Title: ROCK - PAPER - SCISSORES CSM216

https://github.com/surendra7438/rock-paper-scissores

Name: PONNAPALLI SURENDRA VARMA

Registration No: 12309779

Section: K23UP, G2

Roll No: 45

Submission Date: 10-11-24

Submitted to:

Mr. Aman Kumar

Acknowledgment

I would like to express my sincere gratitude to all those who supported and guided me throughout the development of this project, the **ROCK - PAPER - SCISSORES** in Python. Their invaluable assistance and insights have been instrumental in the completion of this project.

Firstly, I would like to thank my Professor Mr. Aman Kumar, for their continuous support, encouragement, and constructive feedback. Their expertise in programming and guidance on structuring the project provided me with the foundation to approach and complete the project successfully.

I would also like to acknowledge the resources provided by Lovely Professional University, which offered valuable reference material and tutorials that significantly helped in understanding the concepts required for game development and implementing the game logic.

Finally, I am grateful to my peers, friends, and family, whose encouragement and belief in my abilities motivated me to overcome challenges and complete this project to the best of my ability.

Ponnapalli Surendra Varma

12309779

Table of Contents

1.	Introduction	4
2.	Objectives and Scope of the Project	4
3.	Application Tools	5
4.	Project Design	6 to 7
5.	Flowchart	7
6.	DFD Chart	8 to 10
7.	Code Implementation	11 to 12
8.	Project Implementation	13 to 14
9.	Testing and Validation	15 to 18
10.	Conclusion	20
11.	References	21

1. Introduction

This project is a 2D shooter game developed using Python's tkinter library. The "Rock, Paper, Scissors Game" is a simple and interactive desktop application built using Python. It allows users to play the classic game against a computer. The project demonstrates the use of graphical user interfaces (GUIs) and randomization in Python to create a fun and engaging user experience. Users can select their choice of Rock, Paper, or Scissors, and the computer generates its choice randomly. The result of the game is displayed in a pop-up window.

Purpose and Significance

The significance of this project lies in its ability to introduce aspiring developers to core game development concepts within a relatively accessible environment. By utilizing Pygame, a versatile and beginner-friendly library, this project bridges the gap between theoretical programming knowledge and hands-on game design experience. Additionally, this game provides a creative space for experimenting with various aspects of game mechanics, including character physics, level design, asset management, and user interface development.

Problem or Objective

The objective of this project is to develop a functional 2D shooter game that offers players a dynamic and challenging experience. The main problem addressed is designing a game that incorporates realistic player controls, responsive AI enemies, and interactive elements such as health, ammo, and grenades. Achieving this requires implementing effective collision detection, ensuring balanced gameplay, and designing an intuitive interface that engages players

2. Objectives and Scope of the Project

Project Objectives

The project is developed with the following key objectives in mind: 1.

User-Friendly Interface:

- To design a simple and intuitive graphical user interface (GUI) that makes it easy for users of all age groups to play and enjoy the classic "Rock, Paper, Scissors" game.
- To ensure the application is accessible and requires no prior technical knowledge to operate.

2. **Showcasing GUI Development with tkinter:**

- To demonstrate the capabilities of the tkinter library for creating interactive desktop applications.
- To provide a practical example of using tkinter components like buttons, labels, and windows to build a real-world project.

3. **Implementation of Game Logic:**
   - To integrate simple but effective game logic that follows the standard rules of "Rock, Paper, Scissors."
     To enable dynamic interactions by evaluating the user's choice against the computer's randomly selected choice, ensuring a fair and engaging game.
4. **Visual Representation Using Images:**
   - To enhance user engagement by incorporating images for each game option (Rock, Paper, Scissors), making the interface visually appealing. ○ To provide a more interactive experience by using graphical elements rather than plain text.
5. **Learning and Skill Development:**
   - To create an educational tool for beginners in programming, helping them understand key concepts such as event handling, image processing, and GUI design. ○ To encourage exploration of Python's libraries and how they can be combined to build functional applications.
6. **Enhanced User Experience:**
   - To ensure that the application is responsive, with quick result display and clear feedback on the game's outcome. ○ To add usability features, such as autoclosing pop-ups, ensuring a seamless gaming experience.

Scope of the Project

The scope of this project is defined by its focus on simplicity, interactivity, and educational value: 1. **Target Audience:**

- This application is specifically designed for beginners and students learning Python programming.
- It serves as a foundational project that introduces the basic principles of GUI design, game development, and Python libraries.

2. **Small-Scale Application:**
   - The project intentionally maintains a small and manageable scope, focusing on a single-player game against the computer. ○ The limited features make it easy to understand, modify, and extend, offering a stepping stone for larger projects.
3. **Core Concepts Highlighted:**

- **Event Handling:** The program demonstrates how user inputs (button clicks) can trigger specific actions. ○ **Randomization:** The use of Python's random library to simulate the unpredictability of the computer's choice. ○ **DecisionMaking Logic:** The implementation of rules to determine the winner showcases fundamental programming constructs like conditional statements. ○ **Image Integration:** The project uses the Pillow library to manipulate and display images, providing practical knowledge of image processing in Python.

4. **Customization and Scalability:**
   - Although simple in its current form, the project can be expanded with additional features such as:
     - Keeping score across multiple rounds.
     - Allowing multiplayer functionality.
     - Adding animations or sound effects for a richer gaming experience.

     The flexibility of Python and the modular design of the code make such enhancements feasible for learners aiming to take their skills to the next level.

3. Application Tools

The following tools, libraries, and resources were used to develop this **2D rock – paper – scissores** project:

1. **Programming Language:**
   - **Python:** The primary language used for coding game logic, character behaviour and interactions within the Tkinter framework.
2. **Integrated Development Environments (IDEs):**
   - **PyCharm:** Used for coding, debugging, and managing the project files. ○ **Visual Studio Code**: An alternative IDE for quick editing, code testing, and project structuring.
3. **Libraries/Packages:**
   - **Tkinter:** The main library for creating is the standard GUI toolkit in Python.
   - **Usage:** It is used to create the graphical components of the application, including buttons, labels, windows, and layouts. ○ **Purpose:** Pillow is an image processing library in Python that is used for opening, manipulating, and saving image files. ○ **Random:** Used to add randomness in enemy behaviour, such as patrol patterns and idle moments. ○ **CSV:** Facilitates reading level data from external CSV files, allowing for flexible and scalable level design.

4. **Additional Tools:**
    - **Button Module:** A custom or third-party module for managing button clicks and user interface elements, enhancing the in-game menu and navigation functionality.

• **Project Design**

This **Rock, Paper, Scissors Game** is designed with a user-centric approach, ensuring a visually appealing and interactive experience. The design incorporates key GUI elements to make the application engaging and straightforward to use.

Main Window: 1.

Purpose:

- The main window serves as the central interface where users can interact with the application and make their choices.

2. **Features:**

Buttons with Images:

- Users can select their choice (Rock, Paper, or Scissors) by clicking on visually intuitive buttons.
- Each button is enhanced with an image representing the choice (rock, paper, or scissors) for better engagement.

Title and Labels:

- A title is displayed at the top of the window to describe the game purpose ("Rock, Paper, Scissors Game").
- Labels below each button provide text descriptions of the options, ensuring accessibility and clarity.

Layout:

- The buttons and labels are arranged in a grid format, with equal spacing and alignment for a balanced appearance.
- The window uses a white background to maintain a clean and professional look, complemented by appropriately colored text and labels.

Secondary Pop-Up Window:

2. **Purpose:**

- After the user makes their selection, the game opens a secondary pop-up window to display the game results.

3. **Features:**

User's Choice:

- The pop-up window shows the user's selection for reinforcement.

Computer's Choice:

- Displays the randomly generated choice of the computer for comparison.

Result Display:

- A bold and distinct message highlights the result (Win, Lose, or Tie). ○

Timed Auto-Close:

- The pop-up window automatically closes after 5 seconds, returning the user to the main window.

Design Details:

- The background color (pink) contrasts with the white main window for differentiation.
- The result text is styled with larger font sizes and bold emphasis, making it visually striking.

Game Logic: The game follows a straightforward logic to determine the winner based on the classic rules of "Rock, Paper, Scissors":

1. **User Input:** The user clicks a button to make their selection (Rock, Paper, or Scissors).
2. **Computer Selection:** The computer makes a random selection using the random.choice() function.
3. **Outcome Determination:** The program compares the user's choice and the computer's choice using the following rules:
   1. **Rock beats Scissors.**
   2. **Scissors beats Paper.**
   3. **Paper beats Rock.**
   4. Identical choices result in a tie.
5. **Result Announcement:** The determined outcome (Win, Lose, or Tie) is displayed in the secondary window.

Visual Design Philosophy:

The application's design prioritizes clarity, simplicity, and engagement: •

Clarity:

- Text descriptions and visual aids ensure the user understands their choices and the result. •

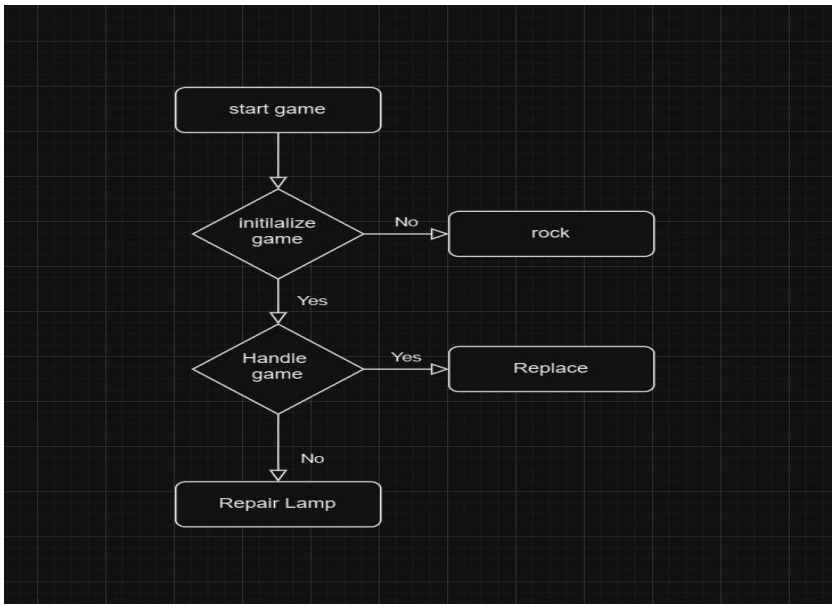
Simplicity:

- Minimalist layouts and intuitive interfaces make the application easy to navigate, even for beginners.

Engagement:

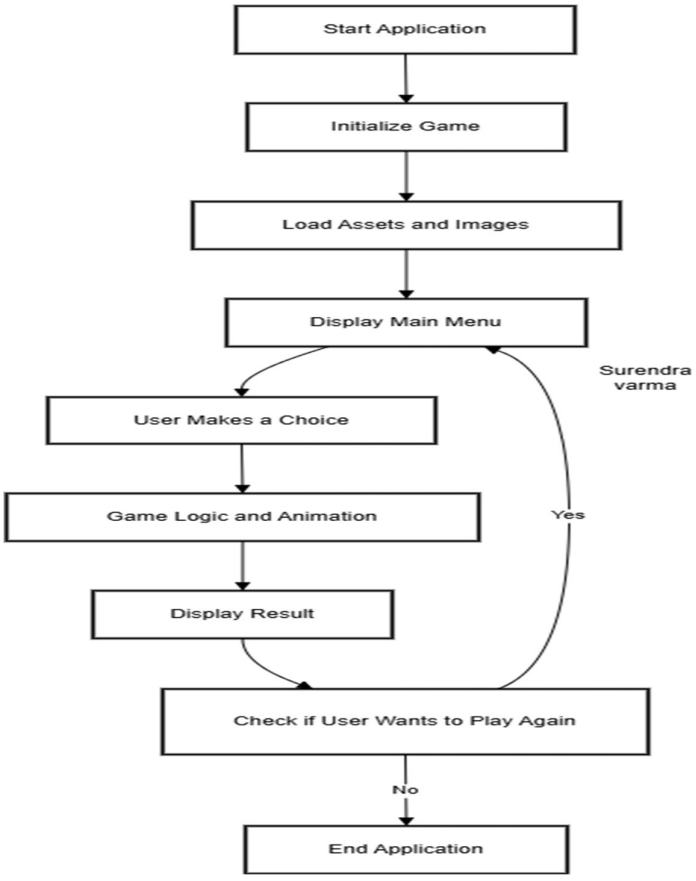
- The use of images, colors, and pop-ups enhances the fun and dynamic nature of the game.

5. Flowchart

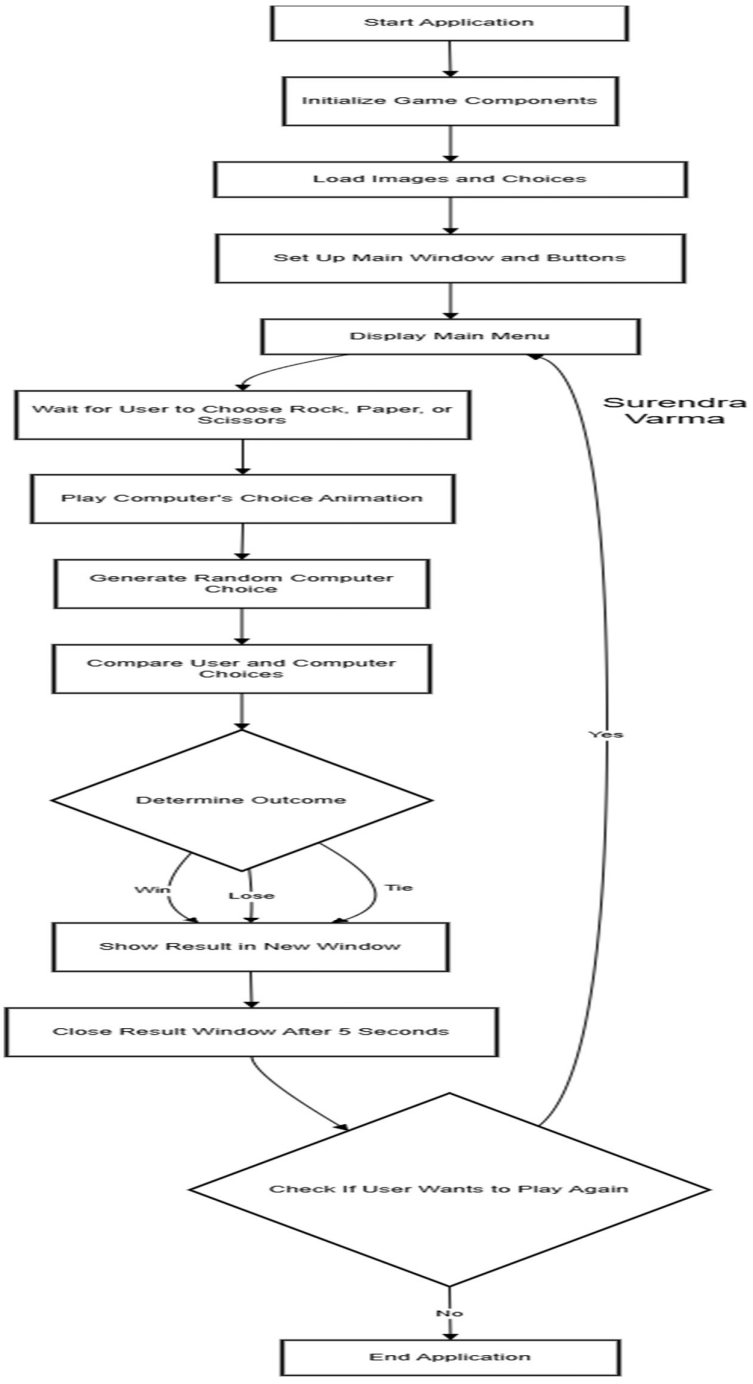


DFD Chart:

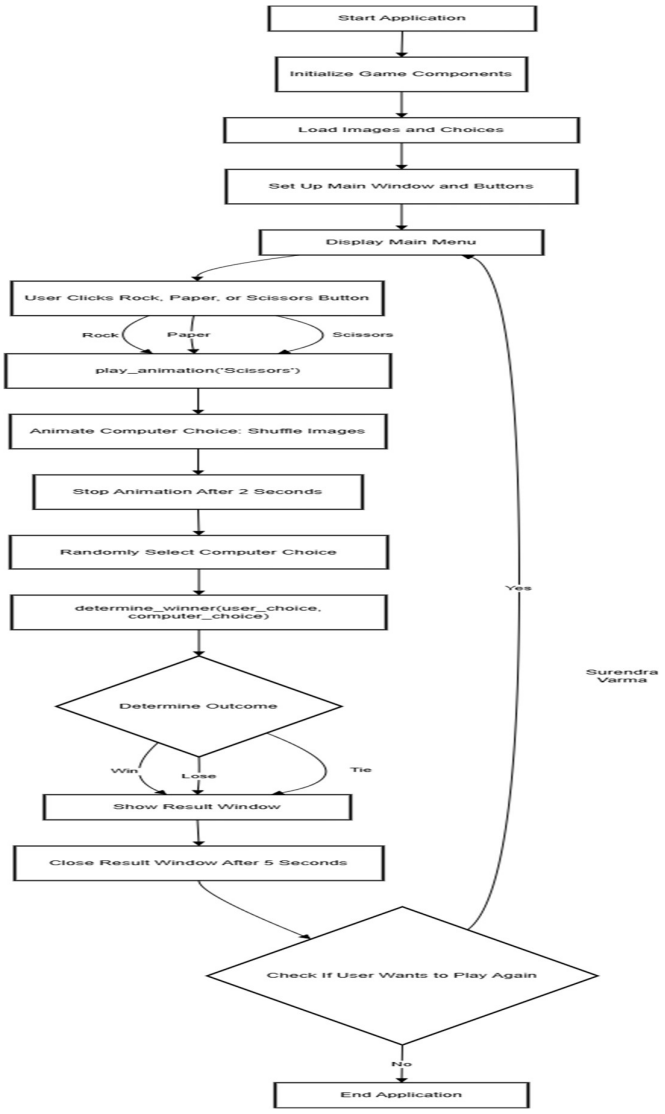
Level 0:



Level 1:


Start Application
Initialize Game Components
Load Images and Choices
Set Up Main Window and Buttons
Display Main Menu
Wait for User to Choose Rock, Paper, or Scissors
Play Computer's Choice Animation
Generate Random Computer Choice
Compare User and Computer Choices
Determine Outcome
Win
Lose
Tie
Show Result in New Window
Close Result Window After 5 Seconds
Check If User Wants to Play Again
Yes
No
End Application
Surendra Varma

Level 2:


Start Application
Initialize Game Components
Load Images and Choices
Set Up Main Window and Buttons
Display Main Menu
User Clicks Rock, Paper, or Scissors Button
Rock
Paper
Scissors
play_animation('Scissors')
Animate Computer Choice: Shuffle Images
Stop Animation After 2 Seconds
Randomly Select Computer Choice
determine_winner(user_choice, computer_choice)
Determine Outcome
Win
Lose
Tie
Show Result Window
Close Result Window After 5 Seconds
Check If User Wants to Play Again
Yes
No
End Application
Surendra Varma

Code Implementation:

```python
import tkinter as tk
from tkinter import Toplevel
from PIL import Image, ImageTk
import random

# Function to start the game
def start_game():  1 usage
    welcome_frame.pack_forget()  # Hide the welcome screen
    game_frame.pack(fill="both", expand=True)  # Show the game screen

# Function to exit the game
def exit_game():  1 usage
    root.destroy()

# Create the main window
root = tk.Tk()
root.title("Game Rock, Paper, Scissors")
root.geometry("500x400")
root.configure(bg="lightblue")

# Define the choices and outcomes
choices = ["Rock", "Paper", "Scissors"]

# Load the images
rock_img = ImageTk.PhotoImage(Image.open("rock.png").resize((100, 100)))
paper_img = ImageTk.PhotoImage(Image.open("paper.png").resize((100, 100)))
scissors_img = ImageTk.PhotoImage(Image.open("scissors.png").resize((100, 100)))

# Store images in a dictionary for easier reference
```

rock paper.py

```python
images = {"Rock": rock_img, "Paper": paper_img, "Scissors": scissors_img}

# Function to create the result window
def show_result(user_choice, computer_choice, result):
    result_window = Toplevel(root)
    result_window.title("Result")
    result_window.geometry("300x300")
    result_window.configure(bg="white")

    user_label = tk.Label(result_window, text=f"Your choice: {user_choice}", font=("Arial", 12), bg="white")
    user_label.pack(pady=10)

    computer_label = tk.Label(result_window, text=f"Computer's choice: {computer_choice}", font=("Arial", 12), bg="white")
    computer_label.pack(pady=10)

    result_label = tk.Label(result_window, text=result, font=("Arial", 16, "bold"), bg="white", fg="black")
    result_label.pack(pady=20)

    result_window.after( ms: 5000, result_window.destroy)

# Function to determine the winner
def determine_winner(user_choice):
    computer_choice = random.choice(choices)
    if user_choice == computer_choice:
        return computer_choice, "It's a Tie!"
    elif (user_choice == "Rock" and computer_choice == "Scissors") or \
         (user_choice == "Scissors" and computer_choice == "Paper") or \
         (user_choice == "Paper" and computer_choice == "Rock"):
        return computer_choice, "You Win!"
```

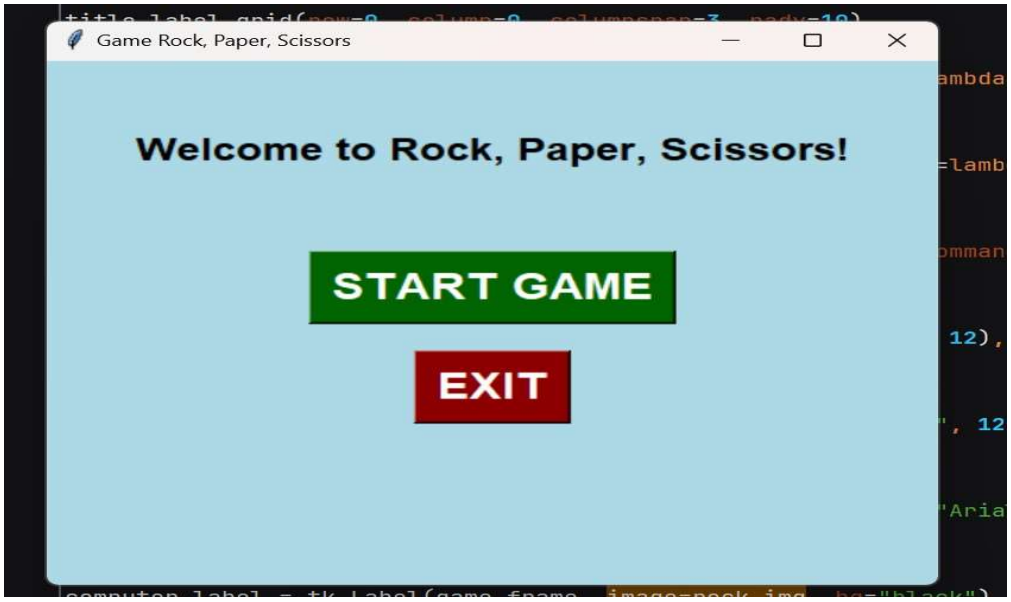
rock paper.py

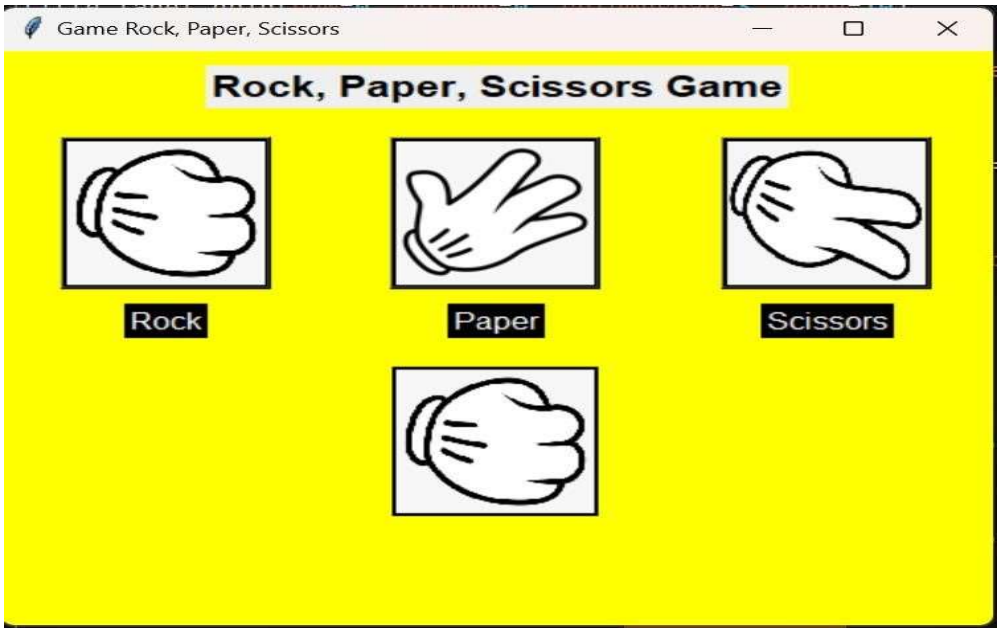
```
51  def determine_winner(user_choice):                                    4 1 2
59      else:
60          return computer_choice, "Computer Wins!"
61
62  # Function to play the animation and show the result
63  def play_animation(user_choice):  3 usages
64      shuffle_choices = ["Rock", "Paper", "Scissors"]
65      shuffle_index = 0
66
67      def animate():
68          nonlocal shuffle_index
69          computer_label.config(image=images[shuffle_choices[shuffle_index]])
70          shuffle_index = (shuffle_index + 1) % len(shuffle_choices)
71          root.after( ms: 100, animate)
72
73      def stop_animation():
74          computer_choice, result = determine_winner(user_choice)
75          computer_label.config(image=images[computer_choice])
76          show_result(user_choice, computer_choice, result)
77
78      animate()
79      root.after( ms: 2000, stop_animation)
80
81  # Create a welcome frame
82  welcome_frame = tk.Frame(root, bg="lightblue")
83  welcome_frame.pack(fill="both", expand=True)
84
85  welcome_label = tk.Label(welcome_frame, text="Welcome to Rock, Paper, Scissors!", bg="lightblue",
86                           fg="black", font=("Arial", 18, "bold"))
```

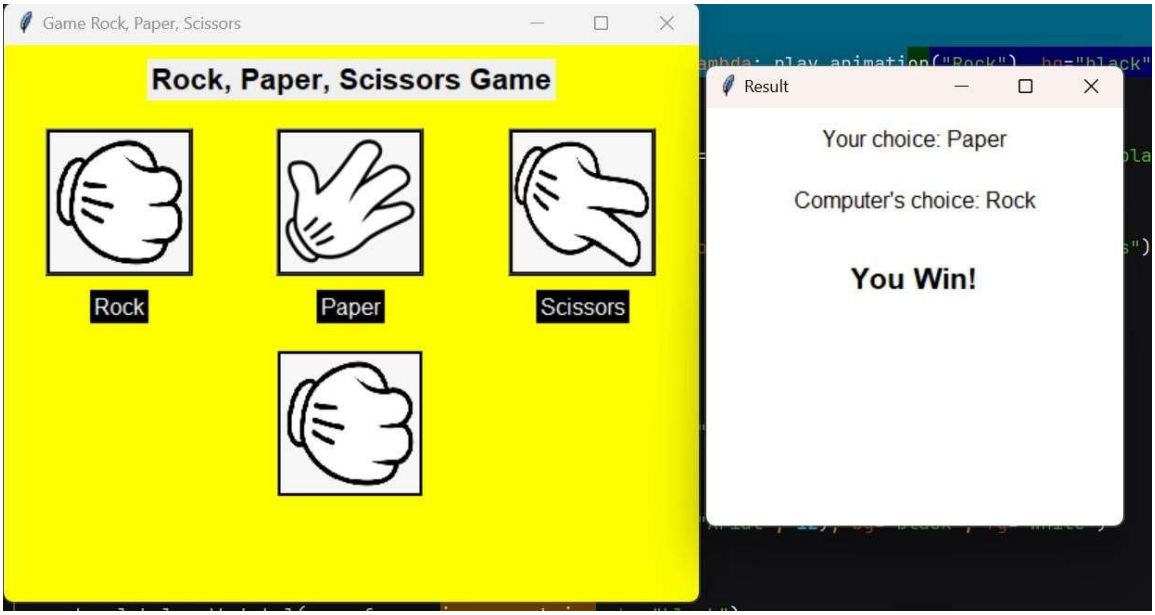
rock paper.py

```
115 game_frame = tk.Frame(root, bg="yellow")                              4
116
117 title_label = tk.Label(game_frame, text="Rock, Paper, Scissors Game", fg="black",
118                          font=("Arial", 16, "bold"))
119 title_label.grid(row=0, column=0, columnspan=3, pady=10)
120
121 rock_button = tk.Button(game_frame, image=rock_img, command=lambda: play_animation("Rock"), bg="black")
122 rock_button.grid(row=1, column=0, padx=10, pady=10)
123
124 paper_button = tk.Button(game_frame, image=paper_img, command=lambda: play_animation("Paper"), bg="black")
125 paper_button.grid(row=1, column=1, padx=10, pady=10)
126
127 scissors_button = tk.Button(game_frame, image=scissors_img, command=lambda: play_animation("Scissors"), bg="black")
128 scissors_button.grid(row=1, column=2, padx=10, pady=10)
129
130 rock_label = tk.Label(game_frame, text="Rock", font=("Arial", 12), bg="black", fg="white")
131 rock_label.grid(row=2, column=0)
132
133 paper_label = tk.Label(game_frame, text="Paper", font=("Arial", 12), bg="black", fg="white")
134 paper_label.grid(row=2, column=1)
135
136 scissors_label = tk.Label(game_frame, text="Scissors", font=("Arial", 12), bg="black", fg="white")
137 scissors_label.grid(row=2, column=2)
138
139 computer_label = tk.Label(game_frame, image=rock_img, bg="black")
140 computer_label.grid(row=3, column=1, pady=20)
141
142 game_frame.grid_columnconfigure( index: 0, weight=1)
143 game_frame.grid_columnconfigure( index: 1, weight=1)
```

Project Implementation:







Testing and Validation

1. Unit Testing:

Unit testing was performed to ensure that individual components of the Rock – Paper - Scissors game functioned as expected. Below is the table detailing the test cases:

Table 1: Movement Validation:

Test Case ID	**Test Description**	**Input**	**Expected Output**	**Actual Output**	**Status**
TC001	Verify "Start Game" button functionality	Click "Start Game" button	Welcome frame disappears; Game frame appears.	Welcome frame disappears; Game frame appears.	Pass
TC002	Verify "Exit" button functionality	Click "Exit" button	Application closes immediately.	Application closes immediately.	Pass
TC003	Test "Rock" button click	Click "Rock" button	Animation plays; result screen shows user as	Animation plays; result screen shows user as	Pass

TC004	Test "Paper" button click	Click "Paper" button	Animation plays; result screen shows user as "Paper," computer choice, and game result.	Animation plays; result screen shows user as "Paper," computer choice, and game result.	Pass
TC005	Test "Scissors" button click	Click "Scissors" button	Animation plays; result screen shows user as "Scissors," computer choice, and game result.	Animation plays; result screen shows user as "Scissors," computer choice, and game result.	Pass
TC006	Verify animation plays before result displays	Click "Rock" button	Images of "Rock," "Paper," and "Scissors" shuffle for ~2 seconds before final result appears.	Images shuffle for ~2 seconds before the final result appears.	Pass
TC007	Check correct result for "Rock vs Scissors"	User: "Rock"; Computer: "Scissors"	User wins; result displays "Rock beats Scissors."	User wins; result displays "Rock beats Scissors."	Pass
TC008	Check tie result	User: "Paper"; Computer: "Paper"	Tie; result displays "It's a Tie!"	Tie; result displays "It's a Tie!"	Pass

System Testing:

- System testing was conducted to validate the functionality of the entire rock – paper - scissore game as an integrated system. The primary goal was to ensure that all components—game logic, user input handling, and interaction rules—function cohesively to deliver a seamless gaming experience.

Test Case ID	Component	Test Description	Input	Expected Output	Actual Output	Status
ST001	Welcome Screen	Verify "Start Game" button functionality	Click "Start Game" button	Welcome screen disappears; game screen appears.	Welcome screen disappears; game screen appears.	Pass
ST002	Welcome Screen	Verify "Exit" button functionality	Click "Exit" Button	Application closes immediately.	Application closes immediately.	Pass
ST003	Game Screen	Test "Rock" button functionality	Click "Rock" Button	Animation plays; result shows "Your choice: Rock," computer choice, and the winner.	Animation plays; result shows "Your choice: Rock," computer choice, and the winner.	Pass
ST004	Animation and Timing	Verify animation before showing results	Click "Rock" button	Images of "Rock," "Paper," and "Scissors" shuffle for ~2 seconds before showing the final result.	Images shuffle for ~2 seconds before showing the final result.	Pass
ST005	Result Window	Verify result window timeout	Wait 5 seconds after result window	Result window closes automatically after 5 secods	Result window closes automatically after 5 seconds	Pass
ST006	Winner Logic	Check correct winner for "Rock vs Paper"	User: "Rock"; Computer: "Paper"	Computer wins; result displays "Paper beats Rock."	Computer wins; result displays "Paper beats Rock."	Pass

ST007	UI Responsiveness	Verify resizing of main window	Resize the main window	UI remains functional and aligned; no overlapping or misalignment of elements	UI remains functional and aligned; no overlapping or misalignment of elements.	Pass

Conclusion

The Rock, Paper, Scissors game implemented with Tkinter provides a functional and engaging graphical user interface for players. Below is the evaluation and summary

User Interface:

- The interface is visually appealing with color-coded buttons and proper labeling.
- Images for "Rock," "Paper," and "Scissors" enhance the game's visual appeal and interactivity.

Game Flow:

- A clear separation between the welcome screen and the game screen ensures intuitive navigation.
- Animation before the result is displayed adds an enjoyable dynamic element to the game.

Logic Implementation:

- The game logic is correctly implemented with all possible scenarios accounted for.
- Randomized computer choices ensure an unpredictable and fair game.

User Feedback:

- A dedicated result window displays outcomes (tie, win, or loss) with both the user's and computer's choices, enhancing clarity.
- Auto-closing the result window after 5 seconds keeps the game flow smooth.

Extensibility:

- The modular structure (e.g., functions for starting the game, determining the winner, showing results) makes the code easy to understand and extend.

Testing:

- GUI automation testing is not implemented but is essential for regression testing.

- Unit testing for the game logic is suggested to validate edge cases programmatically.

User Experience:

- **Responsive Design**: The layout could adjust dynamically for different screen resolutions to improve user experience on larger or smaller screens.
- **Restart Option**: Include a "Restart Game" button to allow players to replay without restarting the application.

Performance:

- The continuous animation loop might cause slight performance issues on lower-spec systems. Optimize by limiting iterations during animation.

Accessibility:

- Provide tooltips or text-based feedback for visually impaired users. • Add a keyboard interface to allow game actions without using a mouse.

References

Below is a list of resources used for the development of the **ROCK - PAPER - SCISSORES** game project: **Books:**

- Sweigart, Al. **"Automate the Boring Stuff with Python"**. (For general Python programming concepts and principles of automation.)

Online Resources:

- YouTube channels such as **"Code with Russ" ▪** Provided step-by-step tutorials for game development using Pygame.
- FreeCodeCamp: Tutorials on Python programming and game development.

Documentation for Tools/Libraries :

Pygame Library:

- Comprehensive documentation and examples for handling graphics, animations, and sounds in Python.