Report On

ROCK - PAPER – SCISSOR

Submitted in partial fulfillment of the requirements of the Course project in Semester IV of Second Year Computer Engineering

by
Kunj Vadhia (Roll No. 56)
Kshitij Vyas (Roll No. 59)
Anujkumar Yadav (Roll No. 61)

Supervisor
Mrs. Sneha Mhatre

University of Mumbai

Vidyavaridhi's College of Engineering &

Technology Department of Computer

Engineering

(2023-24)

Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

CERTIFICATE

This is to certify that the Mini Project entitled “Rock-Paper-Scissor” is a bonafide work of Kunj Vadhia (Roll No.56) Kshitij Vyas (Roll No.59) AnujKumar Yadav (Roll No.61) submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of **“Bachelor of Engineering”** in Semester IV of Second Year **“Computer Engineering” .**

Prof. Sneha Mhatre
Mentor

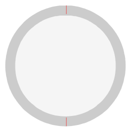
Dr Megha Trivedi
Head of Department

Dr. H.V. Vankudre
Principal

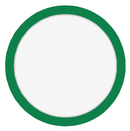
III. PLAGLARISM SCAN REPORT

Dupli Checker

PLAGIARISM SCAN REPORT



0%
Plagiarised



100%
Unique

Date	2024-04-18
Words	525
Characters	4927

Content Checked For Plagiarism

```
# Reset The Game


def reset_game():

      b1["state"] = "active"
                                b2["state"] = "active"
                                b3["state"] = "active"
                                l1.config(text="Player ")
                                l3.config(text="Computer")
                                l4.config(text="")

# Disable the Button


def button_disable():
                       b1["state"] = "disable"
                                b2["state"] = "disable"
                                b3["state"] = "disable"

# If player selected rock
```

```
def isrock():


            c_v = computer_value[str(random.randint(0, 2))]
                                                                                                                                        if c_v == "Rock":
                                  match_result = "Match Draw"
                                                    elif c_v == "Scissor":
                                       match_result = "Player Win"
                                                    else:


match_result = "Computer Win"
                                                    l4.config(text=match_result)
                                                    l1.config(text="Rock ")
           l3.config(text=c_v)
         button_disable()


# If player selected paper



def ispaper():


            c_v = computer_value[str(random.randint(0, 2))]
                                                                                                                                       if c_v == "Paper":
                                   match_result = "Match Draw"
                                                    elif c_v == "Scissor":
                                       match_result = "Computer Win"
                                                  else:


match_result = "Player Win"
                                                    l4.config(text=match_result)
                                                    l1.config(text="Paper ")
                                                    l3.config(text=c_v)
         button_disable()


# If player selected scissor


def isscissor():
```

```
                c_v = computer_value[str(random.randint(0, 2))]
                                                                                                   if c_v == "Rock":
                                   match_result = "Computer Win"
                                              elif c_v == "Scissor":
                                      match_result = "Match Draw"
                                              else:

 match_result = "Player Win"
                                              l4.config(text=match_result)
                                              l1.config(text="Scissor ")
                                              l3.config(text=c_v)
            button_disable()


# Add Labels, Frames and Button


Label(root,


              text="Rock Paper Scissor",
                                              font="normal 20 bold",
                                              fg="blue").pack(pady=20)


frame = Frame(root)
                                   frame.pack()


l1 = Label(frame,
                                text="Player ",


 font=10)


l2 = Label(frame,


 text="VS ",


 font="normal 10 bold")
```

```
l3 = Label(frame, text="Computer", font=10)

l1.pack(side=LEFT)


l2.pack(side=LEFT)
                            l3.pack()


l4 = Label(root,


 text="",


 font="normal 20 bold",
                                    bg="white",


 width=15,


 borderwidth=2,
                         relief="solid")


l4.pack(pady=20)


frame1 = Frame(root)
                                frame1.pack()


b1 = Button(frame1, text="Rock",
                                              font=10, width=7,


 command=isrock)
```

```
b2 = Button(frame1, text="Paper ",  font=10,
width=7,
                              command=ispaper)

b3 = Button(frame1, text="Scissor",
                                                        font=10,
width=7,
                              command=isscissor)

b1.pack(side=LEFT, padx=10)
                                              b2.pack(side=LEFT,
padx=10)
                                              b3.pack(padx=10)

Button(root, text="Reset Game",

              font=10, fg="red",
                                          bg="black",
command=reset_game).pack(pady=20)

# Execute Tkinter
                           root.mainloop()
```

Matched Source

No plagiarism found

IV. Abstract

The Rock-Paper-Scissors (RPS) game has long intrigued scholars, mathematicians, and enthusiasts alike due to its deceptively simple rules yet intricate strategic possibilities. This abstract delves into the dynamics and strategies of the RPS game, examining its significance in various fields including game theory, psychology, and artificial intelligence.The Rock-Paper-Scissors game serves as more than just a pastime; it is a rich domain for exploring strategic interactions, human behavior, and computational algorithms. By unraveling its intricacies, we not only deepen our understanding of fundamental concepts in various disciplines but also uncover practical insights applicable across diverse domains.

Content **Pg No.**

1.Introduction

The game of Rock-Paper-Scissors (RPS) stands as a timeless testament to the captivating allure of simplicity in games. Originating from ancient China and spreading globally, RPS exemplifies a deceptively straightforward yet profoundly engaging form of entertainment. Its rules are elementary: each player simultaneously chooses one of three gestures—rock, paper, or scissors—with the outcome determined by a simple hierarchy where each gesture defeats one and falls to another. Despite its apparent simplicity, RPS has become a subject of fascination for researchers, educators, and enthusiasts alike. Beyond its surface-level amusement, the game serves as a rich platform for exploring strategic decision-making, psychological nuances, and computational algorithms. This introduction sets the stage for delving into the multifaceted dimensions of the RPS game, from its historical roots to its modern-day applications in diverse fields.

2. Problem Statement

The Rock-Paper-Scissors (RPS) game presents an intriguing challenge in understanding strategic decision-making and predicting outcomes in competitive interactions. Despite its apparent simplicity, the game raises questions regarding optimal strategies, behavioral patterns, and the influence of cognitive biases on player choices. This problem statement aims to explore these complexities and develop insights into the dynamics of RPS gameplay. Additionally, it seeks to investigate how advancements in artificial intelligence and game theory can be leveraged to enhance our understanding of RPS and its broader implications in fields such as psychology, education, and conflict resolution. By addressing these challenges, this study aims to shed light on the underlying mechanisms of decision-making in simple yet profound strategic contexts like RPS.

3. MODULE DESCRIPTION

1. Player Input:
    - The module facilitates player interaction by accepting input from two participants, typically represented as human players or AI agents.
    - Players are prompted to select their gesture from the available options: rock, paper, or scissors.
    - Input validation ensures that only valid gestures are accepted, enhancing the reliability and integrity of the gameplay.
2. Game Logic:
    - The core functionality of the module revolves around the implementation of the RPS game logic.
    - It establishes the hierarchical relationships between the gestures (rock defeats scissors, scissors defeats paper, paper defeats rock) to determine the winner of each round.
    - The module includes algorithms for resolving ties and determining overall game outcomes based on the sequence of player gestures.
3. Outcome Determination:
    - After both players have made their selections, the module calculates the outcome of the round based on the established rules of RPS.
    - It identifies the winning gesture and declares the corresponding player as the round winner.
    - In the case of a tie, the module notifies the players and prompts them to make new selections for the next round.
4. Result Visualization:
    - The module offers visual feedback to enhance the user experience and provide clarity on game outcomes.
    - It displays the gestures chosen by both players for each round and highlights the winner.
    - Additionally, the module may include features such as score tracking to maintain a record of wins, losses, and ties throughout the game session.

4. SOFTWARE / HARDWARE

Software:

1. Programming Language: The Rock Paper Scissors game can be implemented using programming languages such as Python or any other language capable of handling user input and implementing game logic.

2. Integrated Development Environment (IDE): Developers can use IDEs like PyCharm, Visual Studio Code, Eclipse, or IntelliJ IDEA to write, debug, and test their code efficiently.

3. Graphics Libraries: If developing a graphical version of the game, graphics libraries like Pygame (Python), SDL (C++), or HTML5 Canvas (JavaScript) can be utilized to create the game interface and visual elements.

Hardware:

1. Computer or Mobile Device: The game can run on any standard computer or mobile device capable of running the chosen programming language and executing the game code.
2. Input Devices: Players can interact with the game using input devices such as keyboards, mice, touchscreens, or game controllers, depending on the platform and implementation.
3. Display Screen: A display screen is required to render the game interface and provide visual feedback to the players. This can be a monitor, laptop screen, smartphone display, or tablet screen.
4. Optional: If developing a physical version of the game, such as a Rock Paper Scissors robot, additional hardware components like microcontrollers, sensors, actuators, and motors may be required to build and control the physical game elements.

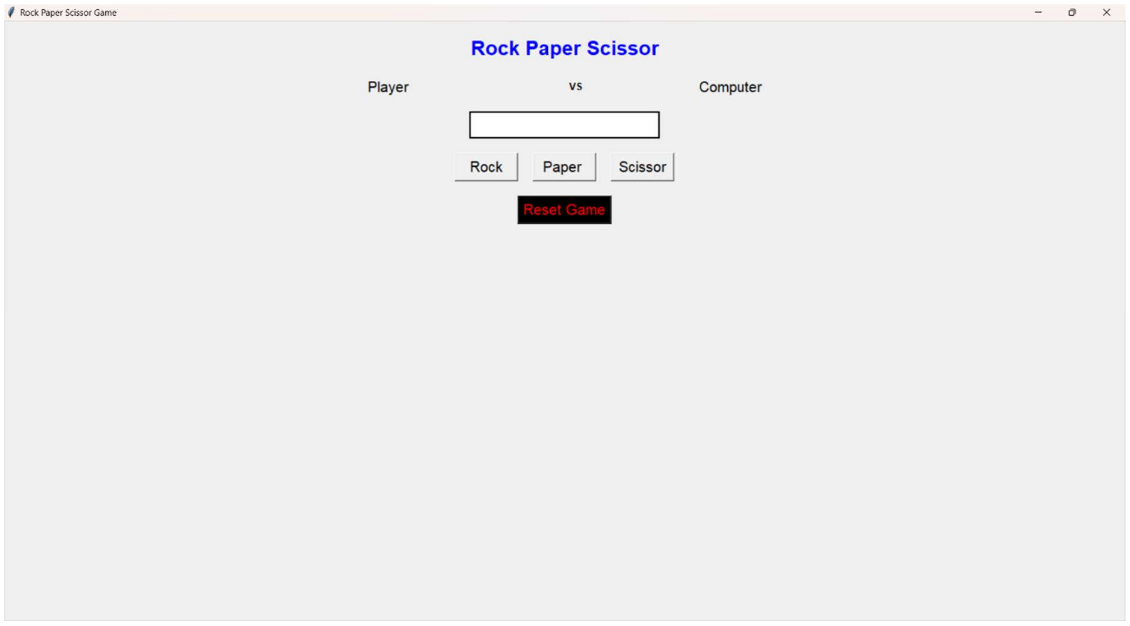
5. CODE

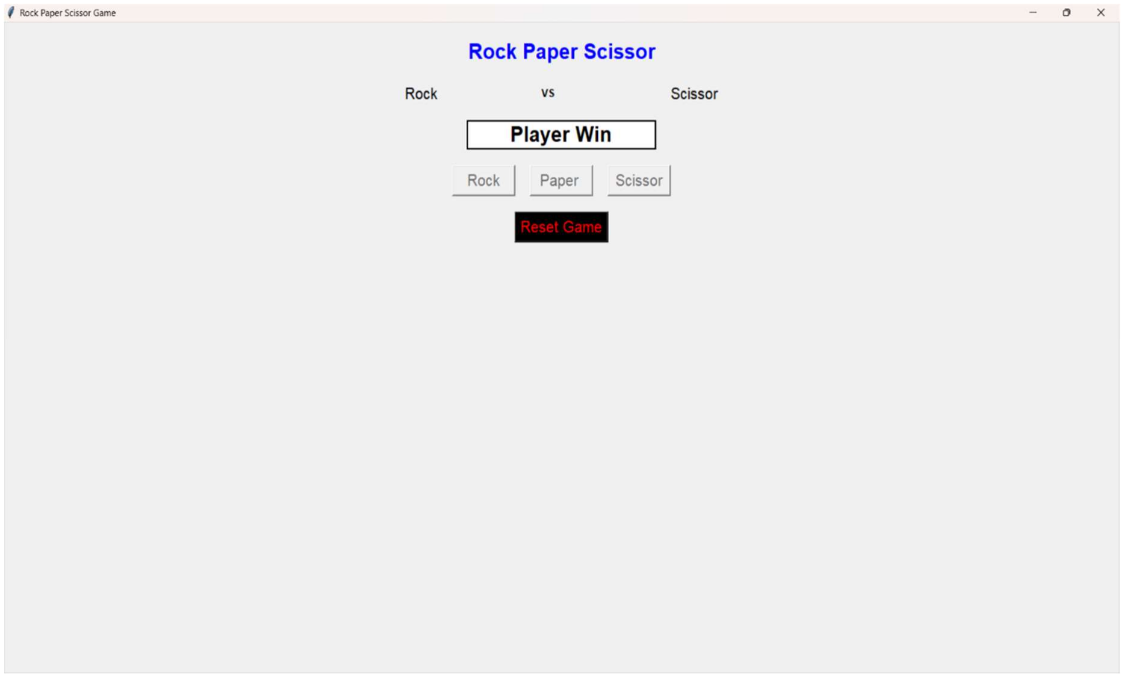
```
# Import Required Library
from tkinter import *
import random

# Create Object
root = Tk()

# Set geometry
root.geometry("300x300")

# Set title
root.title("Rock Paper Scissor Game")

# Computer Value
computer_value = {
	"0": "Rock",
	"1": "Paper",
	"2": "Scissor"
}

# Reset The Game


def reset_game():
	b1["state"] = "active"
	b2["state"] = "active"
	b3["state"] = "active"
	l1.config(text="Player                    ")
	l3.config(text="Computer")
	l4.config(text="")

# Disable the Button


def button_disable():
	b1["state"] = "disable"
	b2["state"] = "disable"
```

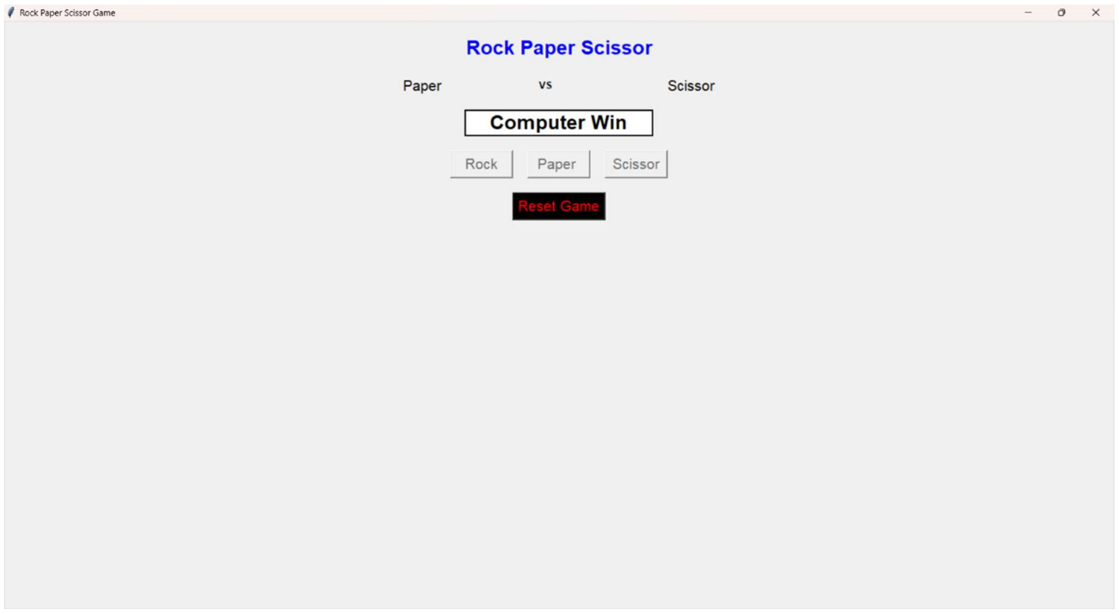
```
        b3["state"] = "disable"

# If player selected rock


def isrock():
        c_v = computer_value[str(random.randint(0, 2))]
        if c_v == "Rock":
                match_result = "Match Draw"
        elif c_v == "Scissor":
                match_result = "Player Win"
        else:
                match_result = "Computer Win"
        l4.config(text=match_result)
        l1.config(text="Rock                ")
        l3.config(text=c_v)
        button_disable()

# If player selected paper


def ispaper():
        c_v = computer_value[str(random.randint(0, 2))]
        if c_v == "Paper":
                match_result = "Match Draw"
        elif c_v == "Scissor":
                match_result = "Computer Win"
        else:
                match_result = "Player Win"
        l4.config(text=match_result)
        l1.config(text="Paper              ")
        l3.config(text=c_v)
        button_disable()

# If player selected scissor


def isscissor():
        c_v = computer_value[str(random.randint(0, 2))]
        if c_v == "Rock":
                match_result = "Computer Win"
        elif c_v == "Scissor":
                match_result = "Match Draw"
        else:
                match_result = "Player Win"
        l4.config(text=match_result)
        l1.config(text="Scissor                    ")
        l3.config(text=c_v)
        button_disable()


# Add Labels, Frames and Button
Label(root,
        text="Rock Paper Scissor",
        font="normal 20 bold",
        fg="blue").pack(pady=20)

frame = Frame(root)
frame.pack()

l1 = Label(frame,
                text="Player                    ",
                font=10)

l2 = Label(frame,
                text="VS                        ",
                font="normal 10 bold")

l3 = Label(frame, text="Computer", font=10)

l1.pack(side=LEFT)
l2.pack(side=LEFT)
l3.pack()

l4 = Label(root,
                text="",
                font="normal 20 bold",
                bg="white",
                width=15,
                borderwidth=2,
                relief="solid")
l4.pack(pady=20)

frame1 = Frame(root)
```

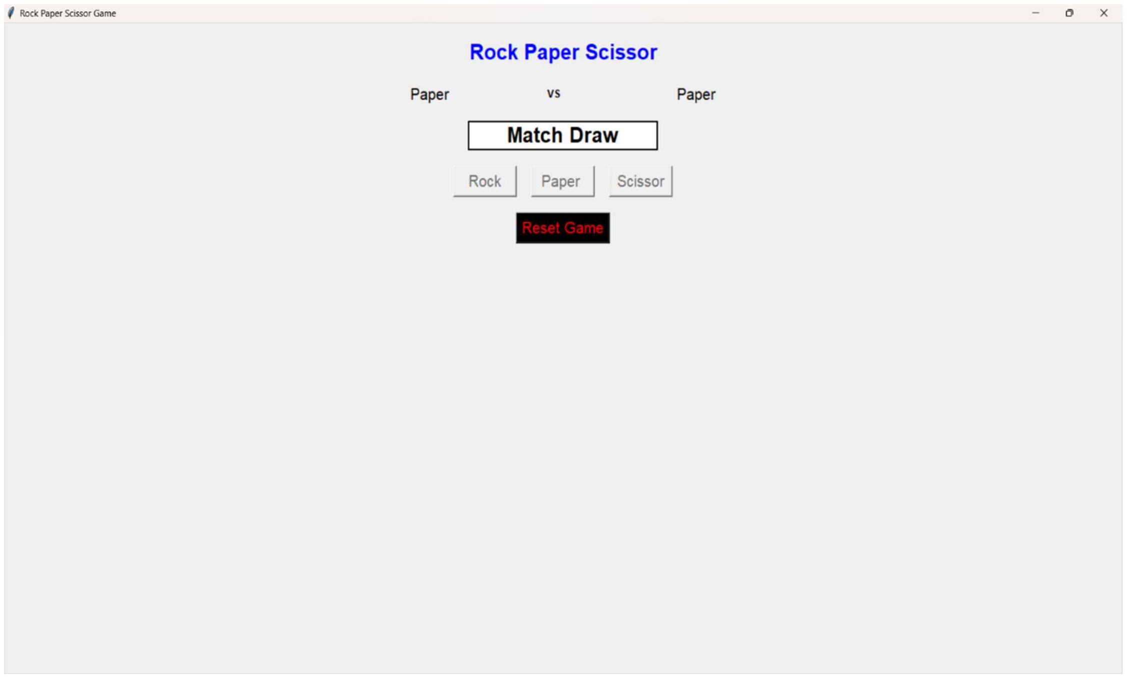
```
frame1.pack()

b1 = Button(frame1, text="Rock",
                              font=10, width=7,
                              command=isrock)

b2 = Button(frame1, text="Paper ",
                              font=10, width=7,
                              command=ispaper)

b3 = Button(frame1, text="Scissor",
                              font=10, width=7,
                              command=isscissor)

b1.pack(side=LEFT, padx=10)
b2.pack(side=LEFT, padx=10)
b3.pack(padx=10)

Button(root, text="Reset Game",
        font=10, fg="red",
        bg="black", command=reset_game).pack(pady=20)

# Execute Tkinter
root.mainloop()
```

6. RESULT AND CONCLUSION






Rock Paper Scissor Game
Rock Paper Scissor
Paper
vs
Scissor
Computer Win
Rock
Paper
Scissor
Reset Game


Rock Paper Scissor Game
Rock Paper Scissor
Paper
vs
Paper
Match Draw
Rock
Paper
Scissor
Reset Game

7. CONCLUSION

The Rock Paper Scissors (RPS) game, despite its simple premise, holds significant value as both a recreational pastime and a subject of academic inquiry. Through this exploration, we've uncovered the multifaceted dimensions of RPS, spanning strategic decision-making, behavioral psychology, computational algorithms, and even physical robotics. From its ancient origins to its modern-day manifestations in digital and physical forms, RPS continues to captivate players and researchers alike.

In conclusion, the Rock Paper Scissors game stands as a timeless testament to the enduring appeal of simplicity and strategy. Whether enjoyed as a casual diversion or analyzed as a microcosm of decision-making dynamics, RPS continues to inspire curiosity, creativity, and camaraderie across cultures and generations. As we continue to unravel its mysteries and leverage its insights, RPS reminds us that even the simplest of games can hold profound lessons for understanding ourselves and the world around us.

8. REFERENCE

[1] Learn Python the Hard Way, 3rd Edition, Zed Shaw's Hard Way Series.

[2] Laura Cassell, Alan Gauld, “Python Projects”, Wrox Publication.