CSCI 3260 Principles of Computer Graphics

Course Project: Space Travel

Due Time: 11:59pm, December 4, 2022

Late submission is NOT allowed

Fail the course if you copy

I. Introduction

In the year 3022, astronomers discover a **planet** with **meteorites** around, where several **space vehicles** 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 project. 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 space vehicles 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 vehicle moving like in the demo video, static position or other movement is all right.
   b) The planet and the local space vehicles should do self-rotation all the time.
   c) The local space vehicles should move in the horizontal axis all the time.
   d) Create a skybox as the background of the virtual scene.
   e) Create a point light source. Basic light rendering (ambient, diffuse and specular) should be obviously observed on all objects. Please properly set your lighting parameters for clear demonstration. Keyboard interaction is allowed for you to tune light parameters during the demonstration.
   f) 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.
   g) The viewpoint should be behind the tail of the spacecraft and relatively stationary with it. The viewing direction should be consistent with the direction to which pointed by the head of the spacecraft. Watch our demo video for an intuitive illustration.
   h) 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.

   i) 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 left.
      2) Keyboard. Please use the following four keys to control the translations of the spacecraft:
         i. Up cursor key: Move the spacecraft forward by a certain distance.
         ii. Down cursor key: Move the spacecraft backward by a certain distance.
         iii. Left cursor key: Move the spacecraft to the left by a certain distance.
         iv. Right cursor key: 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.

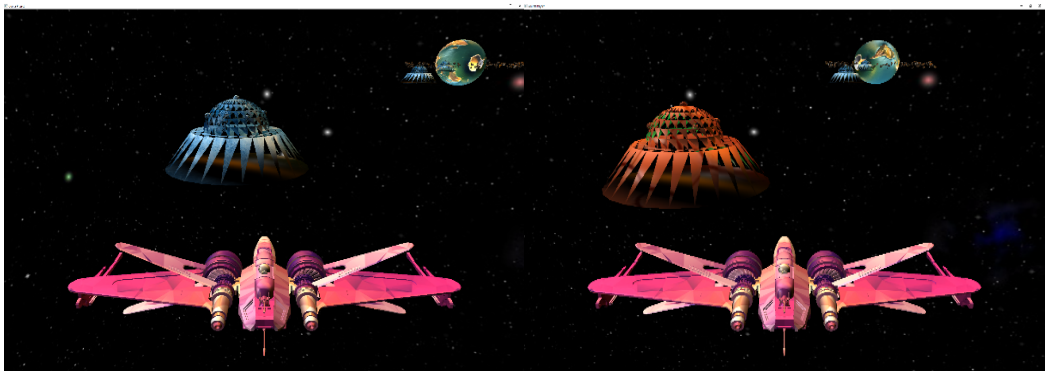

Fig. 3: The example interaction of collision detection and texture changing.

III. Framework and Files

1. We provide the basic .obj files of the objects in this project (planet, spacecraft, rock, space vehicle). 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:
   • FreeGLUT for creating and managing windows containing OpenGL contexts.
   • GLEW for querying and loading OpenGL extensions.
   • GLM which is a C++ mathematics library for graphics software.
   • Assimp for loading 3D object models from .obj files.
   • SOIL for loading textures from images.

IV. Report

Prepare a .pdf file including the following parts:

1. A figure which shows the overall scene like fig. 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.
4. The frames that can represent any bonus features that you have implemented.
5. Some brief and necessary descriptions of your implementation details.

V. Grading Scheme

Your assignment will be graded by the following marking scheme:

	Basic (80%)	
1	Render one planet, one spacecraft and at least one space vehicles	10%
2	Self-rotation for the planet and the local space vehicle	8%
3	Render a skybox	8%
4	Basic light rendering	4%
5	Render an asteroid ring cloud	10%
6	The rotation of the rocks	8%
7	Correct viewpoint	8%
8	Normal mapping for the planet	8%
9	Use mouse to control the self-rotation of the spacecraft	8%
10	Use keyboard to control the translations of the spacecraft	8%
	Bonus (20%)	
1	Add another light source	5%
2	Additional meaningful objects.	10%
3	Interesting and creative interactions.	10%
	Total:	**100%**

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

VI. Project submission and demonstration guidelines

1. Find your group member early. There are at most two members in one group. You should work alone if you cannot find a groupmate.
2. Compress your project files (.h, .cpp, shaders, new obj/bmp if any) and project report in a .zip file. Name it with your student IDs (e.g. 11550XXXXX_11551XXXXX.zip) and submit it to the eLearning Blackboard before 23:59, December 4 (Sunday). Only one student in one group needs to submit. Late submission is not allowed.

3. The project demonstration will on the tutorial hours in the first week after submission. Time slot for each group will be collected later and announced via eLearning Blackboard. If your group is unavailable to demonstrate on that day, please inform us one week earlier with convincing proof for your absence, and we will arrange another day for you.
4. We will announce the demonstration zoom via Blackboard later.
5. Please come to the demonstration zoom earlier before your time slot to test your program. Make sure it can be executed successfully during the demonstration.
6. A few questions will be asked during the demonstration. You will be asked to explain some of the codes in your program and discuss about how the features are implemented.