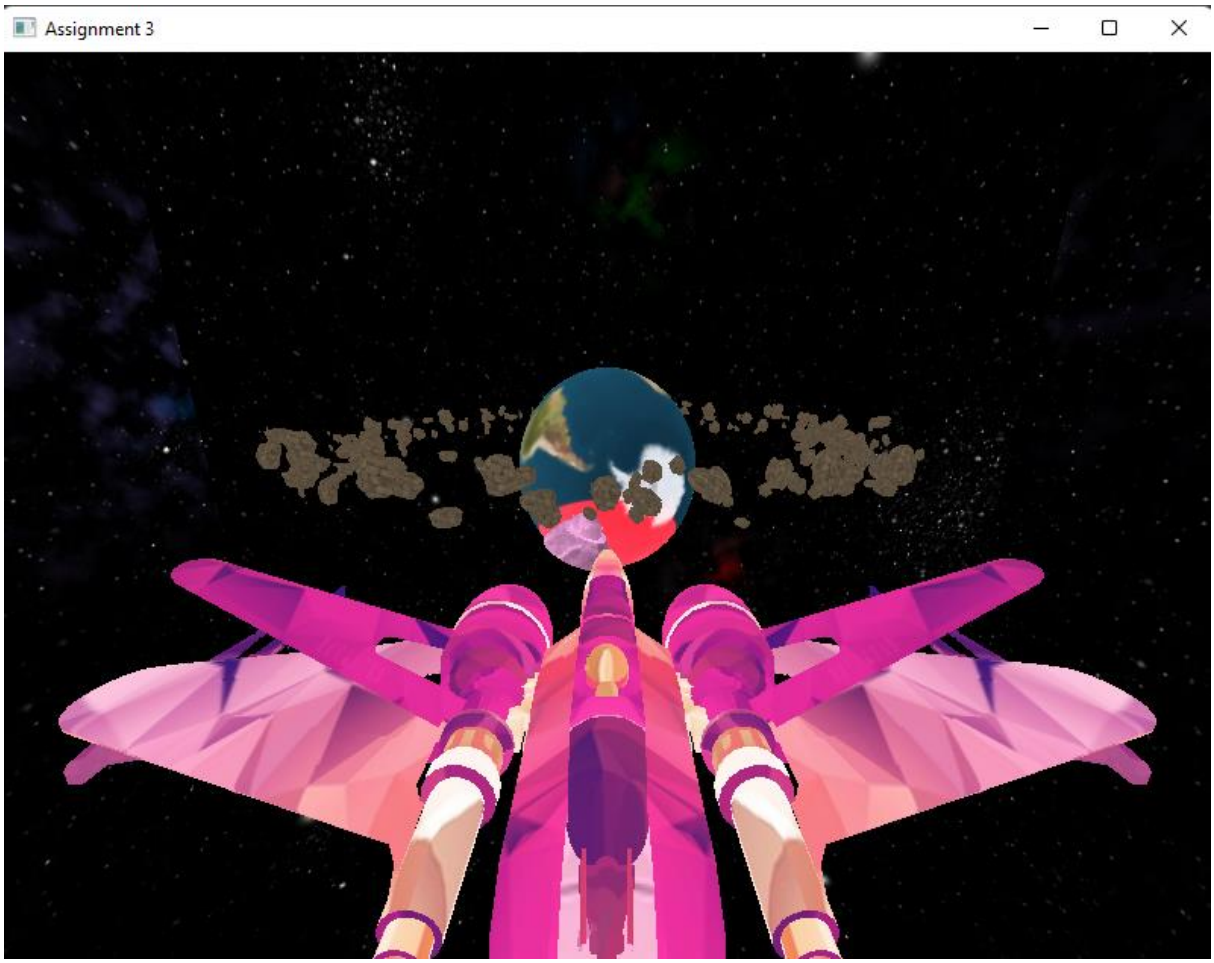
Name: Chan Chun Ming SID: 1155163257 CSCI 3260 Assignment 3

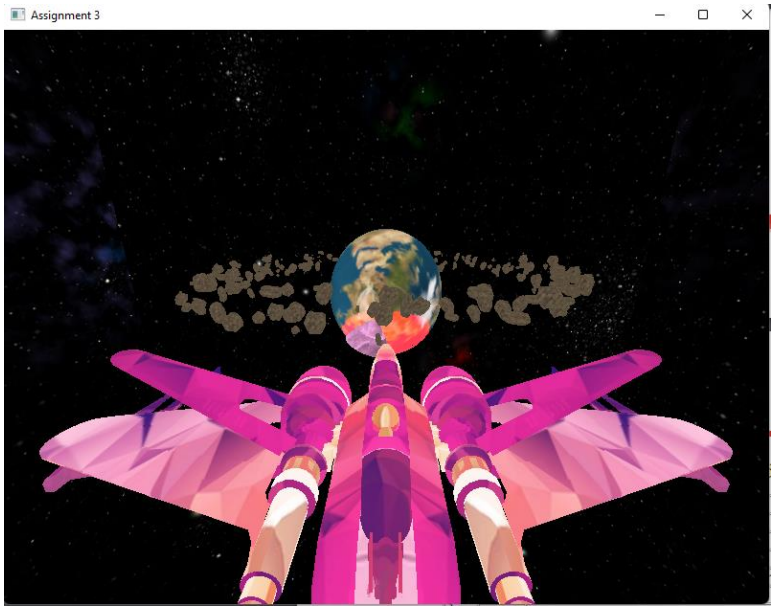
1. A figure which shows the overall scene like Figure 1.



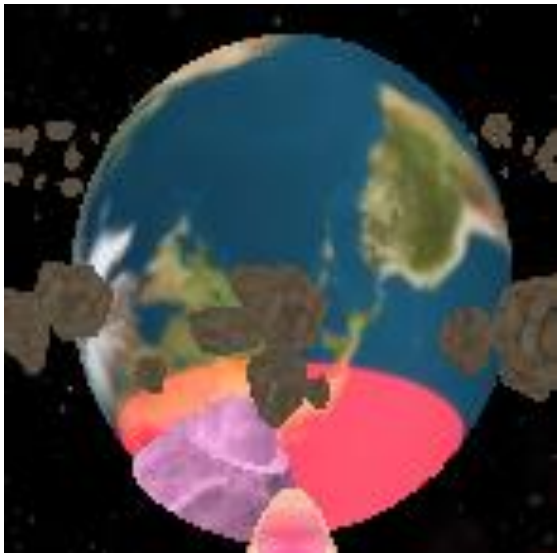
2. Basic light rendering results

There are two main light effects that I have implemented:

2.1 Ambient light for the whole scene



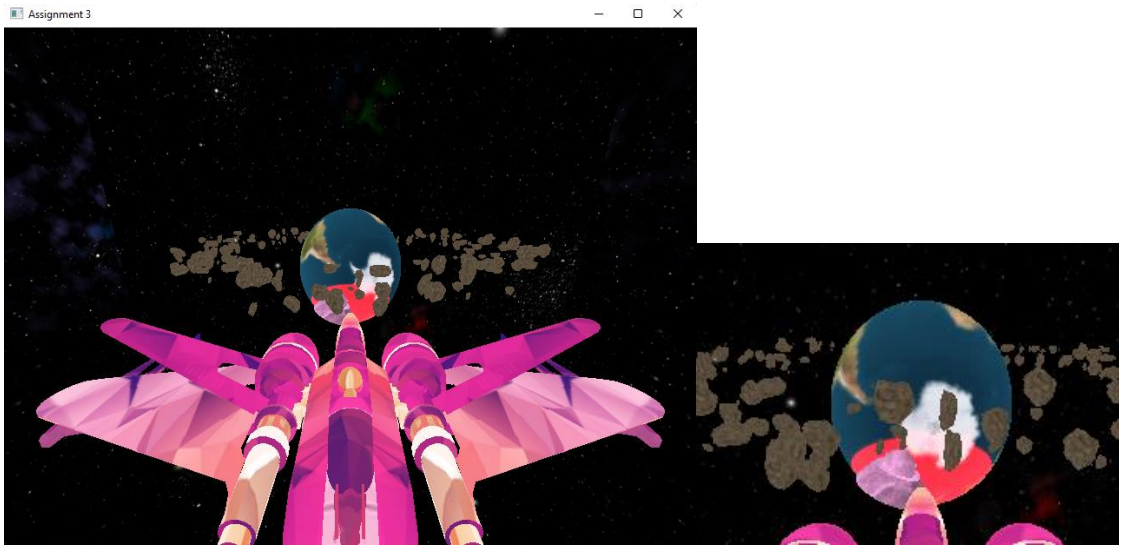
2.2 Point light for the planet (red point light)



3. The frames that can include any basic requirements that you have implemented. Please highlight the correlation between the provided frames and items in the grading scheme.

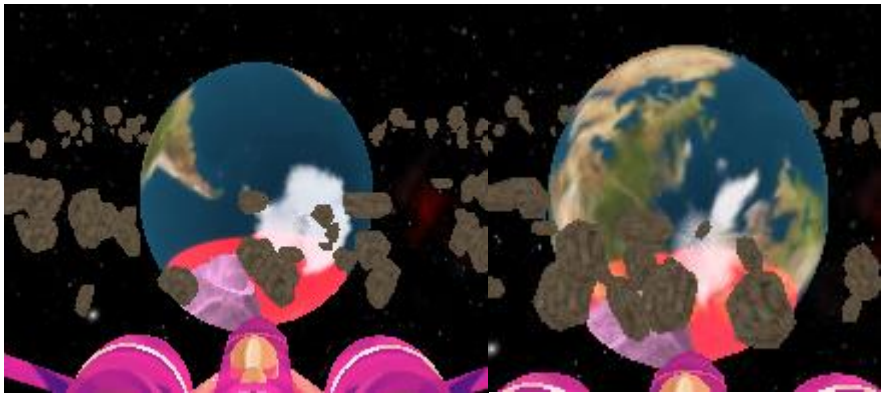
I will introduce my project in the manner of the point form in the grading scheme.

3.1. Render one planet, one spacecraft and at least one craft



In the frame you can see that I have render a planet Earth, a spacecraft, and a craft in pink colour (bottom left of the planet).

3.2 Self-rotation for the planet

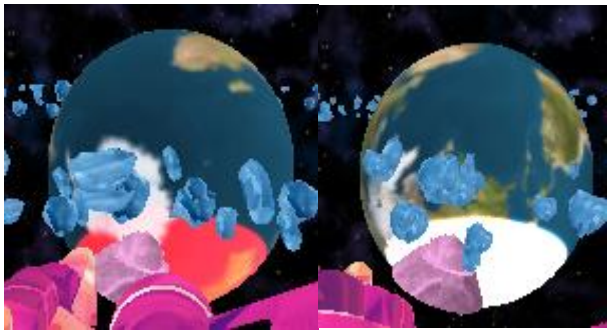


In the left frame you can see the Antarctica is facing towards camera perspective. And in the right frame you can see the North pole is facing towards the camera perspective. And the Earth is set to be rotated in Anti-clockwise manner.

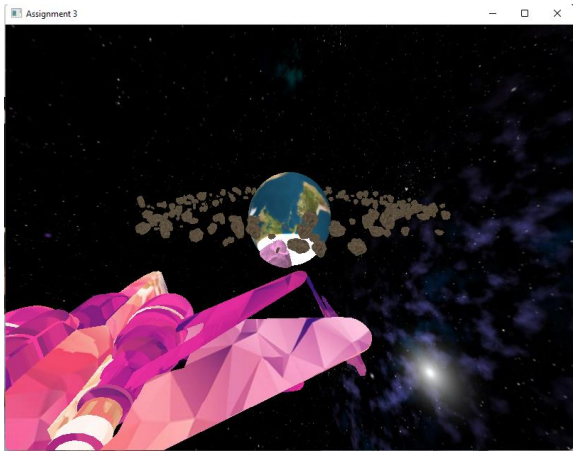
3.3 Render a skybox

You can see the nebulas and stars on the background in both frame which showing the skybox is correctly rendered into the program.

3.4 Basic light rendering

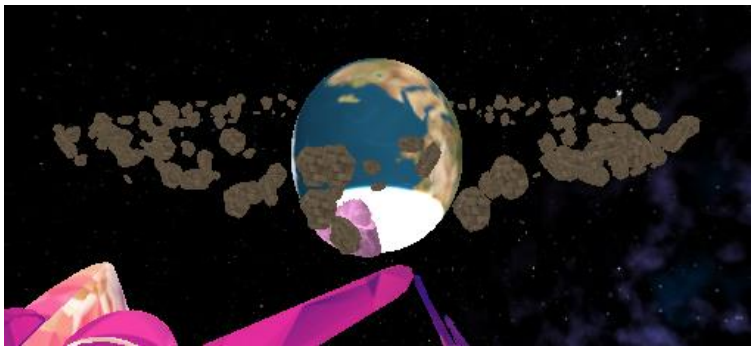


A point light source is directly point to the south pole of the earth and could change brightness and colour by key input.



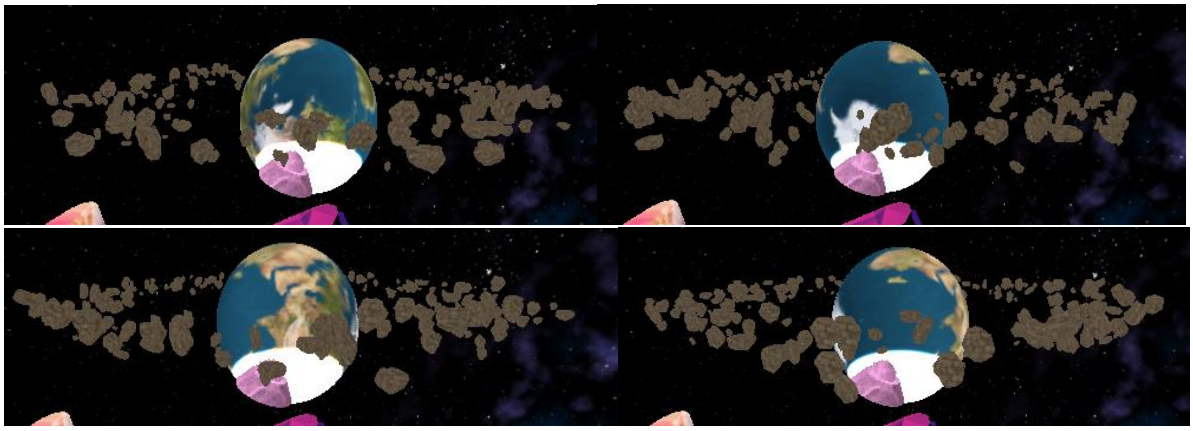
An ambient light source is used in global to make all objects look brighter.

3.5 Render an asteroid ring cloud



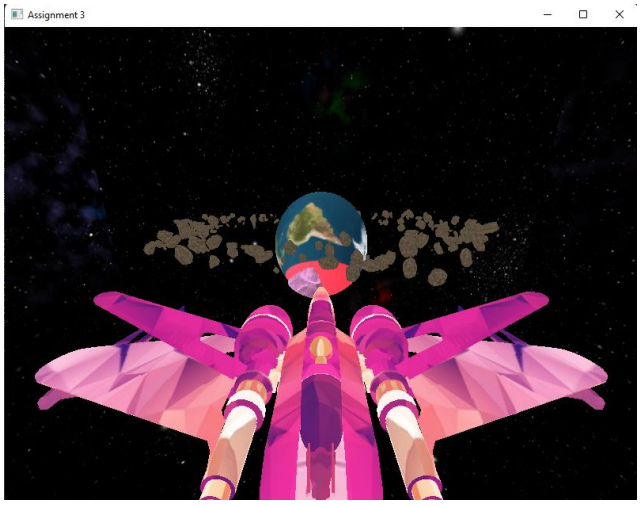
An asteroid ring cloud is implemented around the Earth with the rock texture. In this program, the asteroid ring cloud contains 200 random floating rocks around the planet.

3.6 The rotation of the rocks



Same as the planet, the rocks in the asteroid ring are rotated in an anti-clockwise manner. You can see that the difference of the rock shape in all the frames to prove that the rocks are rotating and not static. They have a limited range and rotate cantered at the planet.

3.7 Correct viewpoint



The viewpoint is behind the tail of the spacecraft and vertically higher than it. You can also see from the initial position the head of the spacecraft is pointing towards the planet.

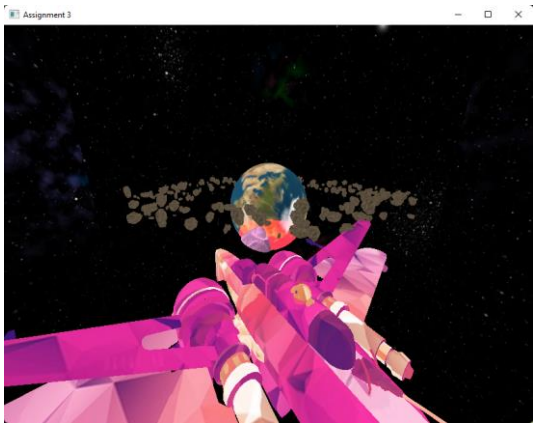
3.8 Normal mapping for the planet

I have tried to import the normal mapping for the planet, but it seems that the normal mapping is unsuccessful in my program.

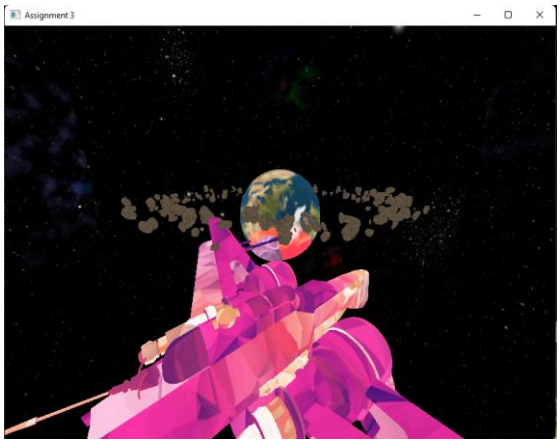
3.9 Use mouse to control the translations of the spacecraft

By clicking the left button of the mouse, and drag the cursor in any position, I can control the rotation of the spacecraft.

For Example, in 4 directions:

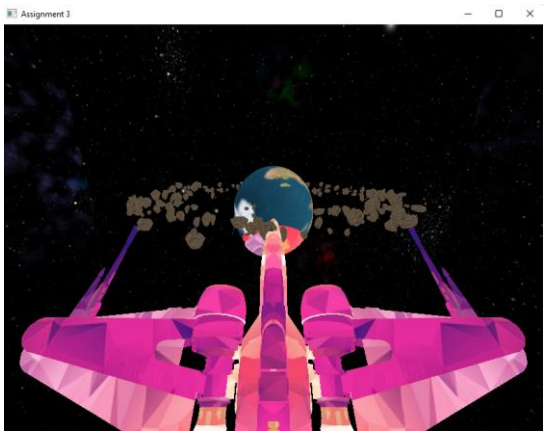


When I drag the cursor to right side (move the mouse to right), the head of the spacecraft will turn left.



When I drag the cursor to left side (move the mouse to left), the head of the spacecraft will turn right.

When I drag the cursor upwards (move the mouse upwards), the head of the spacecraft will become downwards.



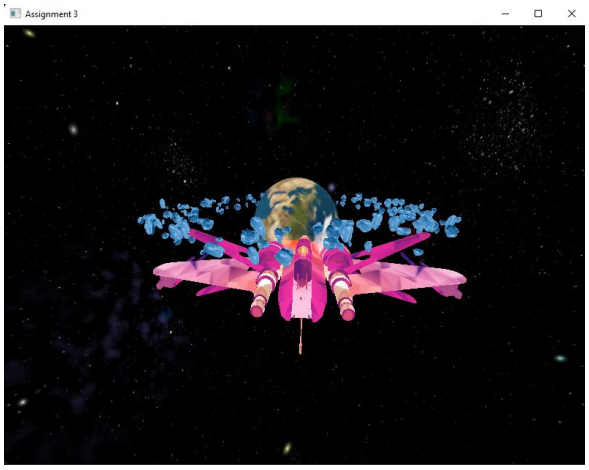
When I drag the cursor downwards (move the mouse downwards), the head of the spacecraft will become upwards.

3.10 Use keyboard to control the translations of the spacecraft

By entering the specific keystroke, I can control the translation of the spacecraft.

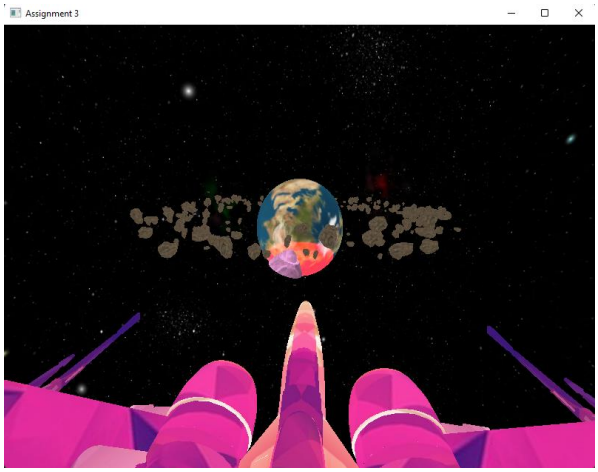
There are 4 keystrokes:

“w”



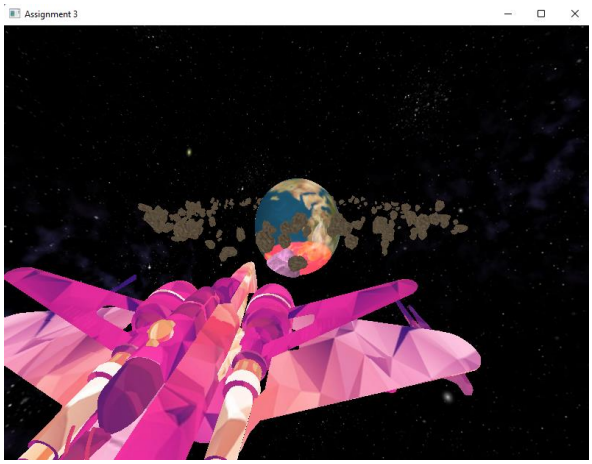
By entering “w”, the spacecraft will move forwards and towards the planet by a certain distance, the skybox will also rotating anti-clockwise correspondingly. The move will continue if the user continuously entering “w”.

“s”



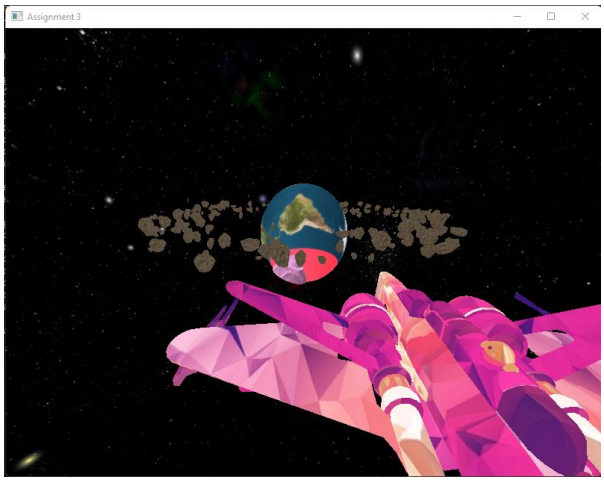
By entering “s”, the spacecraft will move backwards and leave the planet by a certain distance, the skybox will also rotating clockwise correspondingly. The move will continue if the user continuously entering “s”.

“a”



By entering “a”, the spacecraft will move to the left by a certain distance, the skybox will also translate to left correspondingly. The move will continue if the user continuously entering “a”.

“d”

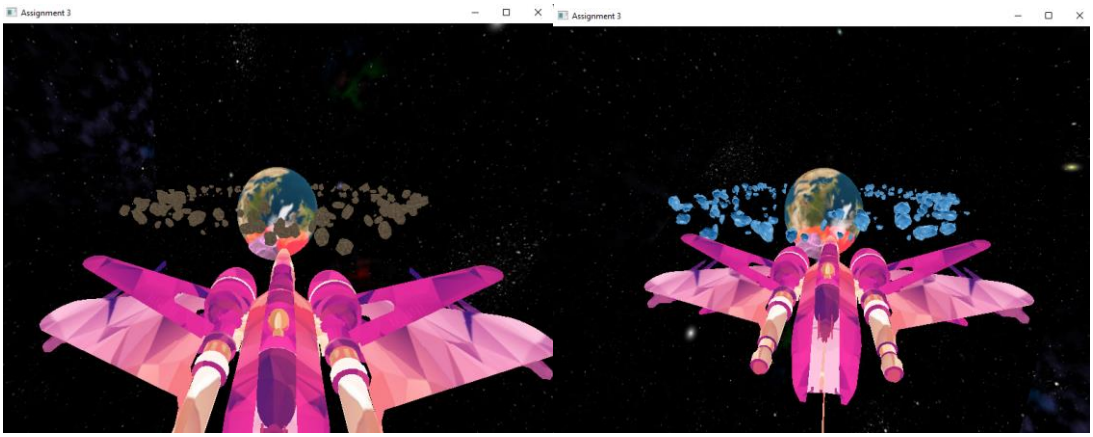


By entering “d”, the spacecraft will move to the right by a certain distance, the skybox will also translate to right correspondingly. The move will continue if the user continuously entering “d”.

3.4. The frames that can represent any bonus features that you have implemented.

Bonus: Distance detection

A simple collision checking system is implemented between spacecraft and asteriod ring.



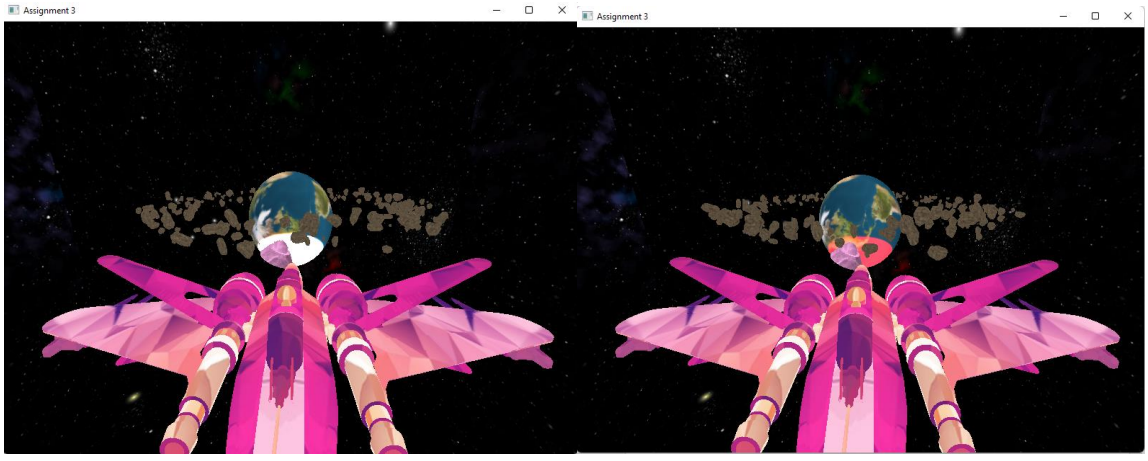
The left frame shows that the spacecraft did not collide into the asteriod ring, thus the asteriod ring remains in the original texture “rockTexture”.

The right frame shows that the system has detected a collision occuers between spacecraft andasteriod ring, which the rock texture in the asteriod ring will change from “rockTexture” to “craftTexture”.

The “craftTexture” texture will hold until the spacecraft have no collision with the asteriod ring cloud.

Bonus: Point Light Brightness

A simple point light brightness tuning is done by keystroke up and down



The left frame shows when point light brightness is set to highest by using keystroke up, the bottom of the earth showing complete white bright light and cannot see the planet texture in the covered area.

The right frame shows when point light brightness is set to lowest by using keystroke down, the bottom of the earth showing no white light and only original red color point light remains.

3.5. Some brief and necessary descriptions of your implementation details.

The program demonstrates a basic program, introducing a 3D space travel simulation using C++ and OpenGL. The simulation includes a spacecraft, a planet, a craft, an asteroid ring made by rock object, and a skybox.

Except normal mapping and other bonus features, all the basic required features that were listed in the marking scheme were properly treated and done. Interactions by using keyboard and mouse have also been implemented nicely.

For the collision feature between spacecraft and asteroid ring cloud, I did it in the most simplistic method, which calculate the distance between the rocks and spacecraft, when the distance between 2 objects reaches 0, it means there are collision, and the texture of the asteroid ring cloud will be changed.