Zhihua Wang G48459695

The code contain two part.

One part is under main directory, its function is to use classes such as Matrix class, Projection class to build animation. This part contain two files, SimpleGLUT.cpp, which is the entry of the program and the main code to build animation, and StdAfx.cpp, which includes stdafx.h file to include some precompiled files to accelerate program.

The other part contains 8 files, which are matrix.h, operation.h, vec3.h, projection.h, parameter.h, camera.h, helper.h and model.h.

matrix.h is the code of my own Matrix class. The routines of my Matrix class include set value of a matrix, add two matrixes and multiply two matrixes.

Most code of vec3.h comes from project CG1\_LAB1. It defines vertex class. I overload one vec3’s operator \*, to define vec3 \* Matrix.

operation.h contains operations needed during program. Xproduct2d is to calculate the cross product of two vectors. Distance2d is to calculate the distance between two points. MultiMatrix can calculate the result of two matrixes. Random can return a float number which is between 0 and 1.

parameter.h contains global parameters, such as number of items, locations of files containing items’ points, directions of items and so on.

projection.h is the code of my own two functions using different parameters to approach perspective.

camera.h is the code of Camera Class. This class support to apply view transformation, which makes it looks like there is a camera look around.

Using 0, 1, 2 can choose one item to control. 0 is to control the car, 1 is to control the shuttle and 2 is to control the cow.

The code of calculating and displaying normal for each vertex is in Model.h.

In LoadModel routine, I add the code calculating normal vertexes. In DrawEdge routine, I add the code displaying normal for each vertex.

Using w, s, a, d, q, e we can control item’s position.

Using up, down arrow we can control car’s scale.

Using left, right arrow and pagedown, pageup we can control car’s rotation.

Using j, k, i, I, u, o we can control camera’s position.

Using n, m can rotate the camera.

Using mouse we can rotate item along with the mouse move direction around y axis. When click left mouse, item start rotating; when release the click, stop rotating.

We store model’s matrix in Model class. Whenever redraw pictures, we first replace current GL\_MODEL\_VIEW matrix of this matrix, then draw pictures. Whenever we apply a transformation on an item, we use a matrix recording this transformation and multiply this matrix with this model’s current matrix and replace current matrix of the result.