Moon Landing

http://o.ooli.ca/en/scratchgames/moon

Can you land the moon lander in this simulation of the low gravity of the moon? Your space ship is fragile. If you touch the rocks or if you land too fast, your ship will fall apart.

When you open this program, you will see a space background and one sprite: the moon lander. In this program, you will write code for simulating what it is like to move in space and what gravity on the moon is like.
(Finished game: http://scratch.mit.edu/projects/30358794/)
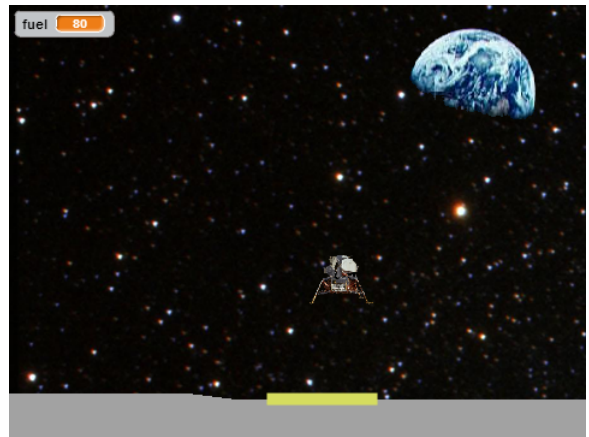
1. Basic Movement

MEDIUM

Start by writing a program that let's you move the space ship in a simple way. Some code has already been written for moving the ship to the left and right. The spaceship uses "x" to show where it is to the left and right. It uses "y" to show where it is up and down. Finish writing the code for moving the ship up and down.

2. Crashing

MEDIUM

Click on the costumes tab of the lunar lander sprite. Look at the different costumes for the sprite. Change your program so that if the ship touches the rocks it will explode. Since all the rocks are grey, your program should check if the ship is touching something grey, and if it is, it should switch to the "explosion" costume and stop the program. Also, when the green flag is pressed to start the game, you should switch to the starting "lunar_lander" costume.

3. Space Movement

HARD

In space, there is no air. That means that when you move in space, there is no air to get in your way and slow you down. If you start moving in one direction, you will keep moving in that direction. You can use variables to show what it's like to move in space. Let's start with moving left and right.

Create a variable called "x speed" for the sprite that keeps track of how fast the ship is moving left or right. During the game, the "x" position of the ship should be regularly changed by the "x speed" that the ship is moving at. This way, if you start moving at a speed of 1 step to the right, the ship will keep moving 1 step to the right again and again.

In space, when you use thrusters, your ship keeps moving even after the thrusters stop. Change your program so that when you press the left and right arrow keys, the "x speed" changes instead of directly moving the ship. That way, the ship will keep moving even when your thrusters are off, just like in real life.

4. Up and Down

HARD

Now that you've written code to move the space ship left and right using space movement, do the same for moving up and down. Use a variable called "y speed" to change the "y" position of the ship.

5. Moon Gravity

HARD

The moon has very weak gravity that pulls things to the ground. Gravity makes things fall to the ground faster and faster until they hit something. Add gravity to your program by having the "y speed" of your ship slowly change by -0.1.

6. Landing

HARD

Change your program so that if the lander touches the yellow landing pad, the game ends.

If you land on the landing pad while moving too fast, the ship will fall apart. Add code to your program so that if your speed is too fast when you touch the landing pad, your ship will explode.

Try It: Fuel

HARD

Space ships have only a small amount of fuel. Astronauts need to land their ship before running out of fuel. Make a variable called "fuel" for your program. When your game starts, your ship should start with 100 fuel. Every time you use your thrusters, you should lose fuel. If you run out of fuel, then your thrusters shouldn't work any more.

Try It: Thrusters

MEDIUM-HARD

The sprite for the spaceship has different costumes showing different thrusters being used. Change your program so that when you press the arrow keys, the correct thruster is shown.

Try It: New Maps

MEDIUM

If you click on the stage, you can go to the backdrops tab to change the terrain on the moon. Can you make your own moon surface?

Useful blocks for this exercise

Motion

`move ( ) steps`

`turn ↻ ( ) degrees`

`go to x: ( ) y: ( )`

`change x by ( )`

`change y by ( )`

Looks

`switch costume to [ ▼ ]`

Data

`set [ speed ▼ ] to [ ]`

`change [ speed ▼ ] by ( )`

Sensing

`touching color [ ]`

`key [ ▼ ] pressed?`

Events

`when ⚑ clicked`

Control

`wait ( ) secs`

`forever ↻`

`if ⬡ then`

`stop [ all ▼ ]`

Operators

`⬡ and ⬡`

`[ ] > [ ]`