render함수

```python
def render():
    global gCamAng, gCamHeight, panningPosX, panningPosY, currentZoomLevel, isOrthoEnabled
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
    glEnable(GL_DEPTH_TEST)
    # glPolygonMode( GL_FRONT_AND_BACK, GL_LINE )
    glLoadIdentity()
    gluPerspective(45, 1, 1,30)

    if isOrthoEnabled:
        glLoadIdentity()
        glOrtho(-3,3, -3,3, -30,30)

    glTranslatef(0,0, currentZoomLevel)
    glTranslatef(panningPosX, panningPosY, 0)

    glRotatef(gCamHeight, 1, 0, 0)
    glRotatef(36.264, 1, 0, 0)
    glRotatef(gCamAng, 0, 1, 0)
    glRotatef(-45, 0, 1, 0)

    drawRectGrid()
    drawFrame()
    drawUnitCube()
```

유닛 큐브를 먼저 그리고, 프레임 그리고, xz 평면 그리드 그린 다음에

카메라상 y축을 기준으로 right handed 관점에서 -45도를 돌리면 사각형의 모서리 부분이 보입니다. 그 상태에서 gCamAng만큼 더 돌려주고(Orbit) 36.264만큼을 카메라상 x축을 기준으로 36.264 만큼 돌려주고 그 다음 gCamHeight만큼 더 돌려줍니다 (Orbit)

panningPosX만큼, panningPosY만큼 앞의 물체를 위, 아래, 왼쪽, 오른쪽으로 옮겨줍니다

currentZoomLevel만큼 앞의 물체를 바라보는 방향 앞, 뒤로 옮겨줍니다.

기본적으로 perspective projection로 물체를 바라보는데 v키를 활용해 isOrthoEnabled의 값을 바꿔주면 orthogonal projection으로 물체를 바라보게 됩니다.

변수들은 인자로 넘겨주기 보다는 전역변수를 활용하여 구현했습니다.

drawUnitCube와 drawFrame은 수업에서 사용하던 코드를 가져와서 활용했습니다. UnitCube의 경우 면들의 차이를 눈으로 보다 쉽게 확인하기 위해 색상 값을 바꿔주었고, drawFrame의 경우는 길이를 조정하였습니다.

drawRectGrid함수

```python
def drawRectGrid():
    glBegin(GL_LINES)
    glColor3ub(255,255,255)

    rows = 30
    columns = 30

    for i in range(rows):
        glVertex3f(i - rows / 2, 0, -columns)
        glVertex3f(i - rows / 2, 0, columns)

    for i in range(columns):
        glVertex3f(-rows, 0, i - columns / 2)
        glVertex3f(rows, 0, i - columns / 2)

    glEnd()
```

x -15부터 15까지 그려주고, z -15부터 15까지 그려주었습니다.

key_callback함수

```python
def key_callback(window, key, scancode, action, mods):
    global isOrthoEnabled
    if key==glfw.KEY_V and action==glfw.PRESS:
        isOrthoEnabled = not isOrthoEnabled
```

v 키를 활용해서 isOrthoEnabled의 값을 바꿔줍니다.

Button_callback함수

```python
def button_callback(window, button, action, mod):
    global isLeftMousePressed, isRightMousePressed
    global initialCamPosX, initialCamPosY, lastCamPosX, lastCamPosY
    global lastPanningPosX, lastPanningPosY

    if button==glfw.MOUSE_BUTTON_LEFT:
        if action==glfw.PRESS:
            isLeftMousePressed = True
            initialCamPosX, initialCamPosY = glfw.get_cursor_pos(window)

        elif action==glfw.RELEASE:
            isLeftMousePressed = False
            lastCamPosX = gCamAng
            lastCamPosY = gCamHeight

    elif button == glfw.MOUSE_BUTTON_RIGHT:
        if action==glfw.PRESS:
            isRightMousePressed = True
            initialCamPosX, initialCamPosY = glfw.get_cursor_pos(window)
        elif action==glfw.RELEASE:
            isRightMousePressed = False
            lastPanningPosX = panningPosX
            lastPanningPosY = panningPosY
```

마우스 왼쪽버튼 눌렀을 때 isLeftMousePressed 값을 True로 바꿔주고 처음으로 찍은 그 마우스 포인터 위치를 저장합니다. 그리고 마우스 버튼을 떼면 isLeftMousePressed 값을 False로 바꿔주고 마지막 캠 앵글을 저장해둡니다.

마우스 오른쪽버튼 눌렀을 때 isRightMousePressed 값을 True로 바꿔주고 처음 포인터 위치 저장하고 버튼을 떼면

마지막 캠 위치를 저장합니다.

Cursor_callback함수

```
def cursor_callback(window, xpos, ypos):
    global gCamAng, gCamHeight, lastCamPosX, lastCamPosY
    global panningPosX, panningPosY, lastPanningPosX, lastPanningPosY
    if isLeftMousePressed:
        gCamAng = lastCamPosX - (initialCamPosX - xpos) / 5
        gCamHeight = lastCamPosY - (initialCamPosY - ypos) / 5
        print('mouse cursor moving: (%d, %d)'%(gCamAng, gCamHeight))

    elif isRightMousePressed:
        panningPosX = lastPanningPosX - (initialCamPosX - xpos) / 100
        panningPosY = lastPanningPosY + (initialCamPosY - ypos) / 100
        print('mouse cursor moving: (%lf, %lf)'%(panningPosX, panningPosY))
```

왼쪽 마우스가 클릭 되었다면 Orbit을 달성하기 위해 마지막 카메라 포지션을 기준으로 처음 포인터에서 xpos, ypos를 각각 빼서 얼마나 떨어졌는지 확인합니다. 5로 나누는 이유는 카메라가 확확 돌아가는 걸 방지하기 위함입니다. (Orbit)

오른쪽 마우스가 클릭되었다면 마지막 패닝 포지션을 기준으로 처음 마우스 포인터에서 xpos, ypos를 빼서 얼마나 떨어졌는지 확인합니다. 100으로 나누는 이유 또한 카메라가 확확 돌아가는 걸 방지하기 위함입니다. (Panning)

scroll_callback함수

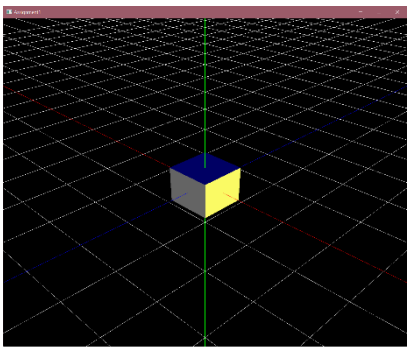
```
def scroll_callback(window, xoffset, yoffset):
    global currentZoomLevel
    currentZoomLevel += yoffset
    print(currentZoomLevel)
    print('mouse wheel scroll: %d, %d'%(xoffset, yoffset))
```

currentZoomlevel을 yoffset만큼 빼주고 더해줍니다. (Zooming)

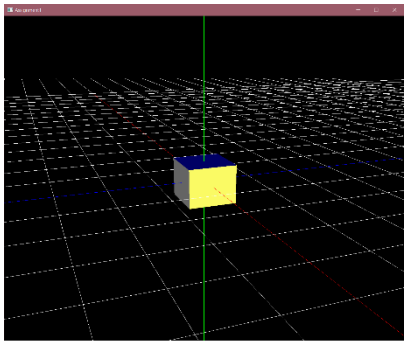
main함수

```
glfw.set_key_callback(window, key_callback)
glfw.set_mouse_button_callback(window, button_callback)
glfw.set_cursor_pos_callback(window, cursor_callback)
glfw.set_scroll_callback(window, scroll_callback)
```

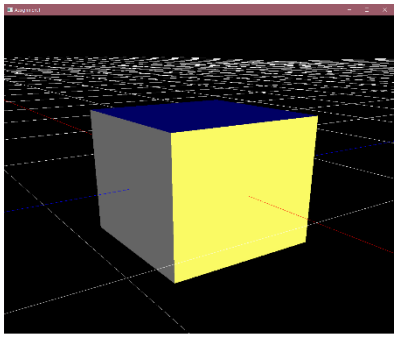
마우스 버튼, 커서, 스크롤, 키보드에 반응하도록 세팅합니다.



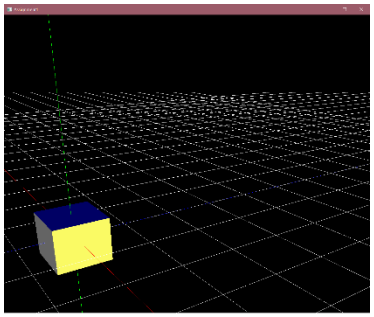
<첫 화면>



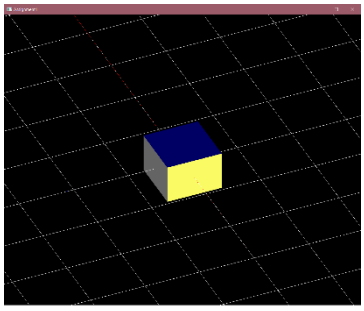
<Orbit>



<Zooming>



<Panning>



<Orthogonal projection>