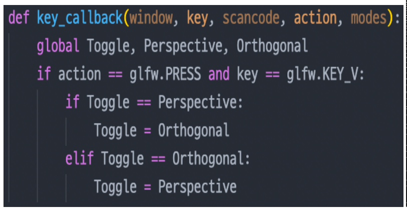
ClassAssignment1 Report

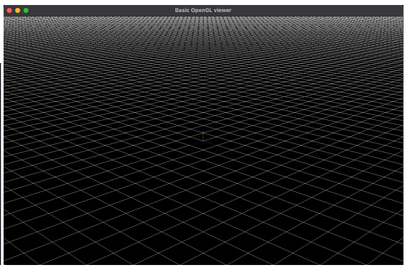
2016025041 하태성

- **Toogle Projection**
  by pressing ‘v’ key

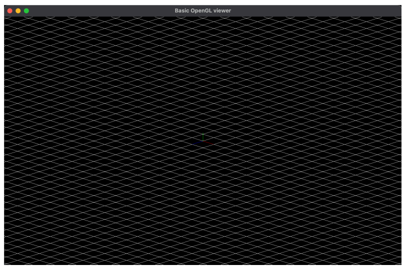


```
def key_callback(window, key, scancode, action, modes):
    global Toggle, Perspective, Orthogonal
    if action == glfw.PRESS and key == glfw.KEY_V:
        if Toggle == Perspective:
            Toggle = Orthogonal
        elif Toggle == Orthogonal:
            Toggle = Perspective
```

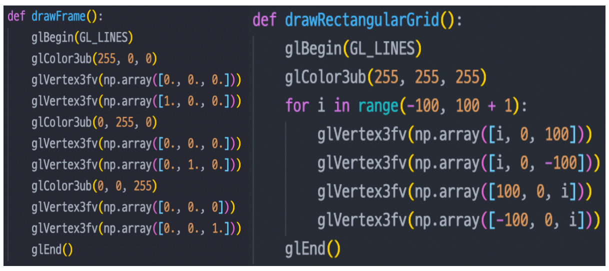
Perspective Projection



Orthogonal Projection



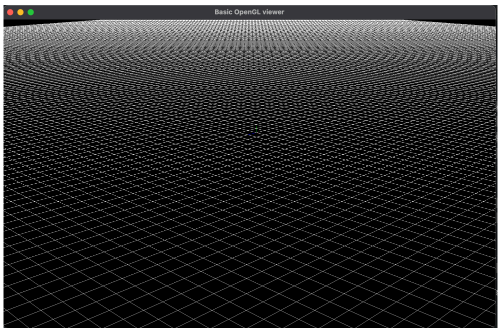
- **Draw Rectangular Grid** on xz plane



```
def drawFrame():
    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()
```

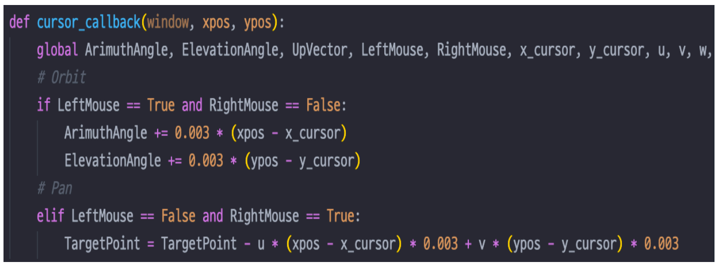
```
def drawRectangularGrid():
    glBegin(GL_LINES)
    glColor3ub(255, 255, 255)
    for i in range(-100, 100 + 1):
        glVertex3fv(np.array([i, 0, 100]))
        glVertex3fv(np.array([i, 0, -100]))
        glVertex3fv(np.array([100, 0, i]))
        glVertex3fv(np.array([-100, 0, i]))
    glEnd()
```

Rectangular Grid



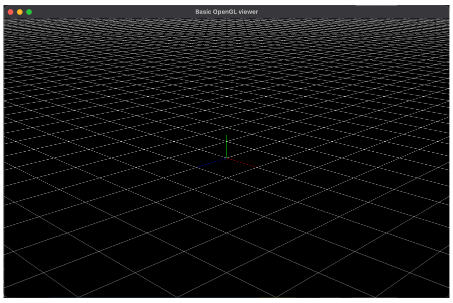
- **Orbit** : Rotate the camera around the target point by changing azimuth / elevation angles(Click mouse left button & drag)

- **Pan** : Move both the target point and camera in left, right, up and down direction of the camera(Click mouse right button & drag)

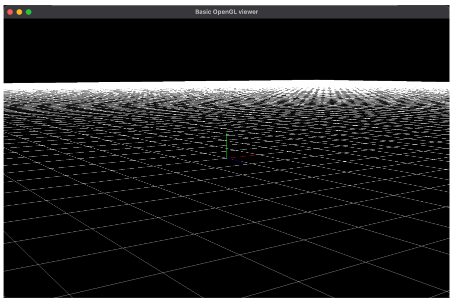


```
def cursor_callback(window, xpos, ypos):
    global ArimuthAngle, ElevationAngle, UpVector, LeftMouse, RightMouse, x_cursor, y_cursor, u, v, w,
    # Orbit
    if LeftMouse == True and RightMouse == False:
        ArimuthAngle += 0.003 * (xpos - x_cursor)
        ElevationAngle += 0.003 * (ypos - y_cursor)
    # Pan
    elif LeftMouse == False and RightMouse == True:
        TargetPoint = TargetPoint - u * (xpos - x_cursor) * 0.003 + v * (ypos - y_cursor) * 0.003
```

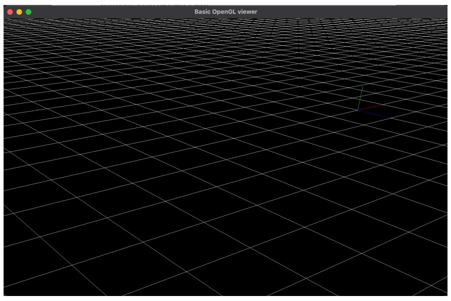
Original



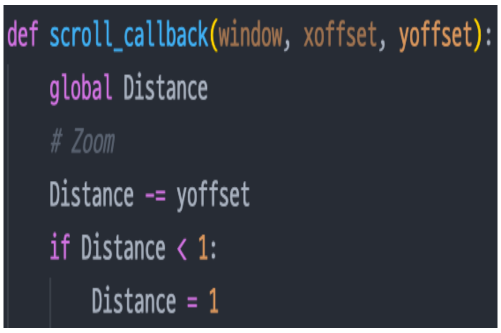
Orbit



Pan

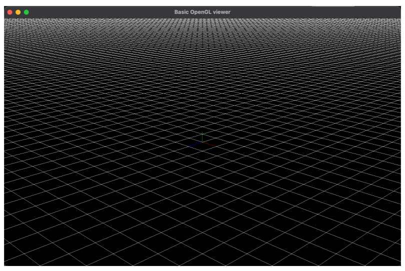


- **Zoom** : Move the camera forward toward the target point (zoom in) and backward away from the target point(Rotate mouse wheel)

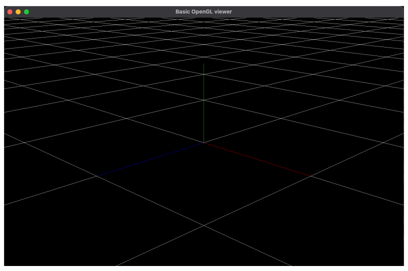


```
def scroll_callback(window, xoffset, yoffset):
    global Distance
    # Zoom
    Distance -= yoffset
    if Distance < 1:
        Distance = 1
```

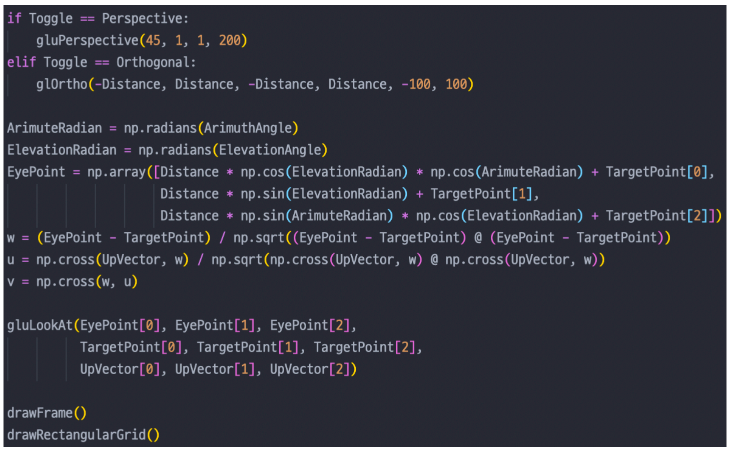
Original



Zoom



- Render Function : Choose Projection and Calculate Camera's coordinate(Eyepoint) and vectors(u, v, w). Then Set Camera, Draw Frame and Grid.



```
if Toggle == Perspective:
    gluPerspective(45, 1, 1, 200)
elif Toggle == Orthogonal:
    glOrtho(-Distance, Distance, -Distance, Distance, -100, 100)

ArimuteRadian = np.radians(ArimuthAngle)
ElevationRadian = np.radians(ElevationAngle)
EyePoint = np.array([Distance * np.cos(ElevationRadian) * np.cos(ArimuteRadian) + TargetPoint[0],
                     Distance * np.sin(ElevationRadian) + TargetPoint[1],
                     Distance * np.sin(ArimuteRadian) * np.cos(ElevationRadian) + TargetPoint[2]])
w = (EyePoint - TargetPoint) / np.sqrt((EyePoint - TargetPoint) @ (EyePoint - TargetPoint))
u = np.cross(UpVector, w) / np.sqrt(np.cross(UpVector, w) @ np.cross(UpVector, w))
v = np.cross(w, u)

gluLookAt(EyePoint[0], EyePoint[1], EyePoint[2],
          TargetPoint[0], TargetPoint[1], TargetPoint[2],
          UpVector[0], UpVector[1], UpVector[2])

drawFrame()
drawRectangularGrid()
```