Computer Graphics Assignment 1: Basic OpenGL viewer & drawing a hierarchical model

Handed out: April 20, 2020

Due: 23:59, May 10, 2020 (NO SCORE for late submissions!)

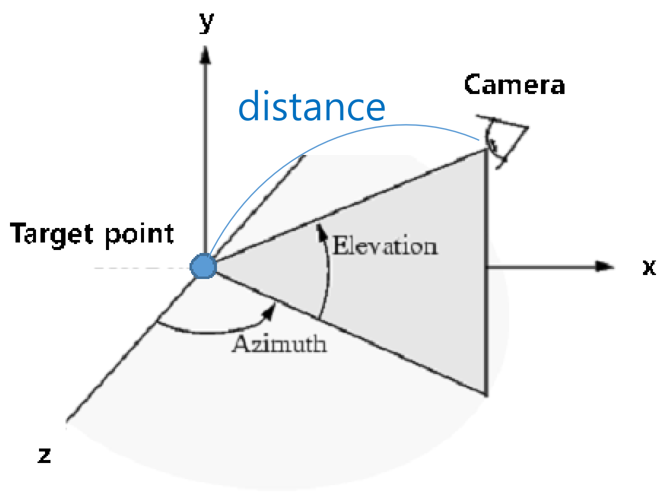
- *Only accept answers submitted via git push to this course project for you at https://hconnect.hanyang.ac.kr (<Year>_<Course no.>_<Class code>/<Year>_<Course no.>_<Student ID>.git).*
- *Place your files under the directory structure* ***<Assignment name>/<your files>*** *just like the following example.*

```
+ 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 a basic OpenGL viewer and show an animation of a hierarchical model using the viewer. This viewer will also be used in future class assignments.
   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 with mouse movement (50 pts)**
      i. Refer the camera manipulation of Blender software.
         1. https://www.blender.org/download/
      ii. The camera of your program should initially look at a target point, similar to that of Blender.
         1. Initialize the target point to the origin (0, 0, 0)



2.

iii. Provide the following three camera control operations.

1. **Orbit**: Rotate the camera around the target point by changing azimuth / elevation angles. (MMB (mouse middle button) in Blender) **(15 pts)**

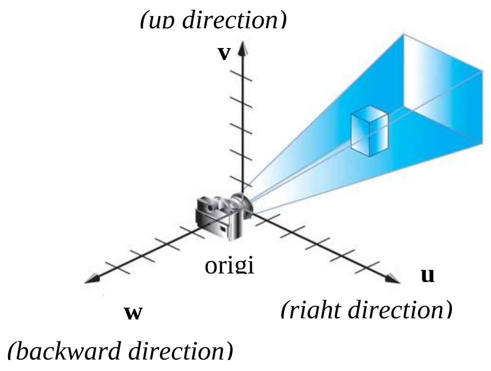
   A. Do not rotate the camera about a vector from the camera to the target point.

2. **Panning**: Move both the target point and camera in left, right, up and down direction of the camera (Shift-MMB in Blender) **(15 pts)**

   A. More specifically, translate both the target point and camera along u axis (left & right) and v axis (up & down) of the camera frame

3. **Zooming**: Move the camera forward toward the target point (zoom in) and backward away from the target point (zoom out) (Ctrl-MMB in Blender) **(15 pts)**

   A. A. More specifically, translate the camera along w axis of the camera frame

   B.

4. You MUST use the following mouse movement:

   A. Orbit: Click mouse left button & drag

B. **Panning: Click mouse right button & drag**

C. **Zooming: Rotate mouse wheel**

D. **Using above mouse movements is essential for scoring your assignment, so if you use any other set of mouse movement or keyboard shortcuts for Orbit / Panning / Zooming, you won't get any score for them.**

iv. Use perspective projection

v. Draw **a rectangular grid with lines (not polygons) on xz plane** as a reference ground plane (similar to Blender). Choose number of rows and columns, size as you want. **(5 pts)**

B. Create an animating hierarchical model using OpenGL matrix stacks (40 pts).

i. The model should consist of **3D primitives** such as boxes and spheres,

ii. You can use drawCube() and drawSphere() in the last page, or your own drawing functions (which should use only numpy, opengl, glfw).

iii. **DO NOT use glut or glu functions to draw 3D primitives** (e.g., glutSoildBox(), gluSphere(),…) because they generate runtime crashes on some systems (maybe problems of some python bindings, but don't use them anyway).

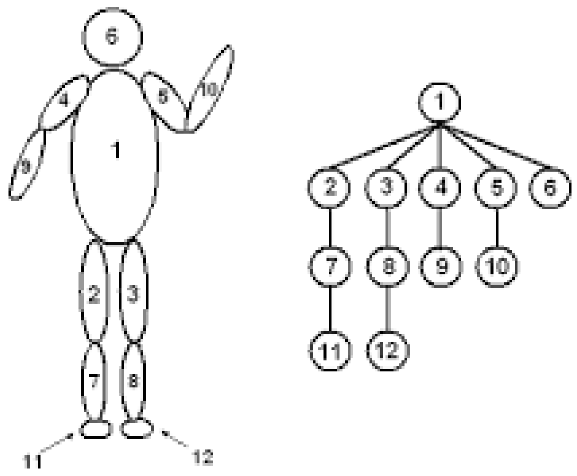
iv. Because we've not covered *shading* yet, just draw your model in **wireframe mode** by calling the following function at the beginning of your render function:

1. glPolygonMode(GL_FRONT_AND_BACK, GL_LINE) # call this at the beginning of your render function
2. You can change the color of your wireframe primitives using glColor*().

v. **You should use OpenGL matrix stack** to draw and animate your hierarchical model.

vi. The model should have a **hierarchy of at least 3 levels (20 pts)**.

1. For example, the following model has a hierarchy of 4 levels.

2. 

vii. **Animate the model** to show the hierarchical structure **(20 pts)**.

1. **Make all child body parts move relative to their parent body part.**
   A. In the above example, part 2, 3, 4, 5, 6 should move relative to part 1, part 7 should move relative to part 2, part 11 should move relative to part 7, … and so on.
   B. If any of the child bodies does not move relative to its parent, you will get -10 pts.
2. You can make any hierarchical system freely. Be creative.
   A. Eg) a hand with fingers bending
   B. Eg) a runner with arms and legs swing
3. The model should be **automatically animated without any mouse or keyboard inputs.**

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 (5 pts)

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 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).**

5. **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**

6. drawCube() and drawSphere() code:

```
# draw a cube of side 2, centered at the origin.
def drawCube():
    glBegin(GL_QUADS)
    glVertex3f( 1.0, 1.0,-1.0)
    glVertex3f(-1.0, 1.0,-1.0)
    glVertex3f(-1.0, 1.0, 1.0)
    glVertex3f( 1.0, 1.0, 1.0)

    glVertex3f( 1.0,-1.0, 1.0)
    glVertex3f(-1.0,-1.0, 1.0)
    glVertex3f(-1.0,-1.0,-1.0)
    glVertex3f( 1.0,-1.0,-1.0)

    glVertex3f( 1.0, 1.0, 1.0)
    glVertex3f(-1.0, 1.0, 1.0)
    glVertex3f(-1.0,-1.0, 1.0)
    glVertex3f( 1.0,-1.0, 1.0)

    glVertex3f( 1.0,-1.0,-1.0)
    glVertex3f(-1.0,-1.0,-1.0)
    glVertex3f(-1.0, 1.0,-1.0)
    glVertex3f( 1.0, 1.0,-1.0)
    glVertex3f(-1.0, 1.0, 1.0)
    glVertex3f(-1.0, 1.0,-1.0)
    glVertex3f(-1.0,-1.0,-1.0)
    glVertex3f(-1.0,-1.0, 1.0)

    glVertex3f( 1.0, 1.0,-1.0)
    glVertex3f( 1.0, 1.0, 1.0)
    glVertex3f( 1.0,-1.0, 1.0)
    glVertex3f( 1.0,-1.0,-1.0)
    glEnd()


# draw a sphere of radius 1, centered at the origin.
# numLats: number of latitude segments
# numLongs: number of longitude segments
def drawSphere(numLats=12, numLongs=12):
    for i in range(0, numLats + 1):
        lat0 = np.pi * (-0.5 + float(float(i - 1) /
float(numLats)))
        z0 = np.sin(lat0)
        zr0 = np.cos(lat0)

        lat1 = np.pi * (-0.5 + float(float(i) / float(numLats)))
        z1 = np.sin(lat1)
        zr1 = np.cos(lat1)

        # Use Quad strips to draw the sphere
        glBegin(GL_QUAD_STRIP)

        for j in range(0, numLongs + 1):
            lng = 2 * np.pi * float(float(j - 1) / float(numLongs))
            x = np.cos(lng)
            y = np.sin(lng)
            glVertex3f(x * zr0, y * zr0, z0)
            glVertex3f(x * zr1, y * zr1, z1)

        glEnd()
```