CSCI 3260 Principles of Computer Graphics

Course Project: Universe Exploration (20%)

Demonstration Date: Dec 2 (Saturday), 2017

Tutor: Xin Yang

Late submission is NOT allowed

No Skeleton Code is provided, so START FROM SCRATCH EARLY!

Fail the course if you copy!

I. Introduction

In this project, you are required to build a universe-like world and enable the interaction based exploration. Once you implement the basic requirements, then you are the director for the rest of the story. The demo implementation is shown as the following picture:

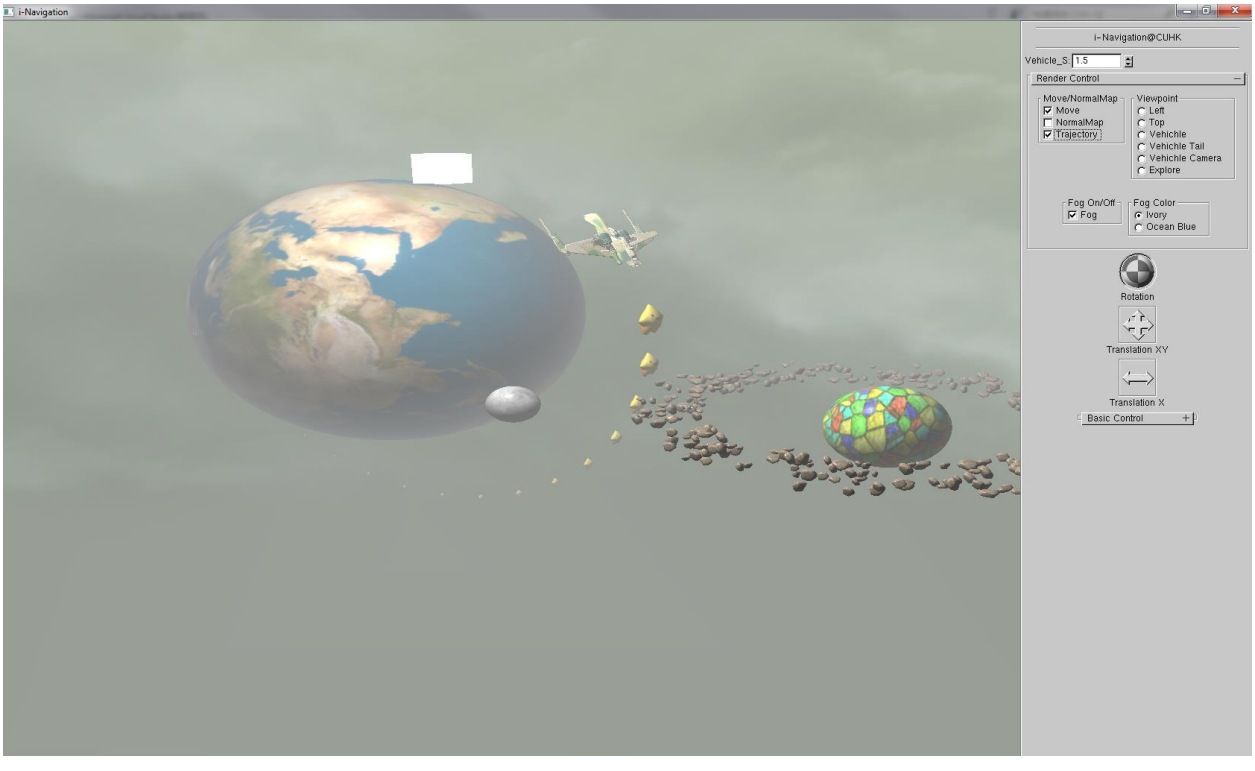
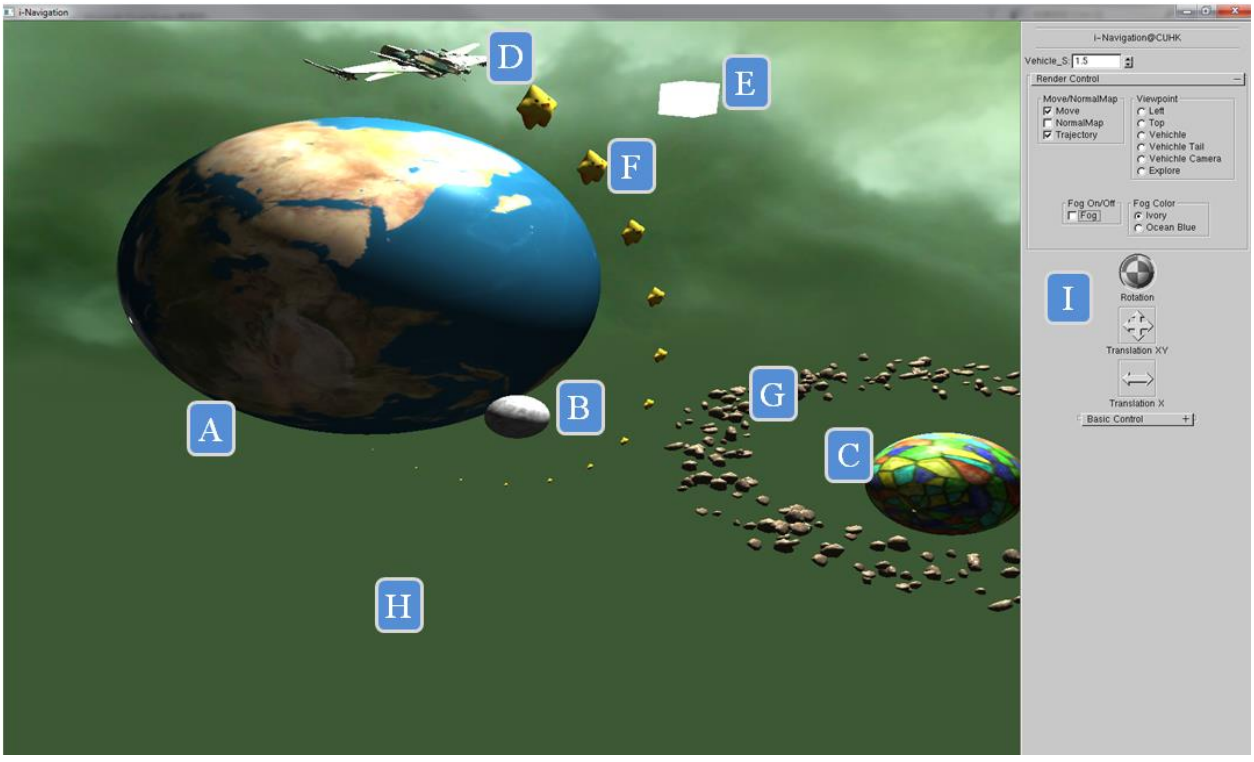


Fig. 1. The demo implementation of course project

You are required to write your own code from scratch to build all the elements in your Universe. All the basic techniques needed have been or will be introduced in our tutorials. Your best skeleton code is your assignments 1 and 2. We will only provide some very short code for necessary support. The program should allow the user to explore the Universe from different aspects. A snapshot to annotate the elements is shown below.



The ultimate objective of this project is to give you an opportunity to practice more with the basic but very important topics in graphics: you will go through *object loading, transformation matrix, viewpoint switch, skybox, lighting, (multiple) texture mapping, multiple shader, normal mapping, fog, tracking, collision detection, instancing, interaction and GUI* before you get a satisfying mark. You have played with most of these basic techniques in assignments 1 and 2, but integrating them together and running them smoothly is really a challenge. Once you get to the destination, you will have a good insight of the rendering pipeline in OpenGL and step further towards the OpenGL based complex rendering.

II. Implementation Details

Basic Requirements:

You should accomplish the following goals to get the basic points.

1. Create 3 planets (A, B, C), 1 light source box (E) and 1 space vehicle (D), and place them properly. *Keep the aspect ratio of the scene correct when we change the window size with "glutReshapeFunc" function. You can load any possible objects or images as your planets, vehicle.*

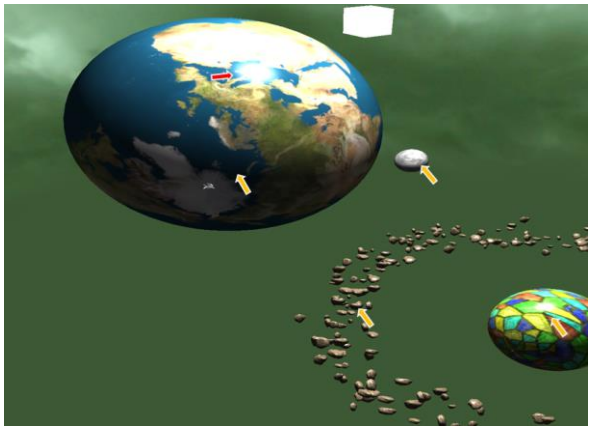
2. Create a Skybox (H) as the background and it should rotate accordingly when you change viewpoint and look around the Universe. *No translation and zoom should happen for Skybox.*

3. Generate an asteroid ring cloud (G) which contains at least 200 random floating rocks around planet C. *Those floating rocks should have random sizes and locations in a limited range.*

4. Conduct single texture mapping and normal mapping for planet A; single texture mapping for planet B, space vehicle D and every rock in G; Conduct multiple texture mapping for planet C. Textures used for A, B, C, D and G should be different from each other. You can use any texture you like, even a chocolate or strawberry.

5. Basic light rendering (ambient, diffuse and specular) should be **obviously** observed on A, B, C, D and G.

Please properly set your lighting parameters for clear demonstration. Keyboard interaction is allowed for you to tune parameters during demonstration.



6. *For planet A and C*, they should do self-rotation all the time. *For planet B*, it should move around A at a designed orbit (circle or other forms as you like). Also, B should have a self-rotation at the same time, just like what moon does against earth.

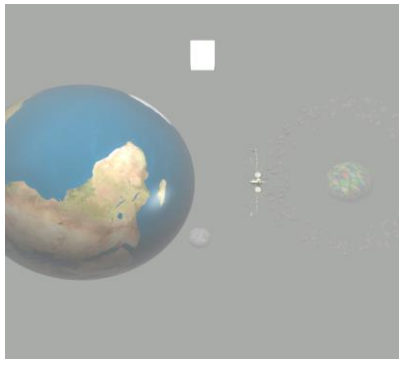
7. *For light source E*, it should move around the center of the universe. It could be better if you can make this movement periodically, just like sunrise and sunset. Here, we represent and visualize the light source with a box at the light position. Basically, only pure color is required for the light source.

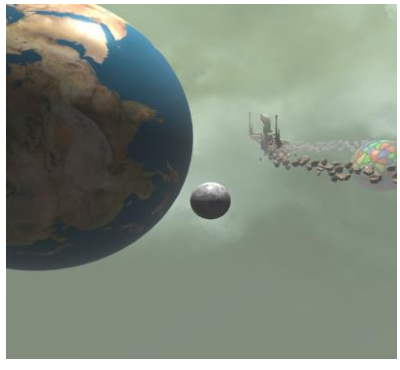
8. *For space vehicle D*, it should basically move around A at a designed orbit.

9. *For asteroid ring cloud G*, all the floating rocks in it should move around planet B simultaneously.

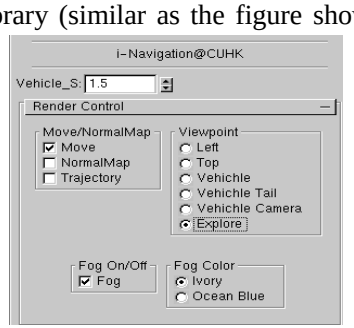
10. *For viewpoint switch*, at least 3 distinctive viewpoints (such as –X, +X, +Y axis) should be provided as choices so that we can explore your Universe from different aspects.

11. *Fog effect*. Fill the whole universe with fog. When we move around, we should be able to experience the different fog density and visibility, shown as figures. Fog color should be able to be changed in GUI part.





12. *Graphical User Interface*. A GUI should be created with the GLUI Library (similar as the figure shows) which includes the module:



(a) A spinner to control the speed of vehicle.

(b) A radio group to set viewpoint.

(c) A checkbox to turn on/off fog effect, a radio group to set fog color.

13. *For interaction*:

a) *Keyboard*. Please set lower case ‘a’, ‘s’, ‘d’ as switch between 3 different viewpoints.

b) *Real-time speed and orbit control.* Please set the up and down arrow keys to control the speed of the space vehicle. Left and right arrow keys to increase/decrease the orbit radius size of the space vehicle.

c) *Mouse*. Please make the setting so that we can zoom in/out when we use the middle wheel of mouse. Control the position of camera by mouse (same as assignment 2) so that we can move around your universe.

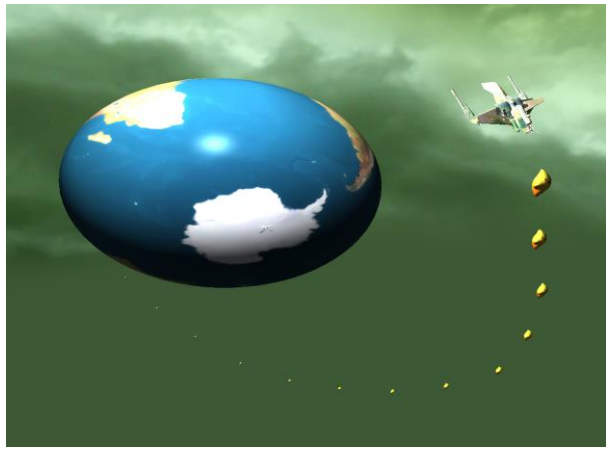
Bonus Features

YOU CAN ACHIEVE ALL BASIC REQUIREMENTS by using the information from the course and tutorials. Presenting your creativity should be more exciting than just finishing the project.

The full mark for bonus part is 25 points.

The bonus points are given depending on the excellence and difficulty of your work. Implementing challenging and diverse visual effects will get you higher score. Here are several suggestions:

- Add another visible light source. All basic light rendering results should then be changed according to the summation property of the Phong Illumination Model. ***(4 points)***
- *Collision Detection.* Detect the collision between any two objects. Once the collision happens, disable the rendering of one object to make it disappear. You can choose **Landmark distance**/**Bounding box**/**Bounding sphere**. ***(****Marking depends on the elegance of your algorithm.* ***Full mark is 6 points)***
- Provide extra viewpoints on the space vehicle. By doing this, we can really enjoy our space travel and have a close touch with the planet and floating rocks. ***(4 points)***
- Represent the real-time trajectory of space vehicle D with a string of stars (F). You need to record and update several real-time positions of the vehicle and render stars on those positions. Shown as the following figure. ***(5 points)***



- Generate the asteroid ring cloud with more than 5000 floating rocks. Some basic generation method will becomes un-renderable as so many repeated objects need to be rendered at the same time. So you are encouraged to explore the amazing instanced rendering. *Code of instancing will be checked during*

demonstration. ***(6 points)***

- Use more "advanced" texture mappings to make objects more realistic, such as shadow mapping environment mapping, displacement mapping etc. ***(7 points)***
- Play background music with *IrrKlang* Library. ***(2 points)***

Other interesting and fantastic bonus idea will be graded accordingly.

III. Framework and Files

1. We will provide the basic .obj files for the planet, vehicle and floating rock, also texture images and skybox. You are encouraged to illustrate your creativities with other extra items you like.
2. Your assignment 1 and 2 should provide you a good starting point, and you can immediately start this project from them. Most tasks in this project can be decomposed into easy tasks that have been taught in our lectures and tutorials, but before you start, several things need to be mentioned:
    - Keep a good knowledge of the transformation matrix. Although no complex rendering algorithms are involved in this project, but a lot of rendering are realized by hierarchical transformation matrices (such as the self-rotation, rotation around other planet, matrix array for floating rock, trajectory and viewpoint switch). So, being good at playing with matrix is important for this project.
    - 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, VBO object to help you figure out, because various information of rendered objects are attached with those items.
    - Get a good practice with multiple shaders and multiple shader program objects. Equip your shader programs for different objects with the right components, and then enable your shader program for specific object to render at the right time.
    - Try to keep clean coding style. Since more than 7 objects need to be rendered, a lot of stuff (lighting, transformation matrix and texture etc.) need to be done before drawing, and those stuff are placed in one paint function, so try to keep your mind clear by clear annotation. Also, try to enclose the repeated codes into functions. Clean coding style is also helpful for debugging.
3. The libraries that you can use are limited as the following list: FreeGLUT, GLEW, GLM, GLUI, IrrKlang, Assimp and SOIL. If you really need more libraries to realize more fantastic rendering effects, please contact tutors for permission.

IV. Report

Prepare key-frame snapshots of your Universe in a report file **(.pdf)**, basically including the following parts. (***Report is important for final marking.)***

1. The frames from different timepoints which can illustrate the movement of all moving objects
2. The frames which can provide close look at the basic light rendering results (zoom in function can facilitate you in this part).
3. The frames which can provide close look at the normal mapping rendering results on planet A.
4. The frames which illustrate the fog effect with different visibilities.
5. The frames which are captured from 3 distinctive viewpoints.
6. The frames that can represent any bonus features that you have implemented. Some brief and necessary descriptions of your implementation details are appreciated.

V. Grading Scheme

Your final project will be graded by the following marking scheme:

	Basic	**73%**
1	Object creation (2+1+1 point for each)	4%
2	Create the skybox (3 points)	3%
3	Generate the asteroid ring cloud (6 points)	6%
4	Texture mapping (3+2+2+2+3 points)	12%
5	Light rendering (2+2+2+2+2 points)	10%
6	Self-rotation of planet A and C (2+2 points). Self-rotation of planet C, rotation around planet A (3+4 points)	9%
7	Movement of Light source box (3 points)	3%
8	Space vehicle D move around planet A (3 points)	3%
9	Floating rocks move around planet C (4 points)	4%
10	Viewpoint switch (3 points)	3%
11	Fog effect with color setting (6 points)	6%
12	GUI (2+2+2)	6%
13	Interaction (1+2+1 points)	4%
	Bonus	**25%**
	Report	**2%**
	Total:	**100%**

Note: Considerable grade deduction will be given if the program is incomplete or fails compilation in demonstration.

VI. Project demonstration and submission guidelines

1) **Find your group member early. At most 2 members in one group.**

2) The project demonstration will start at **10:00 am, 2-Dec-2017**. Time slot for each group will be collected later and announced via eLearning Blackboard. Please upload your project files to eLearning **before 9:30am** of the due date. **No further modification or late submission is allowed**. You cannot update the uploaded codes (.h, .cpp and shaders) in you demonstration. 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.

3) We will announce the demonstration venue via eLearning later.

4) Please come and test your program in the demonstration venue earlier to make sure it can be executed successfully during the project demonstration.

5) 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.

6) **After your demonstration, zip the whole project and your report in a .zip.** ***Your report is important for final marking.*** Name it with your group number (e.g. group01.zip) and submit the .zip via eLearning Blackboard. (https://elearn.cuhk.edu.hk/) (Only one student of each group has to submit)

VII. Acknowledgements

Thanks Prof. Heng, Prof. Philip, Xianzhi Li and Yueming Jin for discussion and contribution.