Middle East Technical University | Department of Computer Engineering

CENG 477

Introduction to Computer Graphics

Fall '2020-2021
Assignment 3 - A Software Rasterizer using OpenGL

Due date: January 3rd, 2021, Sunday, 23:59

1 Objectives

In this assignment, you are going to implement an OpenGL program to render a given scene. The input files for this assignment are very similar to the ones in the first two assignments. A number of point light sources will be provided in the scene file. A number of material properties will be defined and assigned to triangular mesh models. The scene will be rendered using the built-in OpenGL illumination model and the smooth shading mode. The parameters of a perspective view camera are also specified in the input. You will perform modeling transformations to triangular mesh models.

Keywords: *OpenGL, fixed pipeline, modeling transformations, viewing transformations*

2 Specifications

1. The name of the executable will be "hw3".
2. Your executable will take an XML file name from command line. Input file names are not static, they can be anything. XML file has minor differences from first and second assignments. It has only one camera and *Transformations* tag. The only object type included in this homework is *Mesh*.
3. 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. The executable should be run with the "**./hw3 sample_input.xml**" command smoothly. It will not produce any outputs since an OpenGL display window will be used.
4. 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.

5. You will compute normal vectors of vertices as the average of the normal vectors of the triangles incident to this vertex.

6. The scene will only be composed of instances of triangles. With a sequence of several transformations (translation, rotation, scaling) you will be able to move, rotate, or resize a model (i.e., all of the triangles in the model). Transformations will be applied to the models in the order specified in the input file. You may use OpenGL functions to transform your models.

7. Models will be rendered in solid or wireframe modes as specified in the input file. Use the OpenGL function call glPolygonMode(GL FRONT AND BACK, GL FILL); for the solid and the call glPolygonMode(GL FRONT AND BACK, GL LINE); for the wireframe modes, respectively.

8. You will not compute lightings explicitly. It will be done automatically by calling appropriate OpenGL functions.

9. In both wireframe and solid modes, triangles whose backface or frontface to the viewer will be culled according to the parameter in the input file. Backface/frontface culling should be enabled/disabled according to the setting in input file. **CullingEnabled can be 0 (disabled) or 1 (enabled) and CullingFace can be 0 (backface) or 1 (frontface).**

10. Use the smooth shading mode of OpenGL by calling the function glShadeModel(GL SMOOTH);.

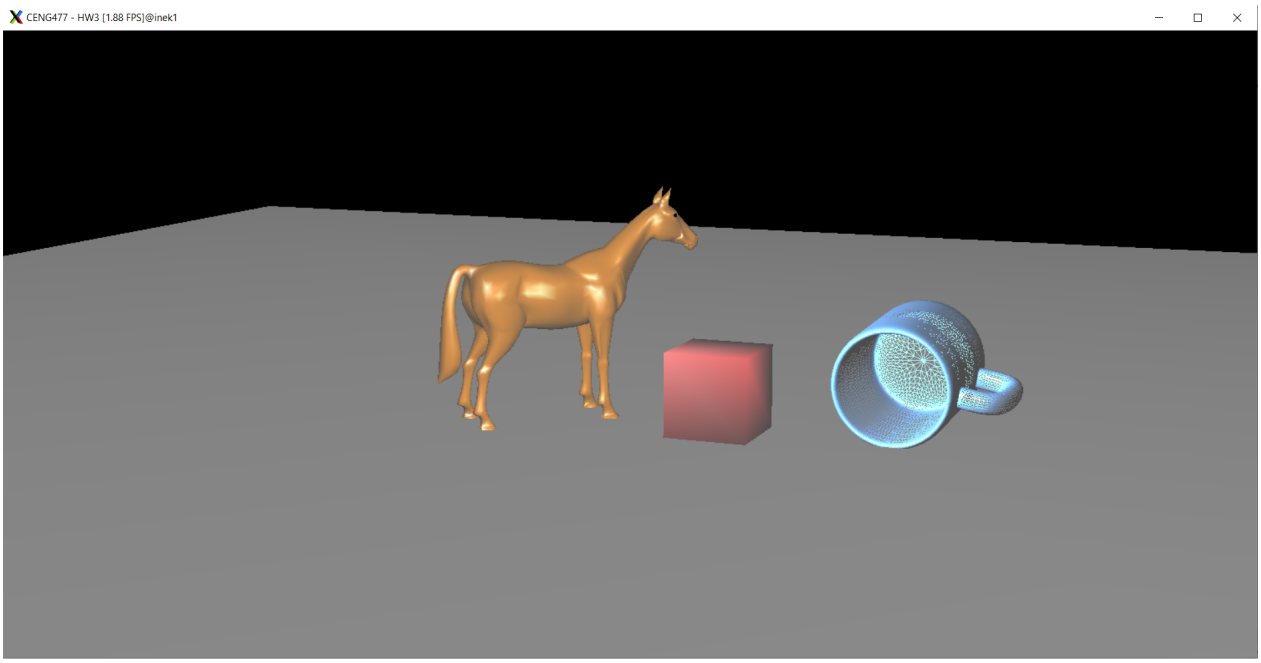
11. To specify the location/parameters of camera, you may use *gluPerspective* function.

12. You will show the FPS (frame-per-second) in the title. The title will look like to the one in the sample image.

13. 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.

3 Sample input/output

Sample input files are given at COW. The initial view of the OpenGL display window for the horse_and_mug input is shown in the image below where the horse is rendered as a filled mesh whereas the mug is rendered as wireframe.



4 Hints & Tips

1. Start early!

2. When lighting is enabled, OpenGL does not use glColor to color the surfaces by default. Instead, surface material properties are defined using the glMaterialfv function. To set the color of surface (or the entire model), use the following piece of code just before drawing the object.

```
GLfloat ambColor[4] = {<ar>, <ag>, <ab>, 1.0};
GLfloat diffColor[4] = {<dr>, <dg>, <db>, 1.0};
GLfloat specColor[4] = {<sr>, <sg>, <sb>, 1.0};
GLfloat specExp[1] = {<specular exponent>};

glMaterialfv(GL_FRONT,GL_AMBIENT,ambColor);
glMaterialfv(GL_FRONT,GL_DIFFUSE,diffColor);
glMaterialfv(GL_FRONT,GL_SPECULAR,specColor);
glMaterialfv(GL_FRONT,GL_SHININESS,specExp);
```

   Finally, to determine the amount of light reflected from the surfaces correctly, you should specify the normal vectors of the vertices.

5 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. But you must notify the assistants about who your partner is within the allowed time frame.
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 "hw3" and should be able to be run using the command "./hw3 scene_file_name.xml". **The .tar.gz file name should be:**

   - If a student works with a partner student:

     `<partner_1_student_id>_<partner_2_student_id>_hw3.tar.gz`

   - If a student works alone:

     `<student_id>_hw3.tar.gz`

   - For example:

     ```
     1234567_2345678_hw3.tar.gz
     1234567_hw3.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.