Computer Graphics HW2

邱名彰 100060007

Objective

This assignment aims to let us practice how to use transformation matrices, viewing matrices and projection matrices to manipulate models and then display them on the screen.

Initial Environment

The models are normalized to [-1,1] in all three axis. Eye position is at (0,0,2), looking at origin, up vector (0,1,0).

Usage

Mouse control:

First, hit 1, 2 or 3 to choose models you want to play.

Second, click left button and drag the model for translation.

Third, click right button and drag the model for rotation.

Fourth, scroll the wheel to scale up and down the model.

Keyboard control:

Hit h for help menu.

Hit 1, 2 or 3 to choose models you want to play.

And then hit ‘g’ or ‘v’ to select geographical transformation or viewing transformation.

In ‘g’, press ‘r’ for rotation, ‘s’ for scaling, ‘t’ for translation and finally hit x/X, y/Y, z/Z to decrease/increase the index value.

In ‘v’, press ‘e’ to change eye position, ‘c’ to change center coordinate, ‘u’ to change up vector. Control of index value is the same as the case of ‘g’.

In the meantime, one can arbitrarily switch between perspective projection and orthogonal projection by pressing ‘p’ and ‘o’. The default is orthogonal projection.

Difficulties

1. Loading multiple objects: I create multiple arrays to store vertices and colors of the other two models. After calculating centers of them and translating them back to the origin, I send the coordinates into drawing model-by-model in “renderScene()”. Notice that we are required to use the same iLoc, iPosition, iMVP to bind models into the same volume.
2. Far, near plane: I had difficult time making the model displayed on the screen since I misunderstood the meaning of far plane, near plane. They are relative distance, so far – near shall always be positive.

Works done

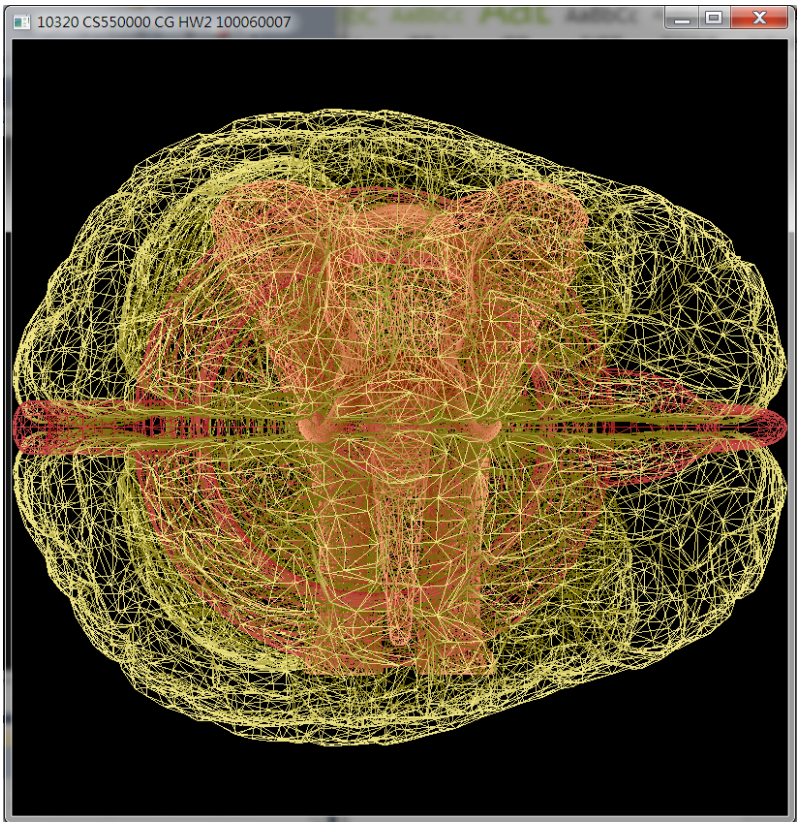
I wrote two C++ classes(MAT and VEC) to declare matrix and do basic matrix operations, and I utilize a resource from the Internet which contains similar functions to perform operations like rotation, translation and so forth.

Most of my efforts reside in the function “processNormalKeys.” In this function, I put almost all the operations, assignments in each case.

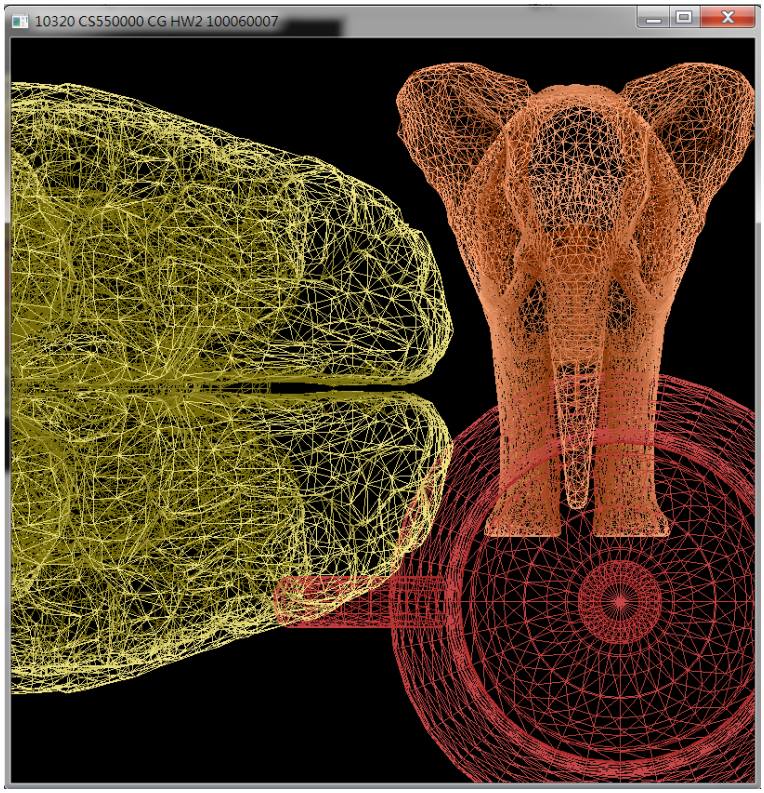
In colorModel(), I set new center to each OBJ->position after setting up transformation matrices. This can make the translation, rotation, scaling in both “processNormalKeys” and “processMouse” easier to implement.

Screenshots

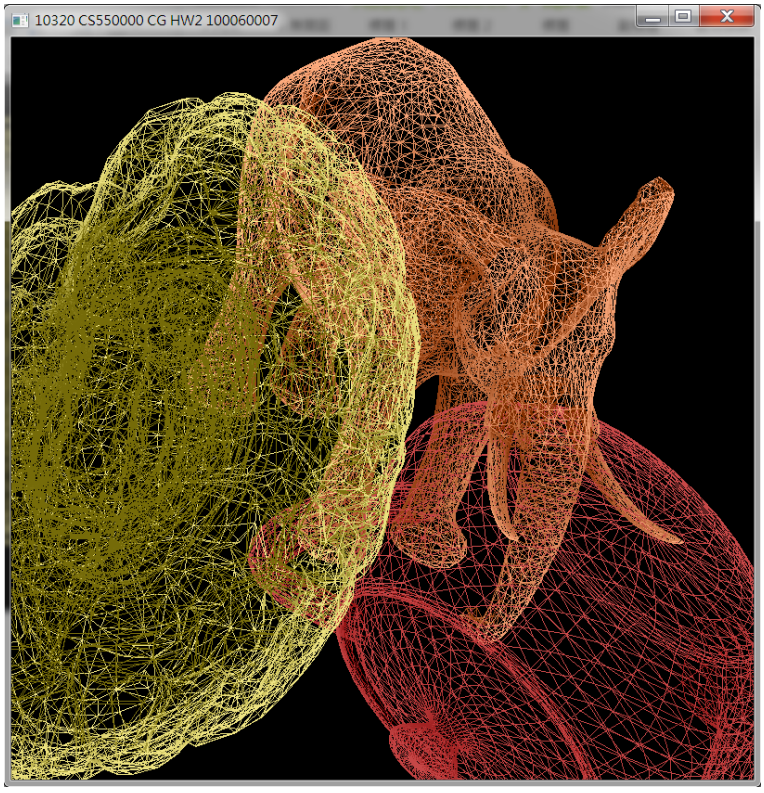
1. Starting window



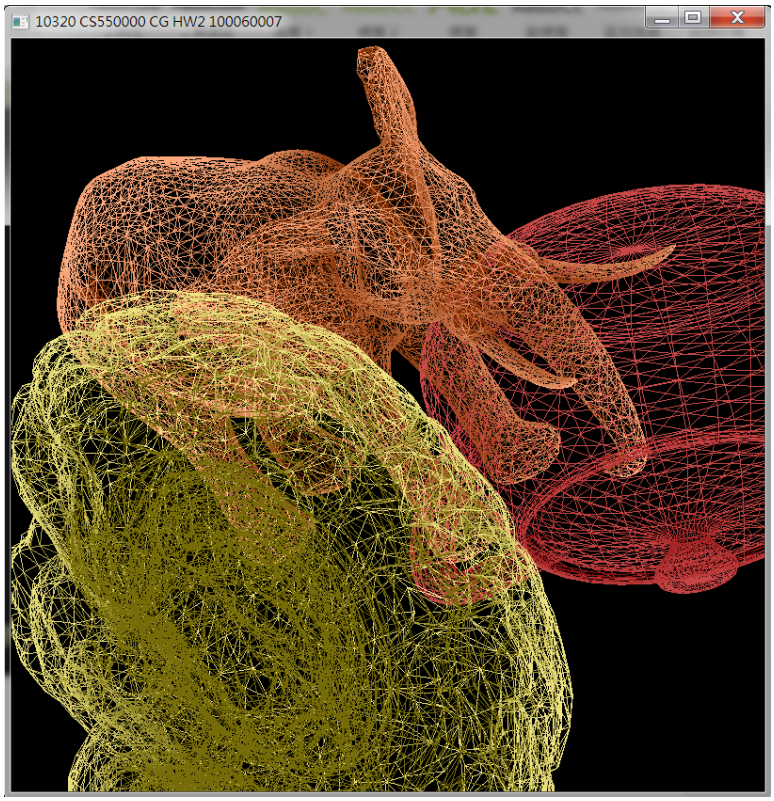
2. Move the models



3. Rotate the models



4. Change up vector



5. Scale the models and change the frame display mode