PROJECT REPORT

On

“Rock Paper Scissor Game”

Submitted By

Vaishnavi Arun Kuhikar

Guided By:

Mr. Ratnesh K. Choudhary

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

S. B. JAIN INSTITUTE OF TECHNOLOGY

MANAGEMENT AND RESEARCH, NAGPUR.

(An Autonomous Institute, Affiliated to RTMNU, Nagpur)

2021-2022

S.B. JAIN INSTITUTE OF TECHNOLOGY MANAGEMENT AND RESEARCH, NAGPUR

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

SESSION 2021-2022

CERTIFICATE

This is to certify that the Project titled **"Rock Paper Scissor Game"** is a bonafide work of **Vaishnavi Arun Kuhikar** carried out for the partial fulfillment of the requirement for the award of Degree of Bachelor of Engineering in **Computer Science & Engineering.**

Mr. Ratnesh K. Choudhary	**Mr. Animesh Tayal**
Assistant Professor	Head of Department

INDEX

LIST OF FIGURE

CHAPTER 1

INTRODUCTION

Rock paper scissors game is also known as stone paper scissors. It is a hand game that is usually played between 2 people, each player can randomly form any one of three from their hand. A player who chooses rock will win by another player who chooses scissors but loose by the player who chooses paper; a player with paper will loose by the player with the scissors.If both players choose the same then the game is tied. Rock paper scissors game is mainly played among kids.The object of the rock-paper-scissor python project is to build a game for a single player that plays with a computer, anywhere, and anytime. This project is base on the rules that:

- rock blunts scissors so rock wins
- scissors cut the paper so scissors win3333
- paper cover rock so paper wins

This project is build using tkinter, random modules, and the basic concept of python.

In this python project, players have to choose any one from rock, paper, and scissors. Then click on the play button will show the result of the game. We will see how we can create a rock paper and scissor game using Tkinter. Rock paper scissor is a hand game usually played between two people, in which each player simultaneously forms one of the three shapes with an outstretched hand. These shapes are “rock”, “paper”, and “scissors”.

Game Winner Conditions:

Let there be a Player who is playing with a computer as an opponent. Now,

- If the player selects Paper and Computer Selects Scissor – Computer wins
- If the player selects Rock and Computer Selects Scissor – Player 1 wins
- If the player selects Paper and Computer Selects Rock – Player 1 wins
- And If the player selects Paper and Computer Selects Paper – Draw
- If the player selects Rock and Computer Selects Rock – Draw
- If the player selects Scissor and Computer Selects Scissor – Draw
  Rock-Paper-Scissors , a game of cyclic dominance, is not merely a popular children’s game but also a basic model system for studying decision-making in non-cooperative strategic interactions. Aimed at students of physics with no background in game theory, this paper introduces the concepts of Nash equilibrium and evolutionarily stable strategy, and reviews some recent theoretical and empirical efforts on the non-equilibrium properties of the iterated RPS, including collective cycling, conditional response patterns, and microscopic mechanisms that facilitate cooperation. We also introduce several dynamical processes to illustrate the applications of RPS as a simplified model of species competition in ecological systems and price cycling in economic markets.

CHAPTER 2
METHODOLOGY

Part 1: Image processing and handling

1. Import Python Module Tkinter to execute GUI applications.

2. From PIL Import ImageTk, Image for image processing.

3. Import random which will help the computer to select options rock, paper, or scissor randomly.

4. Now an object is created root which is the main window object.

- Title of this window -> Rock Paper Scissor
- Dimensions of this window -> “800×680”

5. Create canvas of width=800, height=680

6. Now we Create labels on GUI window

- l1 =Player-> font=’Algerian’, size=25
- l2 =Computer-> font=’Algerian’, size=25
- l3 =Vs font=’Algerian’, size=40

7. Now labels are placed on window

- l1 at x=80, y=20
- l2 at x=560, y=20
- l3 at x=370, y=230

For Default Image:

1. An variable named img.png is used to open the default hand image and resize it to (300,300). This will be shown at default condition in the game at the place of the player.

2. An variable named imgs.png is used to store flipped default image from left to right using transpose function and it is saved in the variable. This will be shown at default condition in the game at the place of the computer’s side.

3. Images img.png and imgs.png are loaded on the canvas now using Tk.PhotoImage

For Rock Image:

1. An variable named rock.png is used to open the rock hand image and resize it to (300,300). This will be shown at the player’s side when the player selects rock in the game.

2) An variable named rocks.png is used to store flipped rock hand image from left to right using transpose function and will be shown at computer’s side when the computer randomly selects rock in the game.

3) Images rock.png and rocks.png are loaded on the canvas now using Tk.PhotoImage.

For Paper Image:

1) An variable named paper.png is used to open the paper hand image and resize it to (300,300). This will be shown at the player’s side when the player selects paper in the game.

2. An variable named papers.png is used to store flipped paper hand images from left to right using the transpose function and will be shown at the computer's side when the computer randomly selects paper in the game.

3. Images paper.png and papers.png is loaded on the canvas now using Tk.PhotoImage.

For Scissors Image

1. An variable named scissor.png is used to open the scissor hand image and resize it to (300,300). This will be shown at player's side when player selects scissor in the game.

2. An variable named scissors.png is used to store flipped scissor hand image from left to right using transpose function and will be shown at computer's side when computer randomly selects scissor in the game.

3. Images scissor.png and scissors.png are loaded on the canvas now using Tk.PhotoImage.

For Selection Image:

1. An variable named img.png is used to open the selection of hand images i.e. Combined image of Rock, Paper & Scissor, and resize it to (300,130).

2. Image imgs.png is loaded on the canvas now using Tk.PhotoImage.

Part 2: Game Implementation

1. A game function is defined in which we have a list named select having values 1, 2, 3 representing rock, paper, and scissors respectively.

2. Here random.choice randomly selects options 1-3 for computer
3. Set image for Player on canvas

- If Player selects 1 (Rock) Show rock image on canvas using create_image.
- If Player selects 2 (Paper) Show paper image on canvas using create_image.
- If Player selects 3 (Scissor) Show scissor image on canvas using create_image.

4. Set image for Computer on canvas

- If Computer selects 1 (Rock) Show rock image on canvas using create_image.
- If Computer selects 2 (Paper) Show paper image on canvas using create_image.
- If Computer selects 3 (Scissor) Show scissor image on canvas using create_image.

5. Obtaining the result

- If Player chooses Rock and computer chooses Rock OR If the player chooses Paper and the computer chooses Paper OR If the player chooses Scissor and the computer chooses Scissor. Result Shown-> Draw
- If the Player chooses Rock and computer choose Scissor OR If the player chooses Paper and computer choose Rock OR If the player chooses Scissor and computer choose Paper. Result Shown -> Player won
- Else Result Shown-> Computer won

6. Buttons

- Clear Button -> Deletes the present result and switches the figures at both sides to its default conditions.
- Rock Button -> Selects choice 1 in function game and Shows an Image of Hand showing Rock On Player side.

- Paper Button -> Selects choice 2 in function game and Shows Image of Hand showing Paper On Player side
- Scissor Button -> Selects choice 3 in function game and Shows an Image of Hand showing the Scissor On the Player side.

Images Used:

Fig 2.1.1 Rock

Fig 2.1.2 Rocks

Fig 2.1.3 Scissor

Fig 2.1.4 Scissors

Fig 2.1.5 Paper

Fig 2.1.6 Papers

Fig 2.1.7 Win

Fig 2.1.8 Loose

Fig 2.1.9 Tie

CHAPTER 3

TOOLS/PLATFORMS

3.1 SOFTWARE REQUIREMENTS

a. **PROGRAMMING LANGUAGE:** Python

b. **IDE:** Visual Studio, Sublime, Pandas

c. **LIBRARY:** Tkinter, Random,PIL

d. **PYTHON VERSION(recommended) :** 2.x or 3.x

e. **OPERATING SYSTEM:** Windows 11

Studio: Visual Studio is an Integrated Development Environment(IDE) developed by Microsoft to develop GUI(Graphical User Interface), console, Web applications, web apps, mobile apps, cloud, and web services, etc. With the help of this IDE, you can create managed code as well as native code. It uses the various platforms of Microsoft software development software like Windows store, Microsoft Silverlight, and Windows API, etc. It is not a language-specific IDE as you can use this to write code in C#, C++, VB(Visual Basic), Python, JavaScript, and many more languages. It provides support for 36 different programming languages. It is available for Windows as well as for macOS.

Evolution of Visual Studio: The first version of VS(Visual Studio) was released in 1997, named as Visual Studio 97 having version number 5.0. The latest version of Visual Studio is 15.0 which was released on March 7, 2017. It is also termed as Visual Studio 2017.

There are 3 editions of Microsoft Visual Studio as follows:

1. Community: It is a free version which is announced in 2014. All other editions are paid. This contains the features similar to Professional edition. Using this edition, any individual developer can develop their own free or paid apps like .Net applications, Web applications and many more.

2. Professional: It is the commercial edition of Visual Studio. It comes in Visual Studio 2010 and later versions. It provides the support for XML and XSLT editing and includes the tool like Server Explorer and integration with Microsoft SQL Server. Microsoft provides a free trial of this edition and after the trial period, the user has to pay to continue using it.

3. Enterprise: It is an integrated, end to end solution for teams of any size with the demanding quality and scale needs. Microsoft provides a 90-days free trial of this edition and after the trial period, the user has to pay to continue using it. The main benefit of this edition is that it is highly scalable and deliver high-quality software.

Pandas: The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008. Pandas is an open-source library that is made mainly for working with relational or labeled data both easily and intuitively. It provides various data structures and operations for manipulating numerical data and time series. This library is built on top of the NumPy library. Pandas is fast and it has high performance & productivity for users.

Series: Pandas Series is a one-dimensional labeled array capable of holding data of any type (integer, string, float, python objects, etc.). The axis labels are collectively called indexes. Pandas Series is nothing but a column in an excel sheet. Labels need not be unique but must be a hashable type.

Pandas DataFrame is a two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns). A Data frame is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns. Pandas DataFrame.

Tkinter: Tkinter tutorial provides basic and advanced concepts of Python Tkinter. Our Tkinter tutorial is designed for beginners and professionals.Python provides the standard library Tkinter for creating the graphical user interface for desktop based applications. Developing desktop based applications with python Tkinter is not a complex task. An empty Tkinter top-level window can be created by using the following steps.

1. import the Tkinter module.
2. Create the main application window.
3. Add the widgets like labels, buttons, frames, etc. to the window.
4. Call the main event loop so that the actions can take place on the user's computer screen.

Python: Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

How Is Python Used?

Python is a general-purpose programming language, which is another way to say that it can be used for nearly everything. Most importantly, it is an interpreted language, which means that the written code is not actually translated to a computer-readable format at runtime. Whereas, most programming languages do this conversion before the program is even run. This type of language is also referred to as a “scripting language” because it was initially meant to be used for trivial projects. The concept of a “scripting language” has changed considerably since its inception, because Python is now used to write large, commercial style applications, instead of just banal ones. This reliance on Python has grown even more so as the internet gained popularity. A large majority of web applications and platforms rely on Python, including Google’s search engine, YouTube, and the weboriented transaction system of the New York Stock Exchange (NYSE). You know the language must be pretty serious when it’s powering a stock exchange system

CHAPTER 4
DESIGN & IMPLEMENTATION

4.1 ALGORITHM

Step1: Start

Step 2: Make a list of 3 options- Rock, Paper, Scissor

Step 3: Ask the player’s name so we can display it on the scoreboard

Step 4: Initialize the scores of player and computer to 0 and number of rounds to 0.

Step 5: Set the gameOn flag to True

Step 6: While gameOn flag is True repeat steps 6 to 12

Step 7: Randomly generate one of the options from the list as ComputerOption

Step 8: Let the player choose any option from Rock, Paper, Scissor as PlayerOption.

Step 9: Let the player choose any option from Rock, Paper, Scissor as PlayerOption.

Step 10: Increase number of rounds by 1

Step 11: If both player and computer, choose the same option then the current round ends in a draw. Else proceed to step 10

Step 12: Determine the winner based on the rules and increase the score of the winner by 1.

Step 13: Choose and display the final winner based on the total score.

Step 14: Stop

4.2 FLOWCHART

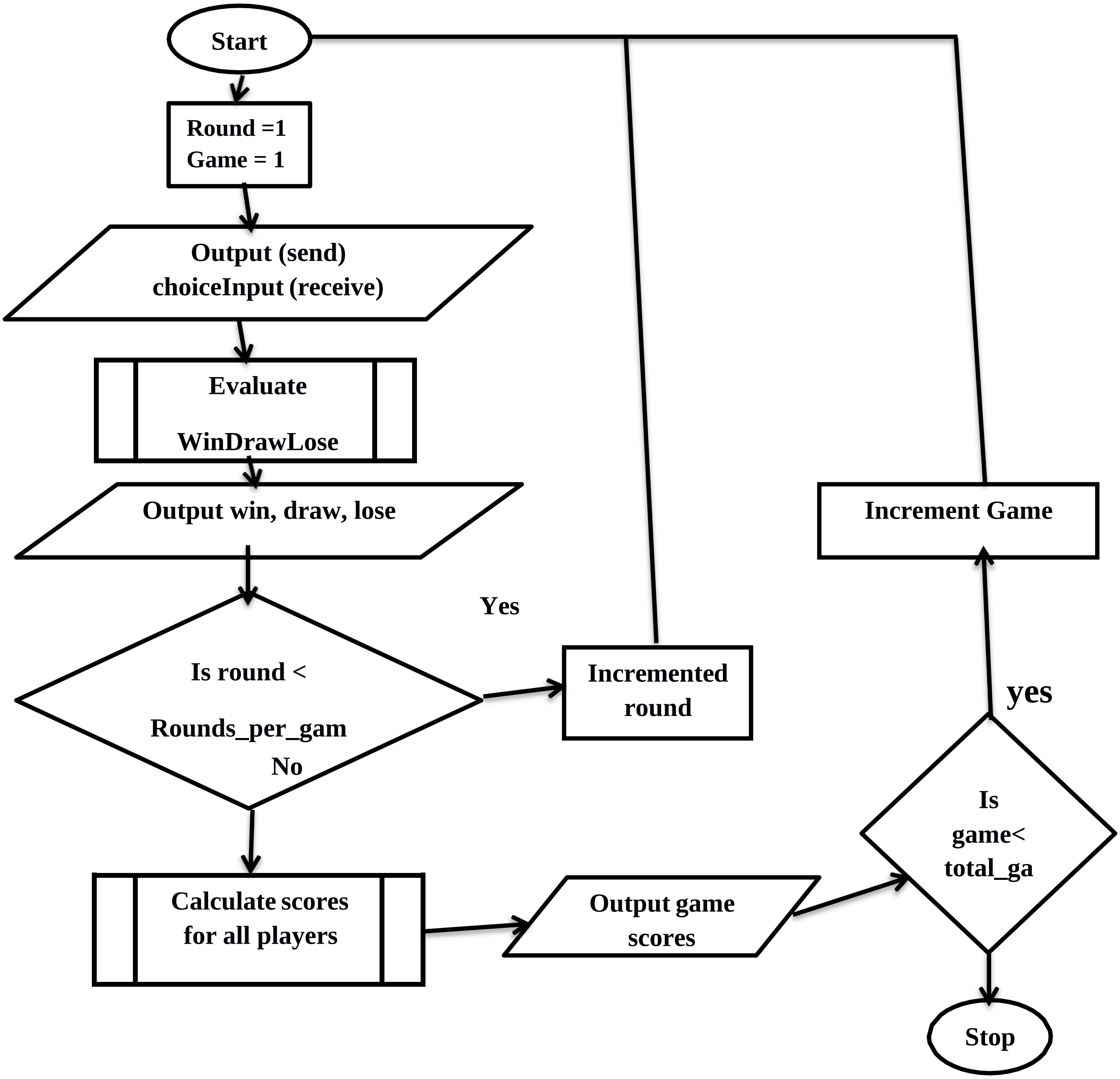


Fig 4.2.1 Flowchart of Rock Paper Scissor Game

The flowchart shows a high-level view of a multi-round game. This broadly matches the Advanced game, although the final implementation deviates from this playing games indefinitely.

4.3 SOURCE CODE

```
#IMPORTING REQUIRED MODULES
from tkinter import *
import random
from tkinter import messagebox
#DEFINING GUI WINDOW
window=Tk()
window.title("Gaming window")
window.geometry("850x400")
#FRAMES FOR GUI WINDOW
frame1=Frame(window)
frame1.pack(side=TOP)

frame2=Frame(window)
frame2.pack(side=TOP)
#DECLARING VARIABLES
ScoreMe_variable=IntVar()
ScoreComp_variable=IntVar()
#INITIALIZING VARIABLES
ScoreMe_variable.set('0')
ScoreComp_variable.set('0')
click=True
Rock=PhotoImage(file="Rock.png")
Paper=PhotoImage(file="Paper.png")
```

```
Scissor=PhotoImage(file="Scissor.png")

Rockpic=PhotoImage(file="rocks.png")

Paperpic=PhotoImage(file="papers.png")

Scissorpic=PhotoImage(file="scissors.png")

Tie=PhotoImage(file="Tie.png")

Youwin=PhotoImage(file="win.png")

Youloose=PhotoImage(file="loose.png")

#DEFINING FUNCTIONS

def Back():

ans=messagebox.askyesnocancel("Confirmation","Are you sure want to back?")

if(ans==True):

window.destroy()

import start

def Comp_Choice():

choice=random.choice(['rock','paper','scissor'])

return choice

def Exit():

ans=messagebox.askyesnocancel("Confirmation","Are you sure want to exit?")

if(ans==True):

window.destroy()

def Endgame(a):

#IF ANYONE(HUMAN OR COMPUTER) REACHES 3 WINS FIRST THEN GAME ENDS
AND WINNER DECLARED
```

```
if(a==3):

b=messagebox.askyesnocancel("WIN","HURRAH YOU WON,Do you want to play again?")

if(b):

window.destroy()

import gui

else:

window.destroy()

import start

else:

b=messagebox.askyesnocancel("LOOSE","OOPS YOU LOOSE,Do you want to play again?")

if(b):

window.destroy()

import gui

else:

window.destroy()

import start

def startgame(Yourchoice):

global click

compchoice=Comp_Choice()

if(click==True):

if(Yourchoice=="rock"):

Button1.configure(image=Rockpic)
```

```
click=False
if(compchoice=="rock"):
Button2.configure(image=Rockpic)
Button3.configure(image=Tie)
elif(compchoice=="paper"):
Button2.configure(image=Paperpic)
Button3.configure(image=Youloose)
ScoreComp_variable.set(ScoreComp_variable.get()+1)
else:
Button2.configure(image=Scissorpic)
Button3.configure(image=Youwin)
ScoreMe_variable.set(ScoreMe_variable.get()+1)
if(Yourchoice=="paper"):
Button1.configure(image=Paperpic)
click=False
if(compchoice=="paper"):
Button2.configure(image=Paperpic)
Button3.configure(image=Tie)
elif(compchoice=="rock"):
Button2.configure(image=Rockpic)
Button3.configure(image=Youwin)
ScoreMe_variable.set(ScoreMe_variable.get()+1)
else:
```

```
Button2.configure(image=Scissorpic)
Button3.configure(image=Youloose)
ScoreComp_variable.set(ScoreComp_variable.get()+1)
if(Yourchoice=="scissor"):
Button1.configure(image=Scissorpic)
click=False
if(compchoice=="scissor"):
Button2.configure(image=Scissorpic)
Button3.configure(image=Tie)
elif(compchoice=="rock"):
Button2.configure(image=Rockpic)
Button3.configure(image=Youloose)
ScoreComp_variable.set(ScoreComp_variable.get()+1)
else:
Button2.configure(image=Paperpic)
Button3.configure(image=Youwin)
ScoreMe_variable.set(ScoreMe_variable.get()+1)
else:
Button1.configure(image=Rock)
Button2.configure(image=Paper)
Button3.configure(image=Scissor)
click=True
```

```
if(ScoreMe_variable.get()==3 or ScoreComp_variable.get()==3):

Endgame(ScoreMe_variable.get())

#DEFINING LABELS ENTRYFIELDS AND BUTTONS

Label1=Label(frame1,fg="BLUE",text="YOUR SCORE:",font=('algerian',10,'bold'))

Label1.grid(row=1,column=1)

Label_MyScore=Label(frame1,textvariable=ScoreMe_variable)

Label_MyScore.grid(row=1,column=2)

Label2=Label(frame1,fg="GREEN",text="COMPUTER SCORE:",font=('algerian',10,'bold'))

Label2.grid(row=2,column=1)

Label_CompScore=Label(frame1,textvariable=ScoreComp_variable)

Label_CompScore.grid(row=2,column=2)

Label3=Label(frame1,fg="BLACK",text="Choose your choice",font=('bold',10,'bold'))

Label3.grid(row=3,column=1)

Button1=Button(frame2,image=Rock,fg="Yellow",bg="Black",activebackground="Green",command=lambda:startgame("rock"))

Button1.grid(row=2,column=1)

Button2=Button(frame2,image=Paper,fg="Yellow",bg="Black",activebackground="Green",command=lambda:startgame("paper"))

Button2.grid(row=2,column=2)

Button3=Button(frame2,image=Scissor,fg="Yellow",bg="Black",activebackground="Green",command=lambda:startgame("scissor"))

Button3.grid(row=2,column=3)

Back_button=Button(frame2,text="BACK",fg="Yellow",bg="Black",activebackground="Green",command=Back)

Back_button.grid(row=3,column=2)
```

```
Exit_button=Button(frame2,text="EXIT",fg="YELLOW",bg="BLACK",activebackground="GREEN",command=Exit)

Exit_button.grid(row=3,column=3)

window.mainloop()
```

4.4 SYSTEM DESIGN

4.4.1 USE-CASE DIAGRAM

Actors-

1.Admin

2.User

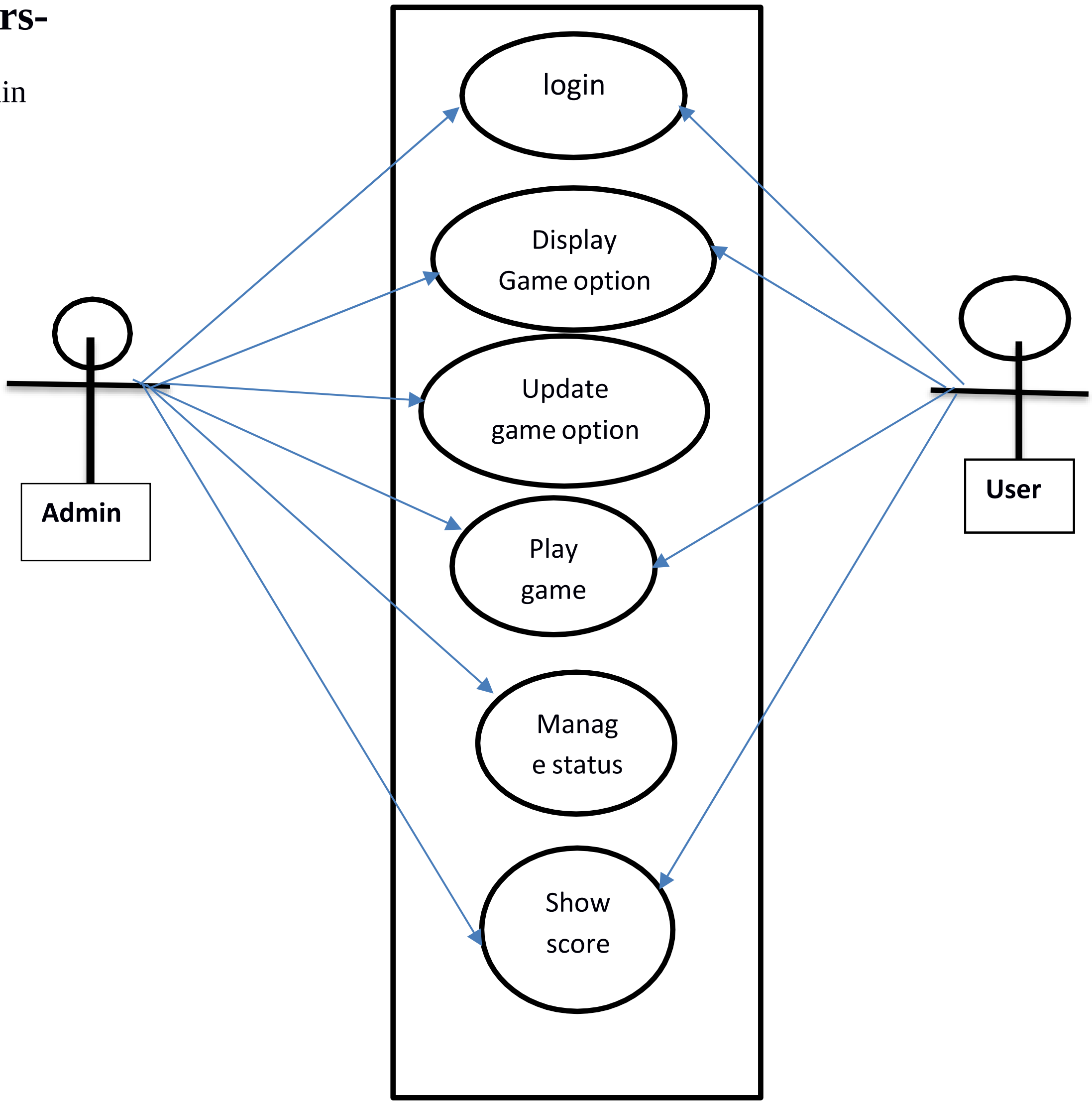
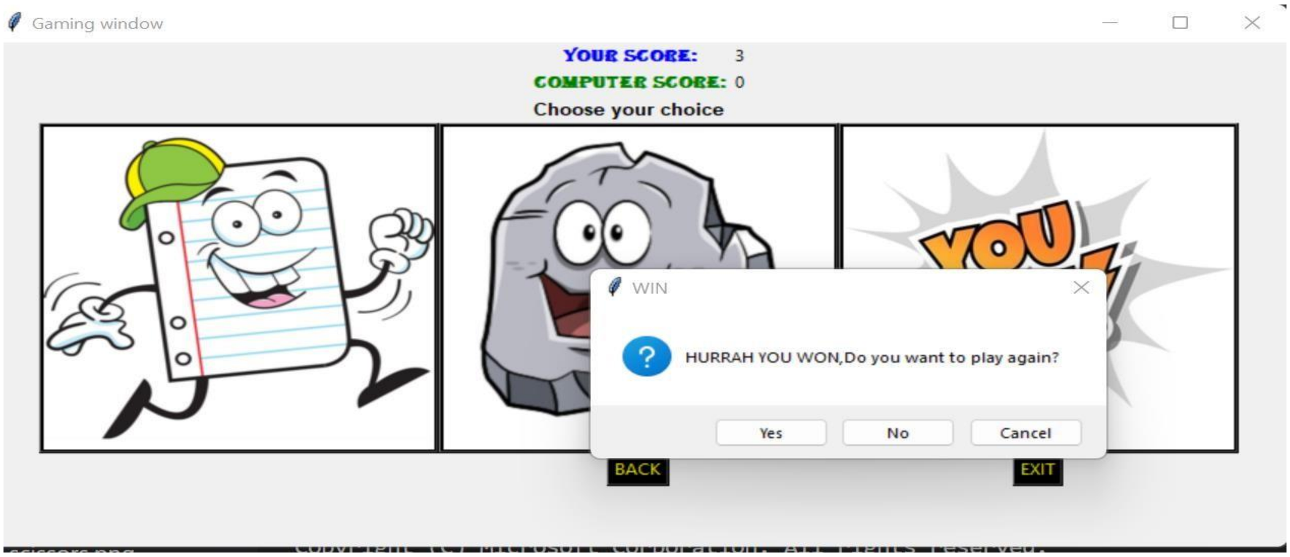


Fig. 4.4.2 User case diagram

CHAPTER 5
RESULT & DISCUSSION

5.1 OUTPUT



Gaming window

YOUR SCORE: 3

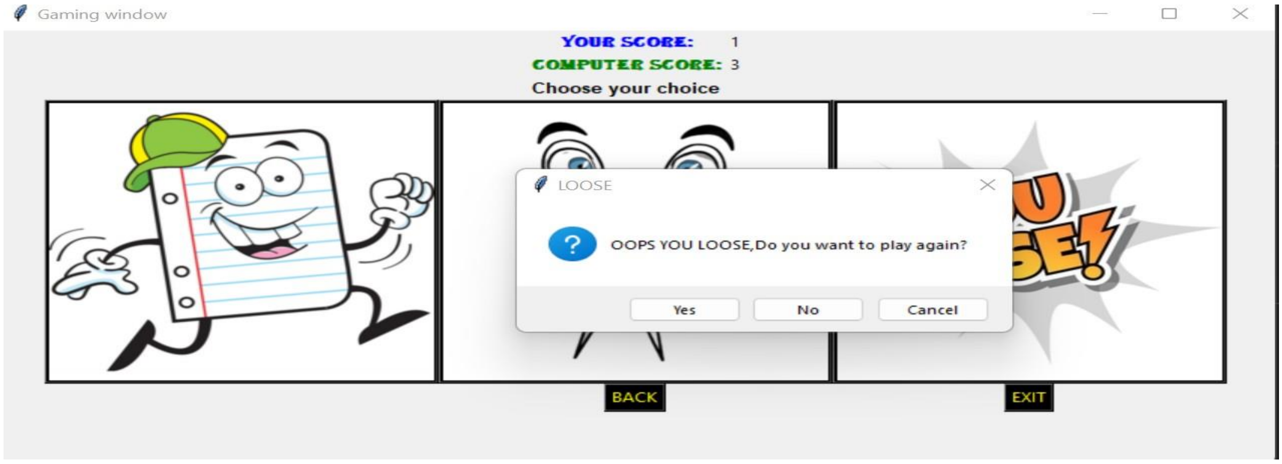
COMPUTER SCORE: 0

Choose your choice

WIN

HURRAH YOU WON,Do you want to play again?

Fig 5.1.1 Output of Win the Game



Gaming window

YOUR SCORE: 1

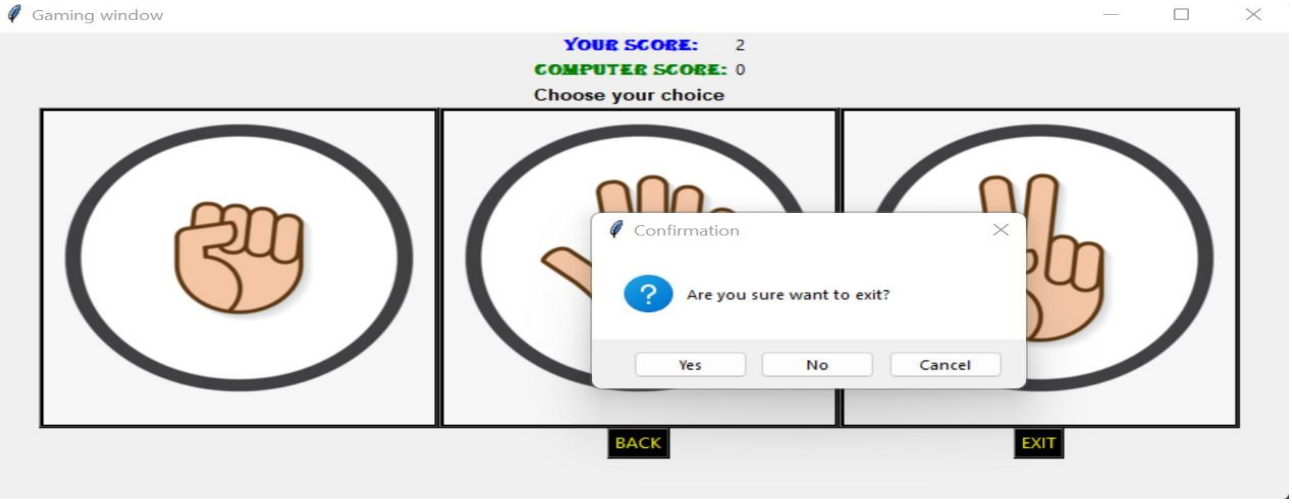
COMPUTER SCORE: 3

Choose your choice

LOOSE

OOPS YOU LOOSE,Do you want to play again?

Fig 5.1.2 Output of Loose the Game



YOUR SCORE: 2

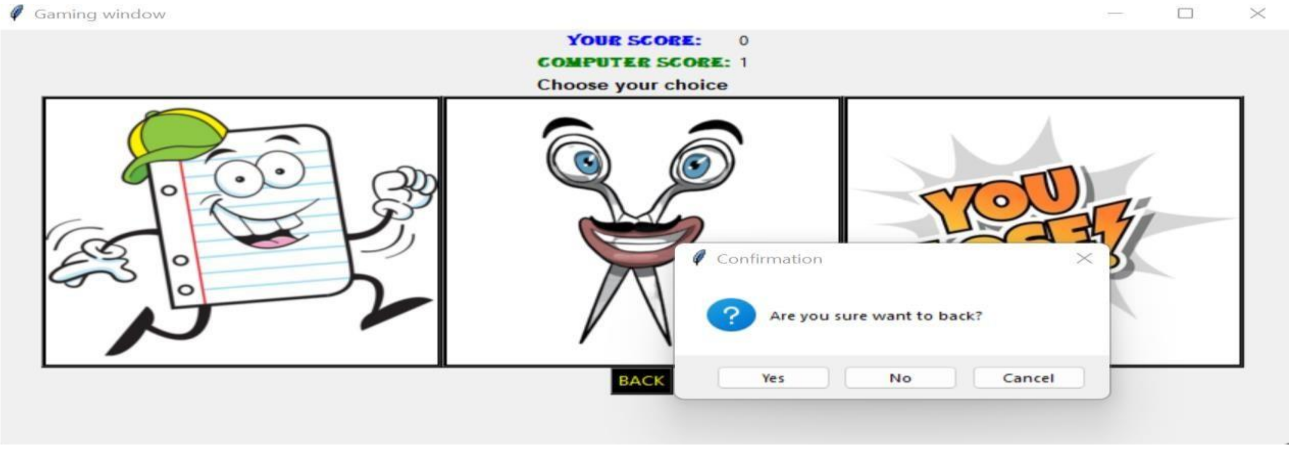
COMPUTER SCORE: 0

Choose your choice

Confirmation

Are you sure want to exit?

Fig 5.1.3 Output to exit the Game



YOUR SCORE: 0

COMPUTER SCORE: 1

Choose your choice

Confirmation

Are you sure want to back?

Fig 5.1.4 Output to back the Game

5.2 DISCUSSION

In Fig 5.1.1, if player choose paper and computer chooses the rock then the result shows / output the player is win and asks do you want play again? In Fig 5.1.2, if player choose paper and computer chooses the scissor then the result shows / output the player is loose and asks do you want play again? In Fig 5.1.3, if do you want to think exit/end the game then click on exit button and conform to exit the game. In Fig 5.1.4, if do you want to think back the game then click on back button and conform to back the game

Rock paper scissors game is also known as stone paper scissors. It is a hand game that is usually played between 2 people, each player can randomly form any one of three from their hand. A player who chooses rock will win by another player who chooses scissors but loose by the player who chooses paper; a player with paper will loose by the player with the scissors. If both players choose the same then the game is tied. Rock paper scissors game is mainly played among kids. We will see how we can create a rock paper and scissor game using Tkinter. Rock paper scissor is a hand game usually played between two people, in which each player simultaneously forms one of the three shapes with an outstretched hand. These shapes are “rock”, “paper”, and “scissors

5.3 APPLICATION

- The Rock, Paper & Scissor game illustrates the basic principle of adaptive artificial intelligence technology. The system learns to identify patterns of a person’s behavior by analyzing their decision strategies in order to predict future behavior.
- Many card-based video games in Japan use the rock paper scissors system as their core fighting system, with the winner of each round being able to carry out their designated attack

CHAPTER 6

CONCLUSION

In this way, I have Successfully developed **Rock paper scissor** game using python **tkinter** which is a hand game usually played between two people, in which each player simultaneously forms one of three shapes with an outstretched hand. These shapes are “rock”, “paper”, and “scissors”.

In conclusion, the Rock Paper Scissors game does not follow difficult play patterns and does not conform to complex and unorthodox customs like that of the **Olympics**. In other words, it offers simplicity in practice. You can play the Rock Paper Scissors game as an amateur to have fun and subscribe to the honor that it brings. You can also pursue a career in it by becoming a **professional RPS player** like in every other professional sport globally.

So, this was an easy and fun way to create a rock paper scissors game. It is customizable, as per a **developer's personal preference**. Not just rock paper scissors, but many more games can be developed easily in Python using various tools and libraries available. The end strategy of a simple Rock Paper Scissors game is to be **random and fast**. Statistically, each attack will tend to occur just as frequently as another, given that each is equally effective.

REFERENCE

- https://www.askpython.com/python/examples/rock-paper-scissors-in-python
- https://data-flair.training/blogs/python-rock-paper-scissors-game/
- https://www.youtube.com/watch?v=GhPZHvhvlsk
- https://tutorialslink.com/Articles/Rock-Paper-Scissors-in-Python/2500