Rock - Paper - Scissors - Lizard - Spock : A Python-Based Interactive Game with Enhanced Decision Making

Project submitted to the

SRM University – AP

for the partial fulfilment of the requirements to award the degree of

Bachelor of Technology

In

Computer Science and Engineering

Submitted By

D. Pavan Kumar - AP22110010208
B. Niteesh Kumar - AP22110010232
P. Pranay Sai - AP22110010240

Under the Guidance of

Dr. Ram Baran Varma

SRM University - AP
Neerukonda, Mangalagiri, Guntur
Andhra Pradesh - 522240

April - 2024

Table of Contents

Rock - Paper - Scissors - Lizard - Spock : A Python-Based Interactive Game with Enhanced Decision - Making

Abstract

The classic Rock-Paper-Scissors game has been a popular choice for decision-making and entertainment for centuries. This project enhances the traditional game by integrating additional choices—Lizard and Spock—based on the extended version popularised by the TV series *The Big Bang Theory*. The game is developed using Python and implemented with a Graphical User Interface (GUI) using Tkinter. The interactive interface allows users to choose between five options, with the computer randomly selecting its move. The program determines the winner based on predefined rules, updating the score dynamically.

A core aspect of the project is its use of conditional logic to evaluate outcomes efficiently. The implementation ensures a fair probability distribution for computer choices, making the game both unpredictable and engaging. The GUI provides an intuitive experience, displaying real-time results and score tracking. Additionally, error handling mechanisms ensure smooth execution without runtime failures.

This project demonstrates fundamental programming principles, including event-driven programming, randomness, and GUI design. It serves as an excellent learning tool for beginners exploring Python and Tkinter. Moreover, it can be extended with additional features such as player statistics, difficulty levels, or multiplayer functionality. The simplicity and interactive nature of this game make it a valuable educational and recreational tool.

Keywords - Rock-Paper-Scissors, Lizard-Spock, Python GUI, Tkinter, Game Development, Randomisation, Event-Driven Programming, Interactive Gaming, Decision-Making Algorithms, User Interface Design

1. Introduction

1. 1. Overview of Rock-Paper-Scissors

Rock-Paper-Scissors (RPS) is a simple hand game that has been played for centuries across different cultures. The game is based on three primary choices—Rock, Paper, and Scissors—where each choice has a predefined interaction with the others. Rock crushes Scissors, Scissors cut Paper, and Paper covers Rock. This cycle of choices ensures a balanced system where no single option is dominant. The game's simplicity and randomness make it an effective decision-making tool in various scenarios.

Over the years, the game has evolved into a competitive format, even being used in professional tournaments and decision-making in sports and business settings. The game's fairness relies on the unpredictability of human choices, but the introduction of artificial intelligence and randomization techniques has further diversified its application. The standard RPS game follows a zero-sum outcome, meaning one player wins while the other loses, or it results in a draw.

In recent times, the game has been adapted into digital formats, making it more interactive and accessible. With the advent of programming languages like Python, developers have been able to create automated versions of the game. These digital implementations use random number generation and decision-making algorithms to simulate a fair and unbiased game experience. This project extends the traditional game by incorporating two additional choices—Lizard and Spock—offering a more complex yet entertaining variation.

1. 2. Evolution to Rock-Paper-Scissors-Lizard-Spock

While the classic RPS game has been widely recognized, it sometimes suffers from frequent ties due to the limited number of choices. To address this, an extended version known as Rock-Paper-Scissors-Lizard-Spock (RPSLS) was introduced by Sam Kass and Karen Bryla and later popularized by *The Big Bang Theory*. This variant expands the number of choices to five, reducing the chances of a tie and adding strategic complexity.

In RPSLS, Rock crushes Scissors and crushes Lizard, Paper covers Rock and disproves Spock, Scissors cut Paper and decapitate Lizard, Lizard eats Paper and poisons Spock, and Spock smashes Scissors and vaporizes Rock. These additional rules create a more intricate web of interactions, making the game more engaging and less predictable. The extended version introduces a level of strategy absent in the traditional three-choice system, requiring players to think beyond simple probabilities.

Implementing RPSLS in a digital format presents a challenge in structuring rules and logic while maintaining fairness. A well-designed program must account for all possible interactions while

ensuring randomness in the computer's choices. By integrating this extended version into a Python-based GUI, the game becomes not only more interactive but also an excellent tool for demonstrating fundamental programming concepts.

1. 3. Importance of Randomization in Game Mechanics

Randomization plays a crucial role in games like RPSLS to ensure fairness and unpredictability. In traditional gameplay, human psychology influences decision-making, often leading to patterns in choice selection. A well-designed digital version must counteract such biases by implementing truly random outcomes. Python’s random module enables this functionality by allowing the computer to select a move randomly from the five available choices.

A game that lacks proper randomization can become predictable, reducing its entertainment value. For instance, if a program unintentionally favors one option over others, players might recognize the pattern and exploit it. To avoid this, the program should ensure an equal probability distribution among the available choices. This maintains a balanced competition between the user and the computer.

Beyond entertainment, randomization in programming has broader applications, including cryptography, simulations, and artificial intelligence. Understanding how to implement randomness in a controlled environment is a fundamental skill for developers. This project’s approach to randomness demonstrates how simple algorithms can create engaging, fair, and replayable experiences for users.

1. 4. Introduction to GUI Development in Python

Graphical User Interfaces (GUIs) allow users to interact with software visually rather than through command-line inputs. Python’s Tkinter library provides a simple yet powerful framework for developing GUI applications. Unlike text-based interfaces, GUIs enhance user experience by making applications more intuitive and visually appealing.

For this project, Tkinter is used to create buttons for user interaction, display results dynamically, and keep track of scores. The interface includes options for selecting Rock, Paper, Scissors, Lizard, or Spock, with the computer generating a response in real-time. The outcome is displayed immediately, along with score updates, making the game engaging and interactive.

A well-structured GUI ensures that the game is accessible to players of all skill levels. By incorporating simple buttons, text labels, and dynamic result updates, users can easily understand and enjoy the game. The project highlights the importance of GUI design principles, such as

usability, layout management, and event-driven programming, which are essential for developing interactive applications.

1. 5. Event-Driven Programming in Python

Event-driven programming is a key concept in software development, particularly in GUI-based applications. Unlike traditional sequential programs, event-driven applications respond to user interactions, such as button clicks and keyboard inputs, triggering specific functions in response. Tkinter follows this event-driven model, allowing developers to link functions to user actions seamlessly.

In this project, each button click is associated with a function that processes the user's choice and determines the game outcome. The program listens for events and executes corresponding actions, making the game responsive. This structure enhances interactivity, ensuring that results are displayed instantly without requiring manual inputs beyond the initial selection.

Event-driven programming is widely used in modern software applications, from web development to mobile apps. Understanding how to handle user interactions effectively is crucial for developers. By implementing event-driven logic in this game, the project provides a practical demonstration of how real-world applications operate.

1. 6. Conditional Logic for Game Decision-Making

The core of the RPSLS game lies in its decision-making process. The program must compare user choices with the computer's selection and determine the winner based on predefined rules. This is achieved using conditional statements (if-elif-else), which allow the program to evaluate multiple conditions efficiently.

Each possible game outcome is mapped to specific conditions, ensuring that all scenarios are accounted for. For example, if the user selects Rock and the computer selects Scissors, the program recognizes Rock's advantage and awards a point to the user. Similarly, losing and draw conditions are handled through structured logic.

Conditional logic is a fundamental concept in programming, used in everything from game development to artificial intelligence. Implementing structured decision-making processes enhances computational efficiency and ensures accurate results. This project demonstrates the application of conditional logic in a fun and interactive way.

1. 7. Score Tracking and User Engagement

To enhance user engagement, the game includes a score-tracking feature. Each time a user wins or loses, the score is updated dynamically. This feature motivates players to continue playing and track their progress over multiple rounds.

Implementing score tracking requires maintaining a global variable that updates each time an event occurs. Tkinter's StringVar and IntVar functions facilitate real-time updates, ensuring that the user always sees the latest score. This adds an element of competition and personal achievement, making the game more appealing.

Score tracking is a simple yet effective feature commonly used in gaming applications. It introduces a progression system that encourages repeated gameplay. By implementing this feature, the project aligns with common practices in game development, making the experience more immersive.

1. 8. Potential Enhancements and Future Scope

While this project provides a functional RPSLS game, there is significant potential for future enhancements. Additional features such as multiplayer mode, AI-driven strategies, and difficulty settings could further enrich the gameplay experience.

One possible improvement is incorporating artificial intelligence to analyze player choices and adapt the computer's strategy accordingly. This would add an element of challenge, making the game more competitive. Additionally, implementing graphical animations or sound effects could enhance the visual appeal and interactivity of the game.

Beyond gaming, the principles applied in this project have real-world applications in areas such as decision-making models, probability studies, and AI-driven automation. By extending the project's capabilities, developers can explore more advanced concepts in programming and game theory.

2. Methodology

The development of the **Rock-Paper-Scissors-Lizard-Spock (RPSLS)** game followed a structured approach to ensure a smooth and efficient implementation. The methodology can be divided into three key stages: **Game Logic and Algorithm Design, GUI Development and Event Handling, and Randomization, Score Tracking, and Testing**. Each stage played a crucial role in creating an engaging and functional interactive game.

2. 1. Game Logic and Algorithm Design

The core of the game lies in its **decision-making algorithm**, which determines the winner based on the predefined rules. Unlike the traditional **Rock-Paper-Scissors** game, the **RPSLS** variation introduces two additional elements—**Lizard** and **Spock**—making the game more dynamic and reducing the probability of a tie.

Game Rules and Winner Determination

The winner is determined based on the following rules:

- **Rock** crushes Scissors and Lizard.
- **Paper** covers Rock and disproves Spock.
- **Scissors** cut Paper and decapitate Lizard.
- **Lizard** eats Paper and poisons Spock.
- **Spock** smashes Scissors and vaporizes Rock.

To implement this logic, a **decision tree** was developed, mapping all possible user and computer choices. The algorithm follows these steps:

1. The user selects a move using a button click.
2. The computer selects a move randomly.
3. The selected choices are compared based on predefined rules.
4. The program determines whether the user **wins, loses, or ties** and updates the output accordingly.

Algorithm Implementation in Python

A dictionary-based approach was used to efficiently compare selections. The relationships between moves were stored in a dictionary structure, reducing the need for multiple conditional statements.

This structured **decision-making process** ensures that every round of the game is fair and that the correct outcomes are displayed based on the user's and computer's choices.

2. 2. GUI Development and Event Handling

A **Graphical User Interface (GUI)** was developed to enhance user interaction and make the game more engaging. Tkinter, Python's built-in library for GUI applications, was chosen for its simplicity and versatility in creating visually appealing interfaces. The GUI incorporates buttons for user inputs, labels to display results, and dynamic text updates to ensure a responsive experience. The event-driven programming approach allows the game to react instantly to user actions, such as button clicks. With a structured layout, the interface is designed to be both intuitive and accessible, providing clear feedback to players after each move. The game's visual elements, including colors and fonts, are carefully selected to improve readability and enhance the user experience. Real-time score updates and outcome displays keep players informed of their progress. This combination of ease of use, aesthetic design, and real-time interaction makes the game enjoyable and accessible to all players.

User Interface Components

To enhance the user experience, the interface was designed with the following elements:

1. **Title and Instructions** – A title label was added at the top, followed by brief instructions on how to play the game.
2. **Choice Buttons** – Five buttons (Rock, Paper, Scissors, Lizard, Spock) were arranged in an easy-to-access layout, allowing users to select their move.
3. **Computer's Choice Display** – A label was included to dynamically update and display the computer's randomly selected move.
4. **Result Display** – A text label was added to show whether the user won, lost, or tied the round.
5. **Score Tracking** – Two labels were used to display the player's and computer's scores, updating dynamically with each round.

Event-Driven Programming for User Interaction

The game follows an **event-driven programming model**, meaning actions occur only when the user interacts with the interface. When a user clicks on a choice button, an **event handler function** is triggered:

1. It retrieves the user's selected move.
2. It generates the computer's move using a random function.
3. It evaluates the result based on game logic and updates the interface accordingly.

The GUI ensures a seamless experience, making the game visually appealing and easy to navigate.

2. 3. Randomisation, Score Tracking, and Testing

To ensure a **fair and unpredictable** gaming experience, a **randomisation technique** was used for the computer's move selection. The random.choice() function was employed to select one of the five possible moves with equal probability.

Ensuring Fairness and Randomness

1. The function random.choice(["Rock", "Paper", "Scissors", "Lizard", "Spock"]) was used to randomly pick the computer's choice.
2. The selection process was designed to provide a **balanced distribution**, preventing bias toward any particular move.
3. Since every move has an equal probability, the game remains **fair and challenging** for the user.

Score Tracking and Dynamic Updates - To enhance engagement, the game includes a score-tracking mechanism. The implementation follows these steps:

1. **Global score variables** were initialised to track user and computer scores.
2. After each round, the score updates dynamically based on the outcome.
3. **Tkinter's StringVar and IntVar** were used to automatically update score labels in the interface.
4. A **reset button** was added to allow players to restart the game at any time.

Testing and Debugging - Before finalising the project, extensive testing was conducted to ensure correct functionality.

Test Case	Expected Outcome	Result
User selects Rock, Computer selects Scissors	User Wins	Passed
User selects Spock, Computer selects Lizard	Computer Wins	Passed
User selects Paper, Computer selects Paper	Draw	Passed
Random selection for Computer	Equal probability of five choices	Passed
GUI responsiveness	Buttons and labels update instantly	Passed

Testing helped identify and resolve issues such as incorrect game logic, GUI misalignment, and score tracking errors.

3. Results and Discussion -

1. **Overview of Game Performance -** The performance of the Rock, Paper, Scissors, Spock, Lizard game can be assessed by evaluating how well it executes the game logic, ensures fair gameplay, and updates the score in real-time. Based on user input, the game correctly calculates outcomes and displays results immediately. The inclusion of additional choices like Spock and Lizard enhances the complexity of the game, making it more engaging and unpredictable. The responsiveness of the game interface allows users to play without experiencing significant delays. The game's efficiency, particularly in terms of computing the random moves for the computer, is handled well by the random module in Python. As the program is lightweight, it runs efficiently even on systems with limited resources. Overall, the GUI offers a smooth, fast-paced gaming experience where players can enjoy playing multiple rounds consecutively without performance lags.

2. **User Interaction and Experience -** The user experience plays a crucial role in making the game both engaging and easy to navigate. With Tkinter's intuitive layout, users can easily interact with the game by clicking buttons for Rock, Paper, Scissors, Spock, or Lizard. The feedback mechanism works effectively, instantly updating the result on the screen and reflecting the current score. Players can immediately understand the consequences of their choices as well as the computer's randomly selected move. The simple button interface makes the game more accessible to users of all ages, and the clear instructions and results allow for an enjoyable experience without confusion. The use of visual feedback to show the game's progress and outcomes in real-time adds to the entertainment value. However, some improvements in visual aesthetics and the overall layout could further enhance the user experience, such as adding more graphic elements or animations.

3. **Randomness and Fairness of the Game -** Randomness is a core element in making the game fair and unpredictable. In this implementation, the computer's choice is determined using Python's random module, which ensures that each of the five possible moves (Rock, Paper, Scissors, Spock, Lizard) has an equal probability of being selected. This introduces an element of chance into the game, preventing users from predicting the computer's actions. By allowing for a broader range of choices, the game prevents repetitive patterns, ensuring that no two rounds are identical. The fairness of the game relies on this randomness to create a balanced experience for the player. If the game logic and random choices are properly implemented, the player has an equal chance of winning, losing, or drawing, regardless of the sequence of previous moves.

4. **Accuracy of Game Outcomes -** The accuracy of the game outcomes is determined by how well the program adheres to the rules of Rock, Paper, Scissors, Spock, Lizard. The game correctly identifies the winner of each round based on the choices made by the user and the computer. The rules of the extended version of the game are applied consistently, ensuring that the right decision is made in every case. For instance, Paper beats Rock, Scissors beats Paper, and so on. The implementation of the decision logic in the function accurately compares the user's input to the computer's input, ensuring fairness and correctness. The results are displayed immediately, and the score is updated accordingly. This accuracy in determining the winner plays a critical role in keeping the game fun and competitive.

5. **Score Tracking and Updates -** Score tracking is a key feature that adds to the game's competitiveness. Every time the player wins, loses, or draws, the score is updated in real-time. The GUI uses an IntVar() in Tkinter to handle the score, which is updated dynamically after each round. The continuous tracking of wins and losses keeps players engaged and motivates them to try again for a higher score. The updates are instantaneous, allowing for smooth transitions between rounds without delay. However, it's important to note that any issue with score handling, such as an incorrect calculation, could impact the player's enjoyment. While the current system works well, additional features such as tracking the best score or adding a reset button for the game could make the scoring system more robust.

6. **GUI Design and Aesthetics -** The design and aesthetics of the Graphical User Interface (GUI) play an important role in creating a visually appealing game. In this game, the use of buttons for each move (Rock, Paper, Scissors, Spock, and Lizard) is efficient, ensuring easy user navigation. The layout is simple but functional, with the result and score displayed clearly in the center of the interface. The color scheme, although minimal, is effective for readability. However, the addition of graphical elements, such as images representing each of the moves, could further improve the visual appeal. A more detailed background or animated transitions might also make the game more immersive. In future versions, further aesthetic improvements, such as custom fonts, dynamic button styling, and animations, could enhance the overall look and feel of the game.

7. **Error Handling and Bug Fixes -** Error handling is a crucial aspect of any application, and this game handles basic user inputs well. In the current implementation, there are no major bugs or crashes, and the program successfully handles all inputs without errors. For example, the user's choices are checked against predefined values (1 to 5), and the game ensures that only valid moves are accepted. However, there is room for improvement in the robustness of the error-

handling system. If a user were to introduce unexpected input, such as non-numeric values, the game could crash. To prevent this, input validation and error messages could be added to inform the user of invalid selections. This would make the game more resilient and error-proof, especially for less experienced users.

8. **Suggestions for Future Enhancements -** While the current implementation of the game is fully functional, there are several areas for potential improvement. One key enhancement would be the addition of sound effects or background music to make the game more immersive. Additionally, integrating animations for the moves and transitions between rounds could improve the overall experience. Another potential upgrade is the addition of difficulty levels, where the computer's choice can be influenced by an AI algorithm to simulate varying levels of skill. Multiplayer support could also be introduced, allowing users to play against other people instead of the computer. Finally, the game could benefit from a more detailed tutorial for first-time players, explaining the rules and mechanics in a simple manner. These improvements would make the game even more engaging and enjoyable for a broader audience.

Conclusion -

The **Rock, Paper, Scissors, Spock, Lizard** game developed with a **Graphical User Interface (GUI)** using **Tkinter** provides an engaging and interactive experience for users. The game follows a straightforward yet captivating logic where players choose one of five options, and the computer randomly selects its choice. The outcome is determined based on predefined rules that ensure fairness, and the score is updated in real-time after each round.

The **GUI design** is simple and user-friendly, featuring clear buttons for each choice and easily readable result and score labels. The interactive elements, such as button presses and real-time updates, contribute to an enjoyable experience for the player. The choice of **Tkinter** for this project was optimal due to its ease of use and sufficient features for developing a lightweight application.

From a functional perspective, the game correctly implements the decision-making process for determining the winner, considering all possible scenarios based on the rules of the game. The game also provides immediate feedback, allowing users to see the result of each round, which enhances the overall gaming experience.

In conclusion, this project successfully combines simple game mechanics with an intuitive interface, offering an enjoyable gaming experience. Future enhancements, such as multiplayer support, sound effects, or a more complex difficulty system, can further elevate the game's appeal. This project demonstrates the potential of GUI-based game development using Python and highlights the power of **Tkinter** in creating interactive applications.

References -

1. Dhruv, A. J., Patel, R., & Doshi, N. (2021). Python: the most advanced programming language for computer science applications. *Science and Technology Publications, Lda*, 292-299.

2. Tadlaoui, M. A., & Chekou, M. (2021). A blended learning approach for teaching python programming language: towards a post pandemic pedagogy. *International Journal of Advanced Computer Research*, *11*(52), 13.

3. Erfina, A., & Nurul, M. R. (2023). Implementation of Naive Bayes classification algorithm for Twitter user sentiment analysis on ChatGPT using Python programming language. *Data and Metadata*, *2*, 45-45.

4. Mukhiddinovna, A. M. (2022). Programming language python methodology for creating and using didactic materials for students. *Galaxy international interdisciplinary research journal*, *10*(5), 63-67.

5. Rajabov, A. (2024). REPLACE OBJECT ORIENTED PROGRAMMING (OOP) IN PYTHON PROGRAMMING LANGUAGE. *Medicine, pedagogy and technology: theory and practice*, *2*(9), 221-229.

6. Kulikov, A., Alabed Alkader, N., Panaedova, G., Ogorodnikov, A., & Rebeka, E. (2023). Modelling Optimal Capital Structure in Gas and Oil Sector by Applying Simulation Theory and Programming Language of Python (Qatar Gas Transport Company). *Energies*, *16*(10), 4067.

7. Leokhin, Y., Fatkhulin, T., & Kozhanov, M. (2024, November). Research of Neural Networks ChatGPT Used to Generate Code in Python Programming Language. In *2024 Intelligent Technologies and Electronic Devices in Vehicle and Road Transport Complex (TIRVED)* (pp. 1-6). IEEE.

8. Thorgeirsson, S., Weidmann, T. B., Weidmann, K. H., & Su, Z. (2024, March). Comparing Cognitive Load Among Undergraduate Students Programming in Python and the Visual Language Algot. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1* (pp. 1328-1334).

9. Sekhar, P. R., & Goud, S. (2024). Collaborative Learning Techniques in Python Programming: A Case Study with CSE Students at Anurag University. *Journal of Engineering Education Transformations*, *38*.