CENG 477

Introduction to Computer Graphics

Fall '2017-2018

Assignment 4

Programmable Shaders in OpenGL

Yusuf Mücahit Çetinkaya, Serhat Şahin
{yusufc, serhat.sahin}@ceng.metu.edu.tr

Due date: January 14, 2018, 23:59

1 Overview

In this assignment, you are going to implement an OpenGL program to render a given scene 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 only an image file. There will be one point light, in a fixed position.

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

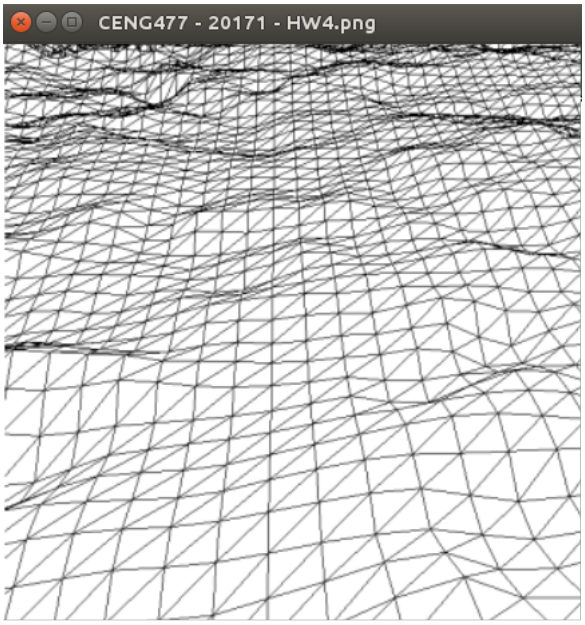
2 Specifications

1. You should name your executable as "hw4".

2. Your executable will take one image file in j̈pg̈format as command line argument. This file will be used as the texture image and to compute the heights on the height map. One should be able to run your executable with the command;

```
>> ./hw4 height_map.jpg
```

3. You will create a flat height map in OpenGL with he 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 height map. The flat height map will be on the xz plane with the corner vertices at (0, 0, 0) and (w, 0, h).



4. The heights, i.e., y-coordinates, of the vertices will be determined in the vertex shader from the corresponding texture color on the image by the following luminance equation:

$$y = 0.2126 \cdot r + 0.7152 \cdot g + 0.0722 \cdot b$$

   This 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 O and L keys.

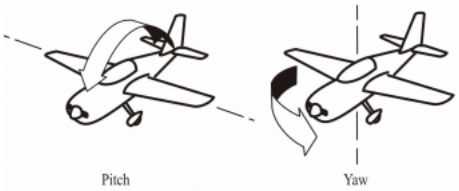
5. There will be one light source in the scene at position $(w/2, w + h, h/2)$. The intensity of the light source is (1.0, 1.0, 1.0) and there will be no attenuation.

6. You will implement Phong shading for determining the color of surface points. The reflectance coefficients and the specular exponent of the surface of the heightmap is provided in the fragment shader contained in the assignment template.

7. The computed surface color will be combined with the texture color do determine the final color of the surface. The method to combine these two colors are also provided in the fragment shader contained in the assignment template.

8. In order 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 thatthis vertex is adjacent to. You will have to query the heights of the neigboring vertices (hint: use texture look-up for this, too) to compute the normals of the adjacent triangles.

9. 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. The camera will also move with some speed at every frame. The speed will increase or decrease with the U and J 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.

10. Window should be switch to full-screen mode with the key F. In addition, your program should support resizing window operation with the frame.

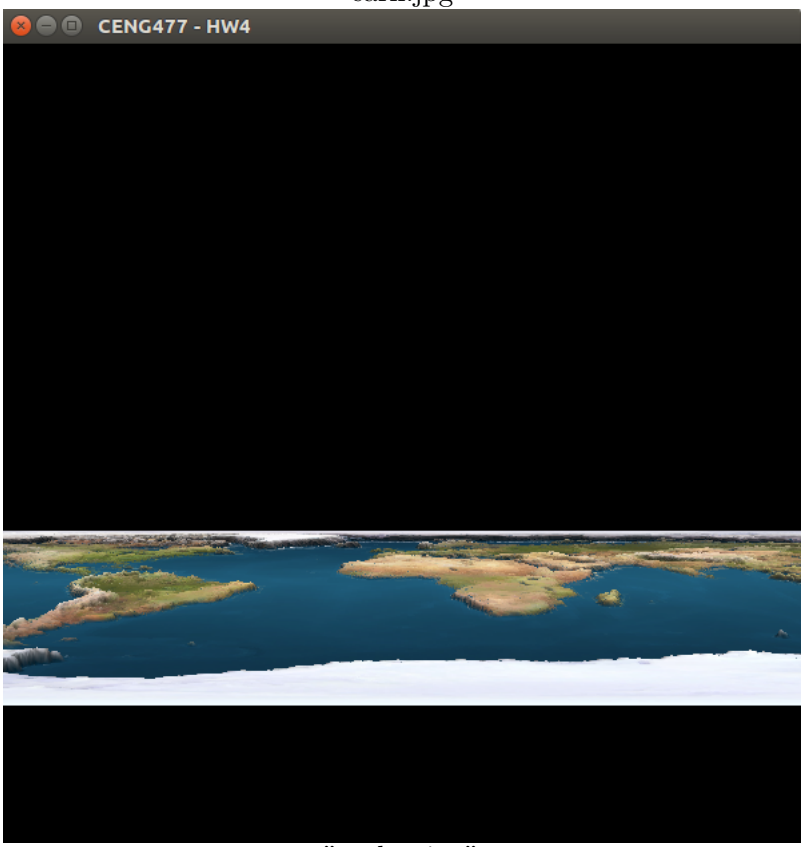
11. There will be perspective projection with an angle of 45 degrees. Aspect ratio will be 1, near and far plane will be 0.1 and 1000 respectively.

12. You will be given sample make file. You can extend it or use as it is. However, be sure, before submitting, it is running on **Inek** machines. It will not produce any outputs since an interactive OpenGL display window will be used.

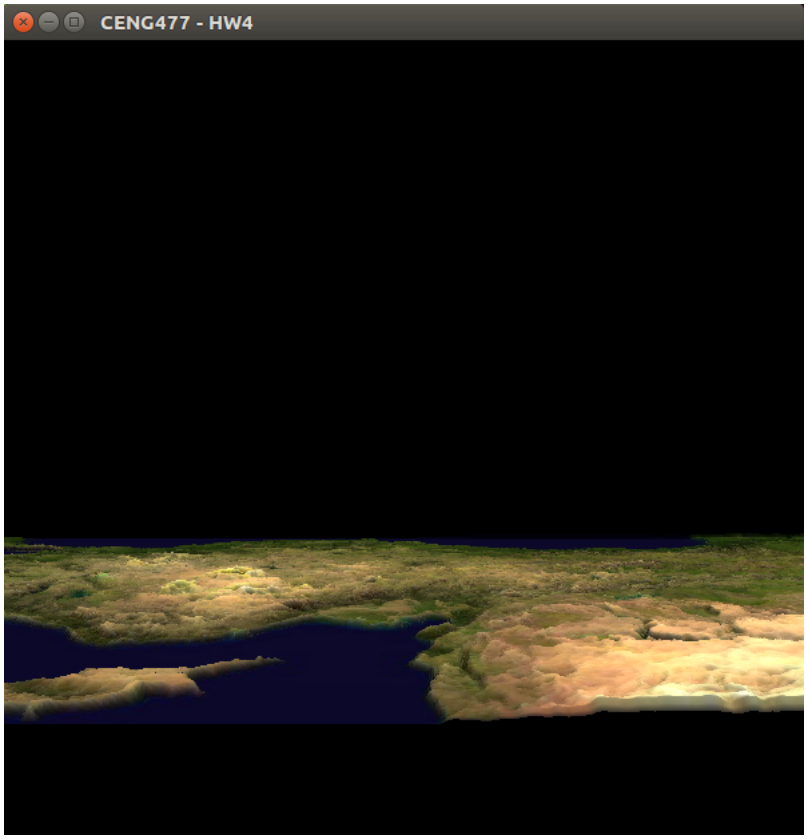
3 Sample input/output

Sample input file is given at COW. The initial view of the OpenGL display window for the "earth.jpg" and "turkey.jpg" inputs are shown in the images below.

"earh.jpg"



"turkey.jpg"



4 Regulations

1. **Programming Language:** C++

2. **Late Submission:** You can submit your codes up to 3 days late. Each late day will be deduced from the total 7 credits for the semester. However, if you fail to submit even after 3 days, you will get 0 regardless of how many late credits you may have left. If you submit late and still get zero, you cannot claim back your late days. As unfortunately the COW system does not have an "unsubmit" option, you must e-mail your assistants if you want your submission to not be evaluated (and therefore preserve your late day credits).

3. **Groups:** You can team-up with another student. But you must notify the assistants about who your partner is.

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. You can discuss algorithmic choices, but sharing code between each other or using third party code is strictly forbidden. Even if you take a "part" of the code from somewhere/somebody else - this is also cheating. Please be aware that there are "very advanced tools" that detect if two codes are similar. So please do not think you can get away with by changing a code obtained from another source.

5. **Newsgroup:** You must follow the newsgroup (news.ceng.metu.edu.tr) for discussions and possible updates on a daily basis.

6. **Submission:** Submission will be done via COW. Create a tar.gz file named "hw4.tar.gz" that contains all your source code files and a makefile. The tar file should not include any subdirectories. The executable should be named "hw4" and should be able to be run using command "./hw4 <path to input jpg file>". If your directory structure is wrong or makefile does not work (or you do not have one) and therefore we will have to manually try to compile your code - there will be an automatic penalty of 10 points. Please do not hesitate to email us if you have any further questions.

7. **Evaluation:** Your codes will be evaluated based on several input files including, but not limited to the test cases given to you as example. We will evaluate your results visually. Therefore, if you have subtle differences (numerically different but visually imperceivable) when running the program, that will not be a problem.