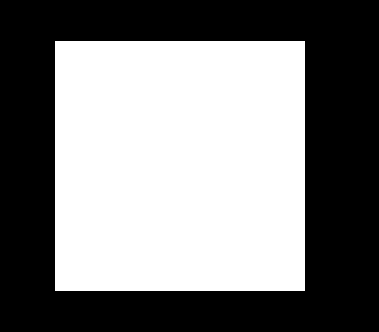
**CS4610 Assignment 2b Report**

By: Nathaniel Callahan

**CS4610 Assignment 2b Report**

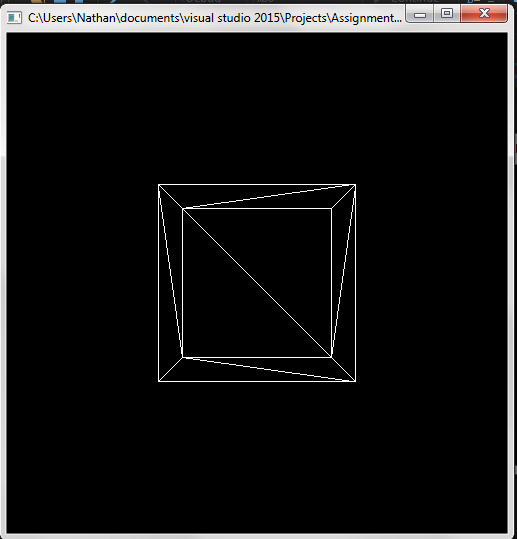
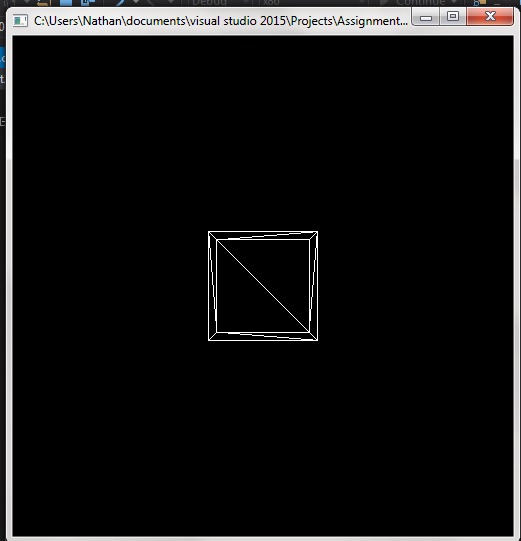
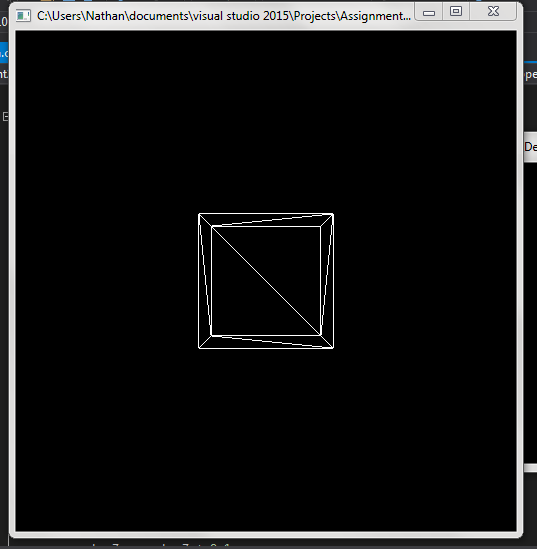
1. On top of Assignment 2a, define a virtual camera in a 3D virtual scene, specifying its position, orientation and field of view.

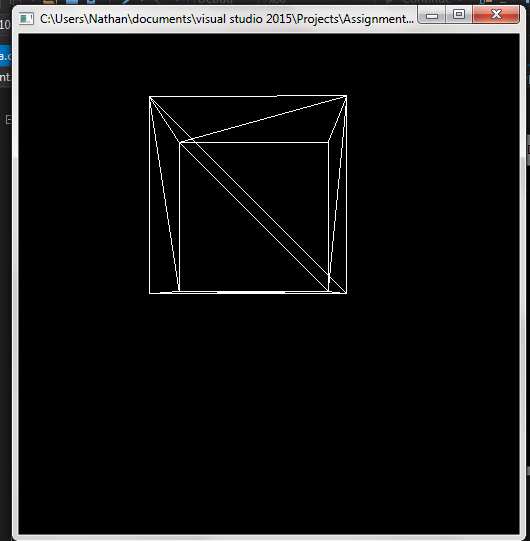
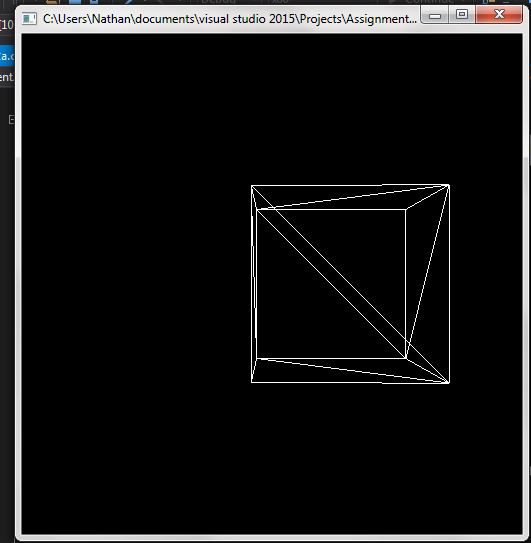
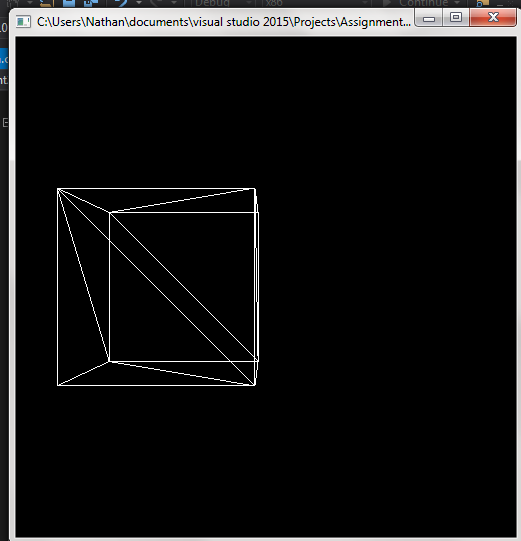
I commented out all of the glOrtho() stuff in the reshape method. I then added gluPerspective(45, w/h, 0.1,100) and gluLookAt(0,0,10,0,0,0,0,1,0) to define the virtual camera. The camera’s position, orientation and field of view are specified by the nine inputs in gluLookAt(). Some of the given values for the inputs for gluLookAt() put the camera too far away from the objects so I changed them so the objects can be seen better.

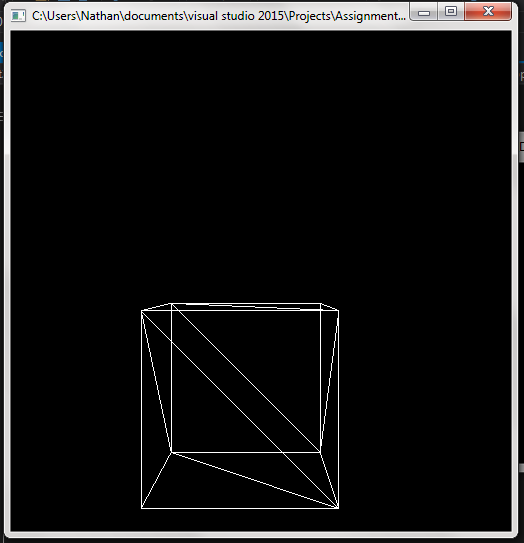


1. Using your graphical user interface (GUI) such as GLUT or equivalent, together with the mouse and keyboard, interactively perform the following tasks:
2. Translate the model / camera in X, Y and Z directions.

For this I used the key board. I calculated the center value for X, Y, and Z of the object. Then, depending on the dimension the user wants to move the object I subtract or add 0.01 from the center then call the display function so the object is redrawn using the new center value

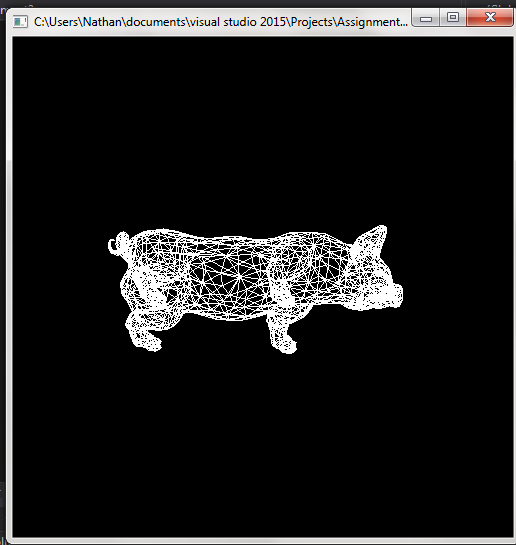
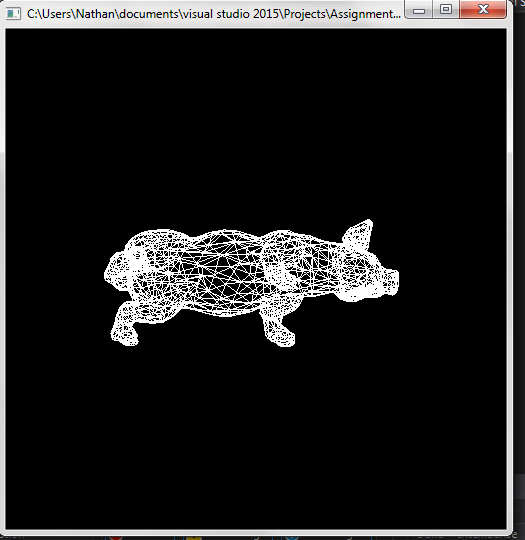
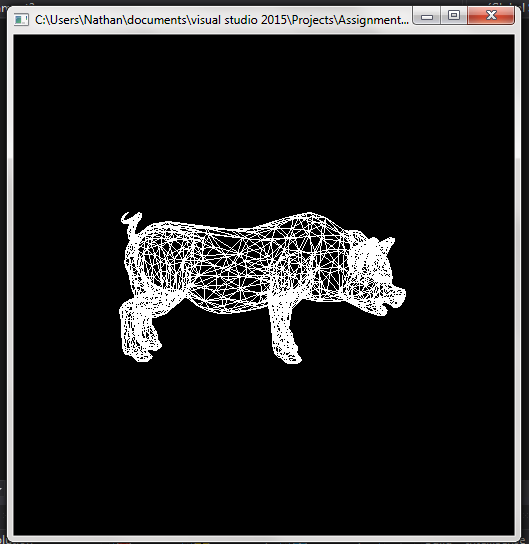


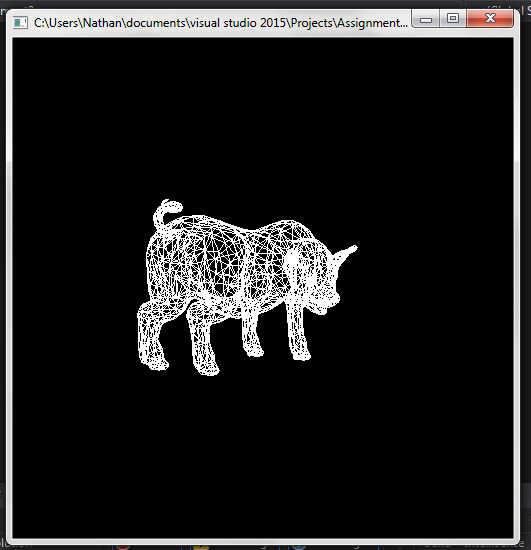
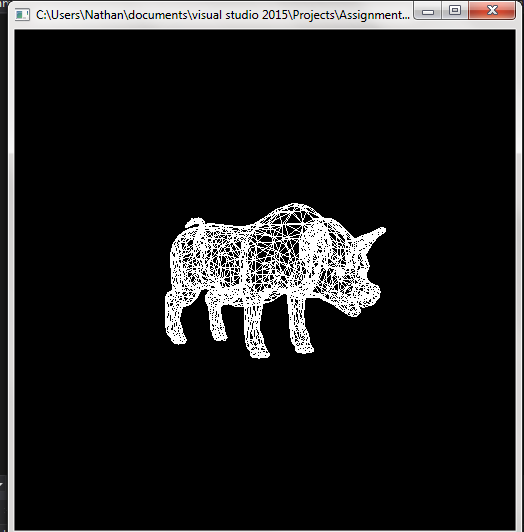




1. Rotate the model / camera around X, Y, and Z axes.

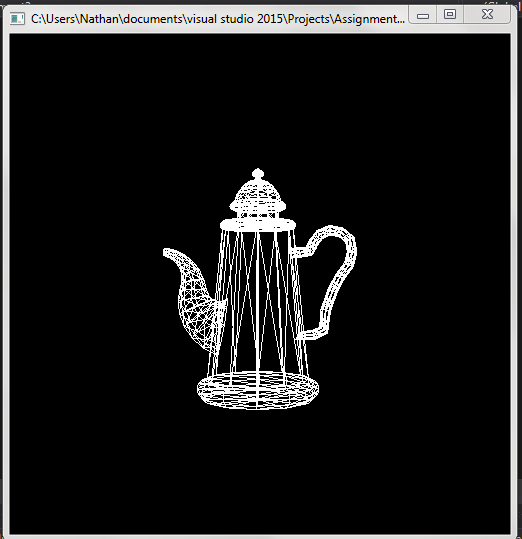
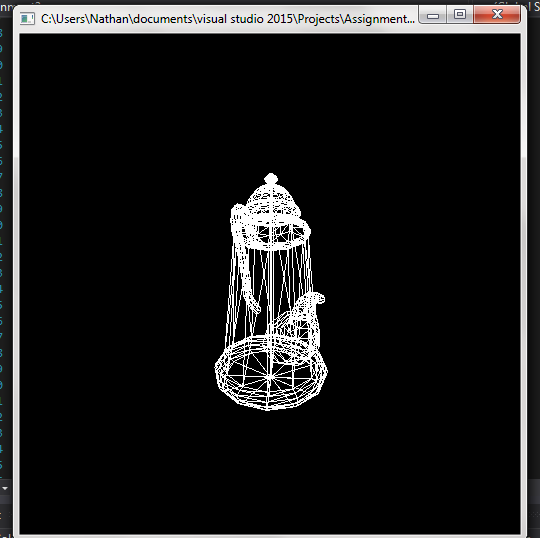
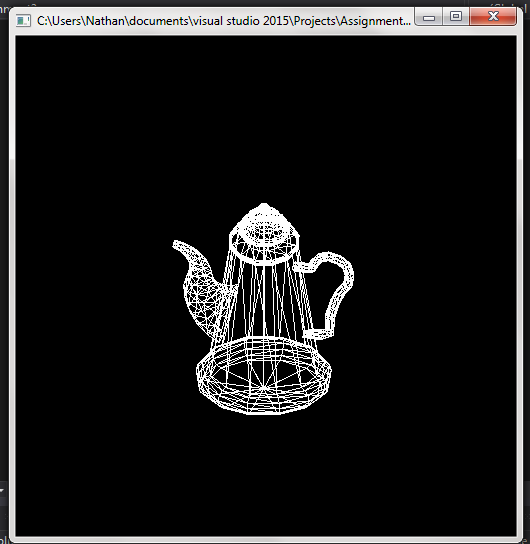
For this I used the key board. I calculated the rotation value for the vertical and horizontal value of the object. Then, depending on the rotation the user wants (vertical or horizontal) I subtract or add 5 from the current angle then call the display function so the object is redrawn, rotated the way the user intended.





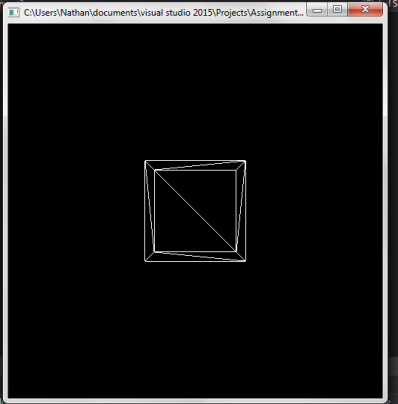
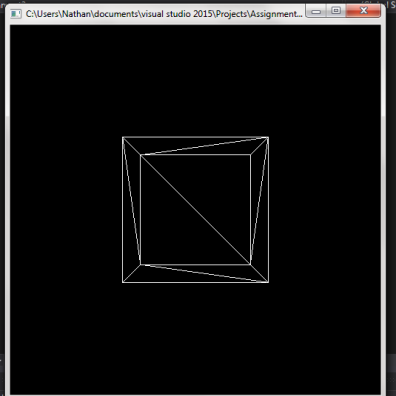
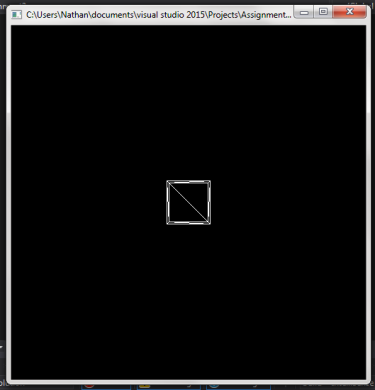
1. Rotate the model /camera according to the moving direction and distance of the mouse.

For this I used the mouse. If the left mouse button is clicked the current X and Y value is stored. When the mouse is dragged, the horizontal and vertical rotation is calculated and it rotates the object while the mouse is left clicked and dragged. When the mouse button is let go, the object stops rotating.

1. Zoom in and zoom out view of the model.

For this I used the key board. I calculated the scale of the object when I get it from the .obj file so that the user can see the entire object in the camera view. If the user wants to zoom out we subtract -0.01 from the scale to shrink the object. If the user wants to zoom in on the object we add 0.01 to the scale to make it bigger. The scale can’t go below zero so if it hits zero it doesn’t get any smaller or else the object will flip in the view.

All tasks can be perfomred on any any of the objecst (cube, pig, teapot) and on any display (points, lines, and surfaces) I chose to show one object pertask to save on ink and wasted space.

To change the scale the buttons to push are: z and x.

To translate the object the buttons to push are: i, o, j, k, n, m.

To rotate the object the buttons to push are: r, t, f, g.

(c and v was extra totating, not needed)

(all in a switch statement)

Clicking the left mouse button and holding it and draging on the window will also rotate the object in the way that the mouse is being draged until the button is let go.

As far as I can tell there are no remaining issues for part b of this assignment to be reported. Everything went pretty smoothly.