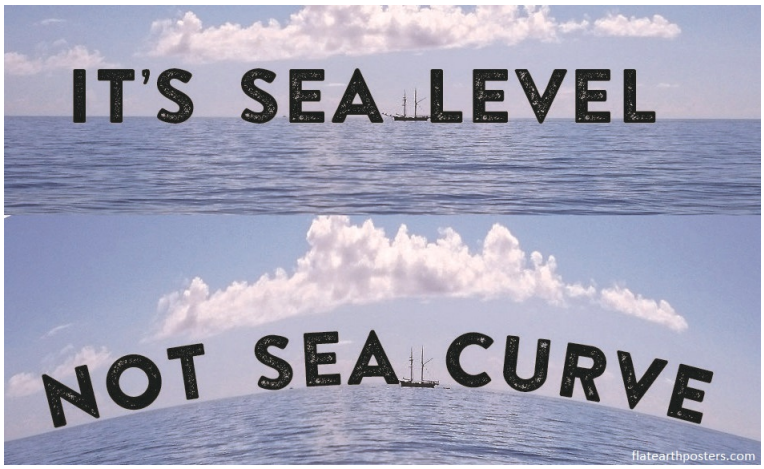
Middle East Technical University | Department of Computer Engineering

CENG 477

Introduction to Computer Graphics

Fall '2020-2021
Assignment 4 - OpenGL Programmable Shaders

Due date: January 21st, 2021, Thursday, 23:59



1 Objectives

The flat Earth model is an archaic conception of Earth's shape as a plane or disk. Many ancient cultures subscribed to a flat Earth cosmography, including Greece until the classical period, the Bronze Age and Iron Age civilizations of the Near East until the Hellenistic period, India until the Gupta period (early centuries AD), and China until the 17th century. Today, there is still a society that tries to prove the concept with some evidence. In this assignment, you are going to implement an OpenGL program to render flat Earth model using vertex and fragment shaders to display a texture image as a height map to fly through the scene interactively by processing keyboard input. The input for this assignment is two image files. Namely, the height map and the texture map. There will be one point light.

Keywords: *OpenGL, programmable shaders, texture mapping, height map, interactive fly-through*

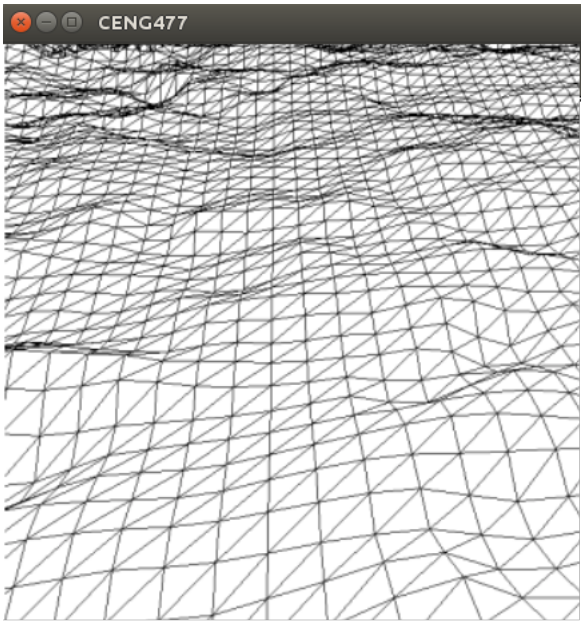
2 Specifications

1. The name of the executable will be "hw4".

2. You will be given a sample makefile. You can extend it or use as it is. However, be sure, before submitting, it is running on **Inek** machines. Your executable will take two image files in "jpg" format as a command-line argument. The first file will be used to compute the heights and the second file is there for texture mapping. Both images will have the same resolution. The executable should be run with the "**./hw4 height_map.jpg texture_map.jpg**" command smoothly. It will not produce any outputs since an OpenGL display window will be used.

3. You can use "X11 forwarding" to work on **Inek** machines. To do so, you need to first connect to **login.ceng.metu.edu.tr** (not ~~external.ceng.metu.edu.tr~~) through ssh with -X flag. After you reached the login lobby machine, you need to connect to any of the inek machines with -X flag.

4. You will create flat earth in OpenGL with the same resolution of the texture image. Each pixel will be represented by two triangles as in the following image. If the width and height of the texture image is w and h, there will be a total of $2 \cdot w \cdot h$ triangles in your heightmap. The flat heightmap will be on the xz plane with the corner vertices at (0, 0, 0) and (w, 0, h).



5. The keys used for moving or changing the position of elements inside the scene should be handled continuously. Namely, it should not be used with the action of any type; while pressing the key, it should execute the related commands repetitively.

6. The heights, i.e., y-coordinates, of the vertices will be determined in the vertex shader from the corresponding texture color(only R channel) on the heightmap image. The computed height will be multiplied with a height factor to determine the final height. The height factor will be initially 10.0 and the user will be able to increase or decrease this factor by 0.5 with the R and F keys.

7. Q and E keys will move the texture and height map along the plane left and right respectively by subtracting and adding 1.

8. There will be one light source in the scene at the initial position of $(w/2, 100, h/2)$. The intensity of the light source is (1.0, 1.0, 1.0) and there will be no attenuation.

9. The light source can be moved using arrow keys. The height of the light source(y-position) is increased and decreased with T and G keys respectively. The position of the light source is changed by 5 for all directions.

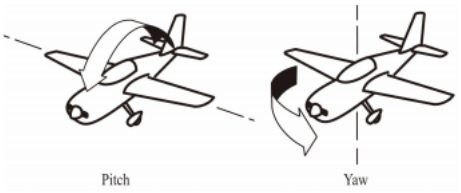
10. You will implement Phong shading for determining the color of surface points.

```
Ambient reflectence coefficient = (0.25, 0.25, 0.25, 1.0)
Ambient light color = (0.3, 0.3, 0.3, 1.0)
Specular reflectence coefficient = (1.0, 1.0, 1.0, 1.0)
Specular light color = (1.0, 1.0, 1.0, 1.0)
Specular exponent = 100
Diffuse reflectence coefficient = (1.0, 1.0, 1.0, 1.0)
Diffuse light color = (1.0, 1.0, 1.0, 1.0)
```

11. The computed surface color will be combined with the texture color do determine the final color of the surface. "clamp" method can be used to combine these two colors.

12. To implement Phong shading, you will have to compute the normal vectors of the vertices at the vertex shader and define the normal vector as a varying variable so that the normal vectors will be interpolated by the rasterizer and passed to the fragment shader. The normal vector of a vertex is defined as the average of the normal vectors of the triangles that this vertex is adjacent to. You will have to query the heights of the neighboring vertices (hint: use texture look-up for this, too) to compute the normals of the adjacent triangles.

13. You will implement a camera flight mode to be able to fly over the visualized terrain. The camera will have a gaze direction, which will be modeled by two angles for pitch and yaw. See the following figure for the definitions of these terms.



Pitch and yaw angles are very closely related to the phi and theta angles that we have used to represent a sphere with parametric equations. The user will be able to change pitch with W and S keys and change yaw with A and D keys where changing is 0.05 unit. W and S keys rotates gaze around the left vector. A and D keys rotates the gaze around the up vector. Do not forget to update up and left vector with normalizing later. The camera will also move with some speed at every frame. The speed will increase or decrease with 0.01 by pressing the Y and H keys on the keyboard. Initially, the speed will be 0 and the camera gaze will be (0, 0, 1). The camera will be positioned initially at (w/2, w/10, -w/4) where w is the width of the texture image. Pressing the X key, the plane will stop. Namely, the speed will be 0.

14. When pressing the I key, the plane will be placed to the initial position with initial configurations of the camera and speed of 0.

15. Window size is initially (1000, 1000) and it should be switch to full-screen mode with the key P. Besides, your program should support resizing window operation with the frame.

16. There will be perspective projection with an angle of 45 degrees. The aspect ratio will be 1, near and far plane will be 0.1 and 1000 respectively.

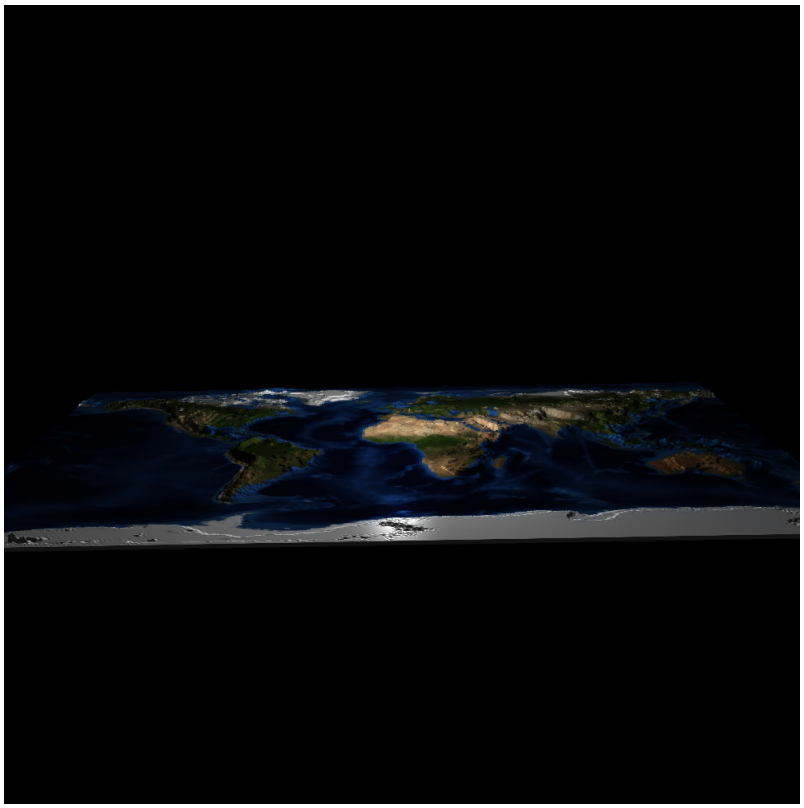
17. Helper functions for reading inputs are given to you. It is **STRONGLY** recommended to inspect types in header files and helper functions, which are given to you in homework stub code.

18. glm files are attached. You can use them with following include statements.

```
#include "glm/glm.hpp"
#include "glm/gtx/transform.hpp"
#include "glm/gtx/rotate_vector.hpp"
#include "glm/gtc/type_ptr.hpp"
```

3 Sample input/output

Sample input files are given at OdtuClass. The initial view of the OpenGL display window for height.jpg and texture.jpg inputs is shown in the image below. You can watch the sample video **here**.



4 Regulations

1. **Programming Language:** C++
2. **Late Submission:** Refer to the syllabus for the late policy.
3. **Groups:** You can team-up with another student. Naming the submission file properly is enough.
4. **Cheating:** We have zero tolerance policy for cheating. People involved in cheating will be punished according to the university regulations and will get 0 from the homework. You can discuss algorithmic choices, but sharing code between groups or using third party code is strictly forbidden. To prevent cheating in this homework, we also compare your codes with online ray tracers and previous years' student solutions. In case a match is found, this will also be considered as cheating. Even if you take only a "part" of the code from somewhere or somebody else, this is also cheating. Please be aware that there are "very advanced tools" that detect if two codes are similar.
5. **Newsgroup:** You must follow the ODTUCLASS forum for discussions, possible updates, and corrections.
6. **Submission:** Submission will be done via ODTUCLASS. Create a "tar.gz" file that contains all your source code files and a makefile. The executable should be named as "hw4" and should be able to be run using the command "./hw4 height_map.jpg texture_map.jpg". **The .tar.gz file name should be:**

   - If a student works with a partner student:

     `<partner_1_student_id>_<partner_2_student_id>_hw4.tar.gz`

   - If a student works alone:

     `<student_id>_hw4.tar.gz`

   - For example:

     `1234567_2345678_hw4.tar.gz`
     `1234567_hw4.tar.gz`

7. **Evaluation:** Your codes will be evaluated on Inek Machines; based on several input files including, but not limited to the test cases given to you as examples. Evaluation will be done manually, small differences might be tolerated. Therefore, if you have subtle differences (numerically different but visually imperceivable) when running the program, that will not be a problem.