



NATIONAL PUBLIC SCHOOL

Kengeri, Bengaluru

Grade 12 Project 2022-23

PONG 2.0

Designed and Developed by:

ADITYA SWADI

ABHIRAM BHAT

SATHWIK BHARADWAJ

Certificate

This is to certify that **Aditya Swadi** of Grade 12-B has prepared the report for his project entitled '**Pong 2.0**'. The report is the result of his efforts and endeavours. The report is found to be worthy of acceptance as the final project report for the subject of Computer Science (083). The project has been successfully completed under the guidance of **Uma Prasad Peddamatham** during the academic session 2022-23 as per the guidelines issued by the **Central Board of Secondary Education**.

Computer Science Teacher

External Examiner

Head of Department

Co-Ordinator

Principal

School Seal

Acknowledgement

We would like to thank our Principal Madam Mrs. Geeta Dikshit for providing us with an opportunity to execute this project at the school and exhibit our talent.

We extend our heartfelt thanks to our Computer Science teacher Mrs. Uma Peddamatham for her encouragement and guidance throughout this project.

Finally, we would like to show our gratitude to our parents for their moral support and for being with us till the completion of this project.

Sincerely,

Aditya Swadi

Index

1. The Project Description

- a. What is Pong 2.0?
- b. About the project
- c. The design of the project
- d. Main Motive

2. Python, PyGame and Minimum Requirements

- a. About Python
- b. About PyGame
- c. System Requirements

3. Source Code

4. Screenshots

5. References

The Project Description

a. What is Pong 2.0?

Pong is one of the first computer games that was ever created, this simple "tennis-like" game features two paddles and a ball.

Pong 2.0 is a timed parody of Pong where 2 players control a red pencil and a blue pencil (the paddles) and bounce an eraser (ball) up and down towards their opponent. The game ends when the timer hits zero and the scores are entered into a csv file.

b. The Design of the Project

Pong 2.0 is a multiplayer game where one player controls the blue pencil, and the other player controls the red pencil. The 'a' key makes the blue pencil go left and the 'd' key makes the bluepencil go right. Similarly, the 'left arrow' key makes the red pencil go left and the 'right arrow'key makes the red pencil go right. The paddle speeds increase over time making the game harder as you continue playing against your opponent.

There is a ticking timer at the top middle part ofthe screen which displays the time left for the game to end. The game ends when the timer hits zero. After this, the scores are entered into the csv file 'MatchHistory.csv' which can be accessed separately.

c. Main Motive

The project game development was a truly valuable experience for the high school students involved. Not only did it allow us to showcase our knowledge of Python programming, but it also helped us develop a variety of other skills. We learned how to work as a team, how to troubleshoot and problem-solve, and how to think creatively and critically.

Additionally, the project provided us with the opportunity to apply our knowledge in a real-world context and see the results of our hard work firsthand.

Overall, the project game development was a helpful and engaging way for us to demonstrate our Python knowledge and skills to the wider community.

About Python

Python is commonly used for developing websites and software, task automation, data analysis and data visualization. Since python is relatively easy to learn, it has been adopted by many non-programmers such as accountants and scientists for a variety of everyday tasks like organising finances and more.

Guido van Rossum began working on Python in the late 1980s as a successor to the ABC programming language and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000 and introduced new features such as list comprehension, cycle-detecting garbage collection, reference counting and Unicode support. Python 3.0 was released in 2008.

- Python consistently ranks as one of the most popular programming languages.
- It is an interpretive, interactive, and object-oriented programming language. It incorporates modules, exceptions, dynamic typing, and very high-level dynamic data types and classes. It supports multiple programming paradigms beyond object-oriented programming, such as procedural and functional programming.

Why Python?

Python is the perfect language for multiple purposes, including

Web app development, Quick prototyping, Scripting, Data science and Database programming.

Python is a language embraced by a large community of coders for its many advantages and features. Many businesses choose python as the main programming language for reasons being:

- It's simple
- It's free
- It's easy to use
- It's highly compatible
- It's object-oriented
- It has a lot of libraries
- It has built-in data structures
- It's easy to learn
- It increases speed and productivity

About PyGame

PyGame is a set of Python modules designed for writing video games. PyGame adds functionality on top of the excellent SDL library. This allows you to create fully featured games and multimedia programs in the Python language. The main features of PyGame are:

- PyGame is extremely simple.
- PyGame is highly portable and runs on nearly every platform and operating system.
- PyGame is free.
- Released under the LGPL licence, you can create open-source, freeware, shareware, and commercial games with it
- Multi-core CPUs can be used easily.
- You control your main loop.
- Millions of people use it.

Minimum System Requirements

OS: 64-bit Windows 7 or later
OSX 10.11 or later

Processor: 1.5GHz or faster

Memory: 2GB (2,048MB)
RAM

Free HDD Space: 1GB

Hardware: Keyboard,
Mouse

Source Code

```
import pygame, csv
from pygame.locals import *
```

```
'''
```

This code imports three modules: pygame, math, and csv. Pygame is a library for creating video games with Python. It allows the user to create graphics, sounds, and control input and output.

CSV (Comma Separated Values) is a file format for storing data in a tabular form. It is commonly used for data export and import because it is easy to read and write. The csv module allows the user to read and write CSV files in Python.

The line "from pygame.locals import *" imports all the constants defined in the pygame.locals module. These constants include constants for key and mouse button codes, as well as flags for surface and window creation.

```
'''
```

```
N1 = input('Player Red, Enter your username: ')
N2 = input('Player Blue, Enter your username: ')
timePeriod = int(input('How long should the game last (in minutes)? ->
'))
timePeriod *= 60
```

```
'''
```

This code prompts the user to enter their usernames and how long they want to play the game for (in min). The first prompt is for the player who is identified as "Red" and the second prompt is for the player who is identified as "Blue". The input given by the user is then stored in the variables N1 and N2, respectively.

```
'''
```

```
pygame.init()
```

```
'''
```

pygame.init() is a function in the Pygame library that initialises all of the Pygame modules. This is necessary before any Pygame functions or features can be used in a program. It is usually called at the beginning of the program before any other Pygame functions are used.

```
'''
```

```
# Loading pictures and assigning values for variables #
```

```
HEIGHT = 620
```

```

WIDTH = 1200
p1, p2 = 0, 0
screen = pygame.display.set_mode((WIDTH,HEIGHT))
pygame.display.set_caption("BREAKOUT")

```

```

bg_img = pygame.image.load(r'Background.png')
bg_img = pygame.transform.scale(bg_img,(WIDTH,HEIGHT))

```

```

counter = 0
test = 0
time = timePeriod

```

```

Start_x = 172
Start_y = 200
End_x = 172

```

```

Bar1_Xcoord = 250
Bar1_Ycoord = 485
Bar2_Xcoord = 250
Bar2_Ycoord = 100
Bar1_velocity = 4
Bar2_velocity = - 4
Bar_speed = 3

```

```

Ball_Xcoord = 300
Ball_Ycoord = 300
Ball_Xvelocity = 0
Ball_Yvelocity = 3.5

```

```

Start=False
clock=pygame.time.Clock()
font_name = pygame.font.match_font('gil sans')

```

```

def timer():
    global counter, time
    if counter % 60 == 0:
        time -= 1
        score(screen,str(time), 25, 475, 60)

```

```

'''

```

The function "timer" is a function that keeps track of the time during a game or other activity. The function uses global variables "counter" and "time" to keep track of the time. If the counter is divisible by 60 (meaning that a minute has passed), the time is decreased by 1. The function also includes a call to the "score" function, which is used to display the current time on the screen.

```

'''

```

```

def ball():
    BallImg=pygame.image.load(r"eraser.png")
    global Ball_Xvelocity, Ball_Yvelocity, Ball_Xcoord, Ball_Ycoord,
    Bar1_Xcoord, Bar1_Ycoord, Start, Bar1_velocity, Bar2_velocity
    global p1, p2

```

```

if Start==True:
    Ball_Xcoord+=Ball_Xvelocity
    Ball_Ycoord+=Ball_Yvelocity
    timer()
else:
    if Ball_Ycoord==123:
        Ball_Xcoord=Bar2_Xcoord+80
    elif Ball_Ycoord==460:
        Ball_Xcoord=Bar1_Xcoord+80

# Ball-to-wall collision
if Ball_Xcoord<= 60 or Ball_Xcoord >=540:
    Ball_Xvelocity = -Ball_Xvelocity

# Ball-to-ground collision
if Ball_Ycoord >= 530:
    Ball_Xcoord=Bar2_Xcoord+80
    Ball_Ycoord=123
    Ball_Xvelocity=0
    Ball_Yvelocity=3
    Start=False
    p2 += 1

# Ball-to-ceiling collision
if Ball_Ycoord <= 10:
    Ball_Xcoord=Bar1_Xcoord+80
    Ball_Ycoord=460
    Ball_Xvelocity=0
    Ball_Yvelocity=3
    Start=False
    p1 += 1

# Collision with Red Paddle / Paddle 1
if 505 >= Ball_Ycoord >= 475 and (Bar1_Xcoord-20) <= Ball_Xcoord <=
(Bar1_Xcoord + 220):
    Ball_Xvelocity = Bar1_velocity
    Ball_Yvelocity = -Ball_Yvelocity-0.3
    if Ball_Xvelocity<0:
        pass
        Ball_Xvelocity -= 0.3
    else:
        Ball_Xvelocity += 0.3
    Bar1_velocity *= 1.001
    Ball_Ycoord-=5

# Collision with Blue Paddle / Paddle 2
if 80 <=Ball_Ycoord <= 120 and (Bar2_Xcoord-20) <= Ball_Xcoord <=
(Bar2_Xcoord + 220):
    Ball_Xvelocity = Bar2_velocity
    Ball_Yvelocity = -Ball_Yvelocity+0.3
    if Ball_Xvelocity<0:
        Ball_Xvelocity -= 0.5
    else:
        Ball_Xvelocity += 0.5

```

```

Ball_Ycoord+=5
Bar2_velocity *= 1.001
screen.blit(BallImg,(Ball_Xcoord,Ball_Ycoord))

```

```

'''

```

The ball function in this code is responsible for moving the ball on the screen and checking for collisions with various objects, such as the paddles and the walls. It uses global variables to track the ball's velocity, coordinates, and the coordinates and velocity of the paddles. If the game has started, the ball's position is updated based on its velocity. If the game has not started, the ball is placed in a specific position depending on which side of the screen it is on.

The function then checks for collisions with the walls, the ground, and the ceiling. If there is a collision, the ball's velocity is reversed and it is placed back in its starting position. The function also checks for collisions with the paddles and adjusts the ball's velocity accordingly. Finally, the function displays the ball on the screen using an image file

```

'''

```

```

def bar():
    Bar1Img=pygame.image.load(r"Red_Pencil.png")
    Bar2Img=pygame.image.load(r"Blue_Pencil.png")
    global Bar1_Xcoord, Bar1_Ycoord, Bar1_velocity,
Bar2_velocity,Bar2_Xcoord, Bar2_Ycoord
    Bar1_Xcoord += Bar1_velocity
    Bar2_Xcoord += Bar2_velocity
    if Bar1_Xcoord <= 50 or Bar1_Xcoord >= 400:
        Bar1_velocity = -Bar1_velocity
    if Bar2_Xcoord <= 50 or Bar2_Xcoord >= 400:
        Bar2_velocity = -Bar2_velocity
    screen.blit(Bar1Img,(Bar1_Xcoord,Bar1_Ycoord))
    screen.blit(Bar2Img,(Bar2_Xcoord,Bar2_Ycoord))

```

```

'''

```

The bar function in this code is responsible for moving the paddles on the screen and checking for collisions with the walls. It uses global variables to track the paddles' coordinates and velocities. The function updates the paddles' positions based on their velocities and checks if they have collided with the walls. If there is a collision, the paddles' velocities are reversed. Finally, the function displays the paddles on the screen using image files.

```

'''

```

```

def score(surf, text, siStarte, x, y):
    font = pygame.font.Font(font_name, siStarte)
    text_surface = font.render(text, True, "TEAL")
    text_rect = text_surface.get_rect()
    text_rect.midtop = (x, y) #60 500
    surf.blit(text_surface, text_rect)

```

```
'''
```

The score function in this code is responsible for displaying a score on the screen. It takes in several arguments: a surface to display the score on, the text of the score, the size of the font, and the coordinates of where the score should be displayed. The function first creates a font object using the specified font name and size. It then creates a surface object for the text by rendering the text using the font object. The function calculates the rectangle that the text surface occupies and centres it at the specified coordinates. Finally, the function displays the text surface on the given surface.

```
'''
```

```
def writeScore(N1, p1, N2, p2):  
    with open('MatchHistory.csv', 'a', newline='') as mh:  
        writer = csv.writer(mh)  
        writer.writerow([N1, str(p1), N2, str(p2)])
```

```
'''
```

The writeScore() function in this code is responsible for writing the scores of a game to a CSV file called 'MatchHistory.csv'. It takes in four arguments: N1, p1, N2, and p2. N1 and N2 represent the names of the two players, and p1 and p2 represent their scores. The function opens the CSV file in append mode, meaning that it will add new rows to the end of the file rather than overwriting it. It then creates a CSV writer object and writes the names and scores of the players as a new row in the file. Finally, it closes the file. This function allows the scores of multiple games to be stored in the same CSV file and can be used to keep track of the scores of multiple players over time.

```
'''
```

```
'''                                #      Main Namespace      #  
'''
```

The main namespace of this code is a while loop that is used to run the game.

The loop begins by setting the variable GameLoop to True. The loop then runs continuously if the value of GameLoop remains True. Inside the loop, other functions and statements are executed to perform various tasks such as drawing the game screen, handling user input, and updating the game objects. The loop continues to run until some event or condition occurs that causes GameLoop to be set to False, at which point the loop will terminate and the game will end. This loop structure allows the game to run continuously until it is manually stopped or a specific event occurs.

```
'''
```

```
GameLoop = True  
while GameLoop:  
    clock.tick(60)  
    counter += 1  
    screen.blit(bg_img,(0,0))  
    for event in pygame.event.get():  
        if event.type==pygame.KEYDOWN:
```

```

    if event.key==K_ESCAPE:
        GameLoop=False
    if event.key==K_RIGHT:
        Bar1_velocity = Bar_speed
    if event.key==K_LEFT:
        Bar1_velocity = -Bar_speed
    if event.key==K_s:
        Bar2_velocity = Bar_speed
    if event.key==K_a:
        Bar2_velocity = -Bar_speed
    if event.key==K_SPACE:
        Start=True
    elif event.type==QUIT:
        GameLoop=False
if event.type==pygame.KEYUP:
    if event.key==K_RIGHT:
        Bar1_velocity = 0
    if event.key==K_LEFT:
        Bar1_velocity = 0
    if event.key==K_a:
        Bar2_velocity = 0
    if event.key==K_s:
        Bar2_velocity = 0
if event.type == pygame.QUIT:
    GameLoop = False

```

...

This block of code is responsible for handling user input and performing actions based on the input. It begins by drawing the background image on the screen.

It then iterates through a list of events that have occurred since the last frame of the game.

If an event is a key press, the code checks the key that was pressed and performs a specific action based on the key.

For example, if the key pressed is the right arrow key, the velocity of Bar1 (one of the paddles) is set to the value of the variable Bar_speed.

If the key pressed is the left arrow key, the velocity of Bar1 is set to the negative of Bar_speed.

Similarly, if the key pressed is the 's' key, the velocity of Bar2 is set to Bar_speed, and if the key pressed is the 'a' key, the velocity of Bar2 is set to the negative of Bar_speed.

If the key pressed is the space bar, the variable Start is set to True.

If the key pressed is the escape key, the variable GameLoop is set to False, which will cause the game to end. If the event is a key release, the velocities of the paddles are set to 0.

If the event is a quit event, the variable GameLoop is also set to False.

This block of code allows the user to control the paddles using the keyboard and start or end the game using specific keys.

...


```

if time == 0:
if test < 1:
    writeScore(N1, p1, N2, p2)
    test += 1
Start = False
pygame.time.wait(2000)
GameLoop = False

```

```

'''

```

This block of code is responsible for ending the game and writing the scores to the CSV file when the time limit is reached.

The if statement checks if the time variable is equal to 0, which means that the time limit has been reached.

If the time limit has been reached, the code enters the if statement. The code then checks if the variable test is less than 1.

If it is, the writeScore function is called to write the scores of the players (N1, p1, N2, and p2) to the CSV file.

The variable test is then incremented by 1.

The variable Start is then set to False, which will cause the game to stop running.

The pygame.time.wait function is called with an argument of 2000, which causes the program to pause for 2 seconds.

Finally, the variable GameLoop is set to False, which will cause the main game loop to terminate and the game to end.

This block of code allows the game to end when the time limit is reached and writes the scores of the players to the CSV file before ending the game.

```

'''

```

```

# calling the image blit functions
ball()
bar()
score(screen,str(p1), 25, 150, 543)
score(screen,str(p2), 25, 153, 65)
# updating the display
pygame.display.update()

```

```

'''

```

This block of code is part of the main game loop and is responsible for updating the game objects and displaying them on the screen.

It calls the ball and bar functions to update the ball and paddles and then calls the score function to display the scores of the players on the screen.

The score function takes four arguments: the screen object, the scores of the players (p1 and p2) as strings, and two coordinates (x and y).

The scores are displayed at the specified coordinates on the screen.

Finally, the pygame.display.update function is called to update the display and show the updated game objects on the screen.

This block of code allows the game objects to be updated and displayed on the screen each frame

```

'''

```

```
pygame.quit()  
print("Thank you for playing!!!!")
```

```
'''
```

The pygame.quit function is a Pygame function that is used to close the Pygame library and release any resources that it is using. This function is typically called at the end of a Pygame program to clean up and close the Pygame window.

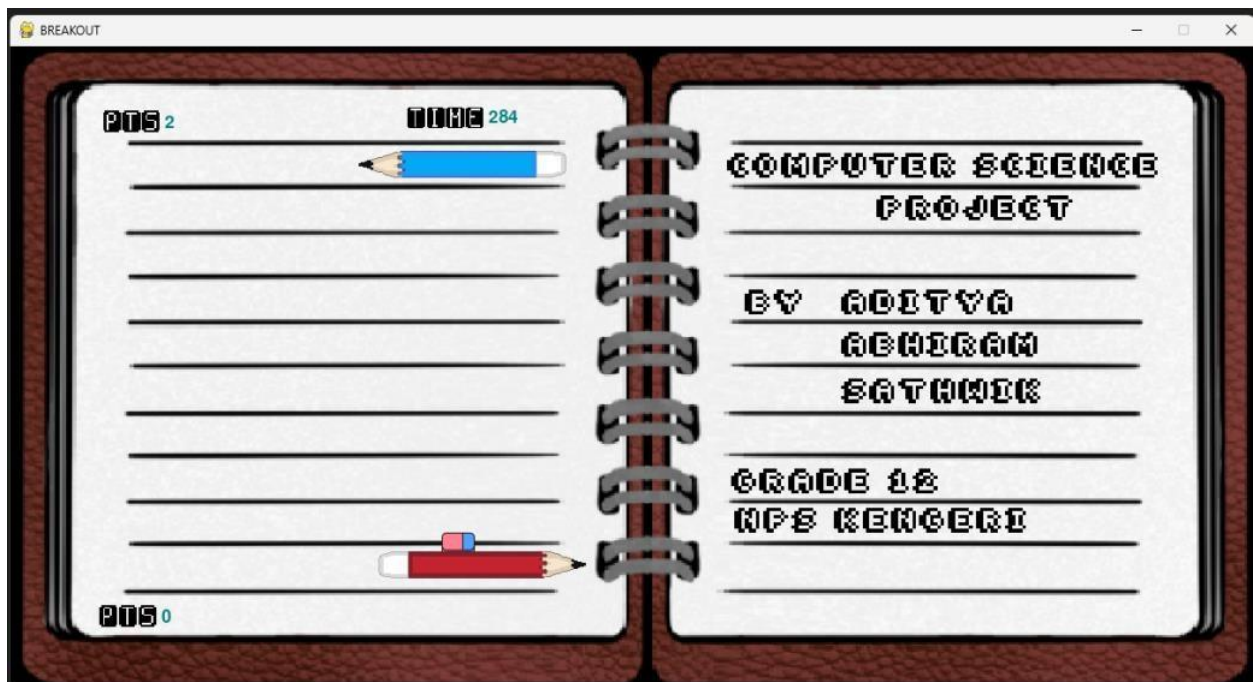
```
'''
```

Screenshots

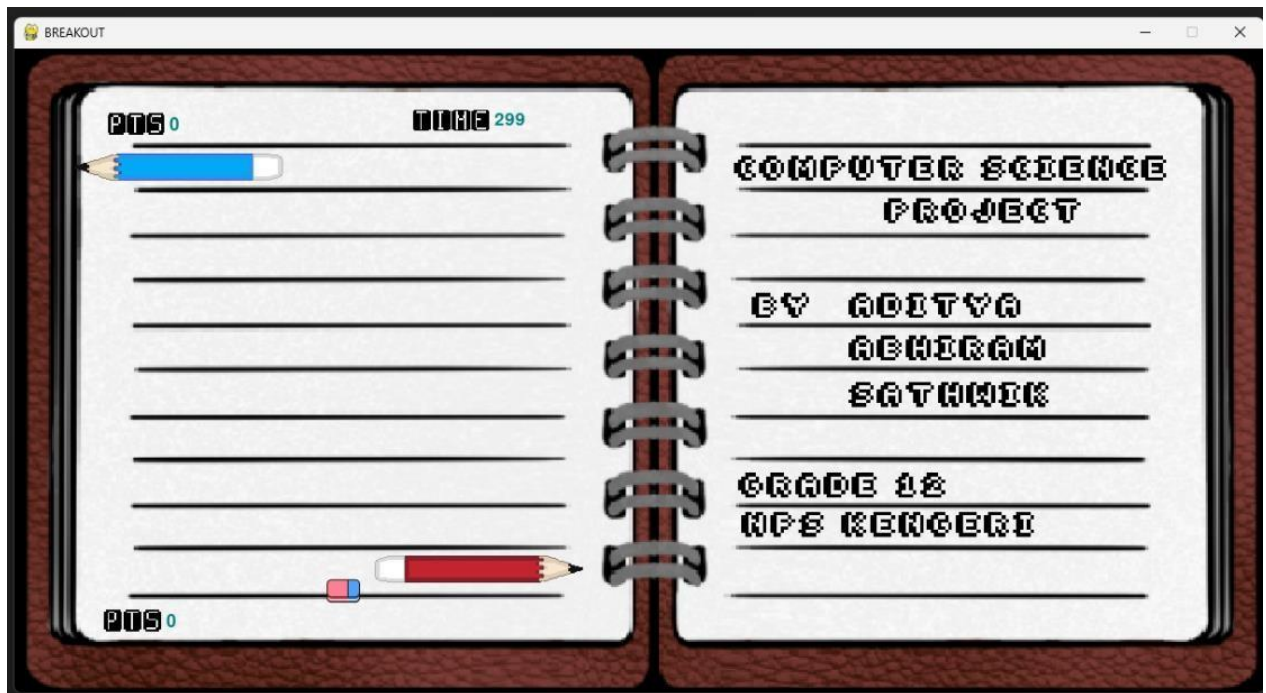
Starting screen on Terminal

```
pygame 2.1.2 (SDL 2.0.18, Python 3.10.5)  
Hello from the pygame community. https://www.pygame.org/contribute.html  
Player Red, Enter your username: Ronaldo  
Player Blue, Enter your username: Messi  
How long should the game last (in minutes)? -> 5
```

Eraser starting on red pencil after blue pencil scored



Blue pencil scoring against Red pencil



Eraser colliding with the wall



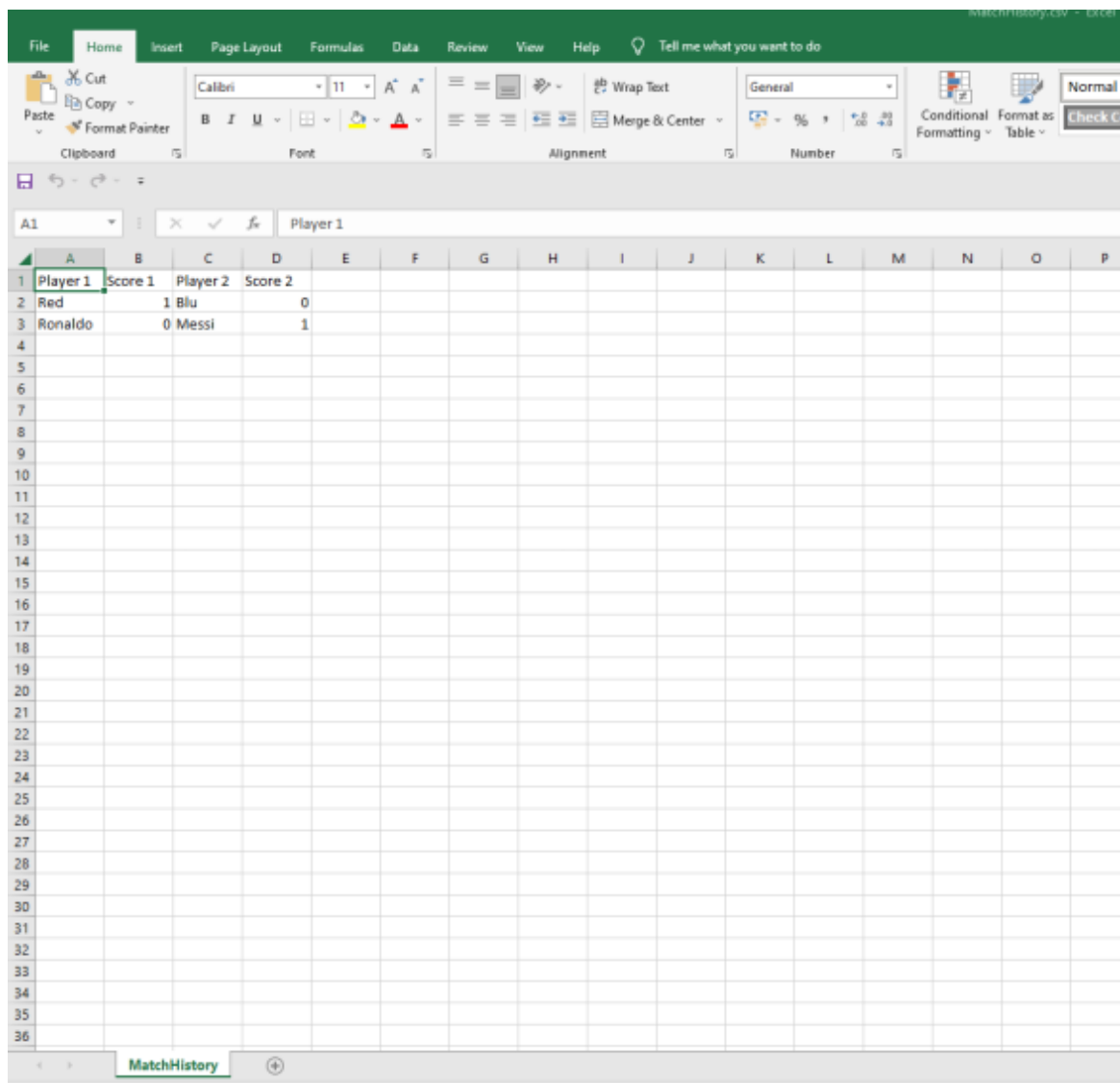
Eraser colliding with red pencil



Eraser colliding with a pencil and wall together



‘MatchHistory.csv’



Player 1	Score 1	Player 2	Score 2
Red	1	Blu	0
Ronaldo	0	Messi	1

References

- pygame.org/docs
- stackoverflow.com
- Sumita Arora Computer Science with Python
- Tech With Tim – Pygame in 90 minutes