Final Report

Introduction

In my final project, I chose to design "Lunar Lander". The core algorithm of the game is a simulation of rocket motion. So I created a class called `Rocket` where contains most of information about a rocket object. It has some fields to store the key information including position, speed and so on. Besides I also design some methods used to simulate the motion of rocket.

Then I design `LunarGame` by using MATLAB App to simply emulate the experience of landing a rocket on the moon's surface. This report outlines the key features and functionalities implemented within the game application.

The process of the game works as follows :

1. It will create some basic elements the game needed, including the rocket image, the McDonald image and some labels which indicate the informations about the rocket. Then they will be show in the GUI
2. Then it will create one of the most important parts of game, the mountain chain which offers the landing plane for rocket. I will describe it in detailed afterward.
3. After all element being created, it will simulate the motion of the rocket. The users can control it by operating the keystone `a`, `w`, `d`. To make it successfully landed on the surface of the moon, you need to make sure it will landed on the plane while the speed of the rocket must be less than 25.
4. Whether the rocket successfully landed or not, you can receive a hint and sound effects to indicate the ending.
5. If you want to try again, you can simply cilck the button "RESTART"

The simulation is extremely **simple**!

- What I Achieve :
  ☑ The simulation of the rocket motion
  ☑ The GUI Interface
  ☑ The winning condition check
  ☑ More chance to try the game
- Needed To Improve :
  ☒ More accurate simulation of the rocket.
  ☒ Rotation of the rocket image
  ☒ More complicated simulation of the environment include adding some traps

Core Components

User Interface (UI)

- **UIFigure**: The main window of the application, providing a black-themed background for the game environment.
- **GamePanel**: A panel within the UIFigure dedicated to displaying the game scene, including the rocket, terrain, and other graphical elements.
- **STARTButton**, **RESTARTButton**, and **HELPButton**: Control buttons allowing users to initiate a new game, restart the current game, and access game instructions, respectively.
- **Image**: A display area for an image, potentially used for branding or decorative purposes.

Game Mechanics

- **Rocket Management**: The game initializes a `Rocket` object with random starting coordinates and velocity parameters. It includes functions to accelerate, turn left or right, and update its state based on physics principles.
- **Terrain Generation**: The `plotLine` method dynamically generates a mountainous terrain with random platforms (representing landing sites) using `uiaxes` for visual representation.
- **User Interactions**:
  - Keyboard inputs (`w`, `a`, `d`) allow players to control the rocket's acceleration and direction.
  - Button clicks trigger game actions such as starting, restarting, and accessing help.
- **Visual Feedback**:
  - `createRocketImage` dynamically places and updates the rocket's image based on its position.
  - `createMcDonald` introduces a McDonald's logo as a target landing spot, further enhancing the game's thematic elements.
  - Labels (`TimeLable`, `FuelLable`, `SpeedLable`, etc.) provide real-time feedback on game metrics like time, fuel, and speed.

Game Logic

I wiil discuss some main parts of the game

- **Simulation Of Rocket Motion :**

  The fucntion will calculate the postion at first. Then it will check if there exist some keystone information that will affect the accelerations used to calulate. If

exists, the fucntion will use the different acceleration and close the information. It will calculate the speed at last .

```
Update Positon :
        obj.pos = obj.pre_pos + obj.speed*t;
Update Speed :
        check keystone informations;
        obj.speed = obj.pre_speed + obj.acceleration*t;
```

- **The Creation Of Mountain Terrain**

  I use the `uiaxes` to create mountain terrain. Firstly I set some scatters. After that, I get some scatter to create plane randomly, I set the value of y axies of two scatters near the plane scatter is the same as plane scatter. Then I make the linear interpolation of the scatters I get above. Some I get the outline of the mountain terrain and painted the in the uiaxes. And I will get the value of the scatter.

```
        Get axies;
        Get some random scatter;
        Set the plane position and create plane;
        Make a linear interpolation;
        Plot them.
```

- **Winning Condition Check**

  It will get the position of the rocket. Then the function will compare the x-value to check it is locate above the plane or not. After that it will compare the y-value with the y-value of the points with same x-value I maked in the line I created in mountain terrrain. If the y-value of the rocket is less than the point in the mountain terrain , It will check if it wins or loses. And I set the speed must be less than 25 if it lands.

```
        Get rocket postion
        if rocket in plane
                if rocket y-value < plane y-value
                        if rocket speed < 25
                                Win
                        else
                                Lose
        else
                if rocket y-value < mountain y-value
                        Lose
```

- **The Whole Part Of Game**

```
Create Elements

While the game is still active:
        Check KeyStone
        Check Wining Or Losing
        Update Rocket Motion
        Update Elements

Showing Ending.
```

Conclusion

The `LunarLander` application offers an engaging and interactive gaming experience centered around the challenge of safely landing a rocket on a lunar surface. With dynamic terrain generation, responsive controls, real-time performance indicators, and climactic audio cues, it encapsulates fundamental game design principles in a MATLAB-based environment. This project showcases the versatility of MATLAB in creating visually appealing and interactive applications beyond traditional data analysis and numerical computation tasks.

What I Get From This Project

Firstly, to be honest, it is not an easy project that can be dealt with alone and I have to commit that I don't spent enough time on this project, which means it is truly simple and owns lots of problems. But I learn how to design GUI in MATLAB and how to use

App designer in MATLAB to satisfy my requirements. And it also improve my ability to write code in MATLAB. But it truly needs further development!!!