# Initializing The Tkinter canvas
root = Tk()
root.title("Game")
canvas = Canvas(root, width=c_width, height=c_height)
canvas['bg'] = '#000000'
canvas.pack()

font = ('Algerian', 30, 'bold italic')
start_text_box = canvas.create_text(c_width / 2,
                                    c_height / 2,
                                    fill="#FFFFFF",
                                    text="Press Enter to Sta
rt The Game",
                                    font=font)
```

```
game_started = False
continue_ = True


def start_game():
    canvas.delete(start_text_box)
    game_started = True


score = 0
score_text = "Score: " + str(score)
score_box = canvas.create_text(c_width - 100,
                               50,
                               fill="#FFFFFF",
                               text=score_text)
started = False
total_time = 120
time_left = total_time
time_text = "Time: " + str(time_left)
time_box = canvas.create_text(c_width / 2, 50, fill="#FFFFFF
", text=time_text)

# get screen width and height
ws = root.winfo_screenwidth()  # width of the screen
hs = root.winfo_screenheight()  # height of the screen

# calculate x and y coordinates for the Tk root window
x = (ws / 2) - (c_width / 2)
y = (hs / 2) - (c_height / 2)

# set the dimensions of the screen
# and where it is placed
root.geometry('%dx%d+%d+%d' % (c_width, c_height, x, y))

prev_mouse_x = mouse_x = c_width / 2

ball = Ball(c_width / 2 + 1, c_height - 100, 0, 10, canvas)
sleep_for_ball = True
paddle = Paddle(431.5, canvas)
```

```
speed_text = "Speed: " + str(ball.speedY)
speed_box = canvas.create_text(100, 50, fill="#FFFFFF", text
=speed_text)

canvas.update()


def motion(event):
    '''To get mouse's X Position'''
    global mouse_x
    mouse_x = event.x


root.bind('<Motion>', motion)


def awake_ball(event):
    '''Makes the ball move again, after being reset, or afte
r the game starts'''
    global sleep_for_ball, start_time, started, time_left, t
ime_box, info
    sleep_for_ball = False

    if (not (started) and game_started):
        started = True
        start_time = perf_counter()
        time_left = total_time
        time_text = "Time: " + str(time_left)
        canvas.itemconfigure(time_box, text=time_text)
    try:
        canvas.delete(info)
    except:
        pass


root.bind('<Button-1>', awake_ball)


def brickTileToIndex(TileCol, TileRow):
```

```
    '''Converts Row-Column Location to Index'''
    return (TileCol + brickCol * TileRow)


def brickPresent(TileCol, TileRow):
    '''Checks if brick's present value is 1(it is present) o
r not(it is removed)'''
    brickIndex = brickTileToIndex(TileCol, TileRow)
    return (brickGrid[brickIndex].present == 1)


def resetBricks():
    '''Resets The Brick wall '''
    global brickGrid, brickCounter
    for col in range(brickCol):
        for row in range(brickRow):
            index = brickTileToIndex(col, row)
            brickGrid[index] = Brick(col, row, 1, brickWidth
 * col,
                                     brickHeight * row, canv
as)
    for i in range(brickCol * 3):
        brickGrid[i].delete(canvas)  # remove the brick
    for i in range(brickCol * (brickRow - 1), brickCol * bri
ckRow):
        brickGrid[i].delete(canvas)  # remove the brick
    brickCounter = brickCol * brickRow


def reset():
    '''Rests The Ball, and Paddle Position, is called when b
all falls down'''
    global sleep_for_ball, score
    sleep_for_ball = True
    paddle.movement(canvas, (c_width / 2) - (paddle.width /
2) - paddle.x, 0)
    ball.movement(canvas, paddle.x + paddle.width / 2 + 1 -
ball.x,
                  c_height - 100 - ball.y)
```

```
    ball.speedX = 0
    moveTo(ws / 2, hs / 2)
    if (game_started):
        '''-5 score on ball falling down'''
        score -= 5
    score_text = "Score: " + str(score)
    canvas.itemconfigure(score_box, text=score_text)


resetBricks()


def removeAndBounceOffBrick():
    '''Removes the brick that the ball collides with, and bo
unce off it'''
    global ball, brickCounter, score, score_text, score_box
    Col = floor(ball.x / brickWidth)
    Row = floor(ball.y / brickHeight)

    if (Col < 0 or Col >= brickCol or Row < 0 or Row >= bric
kRow):
        return  # bail out of function to avoid illegal arra
y positioning usage error to occur
    else:
        brickIndex = brickTileToIndex(Col, Row)
        # so, we know the area we've overlaped has a brick p
resent and not already broken
        if (brickGrid[brickIndex].present == 1):
            prevBallX = ball.x - ball.speedX
            prevBallY = ball.y - ball.speedY
            prevCol = floor(prevBallX / brickWidth)
            prevRow = floor(prevBallY / brickHeight)

            BothTestsFailed = True

            if (prevCol != Col):  # Ball came in horizontall
y
                adjacentBrickIndex = brickTileToIndex(prevCo
l, Row)
```

```
                # make sure reflecting side is not blocked o
ff
                if (brickGrid[adjacentBrickIndex].present !=
 1):
                    BothTestsFailed = False
                    ball.speedX *= -1

            if (prevRow != Row):  # Ball came in vertically
                adjacentBrickIndex = brickTileToIndex(Col, p
revRow)
                # make sure reflecting side is not blocked o
ff
                if (brickGrid[adjacentBrickIndex].present !=
 1):
                    BothTestsFailed = False
                    ball.speedY *= -1
            if (BothTestsFailed):
                ball.speedX *= -1
                ball.speedY *= -1
            brickCounter -= 1
            score += 2
            # +2 Score for every brick broken
            score_text = "Score: " + str(score)
            canvas.itemconfigure(score_box, text=score_text)
            brickGrid[brickIndex].delete(canvas)  # remove t
he brick
            if (brickCounter <= 5):
                for i in brickGrid:
                    i.delete(canvas)
                resetBricks()
                score += 50
                # +50 Score on removing all bricks except 6
                score_text = "Score: " + str(score)
                canvas.itemconfigure(score_box, text=score_t
ext)


def speed_increment(event):
    '''You can decrease the speed of the ball, making
```

```
        it more likely to lose points, but possibly have a h
igher max score.'''
    if (sleep_for_ball):
        ball.speedY += 1
        ball.speedY = min(ball.speedY, ball.max_speed)
        speed_text = "Speed: " + str(ball.speedY)
        canvas.itemconfigure(speed_box, text=speed_text)
        canvas.update()


def speed_decrement(event):
    '''You can decrease the speed of the ball, making
        it less likely to lose points
but have a lower possible max score.'''
    if (sleep_for_ball):
        ball.speedY -= 1
        ball.speedY = max(ball.speedY, ball.min_speed)
        speed_text = "Speed: " + str(ball.speedY)
        canvas.itemconfigure(speed_box, text=speed_text)
        canvas.update()


root.bind('<F>', speed_increment)
root.bind('<S>', speed_decrement)
root.bind('<f>', speed_increment)
root.bind('<s>', speed_decrement)


def move():
    '''Defines the basic physical movement of ball, accordin
g to speed,
    bounces it off the walls and resets the ball when it goe
s below paddle'''
    global ball, brickCounter, paddle, mouse_x, prev_mouse_x
, sleep_for_ball, speed_text, info, continue_
    if (not (sleep_for_ball)):
        ball.movement(canvas, ball.speedX, ball.speedY)
        if (ball.x <= 10 or ball.x >= c_width - 10):
            ball.speedX *= -1
```

```
        if (ball.y <= 0):
            ball.speedY *= -1
        if (ball.y >= c_height - 20):
            if (ball.x >= paddle.x - paddle.width / 2 - 10
                    and ball.x <= paddle.x + paddle.width /
2 + 10):
                ball.speedY *= -1
                X = ball.x - (paddle.x)
                ball.speedX = X * 0.35
            else:
                reset()
        removeAndBounceOffBrick()
        continue_ = False
    elif (not continue_):
        info = canvas.create_text(
            c_width / 2,
            3 * c_height / 4,
            text="Left Click to Continue\nor Press F/S to ch
ange speed",
            font=font,
            justify="center",
            fill='#FFFFFF')
        root.update()
        continue_ = True

    paddle.movement(canvas, mouse_x - paddle.x, 0)
    prev_mouse_x = mouse_x


root.geometry(str(c_width) + 'x' + str(c_height))


def call():
    '''Place holder function, which is run every frame, to c
alculate current postion of things'''
    global time_left, time_box, started, game_started, start
_time, canvas
    if (game_started):
        move()
```

```
        curr_time = perf_counter()
        if (started and time_left > 0):
            time_left = round(total_time - (curr_time - star
t_time))
            time_text = "Time: " + str(time_left)
            canvas.itemconfigure(time_box, text=time_text)
        if (time_left <= 0):
            ball.speedY = 0
            ball.speedX = 0
            sleep_for_ball = True
            for i in brickGrid:
                i.delete(canvas)
                font = ('Algerian', 40, 'bold italic')
            canvas.create_text(c_width / 2, c_height / 2, te
xt="Your Final Score is " + str(score), font=font, justify="
center", fill="#FFFFFF")
            font = ('Algerian', 20, 'bold italic')
            canvas.create_text(c_width / 2, 4 * c_height / 5
, text="Press 'E' to exit", font=font, justify="center", fil
l="#FFFFFF")
    if (not (game_started)):
        # Runs one time, at the start of game, to give the u
ser instructions
        keyboard.wait('enter')
        canvas.delete(start_text_box)

        font = ('Algerian', 15, 'bold italic')
        info = canvas.create_text(c_width / 2, 3 * c_height
/ 4, text="Left Click to Start Moving", font=font, justify="
center", fill='#FFFFFF')
        root.update()
        sleep(2)
        canvas.itemconfigure(info, text="Move your cursor to
 control paddle")
        root.update()
        sleep(2)
```

```
        canvas.itemconfigure(info, text="You have 2 minutes
to score as many\npoints as possible")
        root.update()
        sleep(3)
        canvas.itemconfigure(info, text='''Press F or S to\n
increase/decrese you ball speed\n(You can only change the sp
eed while the ball is still)''')
        root.update()
        sleep(5)
        canvas.itemconfigure(info, text="Press E at any time
 to exit the program.")
        root.update()
        sleep(3)
        canvas.delete(info)
        root.update()
        reset()
        started = False
        game_started = True
        time_left = total_time


while 1:
    # The Clock which runs everything FPS frames per second
    call()
    root.update_idletasks()
    root.update()
    sleep(1 / FPS)

    if keyboard.is_pressed('E'):
        exit()
```

classes.py :

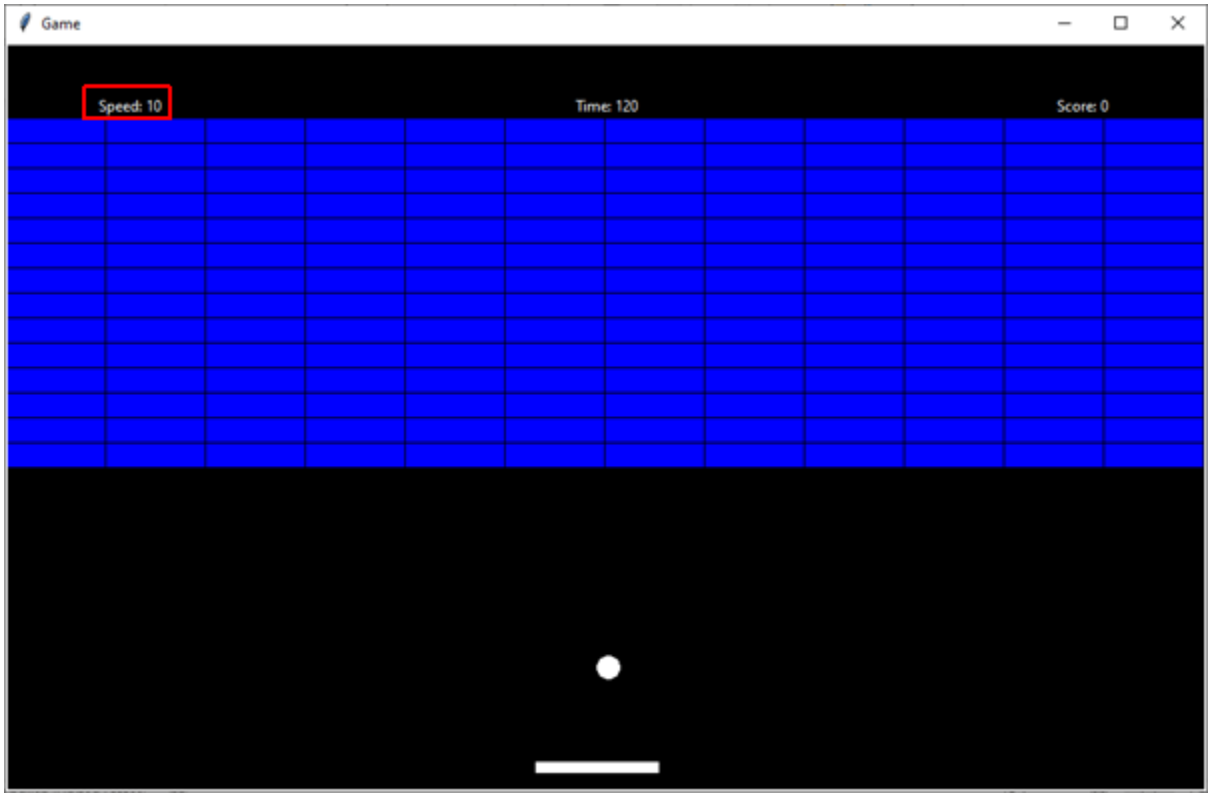
```
# Predefined Canvas Width and Height values
c_height = 600
c_width = 963


class Brick:
    def __init__(self, _col, _row, _present, leftX, topY, ca
nvas):
        self.col = _col
        self.row = _row

        self.width = 80
        self.height = 20
        self.present = _present
        self.id=canvas.create_rectangle(leftX, topY, leftX +
 self.width, topY + self.height, fill='#0000FF')

    def delete(self, canvas):
        '''Deletes the brick'''
        if(self.present):
            canvas.delete(self.id)
            self.present=0



class Paddle:
    def __init__(self, _x, canvas):
        self.x = _x
        self.y = c_height - 20

        self.width = 100
        self.height = 10

        self.id=canvas.create_rectangle(self.x-
self.width/2, self.y - self.height/2, self.x + self.width/2,
 self.y + self.height/2, fill='#FFFFFF')

    def movement(self, canvas, mov_x, mov_y):
```

```
        '''To move the paddle'''
        canvas.move(self.id, mov_x, mov_y)
        self.x+=mov_x
        self.y+=mov_y

class Ball:
    def __init__(self, _x, _y, _speedX, _speedY, canvas):
        self.x = _x
        self.y = _y

        self.r = 10

        self.speedX = _speedX
        self.speedY = _speedY

        self.max_speed = 20
        self.min_speed = 5

        self.id = canvas.create_oval(self.x - self.r, self.y
 - self.r, self.x + self.r, self.y + self.r, fill='#FFFFFF')

    def movement(self, canvas, mov_x, mov_y):
        '''To move the ball'''
        canvas.move(self.id, mov_x, mov_y)
        self.x+=mov_x
        self.y+=mov_y
```

Output:

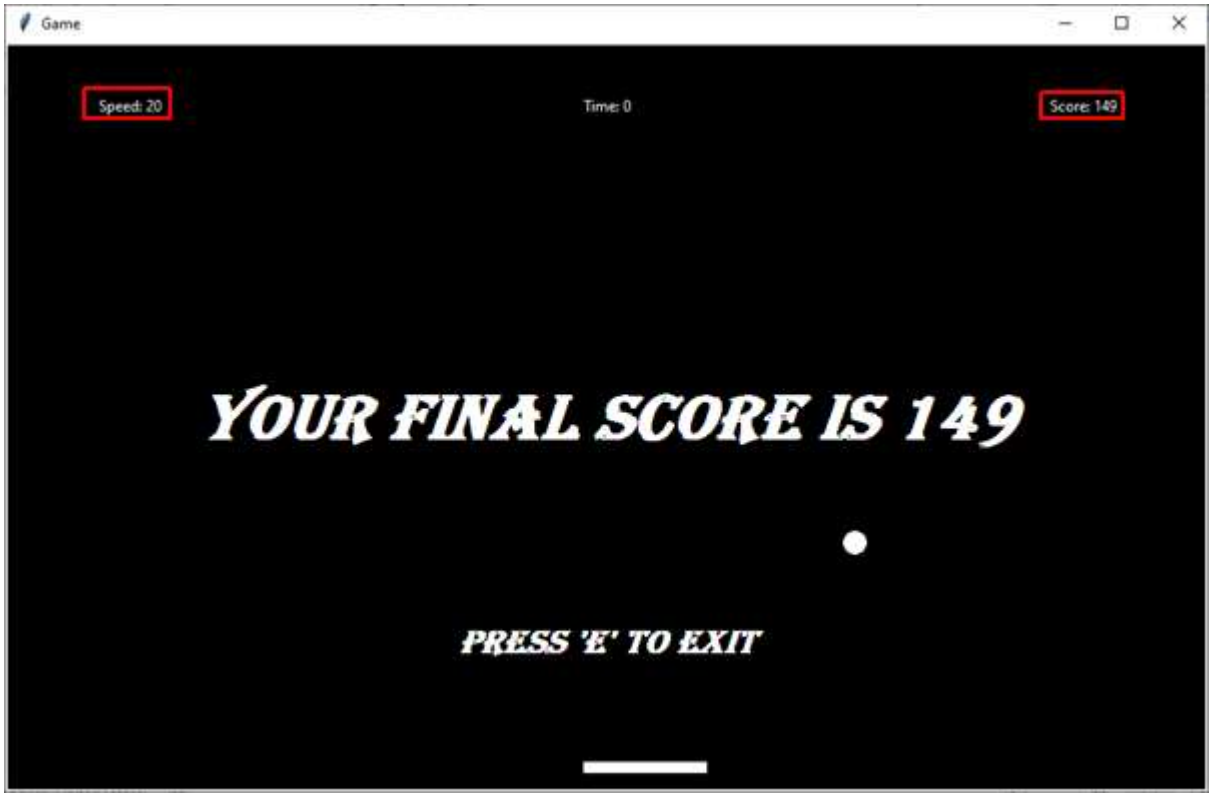
I. Screenshot 1

II. Screenshot 2



III. Screenshot 3

IV. Screenshot 4



4. Limitations

- Keyboard cannot be used to control the paddle
- Speed of the ball cannot be increases/decreased once a round has started

5. Requirements

To run this program, Python 3.0 or above must be installed on the system. Along with Python 3.0 or above, the following modules (other than standard modules) must be installed, if they are not already installed:

- keyboard module
  Installation: **pip install keyboard**
- pyautogui module
  Installation: **pip install pyautogui**

Operating System: any

A **mouse/pointing device** and **keyboard** are required to play the game.

6. Conclusion

Successfully implemented the Brick Breaker game, in Python 3.9.4

As a future scope, a DBMS or file system can be used to keep track of high scores records, as well as more game modes can be implemented into this.

7. Bibliography

- Stack Overflow: https://stackoverflow.com/
- Geeks-For-Geeks: https://www.geeksforgeeks.org/
- Tkinter Module Documentation : https://docs.python.org/3/library/tk.html
- Keyboard module Documentation : https://pypi.org/project/keyboard/