CS-101 Project: Sci-Fi Planet Generator

**Project Report: Planet Explorer and Alien Encounter Simulation**

**Authors: Yazdan Ali Khan, Raja Yawar Abbas, Hammad Shahid**

**1. Introduction**

The "Science Fiction Planet Generator" project is an interactive console-based application designed to engage users in exploring fictitious planets and their unique attributes. The program generates dynamic details about planets, their atmospheres, histories, and the alien races inhabiting them. Players also engage in mini-games to overcome challenges posed by alien inhabitants. This project demonstrates the use of random generation, file I/O, and game logic in C++.

**2. Objectives**

1. Provide an engaging experience for users through the exploration of procedurally generated planets.
2. Demonstrate the application of key programming concepts such as:
   * Randomization and seeding
   * File handling (Saves and Loads planet information from a txt file)
   * Functions
   * Loops
   * IF/ELSE and Switch statements
   * User interaction
3. Develop mini-games to enhance the interactivity of the program.
4. Showcase teamwork and coding best practices.

**3. Features**

**Core Features**

1. **Planet Generation**
   * Randomly generates a planet name using pre-defined syllables and meshing together (2-4) strings stored in an array
   * Assigns a random atmosphere and size from predefined lists.
   * Creates a unique history and events for each planet. While randomly choosing how many history events each planet will house (1-6)
   * Generates Alien species for each planet and cycles number of races (1-5)
2. **Alien Races**
   * Randomly generates alien races with unique appearances, technological levels, and forms of government.
3. **Mini-Games**
   * **Guessing Game:** Users must guess a randomly selected number within a limited number of attempts.
   * **Tic Tac Toe:** Users compete against an AI in a classic game of Tic Tac Toe.
4. **File Handling**
   * Allows users to save and load planet details to store previously visited planet.
5. **User Interface**
   * Interactive menu system

**4. Implementation**

**Technologies Used**

* **Language:** C++
* **Libraries:**
  + <iostream> for input and output.
  + <string> for handling textual data.
  + <ctime> for random number generation.
  + <fstream> for file handling.

**Code Structure**

The project adheres to a modular design to promote code clarity and reusability. Key components include:

1. **Constants and Data Arrays**
   * Predefined lists for atmospheres, sizes, histories, races, and alien details.
2. **Functions**
   * Each function handles a specific task, such as generating planet attributes, saving/loading files, or running mini-games.
3. **Main Program Logic**
   * A loop-based menu system guides the user through exploring new planets, loading saved planets, and exiting the program.

**5. Challenges Encountered**

1. **Randomization**
   * Ensuring that random generation did not repeat patterns too frequently.
   * Solution: Seeded the random number generator using the current time (srand(time(0))).
2. **Game Logic**
   * Developing AI logic for Tic Tac Toe that avoids obvious flaws.
   * Solution: Implemented a basic but effective AI that prioritizes available moves.
3. **File Handling**
   * Handling edge cases such as missing or corrupted files.
   * Solution: Added error handling to ensure graceful failure with informative messages.

**6. Results**

The program successfully achieves its goals:

1. Generates unique and engaging planet profiles with detailed attributes.
2. Provides an interactive user experience through challenging and fun mini-games.
3. Implements a robust file-saving and loading mechanism to preserve user generation.
4. Making an early Dos RPG imitation

**7. Future Possibilities**

1. **Fuel System:**
   * Add a fuel management system to simulate resource constraints during exploration and enhance gameplay loop.
2. **Graphical User Interface (GUI):**
   * Upgrade from a console-based interface to a GUI for improved aesthetics and usability.
3. **Enhanced Mini-Games:**
   * Introduce additional games or improve AI complexity in existing games.
4. **Expanded Planet Details:**
   * Include additional attributes such as gravity, temperature, and ecosystems.
5. **Porting the project on a dedicated Game engine**:
   * Rebuilding the project on an engine like unity or Godot to flesh out the project and make a playable build

**8. Conclusion**

The "Science Fiction Planet Generator" project successfully combines creativity, programming skills, and user interactivity. It serves as a practical demonstration of C++ programming concepts. This project was an excellent opportunity for the team to apply theoretical knowledge in a hands-on environment, resulting in an engaging and functional application.

**9. References**

* C++ Standard Library Documentation
* Online tutorials
* Header files for <ctime>, <fstream>
* Various resources on file handling and random number generation in C++