Computer Graphics Assignment 2: Obj Viewer

Handed out: April 29, 2019

Due date: 23:59, May 20, 2019 (NO SCORE for late submissions!)

Submit your assignment only through hconnect.hanyang.ac.kr.

1. Implement your own obj file viewer for triangle meshes. 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. You can use either a set of glVertex*() & glNormal*() calls or a vertex array (see Extra Credits section) to render the mesh.

3. You get all 40 points for this requirement if your program successfully parse and render triangle meshes (whose all polygons are triangles).

   A. For meshes including polygons with more than 3 vertices, just use first three vertices of such a polygon to draw a triangle covering only a small part of that polygon.

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
6. (Even if your program successfully renders only faces with 3 vertices, you must also print out the number of faces with more than 3 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 3 pages in docx file format (MS Word). Do not exceed the limit.

B. The report should include:

        i. How to run your program
        ii. Which requirements you implemented
        iii. A few screenshot images of your program
            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.
        iv. 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)?

4. **Extra credits**
    A. Use glDrawArrays() or glDrawElements() to render a triangle mesh **(+10 pts)**
    B. Toggle [shading using normal data in obj file] / [forced smooth shading] by pressing **S key (+20 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.

5. **Runtime Environment**
    A. Your program should be able to run on systems only with Python 3.5 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. sphere-tri.obj: A sphere with triangles only
    C. cylinder-tri.obj: A cylinder with triangles only
    D. Basically, your viewer should be able to render cube-tri.obj, sphere-tri.obj, cylinder-tri.obj properly

7. **What you have to submit:**
    A. **A zip file ([studentID]-class2.zip, e.g., 2017123456-class2.zip)** including
        i. **.py files**
            1. You can use multiple .py files for this assignment. In this case, explain how to run the program in the report.
        ii. **.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.