COMP3069 Computer Graphics Coursework Report

Yiming TANG 20126870
2021.12.17

Abstract

In this coursework, we are asked to implement a **scene** using OpenGL.

The project that I have built, is a scene of forest, called "As the Wind". With the breeze blows and the grass shakes, I want to describe a quiet and peaceful space for you to dream and relax.

Figure 1, project screenshot, As the Wind

You can also check the demo video in Attachment.

The full source code can be find in the Moodle submission.

Background

The inspiration of creating such a scene comes from a game, ***Harry Potter: Magic Awakened***. It creates a beautiful and amusing snow scene.

Figure 2, screenshot of Harry Potter: Magic Awakened

Environment

macOS 12.1 (21C52)

Xcode Version 13.0 beta 5 (13A5212g)

Usage

To build this project, first change all the paths in the code from mine to your **absolute paths**. For example, in `MyScene.cpp`, while the code says `Sky *s = new Sky("/Users/yiming/Library/Mobile Documents/com~apple~CloudDocs/Y4/ CGCW素材/sky2.bmp");` in Line 27, the path should be changed according to this file's location in your environment. Note that different cases are tried but Xcode still cannot find the file by **relative paths**. Using absolute paths is a safe approach.

Click `Start the active scheme` in Xcode, and it should be running.

Requirements Meeting

In this chapter I will talk about how this project meets the requirements stated in the issue sheet. Because some parts overlap, such as Transformation and Animations, but in order to correspond to the entries on the issue sheet one by one, I listed them separately and referenced each other.

Researches will be shown in 3D Model.

3D Model

Trees and grasses are built in a creative way. The trees in this project are more realistic compared to others. See Figure 1 for the visualisation.

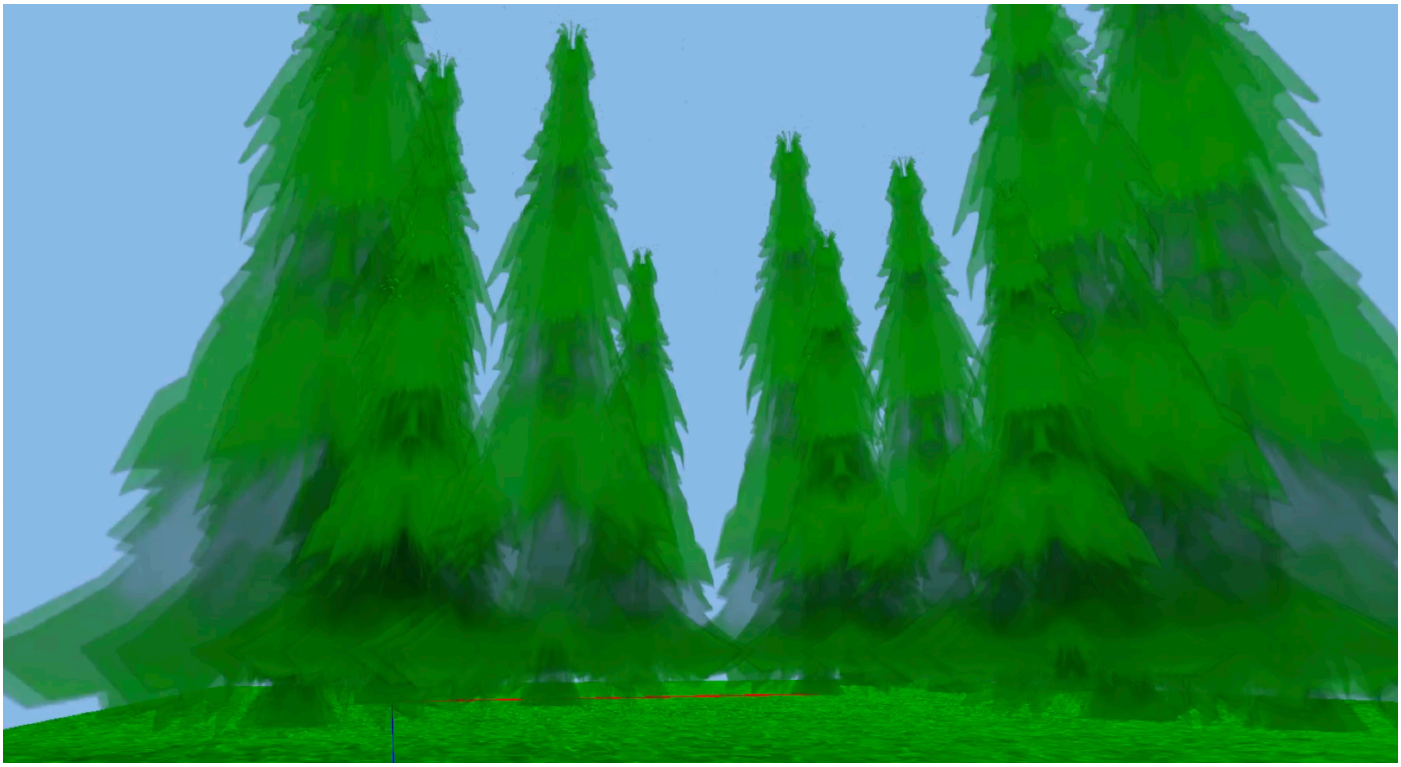

Figure 3, the model of tree

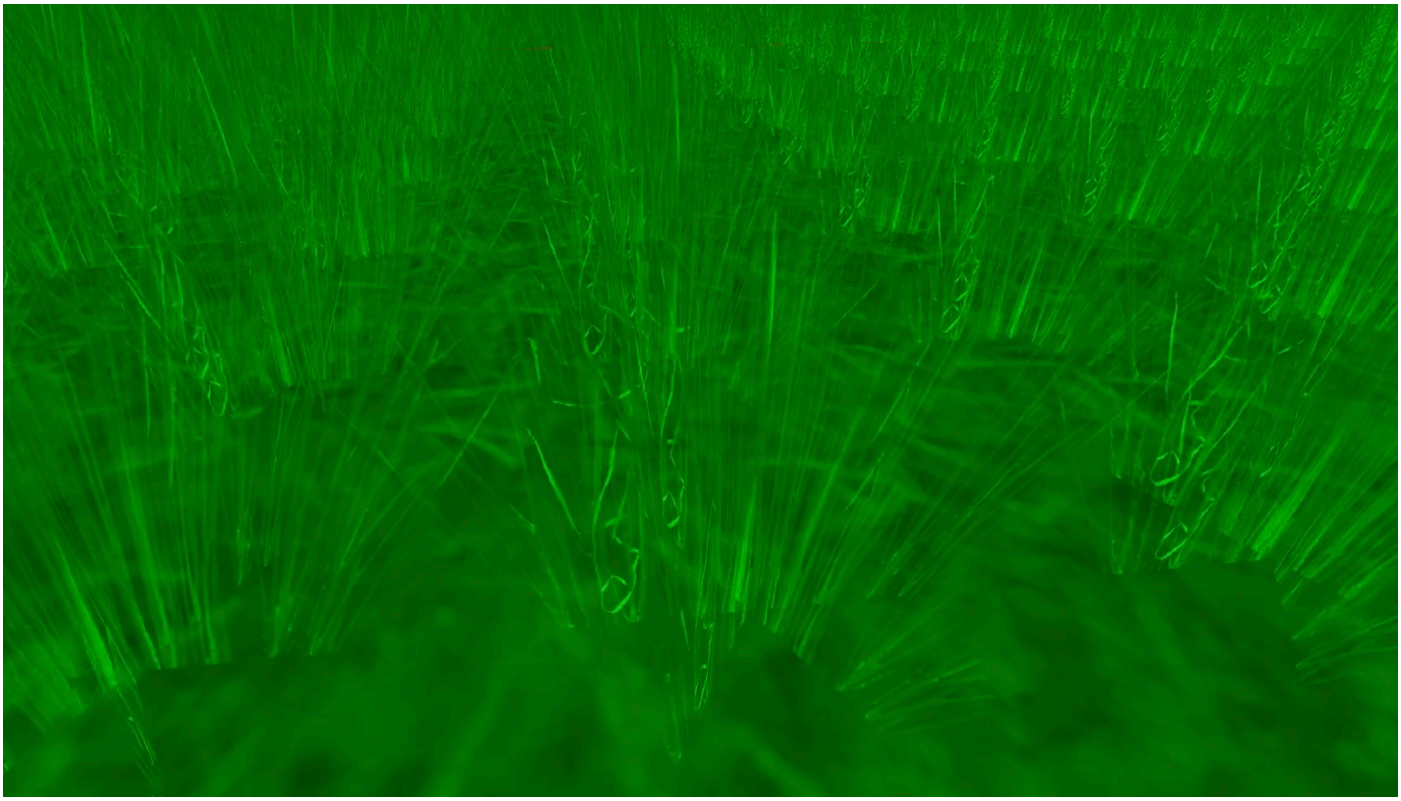

Figure 4, the model of grass

Check the code in Transformation.

Transformation

`glTranslatef()` and `glRotatef()` are used in this project.

When "planting" the trees and grasses, `glTranslatef()` should be used. Here are some essential parts:

In `Tree.cpp`, a sample usage of `glTranslatef()`:

```
void Tree::drawTree(GLfloat x, GLfloat y, GLfloat z){
    // This function draws a tree with texture.
    glColor3f(0, 1, 0);

    glPushMatrix();
    glTranslatef(x, y, z);
    glRotatef(0.4*sin(animateTime)+6*sin(animateTime*1.3), 0, 1, 0);
    glDisable(GL_DEPTH_TEST);  // [IMPORTANT] disable GL_DEPTH_TEST to avoid shelting. see the report.
    for(int i=0;i<360;i+=360/10){  // 10 is the best amount to balance transparency and visualbility
        glPushMatrix();
        glRotatef(i, 0, 1, 0);
        drawRec(4, 16);
        glPopMatrix();
    }
    glEnable(GL_DEPTH_TEST);
    glPopMatrix();

    glColor3f(1, 1, 1);

}

```

In `Grass.cpp`, a sample usage of `glTranslatef()` and `glRotatef()`:

```
void Grass::Display(void){

    glColor3f(0, 1, 0);
    glPushMatrix();
    glTranslatef(0, 0, 0);
    glRotatef(animateRotation,0,1,0);

    glDisable(GL_DEPTH_TEST);

    for(int x=0;x<15;x++){
        for(int z=0;z<10;z++){
            // Grass
            glPushMatrix();
            glTranslatef(x, 0, z);
```

```
            glRotatef(0.4*sin((animateTime-2*z+3*x)*(2-x))+1*sin(animateTime*3), 0, 0, 1);
            glRotatef(3*sin((animateTime+2*z-3*x)*(5+z))+2*sin(animateTime*13), 1, 0, 0);
            for(int i=0;i<360;i+=360/6){
                glPushMatrix();
                glRotatef(i, 0, 1, 0);
                drawRec(0.8, 2);
                glPopMatrix();
            }
            glPopMatrix();
        }
    }

    glEnable(GL_DEPTH_TEST);
    glPopMatrix();

}
```

For other transformations, see Animation for more details.

Different Viewpoint

G53GRA provides basic camera control. By pressing and releasing `W`, `A`, `S` and `D`, you can control the camera's position in XoZ plane. By dragging with left click, you can control the camera's direction.

Animation

Animations are mainly applied in the trees, the grasses, the clouds and the moon.

To implement a realistic effect of grasses' shaking, I applied two trigonometric functions to x and z axes, adjusted their parameters again and again. In order to make each grass act differently, I added their positions, i.e., `x` and `z` into the functions, to make it more realistic.

```
glRotatef(0.4*sin((animateTime-2*z+3*x)*(2-x))+1*sin(animateTime*3), 0, 0, 1);
glRotatef(3*sin((animateTime+2*z-3*x)*(5+z))+2*sin(animateTime*13), 1, 0, 0);
```

The animation of trees are implemented in the same way with simpler trigonometric functions.

```
glRotatef(0.4*sin(animateTime)+6*sin(animateTime*1.3), 0, 1, 0);
```

Check the more complete code in Transformation.

Texturing

While I want to build the trees, one thing I noticed is that our framework supports only BMP images. There are 3 different kinds of BMP images: 16-bit, 24-bit and 32-bit. 32-bit BMP images contain X channels (Alpha channels), which can store transparency values. However, although our framework recognise that, it does not apply the alpha values to the generated textures.

I have tried other ways, and see Presentation for more details. Finally, I have found that if we set the `glColor3f()`'s parameters to `0, 1, 0` (which means pure green) and let the BMP's background colour to be pure green, the background colour will be removed. However, although the image we give it is a tree of white colour, it turns our to be green finally. But green is acceptable: at least as a tree it is not a strange colour.

Figure 5, the image that we use to texture

One important thing is that, to realise the trees, `GL_DEPTH_TEST` must be disabled. Otherwise the 'hidden' background colours will hide other parts of the tree behind it.

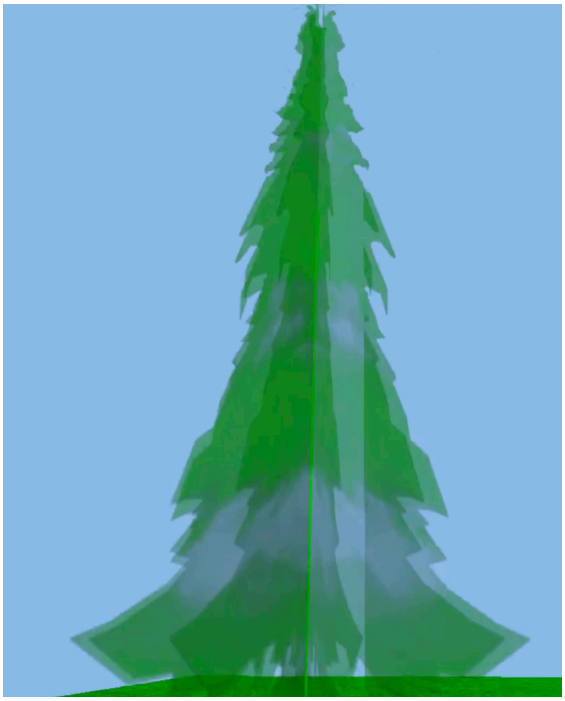

Figure 6, some parts of the tree are hideen, if GL_DEPTH_TEST is enabled

See Creative Idea for more details.

Lighting

Basic lighting that the framework provides, is applied in this project.

In `Engine.cpp`:

```
glLightModeli(GL_LIGHT_MODEL_TWO_SIDE, GL_FALSE);

  /*
// set the ambient light model
GLfloat global_ambient[] = {1.f, 1.f, 1.f, 0.0f};
glLightModelfv(GL_LIGHT_MODEL_AMBIENT, global_ambient);
   */

// enable directional light lighting
// (x, y, z, 0.0) -> directional lighting
// (x, y, z, 1.0) -> positional lighting

glEnable(GL_LIGHTING);
GLfloat ambience[] = {0.0f, 0.0f, 0.0f, 1.0f};
GLfloat diffuse[] = {1.0f, 1.0f, 1.0f, 1.0f};
GLfloat specular[] = {0.0f, 0.0f, 0.0f, 0.0f};
GLfloat position[] = {1.0f, 1.0f, 1.0f, 0.0f};

  /*
  // original below
  GLfloat ambience[] = {0.2f, 0.2f, 0.2f, 1.0f};
  GLfloat diffuse[] = {0.8f, 0.8f, 0.8f, 1.0f};
  GLfloat specular[] = {1.0f, 1.0f, 1.0f, 1.0f};
  GLfloat position[] = {1.0f, 1.0f, 1.0f, 0.0f};
   */
glLightfv(GL_LIGHT0, GL_AMBIENT, ambience);
glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuse);
glLightfv(GL_LIGHT0, GL_SPECULAR, specular);
glLightfv(GL_LIGHT0, GL_POSITION, position);
glEnable(GL_LIGHT0);

// Enable smooth shading from lighting
glShadeModel(GL_SMOOTH);

// Enable automatic normalisation of normal vectors
glEnable(GL_NORMALIZE);
```

Code Readability

The piece of code is written clearly and with proper comments. It follows strictly what the framework wants.

Check the code in Moodle submission for details.

Creative Idea

One of the proudest work I've done, is that I successfully remade the idea from the games that I play. Check Background for more details.

The basic idea is as follows: if we make the midlines of two identical rectangles coincide and make them perpendicular to each other, we will get a three-dimensional cross. If we make the center lines of three identical rectangles coincide and make them form an angle of 120 degrees between each other, we will get a three-dimensional shape similar to "stacked snowflakes".

Support we now have 3 tree images. we **rotate** two of them each 60 degrees by their midlines, with one clockwise and another anti-clockwise, and put them together with their midlines aligned, we can get such a shape.

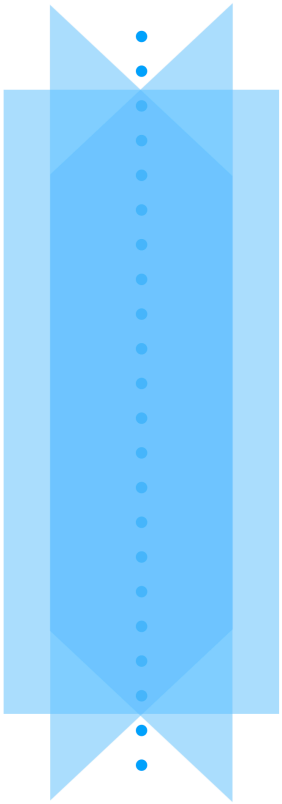

Figure 7, how to create the tree

It looks like stacked snowflakes. This is what we want to do. We put the 'leaves' together to build a tree.

Many games, including ***League of Legends*** and ***PUBG***, present some non-important items in this way. It seems that this is a common way for the 3D games. It costs the GPU and the modeler less, but the visual effect seems to be as good as detailed models.

The grass is built by the same idea.

See Texturing for more details on how to texture this model.

I am confident that this, together with the way that I texture this model, is the creative key to differenciate my project to others' works.

Conclusion

I think the most interesting thing is that I realised that OpenGL is the most basic ones, with no help from game engines or 3D softwares. By doing this project I took experiments on OpenGL and C++, getting deeper understanding about them; it's really a hard time to make a project, thanks to the guys who helped and encouraged me.

Finally, many thanks to you reading such a long report. Thanks for your teaching and marking.

Attachment

Demo Video

You may click this link to see the video demo.

Presentation

The presentation that I have done is a glance of this report. It contains fundamental basic ideas and principles of this project: Background, 3D Model and Creative Idea.

Slides

You may click this link to see the presentation slides.

Notes

You may click this link to see the speech note that I have written for the presentation. You may check this with the slides for convinence.