Class Assignment 1 Report

2019011449 이수빈

[Mouse Movement]

```
# A. Orbit: Click mouse left button & drag
# B. Panning: Click mouse right button & drag
# C. Zooming: Rotate mouse wheel
def button_callback(window, button, action, mod):
    global gPrev_pos
    if button == glfw.MOUSE_BUTTON_LEFT and action == glfw.PRESS:
        gPrev_pos = glfw.get_cursor_pos(window)
    elif button == glfw.MOUSE_BUTTON_RIGHT and action == glfw.PRESS:
        gPrev_pos = glfw.get_cursor_pos(window)

def cursor_callback(window, xpos, ypos):
    global gPrev_pos, gCur_pos
    left_state = glfw.get_mouse_button(window, glfw.MOUSE_BUTTON_LEFT)
    right_state = glfw.get_mouse_button(window, glfw.MOUSE_BUTTON_RIGHT)
    if left_state == True:
        gCur_pos = [xpos, ypos]
        orbit(gPrev_pos, gCur_pos)
        gPrev_pos = gCur_pos
    elif right_state == True:
        gCur_pos = [xpos, ypos]
        panning(gPrev_pos, gCur_pos)
        gPrev_pos = gCur_pos

def scroll_callback(window, xoffset, yoffset):
    zooming(xoffset, yoffset)
```

i) click이 이루어졌을 시 gPrev_pos에 cursor의 위치를 저장한다.

ii) click 상태로 cursor가 움직이면 (drag되면) gCur_pos에 cursor의 위치를 저장하고 각 함수들 (left button이면 orbit, right button이면 panning)에게 인자로 넘겨주며 호출한다.

iii) 함수 호출이 끝나면 gPrev_pos를 gCur_pos로 업데이트한다.

[Toggle]

```
def key_callback(window, key, scancode, action, mods):
    global gToggle
    if key == glfw.KEY_V and action == glfw.PRESS:
        print('v pressed')
        gToggle = not(gToggle)
```

```
# Orthographic
if gToggle == False:
    glOrtho(-10, 10, -10, 10, 1, 100)
# Perspective
elif gToggle == True:
    glFrustum(-1, 1, -1, 1, 1, 100)
```

gToggle이 True면 Perspective projection, False면 Orthogonal projection이 이루어진다. v키가 눌리면 gToggle이 바뀐다.

[Camera Control Operation]

```
def myLookAt(eye, at, up):
    global gAzimuth, gElevation, gDistance, gPanning

    distance = 5
    eye[0] = at[0] + distance * np.cos(gAzimuth)
    eye[1] = at[1] + distance * np.sin(gElevation)
    eye[2] = at[2] + distance * np.sin(gAzimuth)

    w = (eye - at) / np.sqrt(np.dot(eye - at, eye - at))
    u = np.cross(up, w) / np.sqrt(np.dot(np.cross(up, w), np.cross(up, w)))
    v = np.cross(w, u)

    Mv = np.array([[u[0], u[1], u[2], -np.dot(u, eye) + gPanning[0]],
                   [v[0], v[1], v[2], -np.dot(v, eye) + gPanning[1]],
                   [w[0], w[1], w[2], (-np.dot(w, eye)) * gDistance],
                   [0, 0, 0, 1]])

    glMultMatrixf(Mv.T)

def orbit(prev_pos, cur_pos):
    global gAzimuth, gElevation
    # print('orbit')
    gAzimuth += .1 * np.radians(cur_pos[0] - prev_pos[0])
    gElevation += .1 * np.radians(cur_pos[1] - prev_pos[1])

def panning(prev_pos, cur_pos):
    global gPanning
    # print('panning')
    gPanning[0] += .01 * (cur_pos[0] - prev_pos[0])
    gPanning[1] += .01 * -(cur_pos[1] - prev_pos[1])

def zooming(xoffset, yoffset):
    global gDistance
    # print('zooming')
    gDistance += .01 * yoffset
```

-orbit: gAzimuth와 gElevation을 조정해 camera의 방향을 조정한다.

-panning: 마우스 커서의 x축 방향 드래그는 vector u의 방향으로 카메라를 조정하게 하고, y축 방향 드래그는 vector v의 방향으로 카메라를 조정하게 한다.

-zooming: 마우스 스크롤이 입력되면 그 offset만큼 카메라와 물체 사이의 거리를 늘이거나 줄인다.