Student ID: 101062319
Name: 巫承威

Homework 2 - Report

Transformation

• How to operate my program?

As the Figure 1. shown, there are several new header files and source files in my project, please make sure they are all under the project folder.

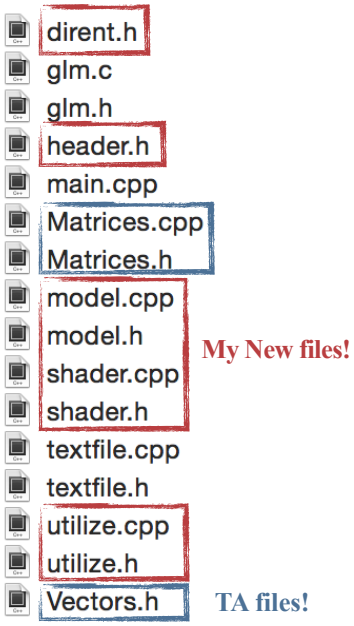

Figure 1.
Header Files and Source Files

Figure 2.
ScreenShot of Microsoft Visual Studio 2013

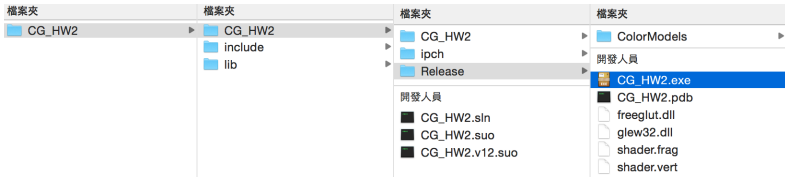

Figure 3.
Path of the "CG_HW2.exe"

All the user need to do to run my program is just open the .sln poject file via Microsoft Visual Studio 2013 (higher version may be fine). Then click the buttons "開始偵錯" or "啟動但不偵錯", and the execution window will appear. Or just execute the "CG_HW2.exe" as the Figure 3. shown. Like the example that TAs provided, user can type some button to control the display window and the object on it. These are the commands(note that I also allow the lowercase case letter to control the model as the uppercase letter does):

h / H - Show Help Menu
c / C - Clear Console
w / W - Switch between Solid / Wired
z Z Left / x X Right - previous / next model
t / T - Start Translation Mode
s / S - Start Scaling Mode
r / R - Start Rotation Mode
e / E - previous / next model
p / P - Parallel / Perspective Projection

• Implementation and problems I met

Based on my previous homework's framework, I add some attributes to my class "Model" and "ModelManager". At first, I store translation values(dx, dy, dz) and scaling factor for the normalization, and then build the matrix as follows:

```
Matrix4 NT = Matrix4(
                    1, 0, 0, -(this->model->dx),
                    0, 1, 0, -(this->model->dy),
                    0, 0, 1, -(this->model->dz),
                    0, 0, 0, 1);
Matrix4 NS = Matrix4(
                    1/this->model->scale, 0, 0, 0,
                    0, 1/this->model->scale, 0, 0,
                    0, 0, 1/this->model->scale, 0,
                    0, 0, 0, 1);
this->model->N = NS * NT;
```

Figure 4.

Normalization Matrix Setting

And times it on the right hand side of the matrix "M". Then, I create six variables for the translation in 3 dimension(tx, ty, tz), scaling in 3 dimension(sx, sy, sz) and the angle of the rotation in 3 dimension(rx, ry, rz). Since I forgot to use the useful library provided by TAs, I store the above values in each model and build 6 matrices(3 for 3 angles of the rotation) for them as follows:

```
Matrix4 T;  // translation matrix
Matrix4 S;  // scaling matrix
Matrix4 R;  // final rotation matrix
Matrix4 RX; // rotation matrix in X-axis
Matrix4 RY; // rotation matrix in Y-axis
Matrix4 RZ; // rotation matrix in Z-axis
```

Figure 5.

Declaration of Geometry Transformation Matrices

And set them like below process:

```
void ModelManager::setTranslationMatrix()
{
    this->model->T.set(
        1, 0, 0, this->model->tx,
        0, 1, 0, this->model->ty,
        0, 0, 1, this->model->tz,
        0, 0, 0, 1);
}
```

Figure 6.

Translation Matrix Setting

```
void ModelManager::setScalingMatrix()
{
    this->model->S.set(
        this->model->sx, 0, 0, 0,
        0, this->model->sy, 0, 0,
        0, 0, this->model->sz, 0,
        0, 0, 0, 1);
}
```

Figure 7.

Scaling Matrix Setting

```
void ModelManager::setRotationMatrix()
{
    this->model->RX.set(
        1, 0, 0, 0,
        0, cos(this->model->rx), -sin(this->model->rx), 0,
        0, sin(this->model->rx),  cos(this->model->rx), 0,
        0, 0, 0, 1);
    this->model->RY.set(
         cos(this->model->ry), 0, sin(this->model->ry), 0,
        0, 1, 0, 0,
        -sin(this->model->ry), 0, cos(this->model->ry), 0,
        0, 0, 0, 1);
    this->model->RZ.set(
        cos(this->model->rz), -sin(this->model->rz), 0, 0,
        sin(this->model->rz),  cos(this->model->rz), 0, 0,
        0, 0, 1, 0,
        0, 0, 0, 1);
    this->model->R = (this->model->RZ) * (this->model->RY) * (this->model->RX);
}
```

Figure 8.

Rotation Matrix Setting

After going through the above steps, I can multiply these matrices to the right hand side of the “Model Transformation Matrix”, and now I finish the geometry transformation of the model.

Second, follow the same steps just like previous processes, I also create 3 matrices for the viewing transformation:

```
Matrix4 VM; // final viewing matrix
Matrix4 VT;
Matrix4 VR;
```

Figure 9.

Declaration of Viewing Transformation Matrices

Then I set the viewing transformation matrix according to the steps on “CG04-Transformation p.72” as shown on the next page:

```
void ModelManager::setViewingMatrix()
{
    this->model->VT.set(
        1, 0, 0, -(this->model->eyePos[0]),
        0, 1, 0, -(this->model->eyePos[1]),
        0, 0, 1, -(this->model->eyePos[2]),
        0, 0, 0, 1);
    this->model->centerVec.set(
        this->model->centerPos[0] - this->model->eyePos[0],
        this->model->centerPos[1] - this->model->eyePos[1],
        this->model->centerPos[2] - this->model->eyePos[2]);
    this->model->upVec.set(
        this->model->upPos[0] - this->model->eyePos[0],
        this->model->upPos[1] - this->model->eyePos[1],
        this->model->upPos[2] - this->model->eyePos[2]);
    Vector3 Rz = (this->model->centerVec).operator/(this->model->centerVec.length());
    Vector3 Rx = this->model->centerVec.cross(this->model->upVec).operator/((this->model->centerVec.cross(this->model->upVec)).length());
    Vector3 Ry = Rx.cross(Rz);
    this->model->VR.set(
         Rx[0],  Rx[1],  Rx[2], 0,
         Ry[0],  Ry[1],  Ry[2], 0,
        -Rz[0], -Rz[1], -Rz[2], 0,
             0,      0,      0, 1);
    this->model->VM = this->model->VR * this->model->VT;
}
```

Figure 10.

Viewing Transformation Matrices Setting

Similar with the geometry transformation, just multiply the matrix “VM” to the right hand side of the viewing transformation matrix “V”, then the step of viewing transformation is finished.

Finally, I calculate the perspective projection matrix at first, and create a flag to represent the different projection method. Then change the projection matrix “P” based on this flag as follows:

```
void ModelManager::setProjectionMatrix(int projectionMode)
{
    if (!projectionMode){   // parallel
        P.set(
            1, 0,  0, 0,
            0, 1,  0, 0,
            0, 0, -1, 0,
            0, 0,  0, 1);
        V.set(
            1, 0, 0, 0,
            0, 1, 0, 0,
            0, 0, 1, 0,
            0, 0, 0, 1);
    }
    else{   // perspective
        P.set(
            1, 0,  0,  0,
            0, 1,  0,  0,
            0, 0, -2, -3,
            0, 0, -1,  0);
        V.set(
            1, 0, 0,  0,
            0, 1, 0,  0,
            0, 0, 1, -2,
            0, 0, 0,  1);
    }
}
```

Figure 11.

Viewing Transformation Matrices Setting

After finishing these processes, now I can just use the same codes from homework1 to draw the models properly. The following segment shows the drawing process:

"…At the beginning, I draw the models via running a for loop to draw the every triangle of the model which means I traverse the model's triangle array to get each triangle's information and to draw it via calling the function "glDrawArrays" in each round of loop. However, this method seems to be an inefficient way since it need to call the function "glDrawArrays" in each round of loop. Due to this concern, I modify the drawing process which will allocate two big float arrays first with the size of the number of the model's triangles times nine. Next, I traverse all the triangles of the model to fill the correct order of all the vertices which will be sent to the drawing function "glVertexAttribPointer" sooner. And it can fix the problem of calling "glDrawArrays" too many times since it only need to call it once. Another problem I met is how to add the keyboard features. At first, I come up with a method that I just only need to store them in a big character array of each file's name and this method is somehow called brute-force method. Fortunately, I saw the discussion on the iLMS's forum which is about the usage of the structure "DIR" and "dirent". I decide to learn how to use it, so I change the previous method. "DIR" and "dirent" are the structure used in Linux system, so if I want to use them in my homework, I need to include the library called "dirent.h". Also, this way can not only get all the models' file name but also can traverse from the given folder name to its subfolders and get all the files' name. It's more powerful and flexible than the brute-force method…"

• Other efforts I have done

Owing to the unfriendly codes which combine all the segment of the program into only one file, and also not so readable for the programmer, I decide to make it looked like the object-oriented program. Nevertheless, this idea cost me almost one afternoon to adjust them…. To finish this task, I need to figure out the operation between the model and the whole program and also how to manipulate each class. I think this is the most difficult part for transforming the original code to object-oriented type.

Another part I have done is to use the structure "DIR" and "dirent" which are already mentioned in above paragraph. It's more powerful and flexible than brute-force method because it can traverse all the files in the given folder's all subfolders. Also, I define new macro called "abs" to help get the length of the model. To avoid the re-define error of so many new header files, I add the "#pragma once" to make compiler correctly include the header file only once when they need.

What's more, I add the feature that my model can be scaled by adjusting the GLUT window. It means that if I change the GL window, my object model will also change its size to fit the size of window just like Figure 13. shown on the next page:

- **Screenshots**

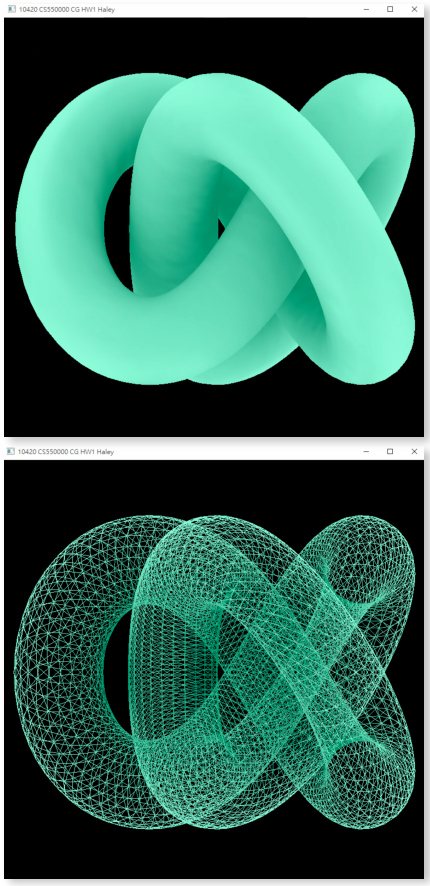

Figure 12.

Fill Mode & Wire Mode of the Model

“Texturedknot11KC.obj”

Figure 13.

Different Scale of the Model

“Texturedknot11KC.obj”

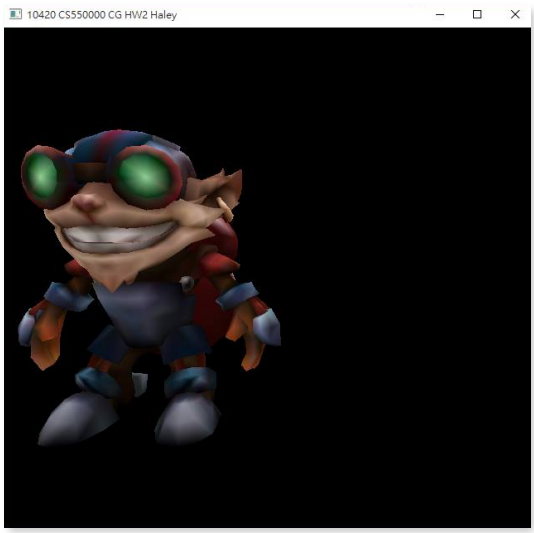

Figure 14.

Fill, Perspective Projection, Rotation of the Model

“ziggs.obj”

Figure 15.

Fill, Perspective Projection, Rotation of the Model

“blitzcrank_incognito.obj”

• Bonus

• Display

I re-organize my program's framework to fit the requirements only. I only read 5 objects in my bonus program, and the manipulation is the same with the basic version. I create a model list to store all 5 objects' attributes and maintain a index to indicate which model is focused on. Most of the functions are based on the basic version but add the additional parameter "index" to help notice that which model is under the control now.

• Screenshots

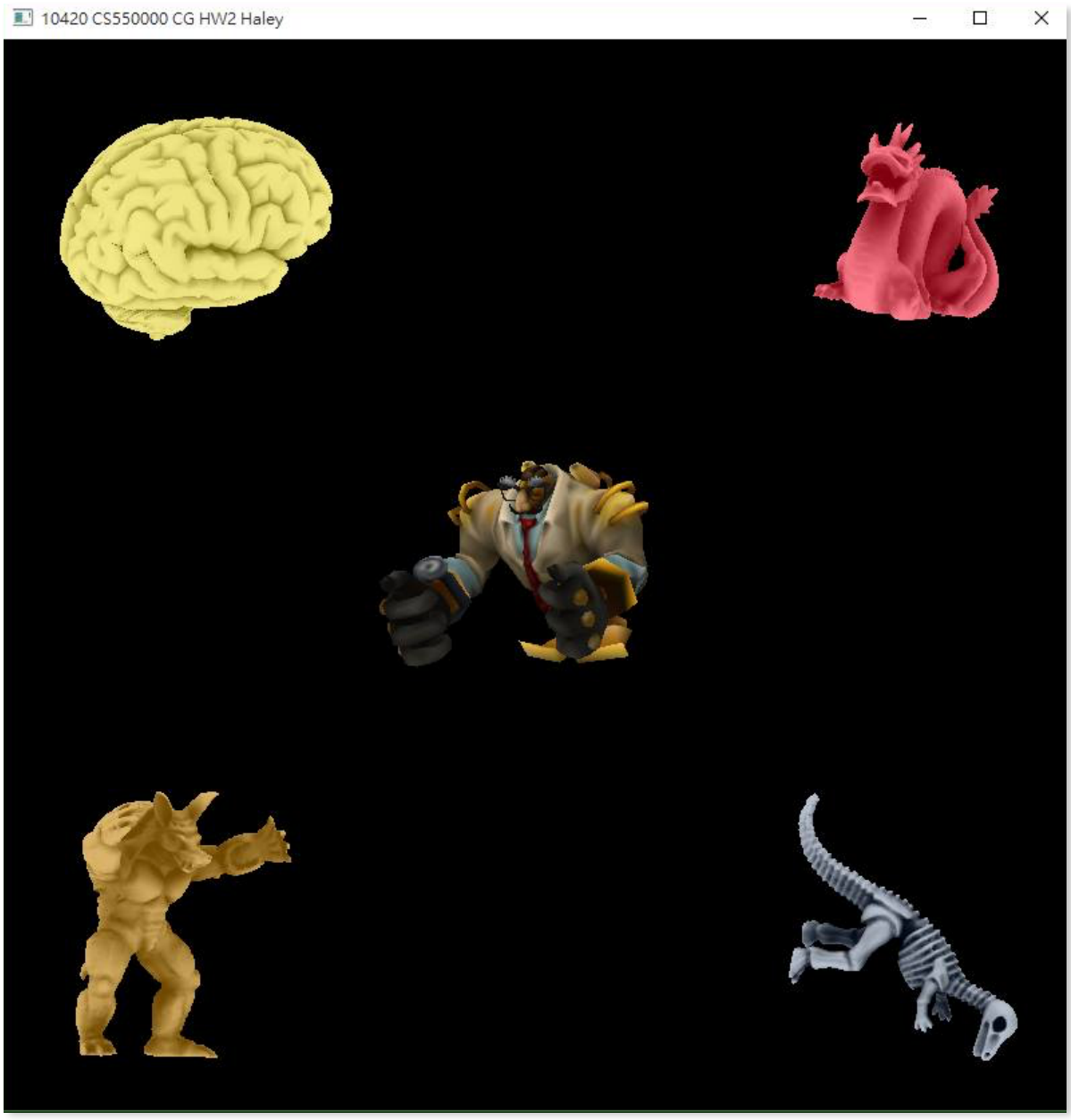

Figure 16.
Overview for the Bonus - Display