Computer Graphics Project 2: Obj viewer

Handed out: April 28, 2025

Due: 23:59, May 18, 2025 (NO SCORE for late submissions!)

- *LMS course home > Weekly Learning (주차학습) > Week 8> Project 2*
- *Compress your files into a zip file as in the following example. The zip file can be named whatever you want.*

```
+ submission.zip
  - main.py
  - ...
  - report.pdf
```

- *Your program may consist of several python source files. But the main module should be in **main.py**. That is, your program should be executed with the following command:*

```
python main.py
```

1. 1. Implement your own obj file viewer showing multiple obj meshes.

   A. You must implement all requirements in a single program. This project DOES NOT require each requirement to be a separate program.

   B. The window size doesn't need to be (800, 800). Use the larger window that is enough to see the details of the viewer.

   C. **Your program must use OpenGL 3.3 Core Profile**, meaning that ...

      i. Your python code must include:

```
glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR, 3)
glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR, 3)
glfwWindowHint(GLFW_OPENGL_PROFILE, GLFW_OPENGL_CORE_PROFILE)
```

      ii. Your shader code must start with:

```
#version 330 core
```

      iii. **You will not get any score for this project (except report) if you do not use OpenGL 3.3 Core Profile.**

D. Total points: 110 pts

2. Requirements

A. Manipulate the camera in the same way as in Project1 using your Project1 code (10 pts).

i. Also draw the reference grid plane.

B. Load and render multiple obj files (70 pts)

i. **Open an obj file** by drag-and-drop to your obj viewer window.

1. Google *glfwSetDropCallback* to see how to do it.

2. **This feature is essential for scoring your assignment, so if not implemented, you won't get any score for "Load and render multiple obj files (70 pts)".**

ii. **Read the obj file and display the mesh** using only vertex positions, vertex normals, faces information **(40 pts)**

1. Ignore texture coordinate, material, group, shading information. In other words, ignore vt, mtllib, usemtl, o, s tags.

2. Meeting this requirement should allow the following sample OBJ files to be correctly loaded and rendered: *cube-tri.obj, sphere-tri.obj, and cylinder-tri.obj.*

iii. **Print out the following information** of the obj file to stdout (terminal) each time an obj file is loaded **(10 pts)**

1. Obj file name
2. Total number of faces
3. Number of faces with 3 vertices
4. Number of faces with 4 vertices
5. Number of faces with more than 4 vertices

iv. **Support for Opening Multiple OBJ Files:** If an OBJ file **B** is drag-and-dropped into the viewer while another OBJ file **A** is already being rendered, the mesh from OBJ file **B** should be added to the 3D space at a fixed offset from the mesh of **A (10 pts)**.

1. Specifically, each newly added mesh should be loaded at a position that is offset

by +2 units along the X-axis relative to the previously loaded mesh (e.g., x position for each mesh: 0 → 2 → 4 → ...).

v. **Support for Meshes with Polygons of Varying Vertex Counts:** Support loading and rendering meshes where polygons do not all have the same number of vertices, using either glDrawArrays() or glDrawElements(). **(10 pts)**

1. For example, the mesh may contain a mix of triangles, quads, and polygons with more than four vertices.
2. To render such meshes using vertex arrays, you may need to convert quads or n-gons into triangles. This requires implementing a basic **triangulation** algorithm.
3. Meeting this requirement should allow the following sample OBJ files to be correctly loaded and rendered: *cube-tri-quad.obj, sphere-tri-quad.obj, and cylinder-tri-quad-n.obj.*

C. Lighting & Etc (20 pts)

i. Render all object using Phong Illumination and Phong shading. **(20 pts).**

1. Choose lighting parameters (light colors, light position, material colors, material shininess, ...) as you want.

3. Report (10 pts)

A. Submit a report of **at most 2 pages** in a **pdf** file. Using MS Word is recommended. Do not exceed the limit.

B. The report should include:

i. Which requirements you implemented (5 pts)

ii. A few screenshot images of your program rendering loaded meshes (5 pts). **Do not use the provided sample obj files for this requirement.**

C. You do not need to try to write a long report. Just write down the required information. Use either English or Korean.

4. Runtime Environment

A. **Your program should be able to run on Python 3.8 with only NumPy, PyOpenGL, glfw, PyGLM installed. Do not use any other additional python modules.**

B. Only **glfw** is allowed for event processing and window & OpenGL context management. **Do not use glut functions for this purpose.**

C. **If your program does not meet this requirement, it will not run on TA's computer so you will not get any score for this project (except report).**

5. **What you have to submit: A zip file including**

A. **.py files -** *Your program may consist of several python source files. But the main module should be in* ***main.py****.*

B. **.pdf report file**

6. **Description of the sample obj files**

A. *cube-tri.obj:* A cube with triangles only

B. *cube-tri-quad.obj:* A cube with triangles and quads

C. *sphere-tri.obj:* A sphere with triangles only

D. sphere-tri-quad.obj: A sphere with triangles and quads

E. *cylinder-tri.obj:* A cylinder with triangles only

F. *cylinder-tri-quad-n.obj:* A cylinder with triangles, quads and polygons with more vertices

7. Additional information

A. *drop_callback* in glfw python binding is slightly different from that of original glfw written in C. *drop_callback* in python takes only two parameters, *window* and *paths*. *paths* is a list of dropped file paths.

B. obj file format reference: https://en.wikipedia.org/wiki/Wavefront_.obj_file

C. Python provides powerful string methods helpful for parsing an obj file. Among them, split() will be most useful.