**Computer Graphics, Lab Assignment 8**

Handed out: April 27, 2021

**Due: 23:59, April 28, 2021 (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.*

```
+ 2021_ITE0000_2019000001
  + LabAssignment8/
    + 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 down a Python program to draw a transformed triangle in a 3D space.
   A. Set the window title to **your student ID** and the window size to (480,480).
   B. Use the following code snippet:

```
gCamAng = 0
gComposedM = np.identity(4)

def render(M, camAng):
    # enable depth test
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
    glEnable(GL_DEPTH_TEST)

    glLoadIdentity()

    # use orthogonal projection
    glOrtho(-1,1, -1,1, -1,1)

    # rotate "camera" position to see this 3D space better
    gluLookAt(.1*np.sin(camAng),.1, .1*np.cos(camAng), 0,0,0, 0,1,0)

     # draw coordinate: x in red, y in green, z in blue
    glBegin(GL_LINES)
    glColor3ub(255, 0, 0)
    glVertex3fv(np.array([0.,0.,0.]))
    glVertex3fv(np.array([1.,0.,0.]))
    glColor3ub(0, 255, 0)
    glVertex3fv(np.array([0.,0.,0.]))
    glVertex3fv(np.array([0.,1.,0.]))
    glColor3ub(0, 0, 255)
    glVertex3fv(np.array([0.,0.,0]))
    glVertex3fv(np.array([0.,0.,1.]))
    glEnd()

    # draw triangle
    glBegin(GL_TRIANGLES)
    glColor3ub(255, 255, 255)
    glVertex3fv((M @ np.array([.0,.5,0.,1.]))[:-1])
    glVertex3fv((M @ np.array([.0,.0,0.,1.]))[:-1])
    glVertex3fv((M @ np.array([.5,.0,0.,1.]))[:-1])
    glEnd()

def key_callback(window, key, scancode, action, mods):
    global gCamAng, gComposedM
    if action==glfw.PRESS or action==glfw.REPEAT:

        if key==glfw.KEY_1:
            gCamAng += np.radians(-10)
        elif key==glfw.KEY_3:
            gCamAng += np.radians(10)
```

C. If you press or repeat a key, the triangle should be transformed as shown in the Table. Note that key 1 and 3 are already implemented in the above code snippet.

| Key | Transformation |
|---|---|
| **Q** | Translate by -0.1 in x direction **w.r.t global coordinate** |
| **E** | Translate by 0.1 in x direction **w.r.t global coordinate** |
| **A** | Rotate about y axis by -10 degrees **w.r.t local coordinate** |
| **D** | Rotate about y axis by +10 degrees **w.r.t local coordinate** |
| **W** | Rotate about x axis by -10 degrees **w.r.t local coordinate** |
| **S** | Rotate about x axis by +10 degrees **w.r.t local coordinate** |

| | |
|---|---|
| **1** | Rotate camera -10 degree |
| **3** | Rotate camera 10 degree |

D. Transformations should be accumulated (composed with previous one).

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

2. Write down a Python program to draw a hierarchical model of boxes.

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

B. Start from the following code skeleton.

```
import glfw
from OpenGL.GL import *
import numpy as np
from OpenGL.GLU import *

def render():
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
    glEnable(GL_DEPTH_TEST)

    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    glOrtho(-2,2, -2,2, -1,1)

    glMatrixMode(GL_MODELVIEW)
    glLoadIdentity()

    drawFrame()
    t = glfw.get_time()

    # blue base transformation
    glPushMatrix()
    glTranslatef(np.sin(t), 0, 0)

    # blue base drawing
    glPushMatrix()
    glScalef(.2, .2, .2)
    glColor3ub(0, 0, 255)
    drawBox()
    glPopMatrix()

    # red arm transformation
    glPushMatrix()
    glRotatef(t*(180/np.pi), 0, 0, 1)
    glTranslatef(.5, 0, .01)

    # red arm drawing
    glPushMatrix()
    glScalef(.5, .1, .1)
    glColor3ub(255, 0, 0)
    drawBox()
    glPopMatrix()

    glPopMatrix()
    glPopMatrix()
```

```
def drawBox():
    glBegin(GL_QUADS)
    glVertex3fv(np.array([1,1,0.]))
    glVertex3fv(np.array([-1,1,0.]))
    glVertex3fv(np.array([-1,-1,0.]))
    glVertex3fv(np.array([1,-1,0.]))
    glEnd()

def drawFrame():
    # draw coordinate: x in red, y in green, z in blue
    glBegin(GL_LINES)
    glColor3ub(255, 0, 0)
    glVertex3fv(np.array([0.,0.,0.]))
    glVertex3fv(np.array([1.,0.,0.]))
    glColor3ub(0, 255, 0)
    glVertex3fv(np.array([0.,0.,0.]))
    glVertex3fv(np.array([0.,1.,0.]))
    glColor3ub(0, 0, 255)
    glVertex3fv(np.array([0.,0.,0]))
    glVertex3fv(np.array([0.,0.,1.]))
    glEnd()

def main():
    if not glfw.init():
        return
    window = glfw.create_window(480,480,'2017123456-lab6-1', None,None)
    if not window:
        glfw.terminate()
        return
    glfw.make_context_current(window)
    glfw.swap_interval(1)

    while not glfw.window_should_close(window):
        glfw.poll_events()
        render()
        glfw.swap_buffers(window)

    glfw.terminate()

if __name__ == "__main__":
    main()
```

**C. Add a green arm at the end of the red arm, and rotate the green arm about its local z axis.**

i. Render the green arm using drawBox().

**D. Also render local frames of the blue base, red arm, green arm using drawFrame().**

E. Expected result: Uploaded LabAssignment8-2.mp4

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