Computer Graphics Assignment 2: Obj Viewer

Handed out: May 11, 2020

Due: 23:59, May 24, 2020 (NO SCORE for late submissions!)

- *Only accept answers submitted via git push to this course project for you at https://hconnect.hanyang.ac.kr (<Year>_<Course no.>_<Class code>/<Year>_<Course no.>_<Student ID>.git).*
- *Place your files under the directory structure **<Assignment name>/<your files>** just like the following example.*

```
+ 2020_ITE0000_2019000001
  + ClassAssignment1/
    - main.py
    - report.docx
```

- *The submission time is determined not when the commit is made but when the git push is made.*

1. Implement your own obj file viewer. The loaded mesh should be rendered with multiple light sources.
   A. You have to implement all requirements in a single program. This assignment DOES NOT require each requirement to be a separate program
   B. The window size doesn't need to be (480, 480). Use the larger window that is enough to see the details of the viewer.

2. Requirements

A. Manipulate the camera in the same way as in ClassAssignment1 using your ClassAssignment1 code (10 pts).

i. Also draw the reference grid plane.

B. Load an obj file and render it (70 pts)

i. Open an obj file by drag-and-drop to your obj viewer window **(10 pts)**

   1. Google *glfwSetDropCallback* to see how to do it
   2. The viewer should render only one obj file at a time. If an obj file B is drag-and-dropped to the viewer while it is rendering another obj file A, the viewer should only render the new obj file B.
   3. **This feature is essential for scoring your assignment, so if not implemented, you won't get any score for "Load an obj file and render it (70 pts)"**

ii. Read the obj file and display the mesh only using 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. Use glDrawArrays() or glDrawElements() to render triangle meshes.
      A. **DO NOT use glVertex*() & glNormal*()**. If you draw meshes using glVertex*() & glNormal*(), you'll get only 10 pts out of 40 pts.

iii. Toggle wireframe / solid mode by pressing **Z key** (similar to pressing Z key in Blender) **(10 pts)**

iv. When open an obj file, print out the following information of the obj file to stdout (console) **(10 pts)**

   1. 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

C. Lighting (10 pts)

i. Use multiple light sources (not a single light) to better visualize the mesh **(10 pts)**

ii. Choose the number of light sources, light source types, light colors, material colors as you want.

3. Report (10 pts)

A. Submit a report of **at most 2 pages** in docx file format (MS Word). 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 with downloaded obj files (5pts)

1. Download cool obj files from the Internet and open them with your viewer. You may download obj files from

A. https://free3d.com/

B. https://www.cgtrader.com/free-3d-models

2. Choose some of the best looking screenshot images of your viewer.

3. Do not use the sample obj files for this requirement. You must use other obj files downloaded from internet to get score for this requirement.

iii. Lighting configuration:

1. How many light sources?

2. Where do you put the light sources?

3. What is the type of each light source (point light or directional light)?

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

4. Extra credits

A. Toggle [shading using normal data in obj file] / [forced smooth shading] by pressing **S key (+10 pts)**

i. In [forced smooth shading] mode, do not use vertex normal in obj. Instead, you have to compute the averaged vertex normal for each vertex as described in the lecture slide and use them for shading.

B. Load & render a mesh that does not have the same number of vertices of all polygons

using glDrawArrays() or glDrawElements() **(+10 pts)**

i. For example, some polygons in the mesh are triangles and the rest are quads or polygons with more vertices

ii. To render this kind of mesh using a vertex array, you might need to render a quad or a n-polygon as a set of triangles. So you may need some kind of "triangulation" algorithm.

5. Runtime Environment

A. Your program should be able to run on systems only with Python 3.7 or later, NumPy, PyOpenGL, glfw. 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 assignment (except report).

6. Test your viewer with 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

G. Basically, your viewer should be able to render cube-tri.obj, sphere-tri.obj, cylinder-tri.obj properly.

H. To meet extra credit requirement B, your viewer should be able to render all above sample obj files properly.

7. What you have to submit:

A. **.py files**

   i. You can use multiple .py files for this assignment. In this case, explain how to run the program in the report.

B. **.docx report file**

8. 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.