**Computer Graphics ClassAssignment1**

2019048586 Afif Danial 아핍

1. **Implementations**

**i) myLookAt()**

-The render was performed by creating a myLookAt() function, calculating u,v,w vector, and multiplying it by the current transformation matrix.

-look at point is located at center.

-w vector was oriented with gElevation and gAzimuth, and the size and location were set to center and zoom.

- We used up vector and w vector to find u vector and v vector and then make a matrix.

- At first, the u,v,w vector was not calculated, so it was implemented in advance before the render so we can prevent errors.

ii) **orbit**

- For orbit, gElevation and gAzimuth are set as global variables, and the degree of movement of the cursor was always stored in the cursor\_callback, and left button of the mouse is **true**, and left button is **false** to reflect the motion value of the cursor only when left is **true**.

**iii) panning**

Panning moved the at point in reverse with u,v, which is the vector of the camera, as the axis. Like orbit, pressing right button makes the right become **true** so that the cursor's motion value is reflected in the center.

**iv) zooming**

The yoffset value is added to the zoom with Scroll\_callback to adjust the size but even if the ortho is zoomed, the distance will increase, and not the object, so the object will disappear when it is zoomed in and out of the volume.

**v) perspective/ ortho**

If v is pressed with Key\_callback, the value of isPerspective will change and glPerspective and glOrtho are implemented in order.

**vi) extras**

The cube is drawn without the vertex array, but the grid is drawn using the vertex array and index array.

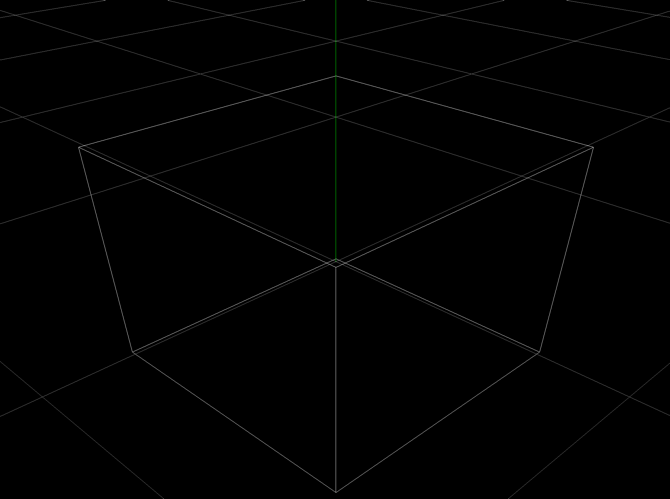
1. **Screenshots**

Initial result

어두운, 검은색이(가) 표시된 사진

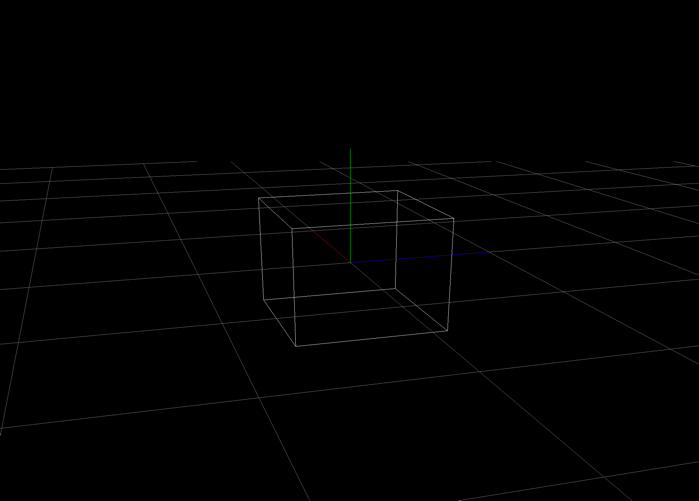
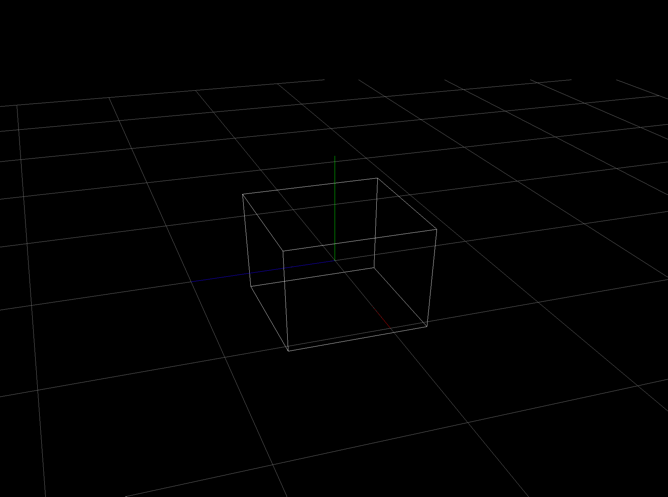
자동 생성된 설명

Zooming

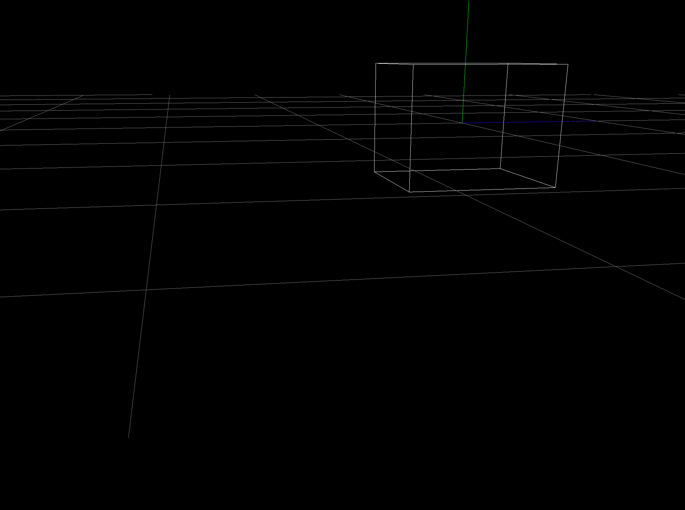
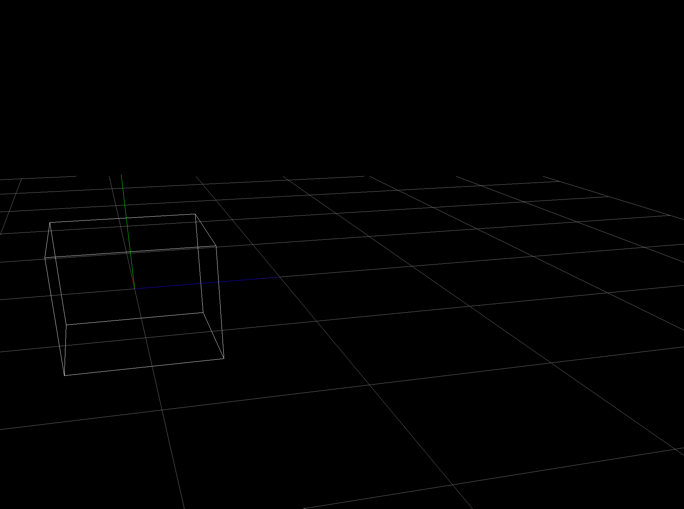
바닥, 어두운, 검은색이(가) 표시된 사진

자동 생성된 설명

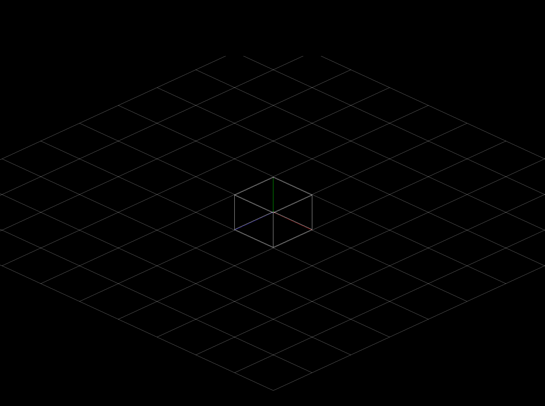
Orbit



Panning



Orthogonal



Perspective projection uniformly applied