CS100433

Computer graphics

3D Scene of the Library

Report of the Course Project

Name : 马骐，徐涵立，蒋明会，刘天颖，梁锦锋

Student ID : 1452276，1452265，1452269，1452268，1452259

Ⅰ. Project Title

3D Scene of the Library

Ⅱ. Team member names

马骐，徐涵立，蒋明会，刘天颖，梁锦锋

Ⅲ. Abstract

With terrain render, the scene in our real world can be performed in the computer. We choose the project to display the library and its surrounding environment. This report will show how we designed and implement the project.

Ⅳ. Motivation

As the library is the landmark of Jiading Campus and the surrounding environment of which is the most beautiful, we want to take advantage of this chance to draft it with OpenGL.

Ⅴ. Goal of the project

1. Revert the environment as far as possible.

2. Realize some physical phenomena, especially the light, such as the reflection of the light.

3. Texture Mapping.

4. Few dynamic effects.

5. User can interactively manipulate the camera parameter.

Ⅵ. Scope of the project

No large-scale terrain render.

Ⅶ. Involved CG techniques

1. Modeling with Sketchup.

2. Rendering with OpenGL.

3. Light treatment with OpenGL.

4. Texture Mapping with OpenGL.

Ⅷ. Project contents

1. Modeling the library and the surrounding environment. Main buildings: the library, the teaching building A & B and the department of electronics and information engineering

2. Importing the model into OpenGL to render it, aiming to make the model more real.

3. Setting up the ambient light and diffuse light. With it we can realize some physical phenomena.

4. Texture Mapping. This technology makes the environment seem like real.

Ⅸ. Implementation

1. Build library prototype by using SketchUp.

2. Import our model(.obj) into opengl project

3. Add light and extra functions

4. Test and improve

Ⅹ. Results

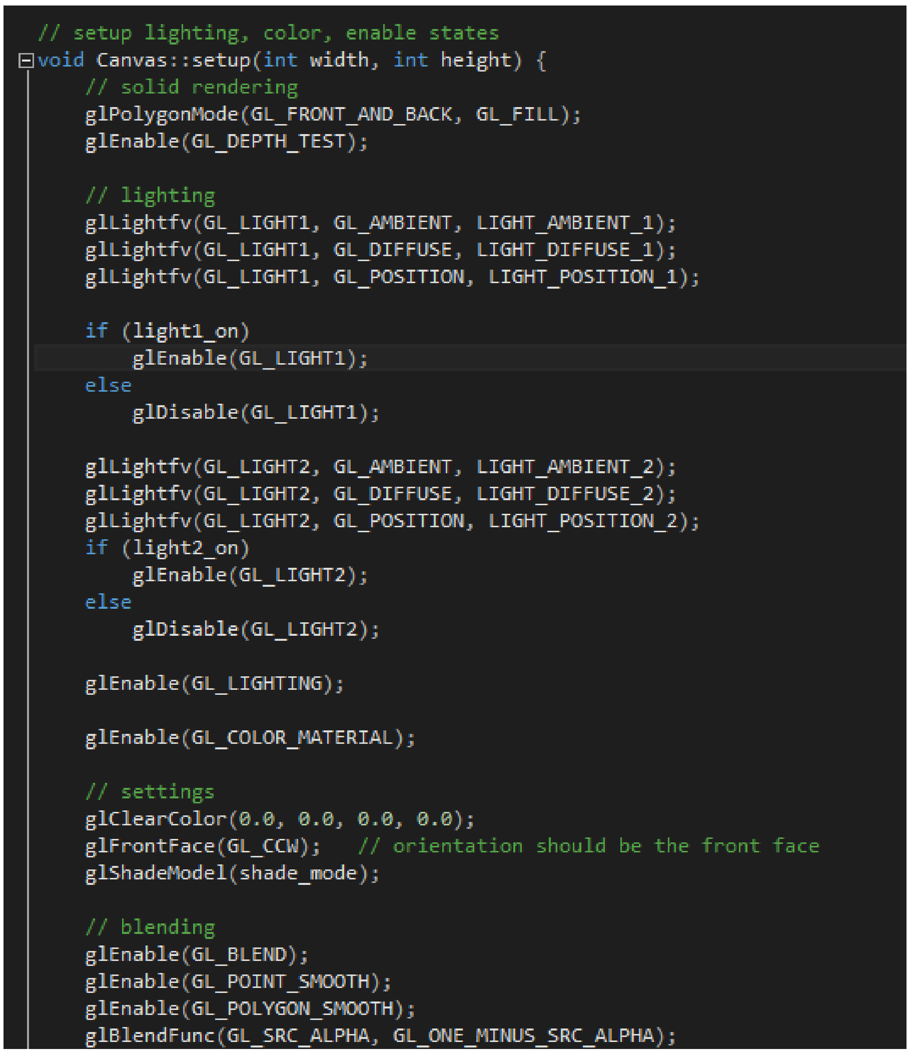
We successfully import our library model and add many extra functions like light-control, keyboard&mouse-control and automatic rotation and so on. However, we didn’t realize texture mapping. In a word, we have done something, but not very well.

Ⅺ. Roles in group

徐涵立 ’s role : Lighting and material settings.

OpenGL light has self luminous (Emitted Light), ambient light (Ambient Light), diffuse reflectance (Diffuse Light) and high light (Specular). The material is represented by light reflectivity. The objects in the scene (Scene) finally reflected in the human eye color is formed of light reflectivity multiplication RGB component and material RGB component after color.

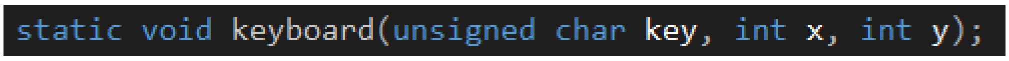
The code is like this:



```
// setup lighting, color, enable states
void Canvas::setup(int width, int height) {
    // solid rendering
    glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
    glEnable(GL_DEPTH_TEST);

    // lighting
    glLightfv(GL_LIGHT1, GL_AMBIENT, LIGHT_AMBIENT_1);
    glLightfv(GL_LIGHT1, GL_DIFFUSE, LIGHT_DIFFUSE_1);
    glLightfv(GL_LIGHT1, GL_POSITION, LIGHT_POSITION_1);

    if (light1_on)
        glEnable(GL_LIGHT1);
    else
        glDisable(GL_LIGHT1);

    glLightfv(GL_LIGHT2, GL_AMBIENT, LIGHT_AMBIENT_2);
    glLightfv(GL_LIGHT2, GL_DIFFUSE, LIGHT_DIFFUSE_2);
    glLightfv(GL_LIGHT2, GL_POSITION, LIGHT_POSITION_2);
    if (light2_on)
        glEnable(GL_LIGHT2);
    else
        glDisable(GL_LIGHT2);

    glEnable(GL_LIGHTING);

    glEnable(GL_COLOR_MATERIAL);

    // settings
    glClearColor(0.0, 0.0, 0.0, 0.0);
    glFrontFace(GL_CCW);    // orientation should be the front face
    glShadeModel(shade_mode);

    // blending
    glEnable(GL_BLEND);
    glEnable(GL_POINT_SMOOTH);
    glEnable(GL_POLYGON_SMOOTH);
    glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
```

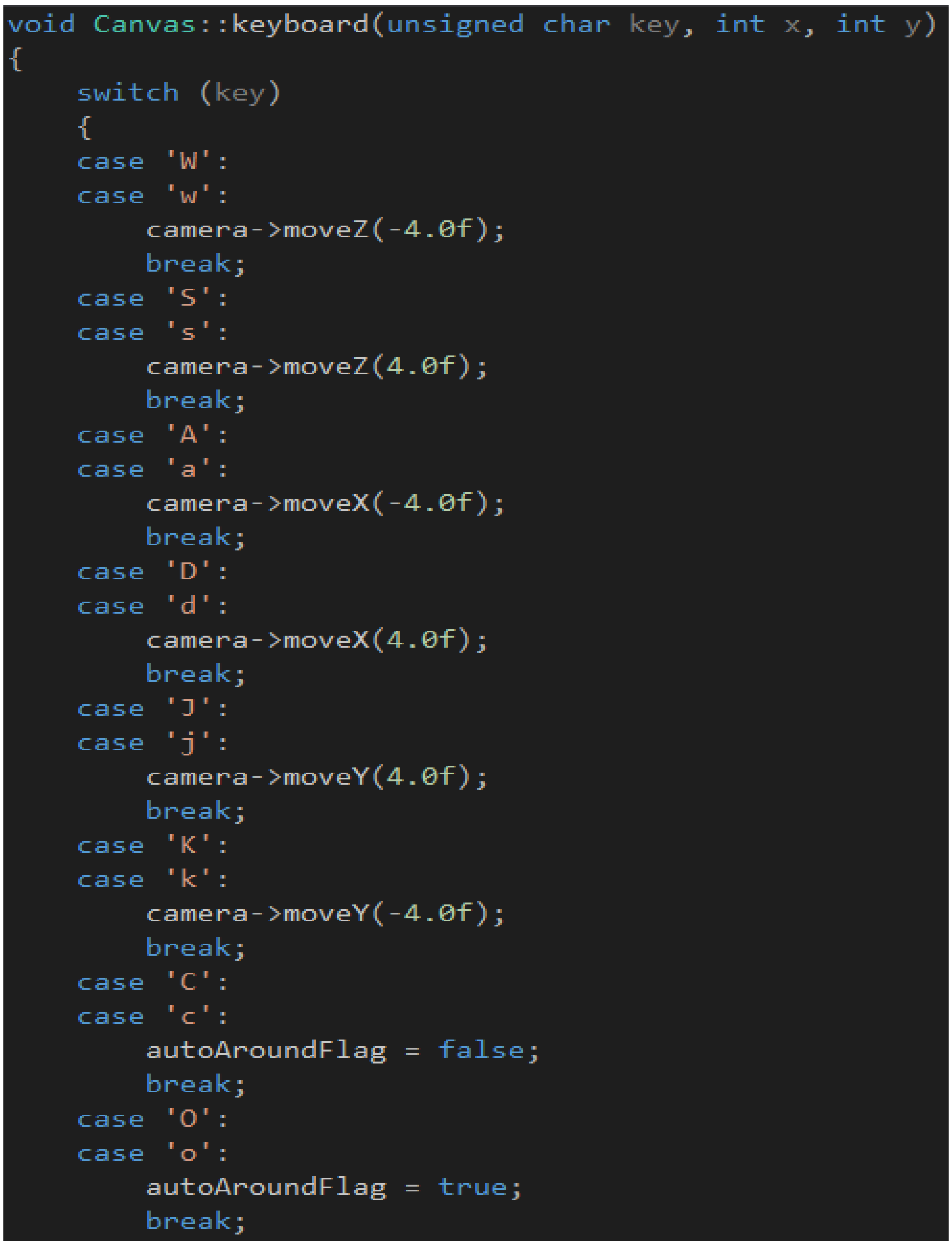
蒋明会’s role: Add keyboard&mouse function, BGM, automatic rotation,etc

keyboard fuction(Define in Canvas.h, actualize in Canvas.cpp):



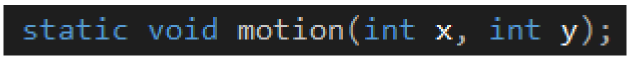
```
static void keyboard(unsigned char key, int x, int y);
```

Canvas::keyboard(): we get the input of keyboard and put it in a char named key and then do the right thing according to the key like follows



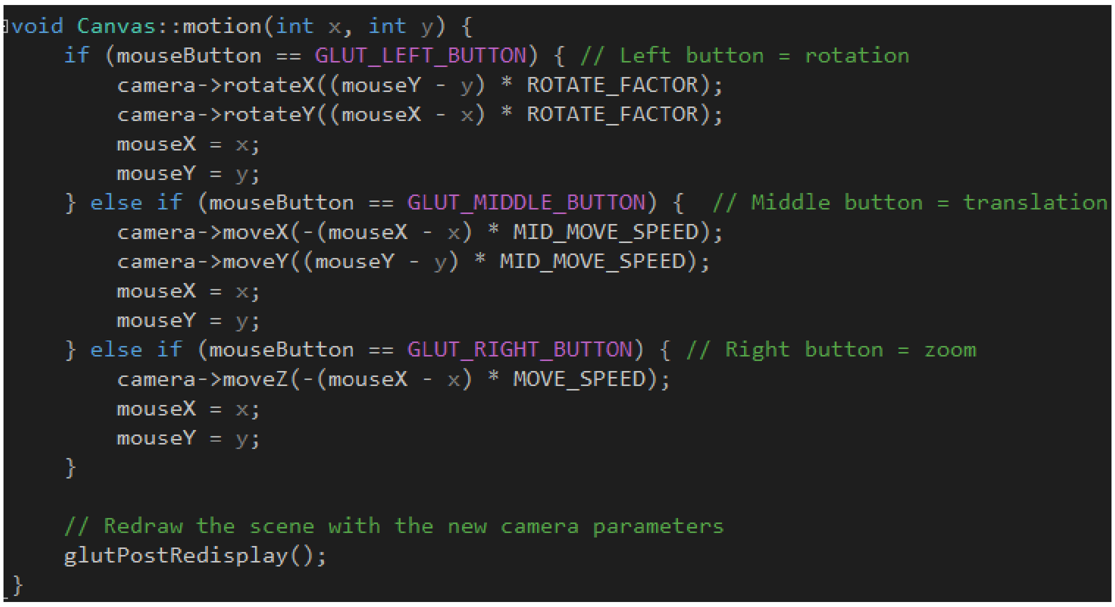
```
void Canvas::keyboard(unsigned char key, int x, int y)
{
    switch (key)
    {
    case 'W':
    case 'w':
        camera->moveZ(-4.0f);
        break;
    case 'S':
    case 's':
        camera->moveZ(4.0f);
        break;
    case 'A':
    case 'a':
        camera->moveX(-4.0f);
        break;
    case 'D':
    case 'd':
        camera->moveX(4.0f);
        break;
    case 'J':
    case 'j':
        camera->moveY(4.0f);
        break;
    case 'K':
    case 'k':
        camera->moveY(-4.0f);
        break;
    case 'C':
    case 'c':
        autoAroundFlag = false;
        break;
    case 'O':
    case 'o':
        autoAroundFlag = true;
        break;
```

mouse fuction(Define in Canvas.h, actualize in Canvas.cpp):



```
static void motion(int x, int y);
```

Canvas::motion(): change the parameters of camera according to the different mouse actions



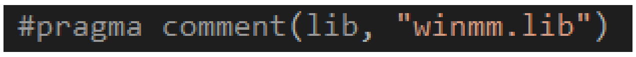
```
void Canvas::motion(int x, int y) {
    if (mouseButton == GLUT_LEFT_BUTTON) { // Left button = rotation
        camera->rotateX((mouseY - y) * ROTATE_FACTOR);
        camera->rotateY((mouseX - x) * ROTATE_FACTOR);
        mouseX = x;
        mouseY = y;
    } else if (mouseButton == GLUT_MIDDLE_BUTTON) {  // Middle button = translation
        camera->moveX(-(mouseX - x) * MID_MOVE_SPEED);
        camera->moveY((mouseY - y) * MID_MOVE_SPEED);
        mouseX = x;
        mouseY = y;
    } else if (mouseButton == GLUT_RIGHT_BUTTON) { // Right button = zoom
        camera->moveZ(-(mouseX - x) * MOVE_SPEED);
        mouseX = x;
        mouseY = y;
    }

    // Redraw the scene with the new camera parameters
    glutPostRedisplay();
}
```

Add backgroundMusic(Actualize in Main.cpp):

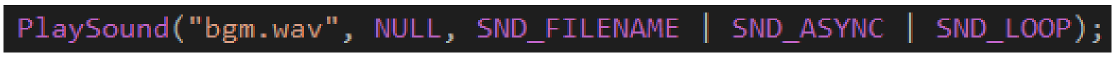
function prototype:

BOOL PlaySound(LPCSTR pszSound, HMODULE hmod,DWORD fdwSound);

add “#pragma comment(lib, "winmm.lib")” before use

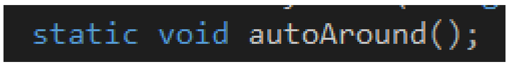


```
#pragma comment(lib, "winmm.lib")
```



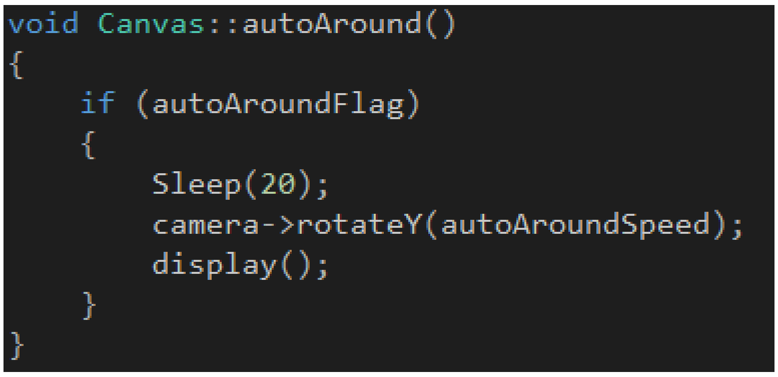
```
PlaySound("bgm.wav", NULL, SND_FILENAME | SND_ASYNC | SND_LOOP);
```

automatic rotation(Define in Canvas.h, actualize in Canvas.cpp):



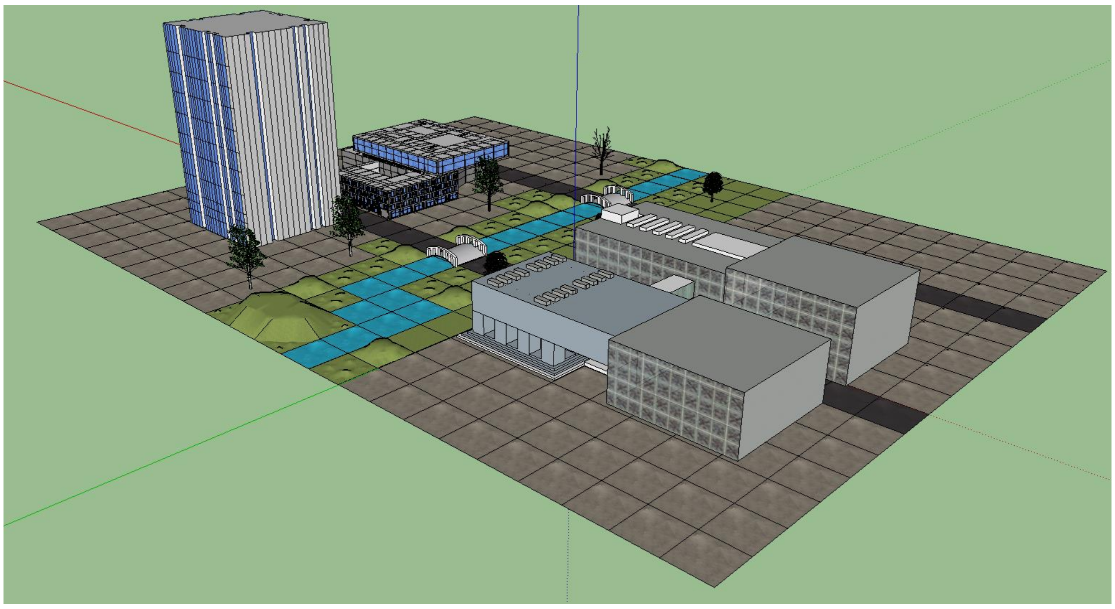
```
 static void autoAround();
```

We have a autoAroundFlag which will be true when user presses button “O” and become false when presses “C”. And we let the secne rotates automatically only when the autoAroundFlag is true.



```
void Canvas::autoAround()
{
    if (autoAroundFlag)
    {
        Sleep(20);
        camera->rotateY(autoAroundSpeed);
        display();
    }
}
```

马骐's role: build the model of terrain and the teaching building A & B with SketchUp.

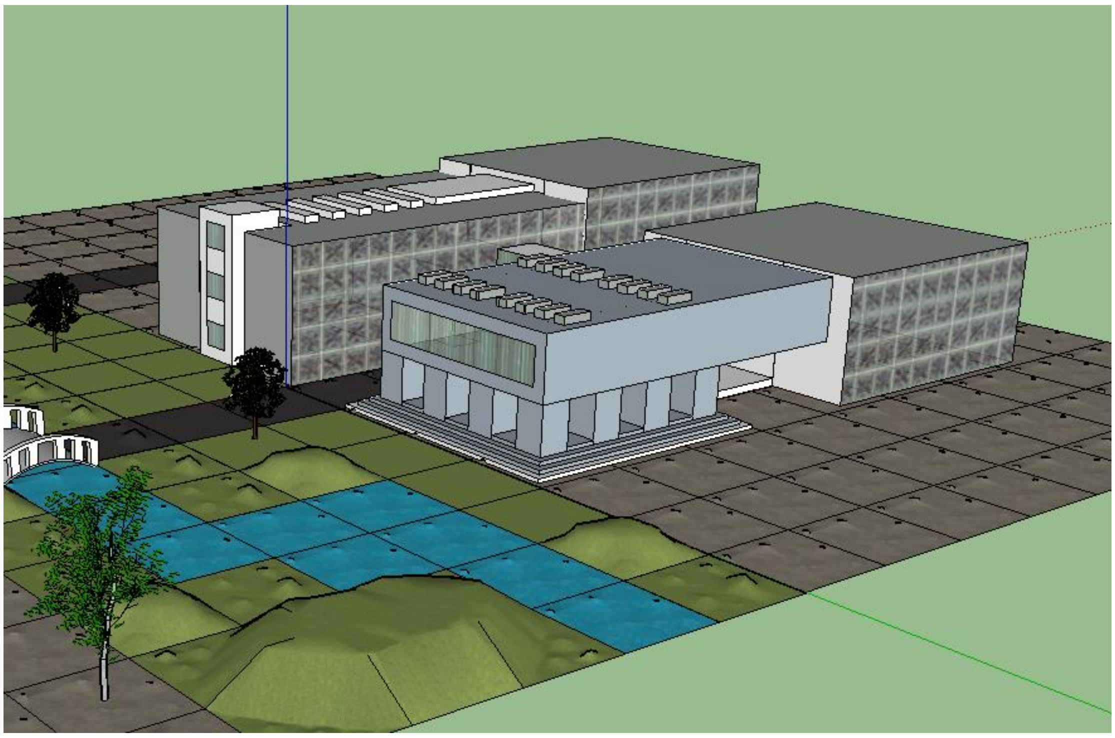


implement details:

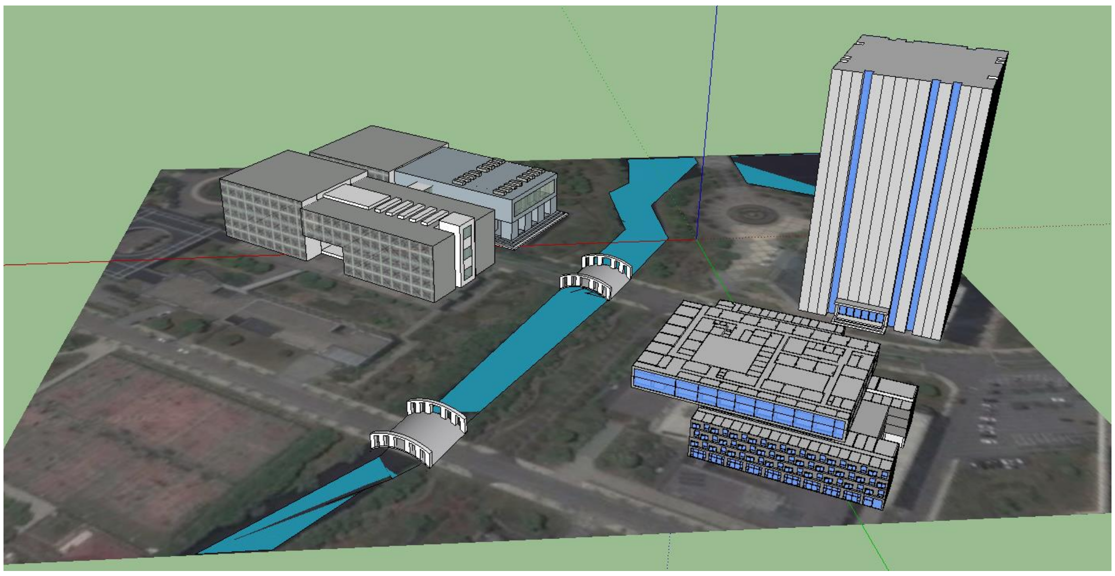
(1) spend some time on learning how to use SketchUp to build 3D model

(2) compare the rough model with the pictures of the builds and take on-the-spot investigation to modify the details of the model.

(3) adopt the model to our original goal and make it suitable for our project



刘天颖's role: build the model of the library, the bridges and the department of electronics and information engineering with SketchUp.



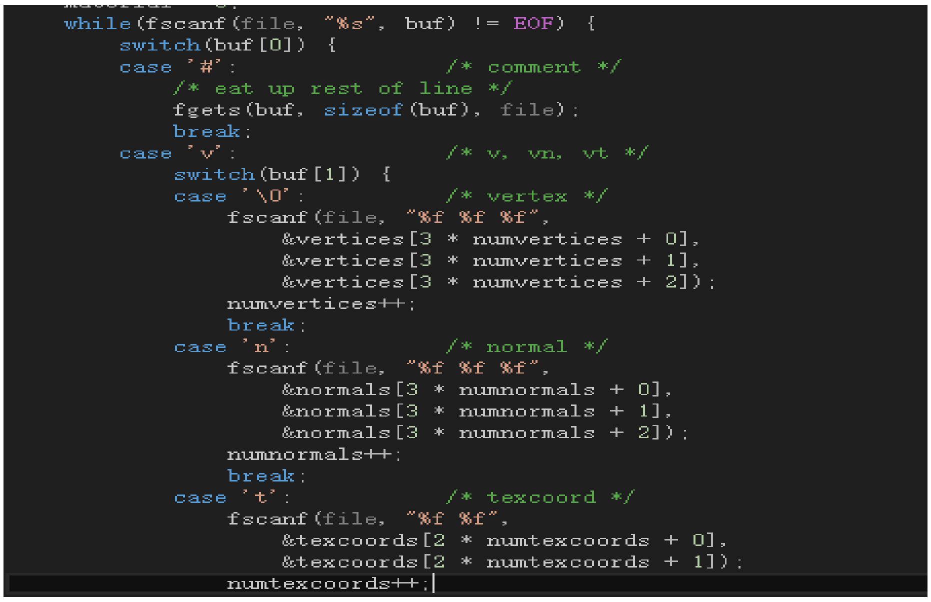
Implement details:

After the main buildings' model built, I chose to add the terrain with Google Earth so I could build 3D terrain conveniently. However, as we can see the picture above, the river is fractured. Teacher Zhao told us the cause is the precision. When the lens is far from the model, the precision can not meet our demand. Another reason that we didn’t choose this model is we didn’t achieve our goals of texture mapping so the terrain will be dark in the final scene. Then we changed our mind to build the terrain with blocks. Although it’s not very real, the texture can be performed good. It’s our pity.

梁锦锋 s role: Import the 3D model and draw the 3D model in OpenGL. It can be divided into 2parts.

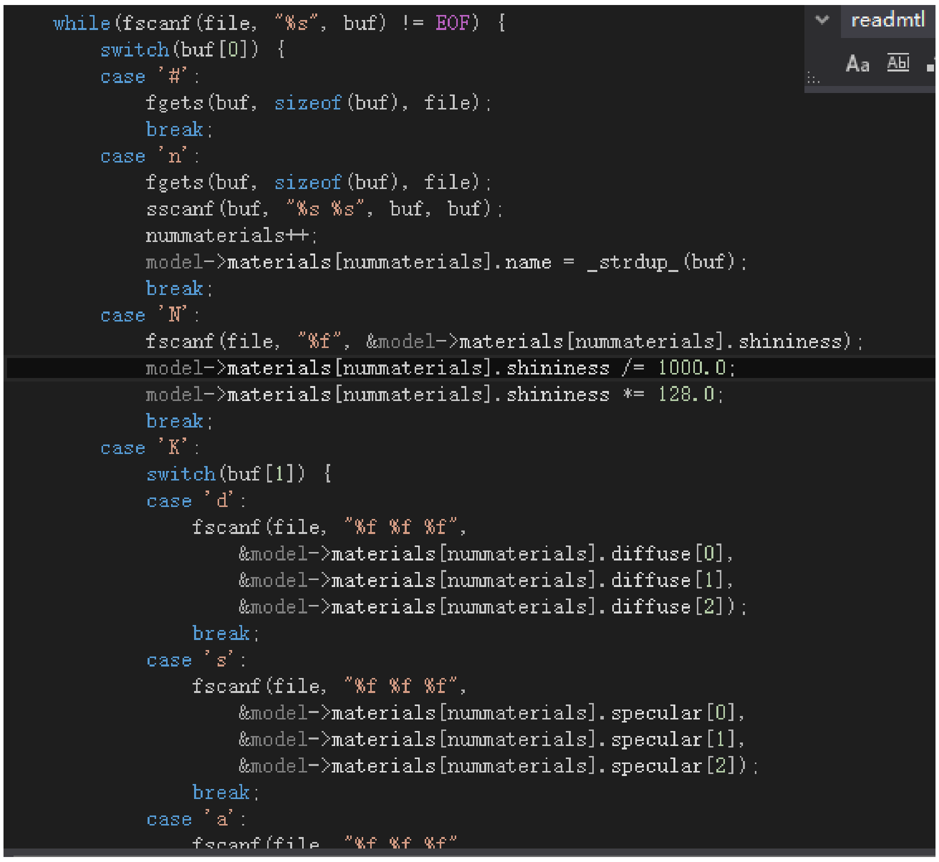
Part 1: Read obj&mtl file. According to the format of obj&mtl file, we can read obj file like the following code. The mtl file also has fixed format and it is assocaited with obj file. Here is how we read obj&mtl file(just a part of the all code):

Read obj file:



```
    material = 0;
    while(fscanf(file, "%s", buf) != EOF) {
        switch(buf[0]) {
        case '#':               /* comment */
            /* eat up rest of line */
            fgets(buf, sizeof(buf), file);
            break;
        case 'v':               /* v, vn, vt */
            switch(buf[1]) {
            case '\0':          /* vertex */
                fscanf(file, "%f %f %f",
                    &vertices[3 * numvertices + 0],
                    &vertices[3 * numvertices + 1],
                    &vertices[3 * numvertices + 2]);
                numvertices++;
                break;
            case 'n':           /* normal */
                fscanf(file, "%f %f %f",
                    &normals[3 * numnormals + 0],
                    &normals[3 * numnormals + 1],
                    &normals[3 * numnormals + 2]);
                numnormals++;
                break;
            case 't':           /* texcoord */
                fscanf(file, "%f %f",
                    &texcoords[2 * numtexcoords + 0],
                    &texcoords[2 * numtexcoords + 1]);
                numtexcoords++;
```

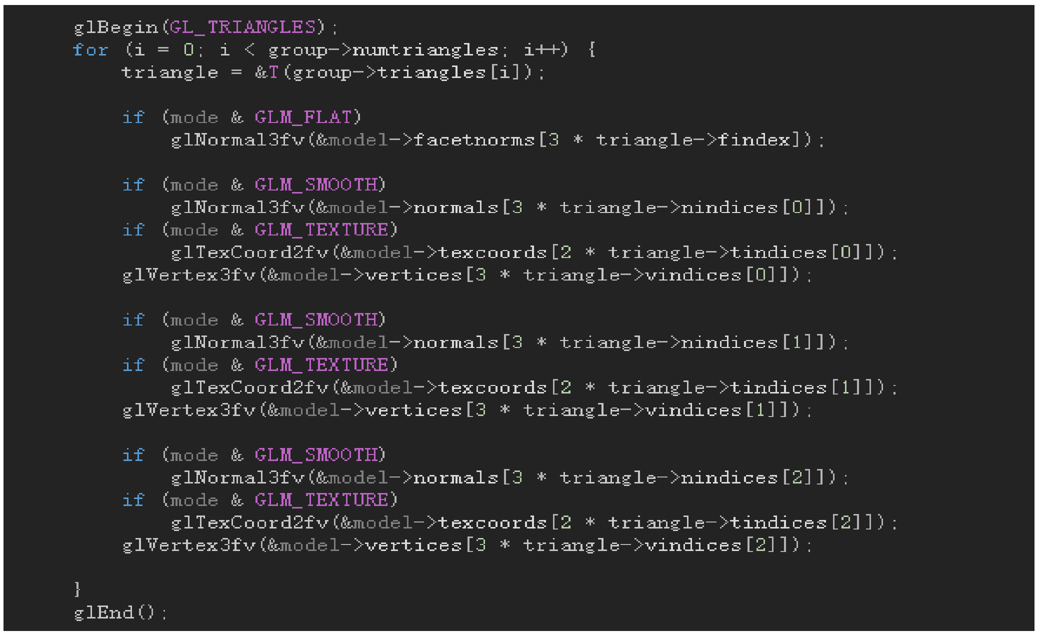
Read mtl file:



```
    while(fscanf(file, "%s", buf) != EOF) {
        switch(buf[0]) {
        case '#':
            fgets(buf, sizeof(buf), file);
            break;
        case 'n':
            fgets(buf, sizeof(buf), file);
            sscanf(buf, "%s %s", buf, buf);
            nummaterials++;
            model->materials[nummaterials].name = _strdup_(buf);
            break;
        case 'N':
            fscanf(file, "%f", &model->materials[nummaterials].shininess);
            model->materials[nummaterials].shininess /= 1000.0;
            model->materials[nummaterials].shininess *= 128.0;
            break;
        case 'K':
            switch(buf[1]) {
            case 'd':
                fscanf(file, "%f %f %f",
                    &model->materials[nummaterials].diffuse[0],
                    &model->materials[nummaterials].diffuse[1],
                    &model->materials[nummaterials].diffuse[2]);
                break;
            case 's':
                fscanf(file, "%f %f %f",
                    &model->materials[nummaterials].specular[0],
                    &model->materials[nummaterials].specular[1],
                    &model->materials[nummaterials].specular[2]);
                break;
            case 'a':
                fscanf(file, "%f %f %f",
```

Part 2:Draw 3D model

Using the methods that OpenGL tell us we can draw the 3D model after we read the obj&mtl file.



```
    glBegin(GL_TRIANGLES);
    for (i = 0; i < group->numtriangles; i++) {
        triangle = &T(group->triangles[i]);

        if (mode & GLM_FLAT)
            glNormal3fv(&model->facetnorms[3 * triangle->findex]);

        if (mode & GLM_SMOOTH)
            glNormal3fv(&model->normals[3 * triangle->nindices[0]]);
        if (mode & GLM_TEXTURE)
            glTexCoord2fv(&model->texcoords[2 * triangle->tindices[0]]);
        glVertex3fv(&model->vertices[3 * triangle->vindices[0]]);

        if (mode & GLM_SMOOTH)
            glNormal3fv(&model->normals[3 * triangle->nindices[1]]);
        if (mode & GLM_TEXTURE)
            glTexCoord2fv(&model->texcoords[2 * triangle->tindices[1]]);
        glVertex3fv(&model->vertices[3 * triangle->vindices[1]]);

        if (mode & GLM_SMOOTH)
            glNormal3fv(&model->normals[3 * triangle->nindices[2]]);
        if (mode & GLM_TEXTURE)
            glTexCoord2fv(&model->texcoords[2 * triangle->tindices[2]]);
        glVertex3fv(&model->vertices[3 * triangle->vindices[2]]);

    }
    glEnd();
```

Ⅻ. References

Donald Hearn, M. Pauline Baker, 2012, Computer Graphics with OpenGL Fourth Edition, Prentice Hall.

Dave Shreiner, Graham Sellers, John Kessenich, Bill Licea-Kane, 2013, OpenGL Programming Guide:the Official Guide to Learning OpenGL, Version 4.3, Eighth Edition, Addison-Wesley Professional.

http://www.sketchup.com/zh-CN/learn