CSCI 3260 Principles of Computer Graphics

Homework 3: Space Travel

Due Time: 11:59pm, 15th, November, 2023

Fail the course if you copy

I. Introduction

In the year 3022, astronomers discover a **planet** with **meteorites** around, where several **crafts** are there. As an outstanding pilot, you are driving a **spacecraft** to visit this planet.

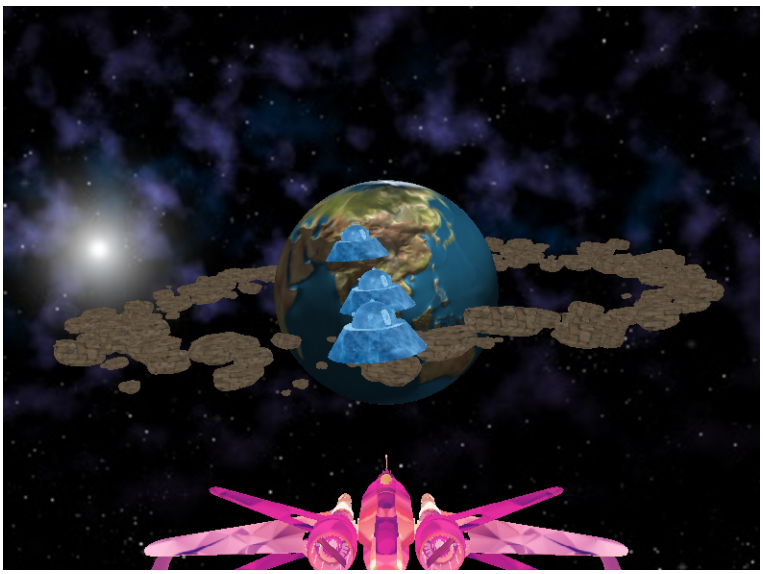

Fig. 1: The scene drawn by the demo program.

You are required to write your own code from scratch to complete this homework. All the basic techniques you need have been or will be introduced in our tutorials. Your best skeleton code is the solution programs of your assignments 1 and 2.

The ultimate objective of this project is to give you an opportunity to practice more with the basic but very important topics in Computer Graphics: you have to go through **object loading, transformation matrix, lighting, texture mapping, skybox, shader, and interaction** before you get a satisfying mark.

II. Implementation Details

1. Basic Requirements:
   a) Render the planet, the spacecraft and at least one local craft with corresponding textures. For simplification, please keep their centroids on a plane that is perpendicular to one axis of the world space. Note that you do not necessarily make the craft moving like in the demo video, relatively static position or other movement is all right. The planet should do self-rotation all the time.

b) Create a skybox as the background of the virtual scene.

c) Create a point light source. Basic light rendering (ambient, diffuse and specular) should be obviously observed on the objects (at least the planet). Please properly set your lighting parameters for clear demonstration. Keyboard interaction is allowed for you to tune light parameters during the demonstration.

d) Generate an asteroid ring cloud that contains at least 200 random floating rocks around the planet. These floating rocks should have random locations in a limited range. The generated asteroid ring cloud should also do a rotation centered at the planet.

e) The viewpoint should be behind the tail of the spacecraft and vertically higher than it. It should always point to the planet.

f) Do normal mapping for the planet. We provide a normal map for the planet. You should load both the planet texture image and the normal map image to use them in the fragment shader.

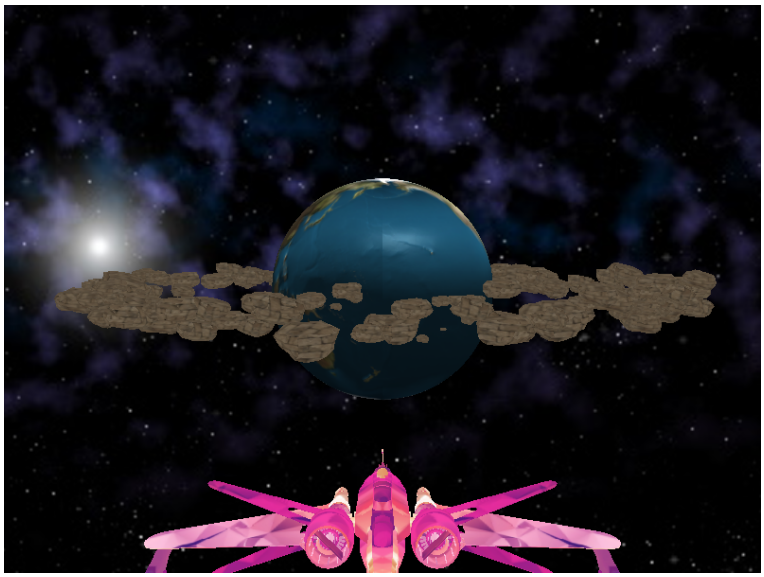

Fig. 2: A closer look at the asteroid ring and the planet with normal mapping.

For interaction:

1) Mouse. Use a mouse to control the rotation of the spacecraft. For example, if you move the mouse to the left, the head of the spacecraft will turn right.

2) Keyboard. Please use the following four keys to control the translations of the spacecraft:

   i. By pressing "W": Move the spacecraft forward (go closer to the planet) by a certain distance.

   ii. By pressing "S": Move the spacecraft backward (leave the planet) by a certain distance.

   iii. By pressing "A": Move the spacecraft to the left by a certain distance.

   iv. By pressing "D": Move the spacecraft to the right by a certain distance.

2. Bonus requirement:

a) Add another light source. The basic light rendering result of two light sources should be determined according to the summation property of the Phong Illumination Model.

b) Additional meaningful objects. You should include other meaningful objects, and the corresponding textures. Anything that might exist in the space is acceptable.

c) Interesting and creative interactions. You should include other meaningful interactions. In the demo video, we provide an example that is when the spacecraft **gets close (collision detection)** to the space vehicle, the **texture changing** is activated. You should create other meaningful scenes for this requirement.

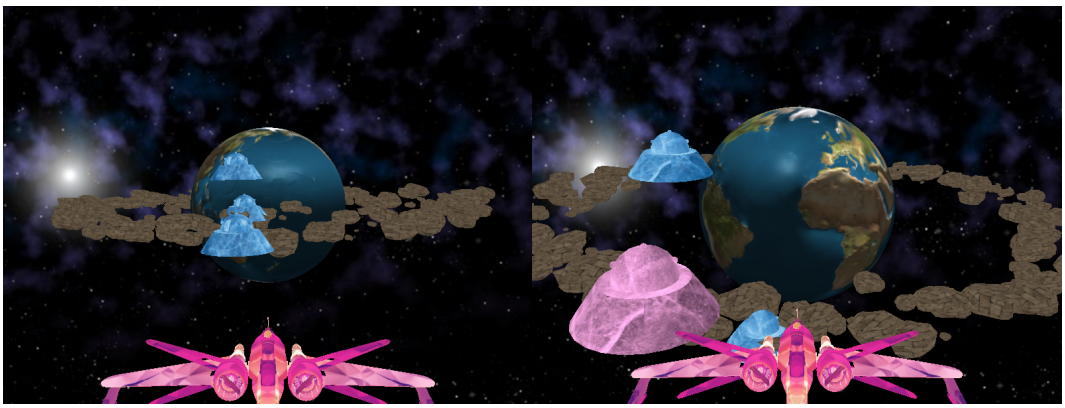

Fig. 3: The example interaction of collision detection and texture changing.

III. Framework and Files

1. We provide the basic .obj files of the objects in this project (planet, spacecraft, rock, craft). Corresponding texture images are also provided.
2. We provide a demo video. Do watch it carefully to fully understand our requirements.
3. Your solution programs of assignment 1 and 2 should provide you a good starting point. Most tasks in this project can be decomposed into easy tasks that have been taught in our lectures and tutorials. We provide some suggestions for you:
   - Keep a good knowledge of the transformations among model coordinate system, world coordinate system, and camera coordinate system.
   - Keep a good knowledge of rendering pipeline, VAO, and VBO. You may be confused by handling so many objects at the same time. Try to use VAO and VBO to help you figure out, because various information of rendered objects can be attached with those items.
   - Try to keep clean coding style. Clean coding style is helpful for debugging. Keep your mind clear by proper annotations. Also, try to enclose the repeated codes into functions.
4. Recommended libraries:
   - GLEW for querying and loading OpenGL extensions.
   - GLM which is a C++ mathematics library for graphics software.

IV. Report

Prepare a .pdf file including the following parts:

1. A figure which shows the overall scene like Figure 1.
2. The frames that provide close look at the basic light rendering results on each kind of the objects.
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.
4. The frames that can represent any bonus features that you have implemented.
5. Some brief and necessary descriptions of your implementation details.

V. Files Need to Code

1. main.cpp: You would mainly code with main.cpp for scene rendering and interaction;
2. nm.vs: You are supposed to implement the vertex shader for the object with normal mapping;
3. nm.fs: You are supposed to implement the fragment shader for object with normal mapping.

Please find the skeleton code and tutorials for more details and hints for implementation.

VI. Grading Scheme

Your assignment will be graded by the following marking scheme:

	Basic (80%)	
1	Render one planet, one spacecraft and at least one craft	10%
2	Self-rotation for the planet	5%
3	Render a skybox	8%
4	Basic light rendering	6%
5	Render an asteroid ring cloud	10%
6	The rotation of the rocks	8%
7	Correct viewpoint	8%
8	Normal mapping for the planet	10%
9	Use mouse to control the translations of the spacecraft	8%
10	Use keyboard to control the translations of the spacecraft	8%
	Bonus (Only two of the following leads to full marks)	20%
a)	Add another light source and do basic light rendering according to the summation of the Phong Illumination Model	10%
b)	Add additional meaningful objects with corresponding textures.	10%
c).	Interesting and creative interactions, distance detection or others.	10%
	Total:	**100%**

Note: Considerable grade deduction will be given if the program is incomplete or fails to compile during the demonstration.

VII. Project submission and demonstration guidelines

1. Homework 3 does not accept group cooperation. Please finish it by yourself. Copy will drive you fail this course.
2. You can write your programs on Windows and macOS. Previously, the official grading platform is Windows with Visual Studio. If we encounter problems when execute/compile your program, you may have to show your demo to the tutor in person.

3. We only accept OpenGL code written in the programmable pipeline. No points will be given if your solution is written in the fixed pipeline.
4. We only accept OpenGL code implemented with GLFW and GLEW. No points will be given if you use other windowing and OpenGL extension libraries (unless you have strong enough reasons).
5. Zip the source code file, the executable file (e.g., openGL.exe if you use windows), the readme file (i.e., readme.txt), the .obj files you create or download and the image files you create or download in a .zip. Name it with your student id (e.g. 1155012345.zip)
6. Please submit through blackboard on time.
7. In case of the multiple submissions, only the latest one will be considered.