Introduction

Background: To land a rocket on a platform is never an easy task. There's air drag, rotations, delaying, limited fuel, unexpected conditions etc. Many companies(like SpaceX) are working on making repeatable landings of rockets.

Our game simulated a comparingly simplified condition, still holding features like air drag, multiple thrusters, limited fuel, and angular momentum. We have done a lot of fine-tuning to make it more playable.

In the game, you are dispatched to manipulate a rocket aimed to land it on specific platforms with an acceptable bank angle. You also need to avoid collisions with meteorites.

Our game is based on MATLAB uifigure(introduced after 2016a). Multiple difficulties choice and score summary provided. Note that program performance may vary due to the different hardware and software of platforms.

Total codes: ~460lLines, 14 files, ~16KB

Total work time: ~60 person x hours

Features & Requirements:

The whole project took about one week for us to fully implemented. Benefiting from reasonable distribution for coding assignments, we only spent about one night before class report after completing our parts, thanks to the unambiguous definition of the main function interface.

Basic requirements:

(a) (a) First, adding a graphical user interface for both the start and end pages is indispensable. Thus, we use uifigure to create more available features. On the start page, we directly insert a picture with instructions and rules of the Lunar Lander Game. Also, whether you successfully land the rocket or not, two different figures will generate, also in the form of an insert. Considering the demand. Considering the demand, we only use four uiButtons are *Easy Mode, Hard Mode, New Game, Quit Game*.

(b) The number of landing pods (marked in red color) is determined by the mode of the game, which means more pods for *Easy Mode* and fewer pods for hard mode. It is worth noting that both terrain and landing pods are generated randomly.

(c) As mentioned above, the terrain can rise and fall even if generated randomly. A function named "crash" was written to judge whether the rocket collides with the wrong landing position and meteorites, which uses arrays of rocket and terrain and function "inpolygon" to evaluate whether there is a point inside the rocket when the main structure loops again and again.

(d) Especially, we draw a fuel gauge above the whole figure; it shows accurate left fuel according to how much time you press the keyboard using the keyboard callback function. When fuel is run out, the rocket can only be dragged by the moon's gravity and crash on the terrain.

Technical Approach and Individualization:

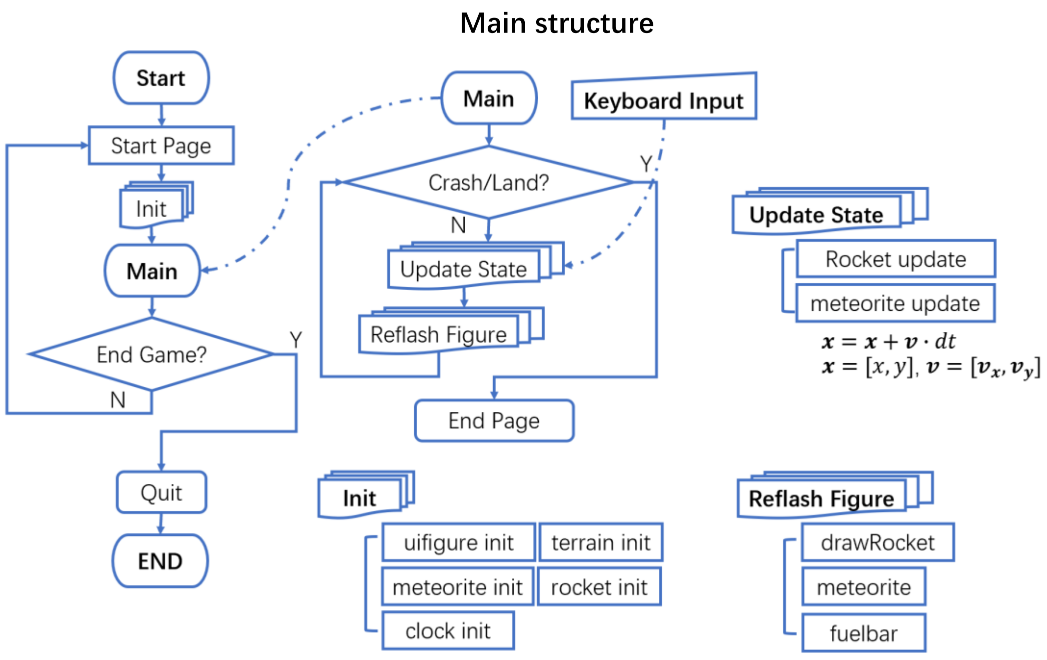
Because all kinds of paraments of the rocket are often used, we define a public class inspired by C++ to manage them quickly and conveniently. It involves size, position, dip angle, velocity, fuel, booster control, angular velocity, and acceleration. All of this is used to simulate the movement process under an authentic environment, which considers resistance of air, gravity, and thrust by three different directions of boosters and the torque they caused.

To decrease the circulated quantity as little as possible, we initially created the terrain and other settings before the loop. Also, we spent time on draw the rocket, clouds and meteorites. Rocket is drawn using functions *patch* and *fill*; its three boosters only display when you press *WASD*. Meteorites have colorful tails because of the setting of *colorbar,* and it will extend length along with fall. Cloud is drawn as decorations to display the usage of transparent degrees.

Additionally, we add some details, such as the explode effect when the function "crash" returns 'true'. The most important thing is the adjustment of the pause time for each cycle. Only when 'tstep' is about 0.1, can we use the keyboard control smoothly, and the refresh of the whole figure is faster than people's eyes which means you will feel it is continuous rather than displayed frame by frame. Luckily, it doesn't cause too much hysteresis.

In conclusion, the efficient process of finishing all Matlab code depends on the assignment of the tasks. One of us mainly wrote the control part and others wrote the drawing part, GUI setting and so on. We packed our code as a function after testing them in a simple script loop. In this way, when we get together to merge them, we need to tell each other what the input and output are because the inside code is already tested feasible. Thus, we have more time to adjust relevant paraments and proportions to correctly display all plots belonging to one "Parent" axis or figure.

Pseudo-code and description in flow chart form



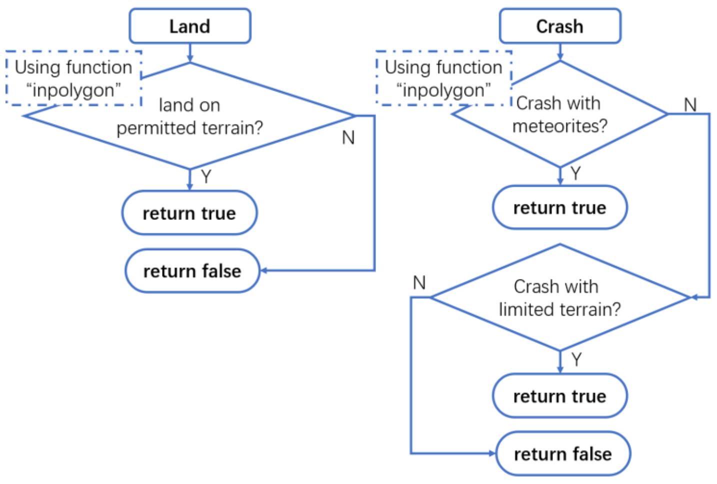
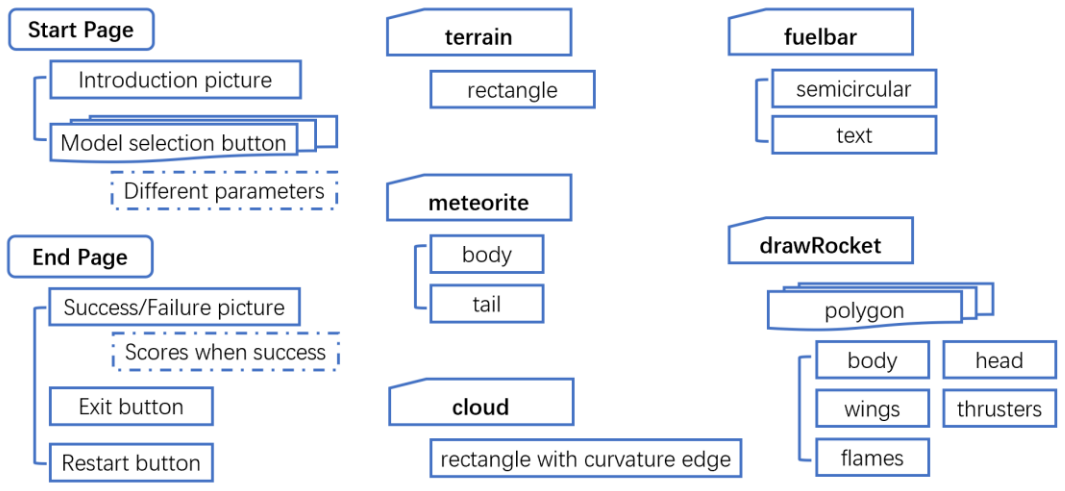
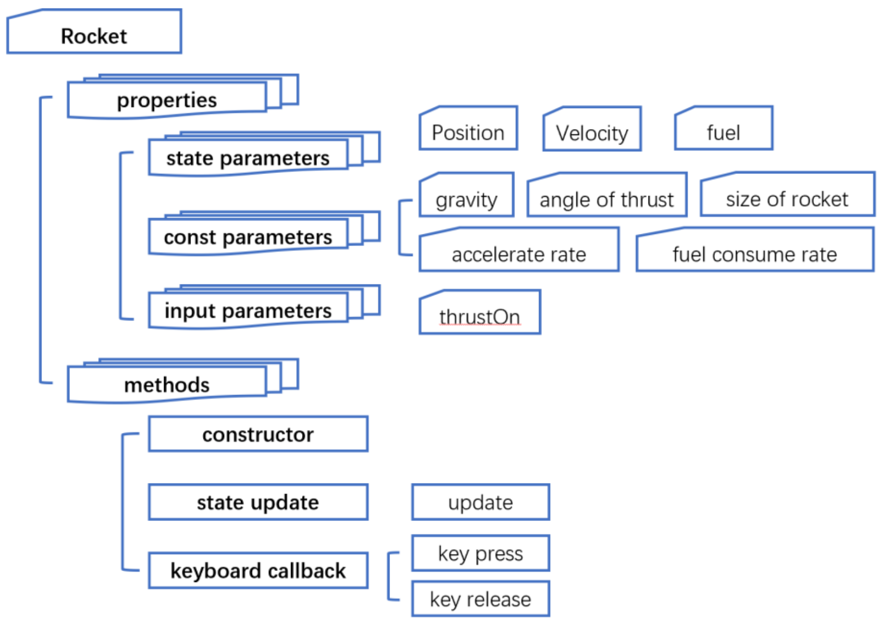
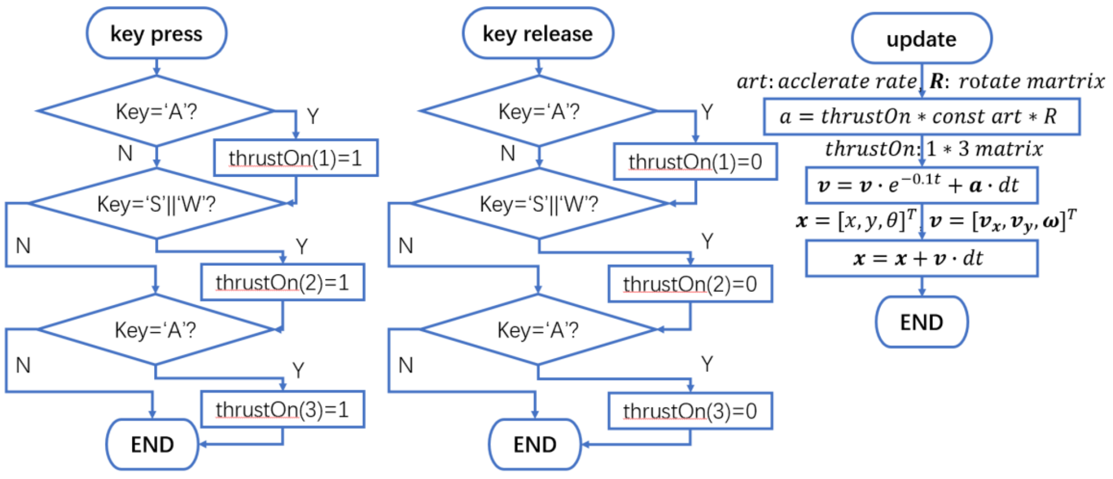

Land
Using function "inpolygon"
land on permitted terrain?
N
Y
return true
return false
Crash
Using function "inpolygon"
Crash with meteorites?
N
Y
return true
Crash with limited terrain?
N
Y
return true
return false

Figure objects



Class rocket




key press
Key='A'?
Y
N
thrustOn(1)=1
Key='S'||'W'?
N
Y
thrustOn(2)=1
Key='A'?
N
Y
thrustOn(3)=1
END
key release
Key='A'?
Y
N
thrustOn(1)=0
Key='S'||'W'?
N
Y
thrustOn(2)=0
Key='A'?
N
Y
thrustOn(3)=0
END
update
art: $acclerate\ rate$, $\boldsymbol{R}$: $rotate\ martrix$
$a = thrustOn * const\ art * R$
$thrustOn$: $1 * 3\ matrix$
$\boldsymbol{v} = \boldsymbol{v} \cdot e^{-0.1t} + \boldsymbol{a} \cdot dt$
$\boldsymbol{x} = [x, y, \theta]^T$, $\boldsymbol{v} = [\boldsymbol{v_x}, \boldsymbol{v_y}, \boldsymbol{\omega}]^T$
$\boldsymbol{x} = \boldsymbol{x} + \boldsymbol{v} \cdot dt$
END

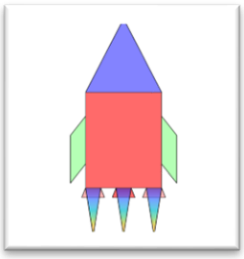
Conclusion

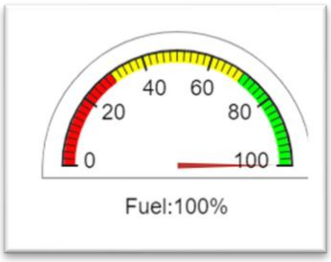
Our group put a lot of effort into polishing this program. During the teamwork we gained a deeper understanding of object-oriented programming in matlab and learned a lot about the usage of uifigure and its components. The game ended up with a satisfying UI with attractive coloring and a fair degree of playability. The final presentation was also satisfying. We are proud and excited about this work.

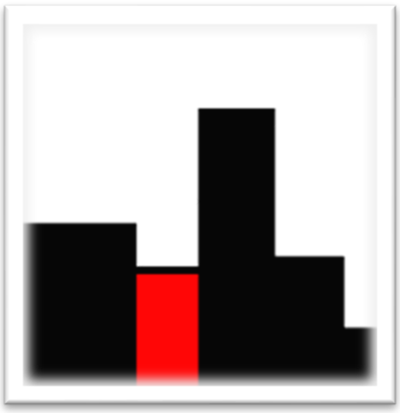
Hopefully, this collaboration will be a doorstop for the group to venture into the field of scientific computing!

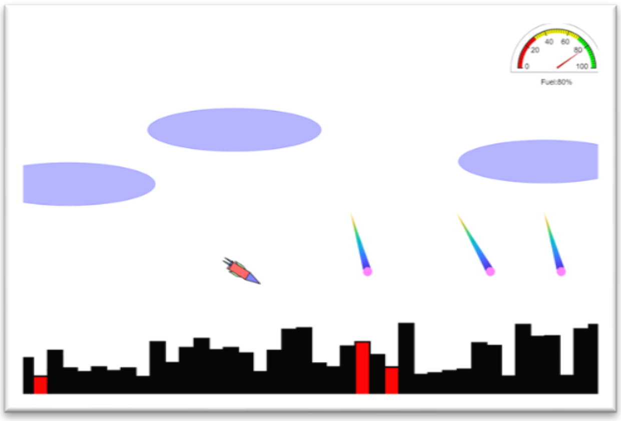
And the final segment followed is a gallery containing some screenshots of the game. Thanks for your appreciation.

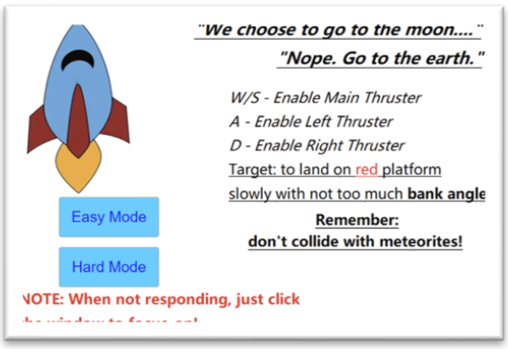
Gallery











"We choose to go to the moon...."

"Nope. Go to the earth."

W/S - Enable Main Thruster

A - Enable Left Thruster

D - Enable Right Thruster

Target: to land on red platform slowly with not too much **bank angle**

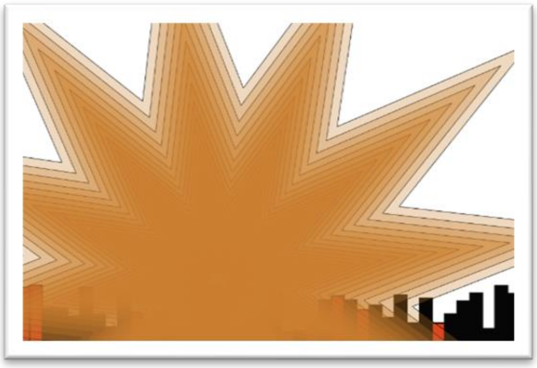
Remember:

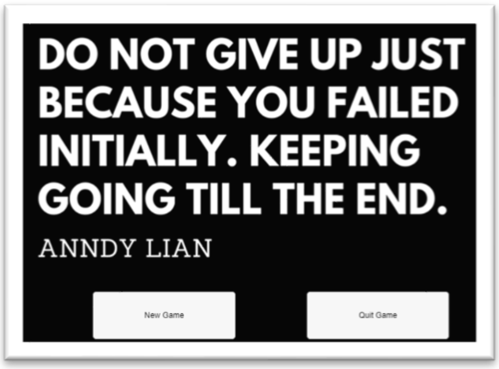
don't collide with meteorites!

Easy Mode

Hard Mode

NOTE: When not responding, just click



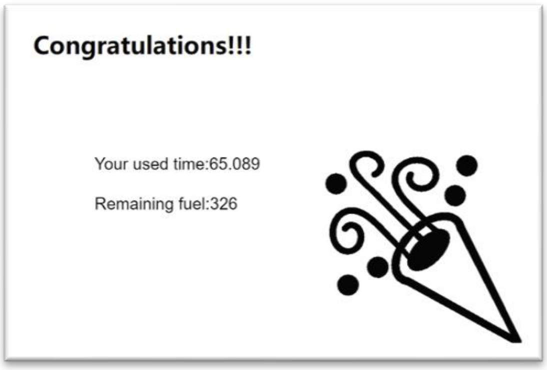


DO NOT GIVE UP JUST BECAUSE YOU FAILED INITIALLY. KEEPING GOING TILL THE END.

ANNDY LIAN

New Game

Quit Game



Congratulations!!!

Your used time:65.089

Remaining fuel:326