DICE ROLLING SIMULATOR

A PROJECT REPORT

Submitted by

C.SAI MURALI [192211951]

S. SARATH SAI [192210051]

M.ROHIT [192211415]

Under the guidance of

R. YUVARANI

in partial fulfilment for the completion of

Course CSA0880-PYTHON PROGRAMMING

FOR WEB APPLICATION

SIMATS ENGINEERING

THANDALAM

JUNE 2024

BONAFIDE CERTIFICATE

Certified that this project report titled **"DICE ROLLING SIMULATOR"**

Is the bonafide work of **"C. SAIMURALI [192211951],S. SARATH SAI [192210051],M.ROHIT [192211415]"** who carried out the project work under my supervision as a batch. Certified further, that to the best of my knowledge the work reported herein does not form any other project report.

Date: Project Supervisor Head of the Department

ABSTRACT

Dice rolling simulator is a software that mimics the action of rolling dice. It strives to faithfully recreate the element of chance associated with real life dice rolling through accurate simulations. This feature allows user to experience a realistic and interactive dice rolling experience.

A virtual dice roller that generates random numbers like a real dice. Once you hit on the roll button, the simulator picks a random number that match those dice faces. The result of each roll is shown to user.

The dice rolling simulator offers a versatile and immersive platform for simulating dice rolls with precision and realism. Whether for recreational enjoyment, educational purpose, or statistical experimentation, this software application provides user with a valuable tool for exploring the fascinating world of probability.

PRNG algorithm that provides sufficient randomness for the intended application. The choice of algorithm can impact the stastical properties of the simulated dice rolls, so it's essential to select an algorithm that meets the requirements of the specific use case.

KEY WORDS: dice rolling, random number, simulator, games, platform.

INTRODUCTION:

A standard die is a cube with six faces that display varying numbers of dots, or pip, ranging from 1 to 6. when the dice is thrown or rolled, it comes to rest with a random integer between one and six shown on its upper surface, each value having an equal chance of occurring.

Numerous other such devices are sometimes referred to as dice; these specialized dice may have faces marked with symbols rather than numbers and may have polyhedral or irregular shapes. They could be applied to generate more than one through six outcomes. Dice that are loaded and skewed are intended to Favor some outcomes over others for entertainment or cheating. When playing board games or gambling, a dice tray is commonly used to catch thrown dice.

A platform for engaging with probability theory, statistical analysis, and gaming scenarios is offered by the Dice Rolling Simulator, a software application that simulates the rolling of one or more dice. The simulator provides a flexible tool for users to investigate the dynamics of chance and randomness, whether for entertainment, educational reasons, or statistical investigation.

In the realm of gaming and decision-making, the rolling of dice has long been a symbol of chance and uncertainty. From tabletop games to casino halls, the simple act of rolling dice carries with it a sense of excitement and anticipation. In this digital age, we bring this timeless experience to life through the creation of a dice rolling simulator.

METHODOLOGY

- **Requirement Analysis:** The dice rolling simulator should provide a user-friendly interface for rolling various types of dice, ensuring random and unbiased results. It should also support modifiers and display a history of past rolls for user reference.

- **Technology Selection:** Web technologies like HTML, CSS, and JavaScript could be used to create the dice rolling simulator to ensure cross-platform compatibility and user-friendliness. Alternatively, a graphical user interface and random number generating library like Tkinter or PyQt might be used to build a desktop program in a language like Python.

- **Data Sources Identification:** The dice rolling simulator is entirely dependent on internal logic and random number generating techniques; it does not require external data sources to work. Consequently, no particular data sources need to be found in order for it to develop.

- **Data Retrieval Mechanism:** Since the dice rolling simulator uses random number generation algorithms to generate outcomes internally, it does not require data retrieval procedures. As a result, it functions without reliance on other data sources, making implementation easier and less complicated.

- **Error Handling:** Error handling in the dice rolling simulator involves validating inputs for accuracy and providing clear error messages for guidance, ensuring smooth user experience and accurate results.

- **User Interface Design:** Prioritizing simplicity and ease of use, the dice rolling simulator's user interface should have simple controls for choosing different types of dice, rolling methods, and showing outcomes.

- **Testing and Validation:** ensuring the reliability and efficacy of the application, extensive unit testing is conducted as part of the dice rolling simulator's testing and validation process to confirm the precision of user inputs and dice rolls.

- **Deployment and Maintenance:** Deployment and maintenance for the dice rolling simulator entail ensuring seamless distribution across platforms and regular updates to address any bugs or improve functionality, ensuring a smooth user experience over time.

- **Documentation:** The dice rolling simulator's documentation ensures ease of understanding and future maintenance by providing clear instructions on how to use the application, a description of its features and functionalities, and details on its implementation for developers.

IMPLEMENTATION:

1. Set Up Development Environment:

Choose a programming language and environment (e.g., Python with Tkinter for desktop app, HTML/CSS/JavaScript for web app). Set up your development environment with necessary tools and libraries.

2. Design User Interface (UI):

Design a simple and intuitive UI for the simulator. Include options for selecting dice types, rolling mechanism, displaying results, and any additional features like modifiers or history.

3. Implement Dice Rolling Logic:

Write code to generate random numbers based on the selected dice type and any modifiers. Ensure randomness and fairness in dice rolls.

4. Handle User Inputs:

Implement mechanisms to capture user inputs such as selected dice type and any modifiers. Validate user inputs to prevent errors or invalid selections.

5. Display Results:

Write code to display the results of dice rolls in the UI. Ensure clear presentation of results for easy interpretation by users.

6. Add Additional Features (Optional):

Implement features like rolling multiple dice at once, adding modifiers to rolls, or displaying a history of past rolls if desired.

7. Test the Application:

Perform thorough testing to ensure the simulator works as intended. Test for different scenarios including valid and invalid inputs, edge cases, and various combinations of settings.

8. Refine and Debug:

Debug any issues encountered during testing. Refine the code and UI for better performance and user experience.

9. Document the Implementation:

Document the code, including comments and explanations for future reference. Create user documentation explaining how to use the simulator.

CODE:

```
import random
import time


def roll_dice():
    return random.randint(1, 6)


def print_dice(number):
    if number == 1:
        print(" ------- ")
        print("|       |")
        print("|   •   |")
        print("|       |")
        print(" ------- ")
    elif number == 2:
        print(" ------- ")
        print("| •     |")
        print("|       |")
        print("|     • |")
        print(" ------- ")
    elif number == 3:
        print(" ------- ")
```

```
    print("| •     |")
    print("|   •   |")
    print("|     • |")
    print(" ------- ")
elif number == 4:
    print(" ------- ")
    print("| •   • |")
    print("|       |")
    print("| •   • |")
    print(" ------- ")
elif number == 5:
    print(" ------- ")
    print("| •   • |")
    print("|   •   |")
    print("| •   • |")
    print(" ------- ")
elif number == 6:
    print(" ------- ")
    print("| •   • |")
    print("| •   • |")
    print("| •   • |")
    print(" ------- ")
```

```
def main():
    print("Welcome to the Dice Rolling Simulator!")
    while True:
        input("Press Enter to roll the dice (Press 'q' to quit): ")
        dice_number = roll_dice()
        print("\nRolling the dice...")
        time.sleep(1)
        print_dice(dice_number)
        print("\nYou rolled:", str(dice_number))
        if input("\nRoll again? (y/n): ").lower() == 'n':
          print("Thanks for playing!")
          break

if __name__ == "__main__":
    main()
```

OUTPUT:

```
Welcome to the Dice Rolling Simulator!
Press Enter to roll the dice (Press 'q' to quit):

Rolling the dice...
 -------
| •   • |
|       |
| •   • |
 -------

You rolled: 4

Roll again? (y/n): y
Press Enter to roll the dice (Press 'q' to quit):

Rolling the dice...
 -------
| •     |
|       |
|     • |
 -------

You rolled: 2

Roll again? (y/n): n
Thanks for playing!
```

RESULT AND DISCUSSION:

The experiment with the dice rolling simulator demonstrates how well it generates random results that resemble conventional dice rolls. After a great deal of testing, the simulator continuously generated a wide range of outcomes that were representative of the inherent unpredictability of dice-based games. The distribution of roll results verified the quality of the simulator's random number generation process, as it nearly matched the expected probability for each face of the dice. Nonetheless, few disparities were observed, emphasizing the intrinsic fluctuation in probabilistic procedures and the necessity for more improvement in subsequent rounds.

The simulator's performance highlights how useful it could be for statistical modeling, gaming, and educational simulations. Although the simulator does a great job at offering an easy-to-use platform for modeling dice-based scenarios, there is room for improvement in terms of response times and the way dice roll outcomes are displayed visually. Additionally, thinking about adding more sophisticated features to the simulator, including multiplayer support, could increase its usefulness in a variety of scenarios. In general, the experiment highlights how crucial strong randomization algorithms and user-friendly interfaces are to the creation of dice rolling simulators that work.

CONCLUSION AND FUTURE ENHANCEMENT

The dice rolling simulator experiment has demonstrated its ability to accurately simulate random dice rolls, providing a reliable tool for various applications such as gaming and statistical analysis. The experiment revealed that the simulator effectively generates outcomes reflective of the probabilities associated with traditional dice, albeit with occasional minor discrepancies. Despite these minor imperfections, the simulator's overall performance and user experience were deemed satisfactory, showcasing its potential value as a versatile tool for simulating random events.

Looking towards future enhancements, several opportunities for improvement have been identified. Firstly, refining the simulator's random number generation algorithm to minimize deviations from expected probabilities could enhance its accuracy. Additionally, incorporating advanced features such as multiplayer support, customizable dice sets, and more intricate visual representations of dice faces could expand the simulator's capabilities and appeal to a wider audience. Furthermore, exploring applications beyond recreational gaming, such as educational simulations and probabilistic modelling, could unlock new avenues for the simulator's utilization. By continually refining and expanding upon its features, the dice rolling simulator can evolve into a versatile and indispensable tool for various domains requiring random event simulation.

REFERENCES:

[1] Ruiz-Ezquerro, A., 2021. Rolling Dice and Learning--Using Role-Playing Games as Pedagogy Tools. *Journal of Campus Activities Practice and Scholarship*, *3*(2), pp.50-56.

[2] Chetcuti-Sperandio, N., Delorme, F., Lagrue, S. and Stackowiak, D., 2008, December. Determination and evaluation of efficient strategies for a stop or roll dice game: Heckmeck am bratwurmeck (pickomino). In *2008 IEEE Symposium On Computational Intelligence and Games* (pp. 175-182). IEEE.

[3] Freda, A., 1998. Roll THE DICE—an introduction to Probability. *Mathematics Teaching in the Middle School*, *4*(2), pp.85-89.

[4] Neller, T.W. and Presser, C.G., 2010. Practical play of the dice game pig. *The UMAP Journal*, *31*(1).

[5] Langtangen, H.P. and Langtangen, H.P., 2011. Random numbers and simple games. *A Primer on Scientific Programming with Python*, pp.375-436.