Computer Graphics, Lab Assignment 6

Handed out: April 22, 2020

Due: 23:59, April 22, 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>/<Problem no.>/<your file> just like the following example.*

```
+ 2020_ITE0000_2019000001
  + LabAssignment2/
    + 1/
      - 1.py
    + 2/
      - 2.py
    + 3/
      - 3.py
```

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

1. Write your own myLookAt() and myOrtho() functions (of the following form) that behaves exactly same as gluLookAt() and glOrtho().

   A.
   ```
   def myLookAt(eye, at, up): # eye, at, up are 1D numpy array of length 3
   def myOrtho(left, right, bottom, top, zNear, zFar):
   ```

   B. Set the window title to **your student ID** and the window size to (480,480).

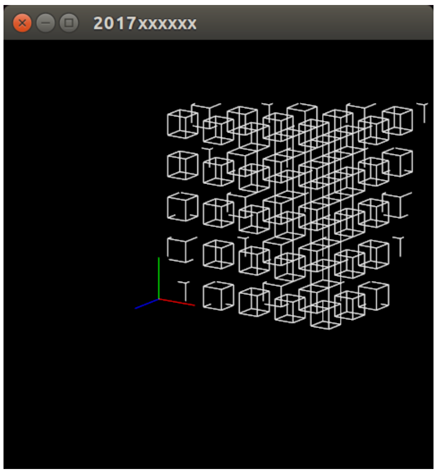
   C. Code skeleton

```
def render():
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
    glEnable(GL_DEPTH_TEST)
    glPolygonMode( GL_FRONT_AND_BACK, GL_LINE )
    glLoadIdentity()

    myOrtho(-5,5, -5,5, -8,8)
    myLookAt(np.array([5,3,5]), np.array([1,1,-1]), np.array([0,1,0]))

    # Above two lines must behaves exactly same as the below two lines

    #glOrtho(-5,5, -5,5, -8,8)
    #gluLookAt(5,3,5, 1,1,-1, 0,1,0)

    drawFrame()

    glColor3ub(255, 255, 255)
    drawCubeArray()

def myOrtho(left, right, bottom, top, near, far):
    # implement here

def myLookAt(eye, at, up):
    # implement here
```

D. Find code for drawFrame(), drawCubeArray() from *6-Viewing,Projection* slides.

E. DO NOT use gluLookAt() inside myLookAt() and glOrtho() inside myOrtho()!

F. Your program should render the following scene:

i. 

G. Hint:

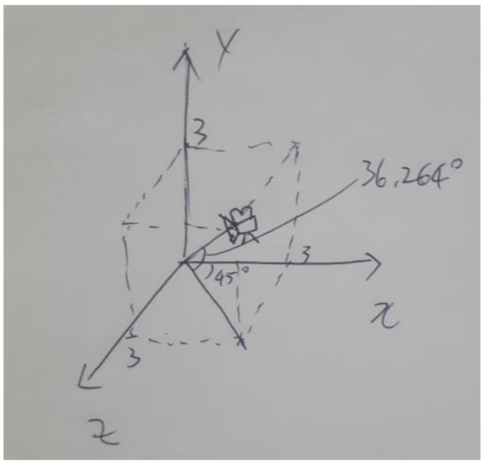
1. Everything you need to write code is in *6-Viewing,Projection* slides.
2. l2 norm of **v** : ||**v**|| = np.sqrt(np.dot(**v**, **v**))
3. **a** x **b** (cross product) : np.cross(**a**, **b**)
4. **a** · **b** (inner product) : np.dot(**a**, **b**) or **a**@**b**

5. Use glMultMatrixf() to multiply your projection matrix and viewing matrix to the current transformation matrix.

H. Files to submit: A Python source file (Name the file whatever you want (in English). Extension should be .py)

2. As mentioned in the lecture, “moving camera” and “moving world” are two equivalent operations. Based on the following figure, replace the gluLookAt call() in the following code with **two glRotatef() calls and one glTranslatef() call** and complete the program.

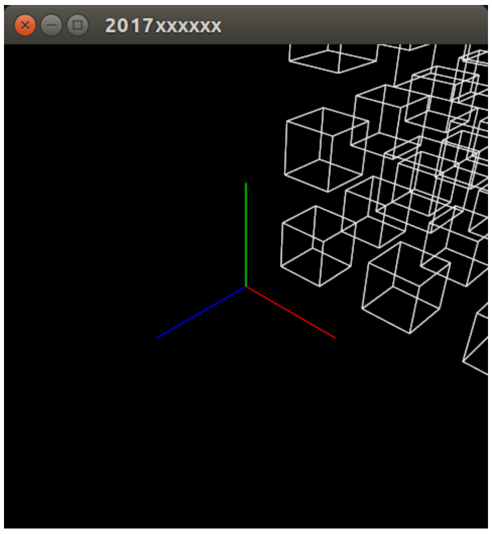
A.
```
def render():
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
    glEnable(GL_DEPTH_TEST)
    glPolygonMode( GL_FRONT_AND_BACK, GL_LINE )
    glLoadIdentity()

    gluPerspective(45, 1, 1,10)

    # Replace this call with two glRotatef() calls and one
glTranslatef() call
    gluLookAt(3,3,3, 0,0,0, 0,1,0)

    drawFrame()

    glColor3ub(255, 255, 255)
    drawCubeArray()
```

B. 

C. Set the window title to **your student ID** and the window size to (480,480).

D. Find code for drawFrame(), drawCubeArray() from 5-RenderingPipeline slides.

E. Your program should render the following scene:

i. 

F. Files to submit: A Python source file (Name the file whatever you want (in English). Extension should be .py)