**PROJECT SYNOPSIS**

**ON**

**ROCK-PAPER-SCISSOR GAME**

**SUBMITTED BY**

**RIMJHIM SAXENA**

****

**SUBMITTED TO**

**PIEINFOTECH**

**TABLE OF CONTENT**

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Topic** | **Page No.** |
| 1. | Introduction | 3 |
| 2. | Objective | 4-5 |
| 3. | Background | 6-8 |
| 4. | Hardware and Software Requirements | 9 |
| 5. | Coding | 10-12 |
| 6. | Output Screenshot | 13 |
| 7. | Future Scope | 14-15 |
| 7. | Future Scope | 16 |
| 9. | References and Bibliography | 17 |

**INTRODUCTION**

Games were an essential aspect of civilisational development from the earliest days and are a form factor that crosses countries and cultures; vibrant, engaging, fun, collaborative, strategic, and creative. The evolution of gaming, from early board games to contemporary video games, mirrors the larger technological progress of society. One of those is Rock-Paper-Scissors, a simple game that has been used for centuries to produce a fair resolution to a dispute or event while also allowing the participants to enjoy an agreed play space.

Its universal appeal lies in its simplicity. In a contest played with just three choices — rock, paper and scissors — players compete according to an uncomplicated rule: Rock crushes scissors, scissors cut paper and paper covers rock. The game is simple but adds an exciting level of strategy and an element of randomness and psychological play that are, to this day, classic game elements.

Modern versions of old-fashioned games are designed as software applications for people on the go. Rock-Paper-Scissors game in particular, is an exercise in computer programming that has gained popularity for its versatility as an accessible project while teaching core programming principles like conditional statements, randomisation, and interactions with the user.

In this project, we will take the game Rock-Paper-Scissors and code it up in Python, a powerful, widely used, and beginner-friendly language. The application uses a graphical user interface (GUI) created with Python's Tkinter library. This library enables the creation of interactive interfaces, which makes it popular for building games and applications with user interaction.

The main goal of this project is to create a working Rock-Paper-Scissors game for a user to play against a computer. The choice of computer is decided using the random module (completely random and fair in Python). This app handles the scoreboard of the user as well as the computer so that the user feels like a gamer.

It's not only a project to make a game but also a working example of the implementation of programming concepts. In writing the code for this application, you will learn how code can be translated into the execution of logical operations visibly and interactively. The project includes topics like global variables, event-driven programming, and GUI.

Another great place for newbie Python/programming starters is the Rock-Paper-Scissors project. It is a good challenge to logical and creative thinking, and it is not too challenging to keep from being frustrated. Users can further extend a simple project into an application that does quite a lot with additional features (animations, sound effects) and the like as they get better.

Additionally, this project shows how an ancient game can be transformed into a more contemporary online game for mass appeal. The game could be further extended to allow multiplayer, platform web/mobile, and even AI against different user strategies.

With this project, we hope to close the distance between classic entertainment and the digital age and enjoy our minds working.**OBJECTIVE**

1. Design the UI for the user to play the game:

* The Tkinter module of Python is used to create a graphical user interface (GUI); this helps in making our game interactive and fun, creating an amazing game experience.
* Keep your navigation simple and buttons so you select Rock/Paper/Scissors.

2. Do Basic Game Logic:

* This code demos the game of Rock-Paper-Scissors: rock crushes scissors, paper covers scissors, scissors cut paper.
* 15. Facilitate modular programming that enables organizing the logic in an organized and reusable manner.

3. Player vs. Computer Play:

* Use the `random random` module in Python to randomly choose an answer for the computer so that it would not be biased.
* Develop a basic player for live interaction with user-cypher. The experience is going to reduce monotony.

4. Record Game Score:

* Global Variables (unanchored) Useful to store user and computer scores for the lifetime of the session
* Dynamic interface refreshes the scores every time after a round for that competitive spark in you.

5. Respond to Gameplay:

Either a visual/textual feedback mechanism that tells both players and their respective opponents what has decided the outcome in every round (win, loss, or tie)

6. Lay the Groundwork for Future Improvements:

* Give a solid footing for future feature implementations like:
* Competition through real-time multiplayer mode for multiple users.
* Animate user experience features as well.
* Add hard difficulty to make the game more challenging.
* More immersive sound effects

7. Develop a Scalable Application at the Technological Level:

* Make sure that the code architecture scales to different places of deployment, i.e., mobile devices or web browsers as well.
* With this project, advance towards implementing artificial intelligence (AI) and really smart gameplay by adapting to user patterns in strategic decisions.

8. Motivate for Innovation in Technology:

* Motivate the user and developers to see a creative combination between traditional games and modern programming.
* Show that basic is turning into good software solutions.

**BACKGROUND**

1. Rock-Paper-Scissors in History:

* According to one version of the etymology, the game Rock-Paper-Scissors was called in ancient China during the Han Dynasty (206 BCE — 220 CE) under the name shoushiling.
* It travelled to Japan and became jan-ken, with gestures close to but not identical to modern ones.
* A Tokens-for-a-decision; is the same as tossing a coin or pulling straws.
* And its simplicity has meant it transcends generations and continents

2. Evolutions in Present Times:

* The game has also been converted to digital, such as mobile games, the cloud and microservices.
* There exist even world-level Rock-Paper-Scissors competitions focused on strategy and cognition.

3. Educational Applications:

* Code Smile is useful as an example to illustrate common programming concepts in the game Rock-Paper-Scissors, such as:
* Conditional logic (if-elif-else statements).
* Random number generation.
* Handling events for graphical user interfaces.
* Rock-Paper-Scissors outcomes analysis and strategy is how players start to learn about simple probability.

4. Technological Context:

* Simple Rules—Making the game so there are simple programming projects for novices.
* Belies the strength of automation in decision-making.
* Combining logical programming with user-centric design in between projects was achieved by the development of a graphical version of our game using Python's TKinter.
* Python's random module will be used for the computer to get a random number that embeds (at least among other things) fairness into how the computer "chooses," needed for the unbiasedness of the game.

5. Learn Python and Tkinter:

* Python for small-scale projects (like Rock-Paper-Scissors) is an understandable, easy-to-start programming language.
* Random for the unpredictability and libraries such as Tkinter for making GUIs
* Standard lib Python is used to create a simple and interactive GUI.
* Games and applications need widgets like buttons, labels, text areas, etc.
* The event-based programming paradigm of Tkinter is reliable in the case of turn-based gaming, as in the example of Rock-Paper-Scissors.

6. Game Logic Design:

* Rock crushes scissors, scissors beat paper, and paper covers rock.
* The very first thing you need to do when writing a game is make sure these rules are implemented properly.
* (Translating user's choices, for example, "rock," into numeric values so that my code can easily compare them)
* Some of this workflow is automated by choice\_to\_number and number\_to\_choice.
* Comparing user & computer choices to decide the result of each round.
* The winner with modulo arithmetic simplified logic.

7. User Experience (UX) Design:

* The game is designed so that buttons are used to select rock, paper, and scissors on the interface as a user interaction point.
* When dynamically maintaining and showing scores, user engagement can be improved.
* Immediate feedback on the result of each round will make players receive the feedback, and hence it will be the best part of the gaming.

8. Outside of Gaming Applications:

* Rock-paper-scissors become the base of decision-making algorithms used in software.
* With AI, the integration could be done by training models to assess trends in user selection for counter-strategies, thus improving gameplay.
* It is simple, so this is a great way of having games featuring psychology studies and how people will make decisions.

9. Learning Outcomes from the Project

* Learn the basics of conditional logic, random, and GUI (with the help of Tkinter).
* Dissecting code and finding efficient algorithms to deduce game output
* Building an interface for usability and engagement first.

10. Challenges in Development:

* Ensure the game's game logic does not contain any bugs and is implemented correctly.
* Creating a simple and beautiful interface is often difficult in the beginning.
* Ensuring that the software choices are random and not slanted.

11. On The Relevance of Technological Trends:

* Interactive Rock-Paper-Scissors games have become a more common practice in educational platforms to gamify learning.
* These projects make room for input/modification by the programming community, a kind of pressure to do things right and innovate.

12. Future Prospects:

* Expanding in gaming for multiplayer play over a network.
* Use machine learning algorithms to make the game so challenging that it learns from users' behaviour and adapts.
* Creating mobile and web versions to attract more audience.

**HARDWARE AND SOFTWARE REQUIREMENTS**

**Hardware Requirements:**

1. Processor: Intel Core i3 or higher

2. RAM: 4 GB minimum

3. Storage: 500 MB of free disk space

4. Display: 1024x768 resolution or higher

5. Input Devices: Keyboard and mouse

6. Power Supply: Standard laptop or desktop power supply

7. OS Compatibility: Windows/Linux/MacOS

**Software Requirements:**

1. Operating System: Windows 10 or higher

2. Python: Version 3.8 or later

3. IDE: PyCharm/VS Code/IDLE/Jupyter Notebook

4. Libraries: Tkinter, random

**CODING**

import random

import tkinter as tk

window = tk.Tk()

window.geometry("500x500")

window.title("Rock Paper Scissors Game")

USER\_SCORE = 0

COMP\_SCORE = 0

USER\_CHOICE = ""

COMP\_CHOICE = ""

def choice\_to\_number(choice):

rps = {'rock':0,'paper':1,'scissor':2}

return rps[choice]

def number\_to\_choice(num):

rps={0:'rock',1:'paper',2:'scissor'}

return rps[num]

def random\_computer\_choice():

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

def result(human\_choice,comp\_choice):

global USER\_SCORE

global COMP\_SCORE

user=choice\_to\_number(human\_choice)

comp=choice\_to\_number(comp\_choice)

if(user==comp):

print("Tie")

elif((user-comp)%3==1):

print("You win")

USER\_SCORE+=1

else:

print("Comp wins")

COMP\_SCORE+=1

text\_area = tk.Text(master=window,height=12,width=30,bg="#73a99e")

text\_area.grid(column=0,row=4)

answer = "RIMJHIM's Choice:{uc}\nComputer's Choice:{cc}\n RIMJHIM's Score:{u}\n Computer Score:{c}".format(uc=USER\_CHOICE,cc=COMP\_CHOICE,u=USER\_SCORE,c=COMP\_SCORE)

text\_area.insert(tk.END,answer)

def rock():

global USER\_CHOICE

global COMP\_CHOICE

USER\_CHOICE='rock'

COMP\_CHOICE=random\_computer\_choice()

result(USER\_CHOICE,COMP\_CHOICE)

def paper():

global USER\_CHOICE

global COMP\_CHOICE

USER\_CHOICE='paper'

COMP\_CHOICE=random\_computer\_choice()

result(USER\_CHOICE,COMP\_CHOICE)

def scissor():

global USER\_CHOICE

global COMP\_CHOICE

USER\_CHOICE='scissor'

COMP\_CHOICE=random\_computer\_choice()

result(USER\_CHOICE,COMP\_CHOICE)

button1 = tk.Button(text=" Rock ",bg="pink",command=rock)

button1.grid(column=0,row=1)

button2 = tk.Button(text=" Paper ",bg="lightgreen",command=paper)

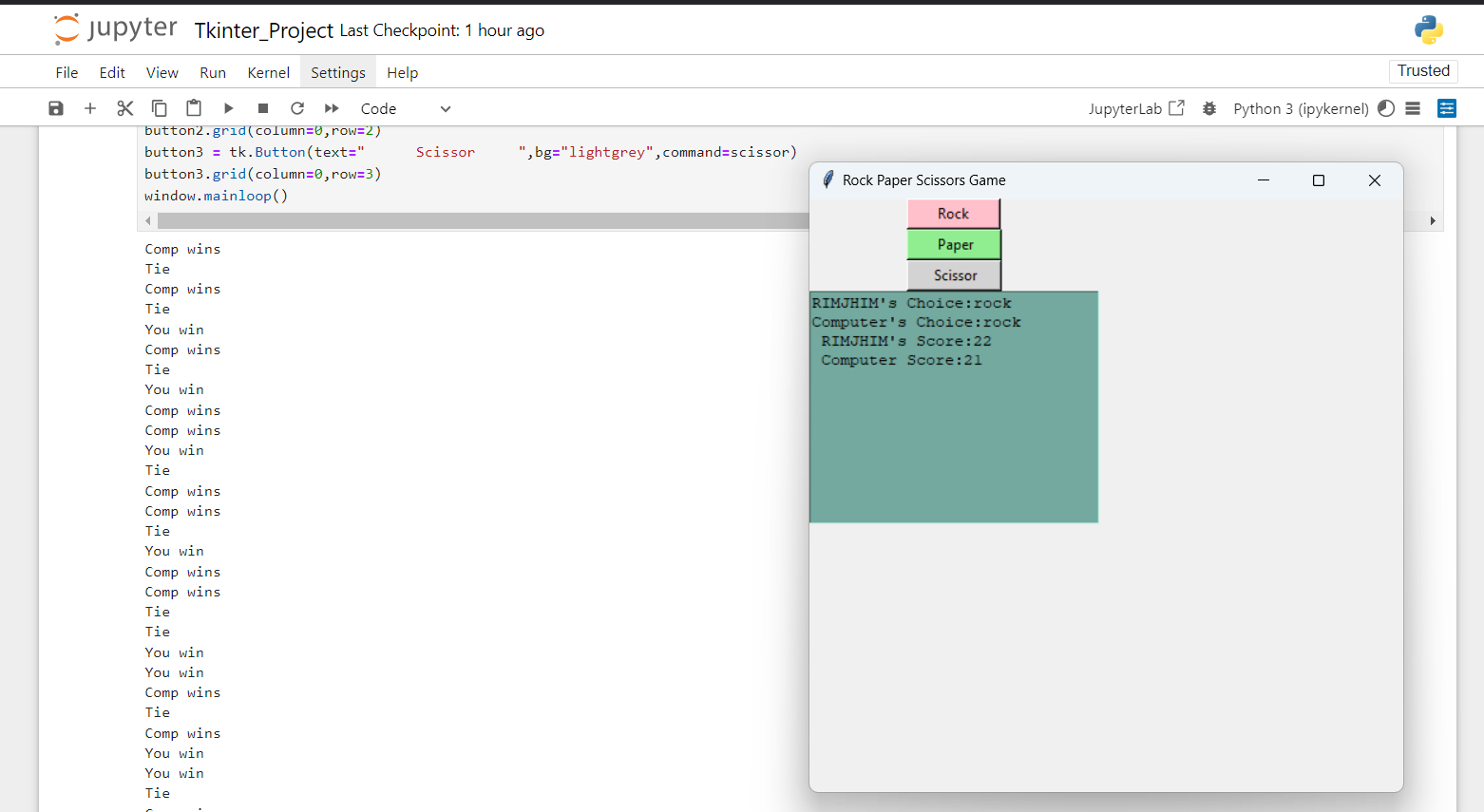
button2.grid(column=0,row=2)

button3 = tk.Button(text=" Scissor ",bg="lightgrey",command=scissor)

button3.grid(column=0,row=3)

window.mainloop()

**OUTPUT SCREENSHOT**



**FUTURE SCOPE**

1. Integration of Multiplayer Mode:

* Create online multiplayer abilities to engage with players on a network.
* Create a traditional real-time gameplay & score synchronization server-client architecture.

2. Better AI and Computer Opponent on Enhanced Mode:

* Use a more clever artificial intelligence to have the computer opponent be wiser and much more difficult.
* Machine Learning for evolving adaptive AI, with its strategy-based decision-making and game tweaking based on user strategies.

3. Mobile and Web Deployment:

* Packaging the game for both mobile (iOS, Android)
* Turn a game into an online app now that can be directly played using a browser without installing.
* Employ Flask for the web version and React Native for the mobile app version.

4. Graphics and Updates to the UI:

* Improve the game with animations, dynamic backgrounds, and sound effects (game visuals).
* Users will be able to choose different themes, which would provide a more user-friendly experience.

5. Expansion beyond Other Standard Games:

* Program variants of other games whose game structure is similar (for example, in Tic-Tac-Toe, Connect Four, or Chess).
* Bring in a collection of games all using the same engine so the utility is broadened.

6. Gamification Elements:

* Include achievements (or stars if you are from SOLO), badges, and leaderboards to indicate a player's performance to generate competitiveness.
* Do what we do for daily challenges and quests; something can make it interesting as a game and could be addictive enough to replay it.

7. Social Media Integration:

* To enable users to share their scores and achievements with social media.
* Create a social/hangout feature where users can talk with each other, trade tactics, and play tournaments.

8. Cross-Platform Multiplayer and Data Synchronization:

* Allow cross-platform play so that players will be able to start a game with one device, and co-players can join on different devices.
* Use cloud-based storage to store game data and enable users to not lose a game automatically on switching devices.

9. Accessibility Features:

* Enhance the function of the gaming system to be more accessible for a person with disabilities by utilizing voice commands, screen readers, or changing keyboard shortcuts.
* Introduce adaptive difficulty levels to meet players with different skill levels.

10. On-Going Support and Updates:

* Continuously release new content (campaigns and others) into the game and fix bugs based on our user feedback.
* Keep a snapping development roadmap in place that keeps the game relevant and interesting to potential players.

**CONCLUSION**

The Rock-Paper-Scissors project is a great example of marrying traditional games with contemporary coding and digital technologies, illustrating how something relatively simple can be turned into a strong educational tool and immersive digital playground.

This project encapsulates a classic into a digital form, but it also helps to get acquainted with the fundamentals of programming—that would be perfect for absolute beginners in Python and Tkinter (for those who want just an intro to GUIs + some great algorithm examples).

This project (developed throughout) was to create an interactive or at least engaging game using the fundamentals of programming (logic, decisions, and randomness). In a minimalistic way, the project uses the Tkinter library for Python, and to generate a computer choice, it utilizes the random module, demonstrating the basics of game development. Tracking scores, providing instant feedback, and balancing simplicity with difficulty—the practical uses of programming that create functional, user-centered software.

Furthermore, the Rock-Paper-Scissors project is the base for things related to gamification, user engagement, and interface design. It, for instance, shows how interactivity can be particularly well used to enrich user experience with achievements, leaderboards, and social media integrations. The elements serve to not just make the game fun but also build a community among players and build competition that increases its virality.

This project proves to us how easy concepts with modern programming techniques can make amazing results; it's evidence of how old games can be recreated and modernized. This project, as learners and developers work on this project, they aren't just learning new technical skills; they also are learning about the relationship between gaming and technology.

For better or worse, this project captures the harmonious collision of gaming and technology and provides a very cool educational element to expand on. Growing on this foundation, developers and teachers can advance to new heights in the still-young digital gaming industry and start doing things that would have been impossible with traditional games as dead-end projects for modern technology. This project is just the beginning, and enormous opportunities lie in the future of game design and software development revealed from the impact of this project.

**REFERENCES AND BIBLIOGRAPHY**

* Monty, A., Briggs, C. (2020). “Learning Python: Building Games and Visualizing Data”. This book offers practical examples of game development in Python, including tutorials on creating interactive applications like Rock-Paper-Scissors using Tkinter.
* Stack Overflow Community. (2024). “How to Implement Rock Paper Scissors in Python with Tkinter”. Available at: https://stackoverflow.com. This community-driven resource provides code snippets, solutions, and discussions relevant to the development of the Rock-Paper-Scissors game.
* Ko, A., Rose, L. (2011). “Programming Games with Python and Pygame”. This book covers fundamental concepts in game programming with Python that could be valuable for expanding the game beyond its current scope.
* Computer Learning and AI Integration: Further research focused on integrating artificial intelligence (AI) into Rock-Paper-Scissors game systems, where the computer learns player patterns and adjusts its choices accordingly. This AI model can predict future moves based on past player behavior, making the game more challenging for user. (IARJSET)

.