521140S Computer Graphics (2019) Programming Assignment III

General information

In this assignment, the goal is animate the solar system, techniques including hierarchical modeling, light shading and input-output device control will be used to fulfill the task.

Deadline for returning the work is 2019-05-16 24:00. If you need an extension for any reason, please apply for it before the deadline.

The maximum number of points for this assignment is 8, the minimum requirement is to get 5 points to pass the assignment. Missing or incorrect functionality will reduce the points.

Introductions in the task guidelines are based on the Windows platform. Programming on Linux or OS system is possible. Please find the Platform-Specific Information in Common additional material page at Noppa. A submission working on one platform is enough.

The assignment is to be finished alone. Feel free to discuss the problems but sharing code is not allowed.

Do not use any existing libraries to solve the rendering part of the assignment outside what is used in the framework itself. The purpose of this course is to learn how to implement graphics engines and not using existing ones.

Workstations. The framework has all the required libraries and a project file for MS Visual Studio 2013+ included for both 32- and 64-bit platforms. PC classrooms (TS135 and TS137) have Visual Studio 2015 installed so it is possible to work there.

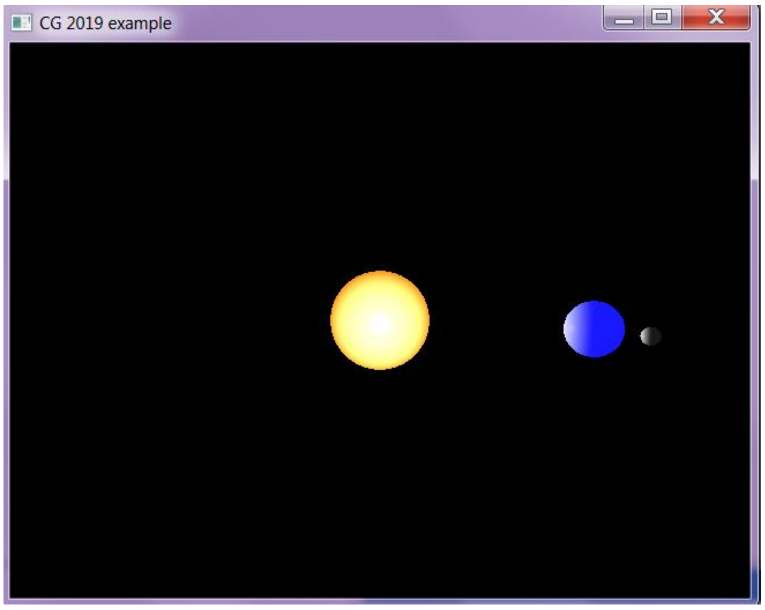

Figure 1. Example scene.

Tasks

1. Download the code package from Noppa, and run the assignment3 scene. There is a sphere with Gouraud shading. Open the vertex shader sample.vs, change the following parameters and report how these parameters influence the lighting effect. (1 p)
    a. const vec4 ambientProduct;
    b. const vec4 diffuseProduct;
    c. const vec4 specularProduct;
    d. const float shininess;

2. Render your own scene to animate a solar system with revolutions and rotations as in Figure 1. In the scene there are three objects. The transformations should be bug free to get full points.
    a. The sun. Rotate on its own. (1 p)
    b. The earth. Rotate on its own and rotate around the sun. (1 p)
    c. The moon. Rotate on its own and rotate around the earth. (1 p)

   Hints: (1) The initiation of the three objects has been created in the bool assignment3::init(). You mainly need to implement the void assignment3::render() part. (2) Reasonably select and order glm::rotate(), glm::translate() and glm::scale() for object transformations. (3) You need to use hierarchical modeling to render the scene, as the animations of the objects are like a tree structure. That is, the moon animates based on the transformations of the earth while the earth moves based on the transformations of the sun. Thus you need to stack the model matrix such that the transformations from the parent can be stored. In the *assignment3.cpp*, the class <stack> has been included and std::stack<glm::mat4> glm_ModelViewMatrix; is defined to stack the modelView matrix. Use glm_ModelViewMatrix.push() when the model view matrix needs to be stored, and use glm_ModelViewMatrix.pop() to release the recent storage. The following example code is based on the hierarchical structure in Figure 2.

```
    glm_ModelViewMatrix.push(glm::mat4());
    glm_ModelViewMatrix.top() = viewMat*glm_ModelViewMatrix.top();

render obj1

    glm_ModelViewMatrix.top() = glm::translate(glm_ModelViewMatrix.top(), …)
    glm_ModelViewMatrix.top() = glm::rotate(glm_ModelViewMatrix.top(), …)

    ……

    glm_ModelViewMatrix.push(glm_ModelViewMatrix.top());

render obj2

    glm_ModelViewMatrix.top() = glm::rotate(glm_ModelViewMatrix.top(), …)

    ……

    glm_ModelViewMatrix.push(glm_ModelViewMatrix.top());

render obj3

    glm_ModelViewMatrix.top() = glm::rotate(glm_ModelViewMatrix.top(), …)

    ……
```

```
glm_ModelViewMatrix.push(glm_ModelViewMatrix.top());

render obj4

glm_ModelViewMatrix.pop();

glm_ModelViewMatrix.push(glm_ModelViewMatrix.top());

……
```

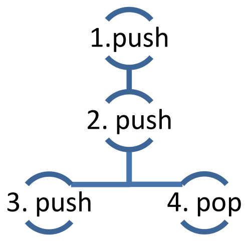

Figure 2. Render hierarchical objects from 1 to 4. ‘push’ means that the model matrix needs to be stored for children transformation. ‘pop’ means that the transformation is not based on the previous object but based on its parent.

3. Add light shading effects to at least one object on the scene.
   a. Modify the sample shaders to involve object color information to the shading. (1 p)
   b. Modify the shaders to change the Gouraud shading to Blinn Phong shading. (2 p)

   Hints: To involve object color to the light shading, you just need to add the object color to the calculated intensity in the shaders. The difference between the Gouraud shading and the Blinn Phong shading can be found in https://www.haroldserrano.com/blog/what-is-the-difference-between-gouraud-and-phong-shading, where you should calculate the output colors in the fragment shader. An example of implementing Blinn Phong shading can be found in https://en.wikipedia.org/wiki/Blinn%E2%80%93Phong_shading_model.

4. Drag the mouse to move the solar system. Left click the mouse and drag the solar system, the solar system reallocates at the position where the mouse button up. (1 p)

You can use any of the example code included in the framework directly in your assignment if you find them useful. You can also freely reuse any of the previous code from earlier assignments that you created yourself.

What to return

Please send the following to any one of the TAs using **title format "CG3 _<student id>_ <given name> _<family name>"** during the given deadline date at the latest (Deadline date is at first page, assume Finnish time).

The submission should contain the following documents:

1. A PDF document describing your programming work and how you solved your assignment. Make sure your document contains at least the following information:
   a. Your name and student id number
   b. What operating system(s) was/were used to develop and test the software.
   c. Number of hours needed to finish the assignment
   d. Description of how you solved the programming tasks (General description of the method, any used external information sources used etc.) Please include information of any problems you encountered and how you managed to solve or work around them.
   e. Any feedback you wish to provide.
2. A zip file containing **assignment3.cpp, assignment3.h, and all the shader files.**

Contacts

Please contact the TAs for submission or for help if you face with problems. The TAs for the programming assignments are as follow:

Tuomas Varanka: tuomas.varanka@oulu.fi (Finnish & English)
Yingyue Xu: yingyue.xu@oulu.fi (English)
Nhat Vo: nhat.vo@oulu.fi (English)