Homework 5 - Camera

陈曦 16340036

1.投影(Projection):

- **把上次作业绘制的cube放置在(-1.5, 0.5, -1.5)位置，要求6个面颜色不一致**
    - **顶点着色器**

```
//GLSL顶点着色器
const char *vertexShaderSource = "#version 330 core\n"
"layout(location = 0) in vec3 aPos;\n"
"layout (location = 1) in vec3 aColor;\n"
"out vec3 outColor;\n"
"uniform mat4 model;\n"
"uniform mat4 view;\n"
"uniform mat4 projection;\n"
"void main()\n"
"{\n"
"   gl_Position = projection * view * model * vec4(aPos, 1.0);\n"
"   outColor = aColor;\n"
"}\0";
```

    - **片段着色器**

```
//片段着色器
const char *fragmentShaderSource = "#version 330 core\n"
"out vec4 FragColor;\n"
"in vec3 outColor;\n"
"void main()\n"
"{\n"
"   FragColor = vec4(outColor, 1.0);\n"
"}\n\0";
```

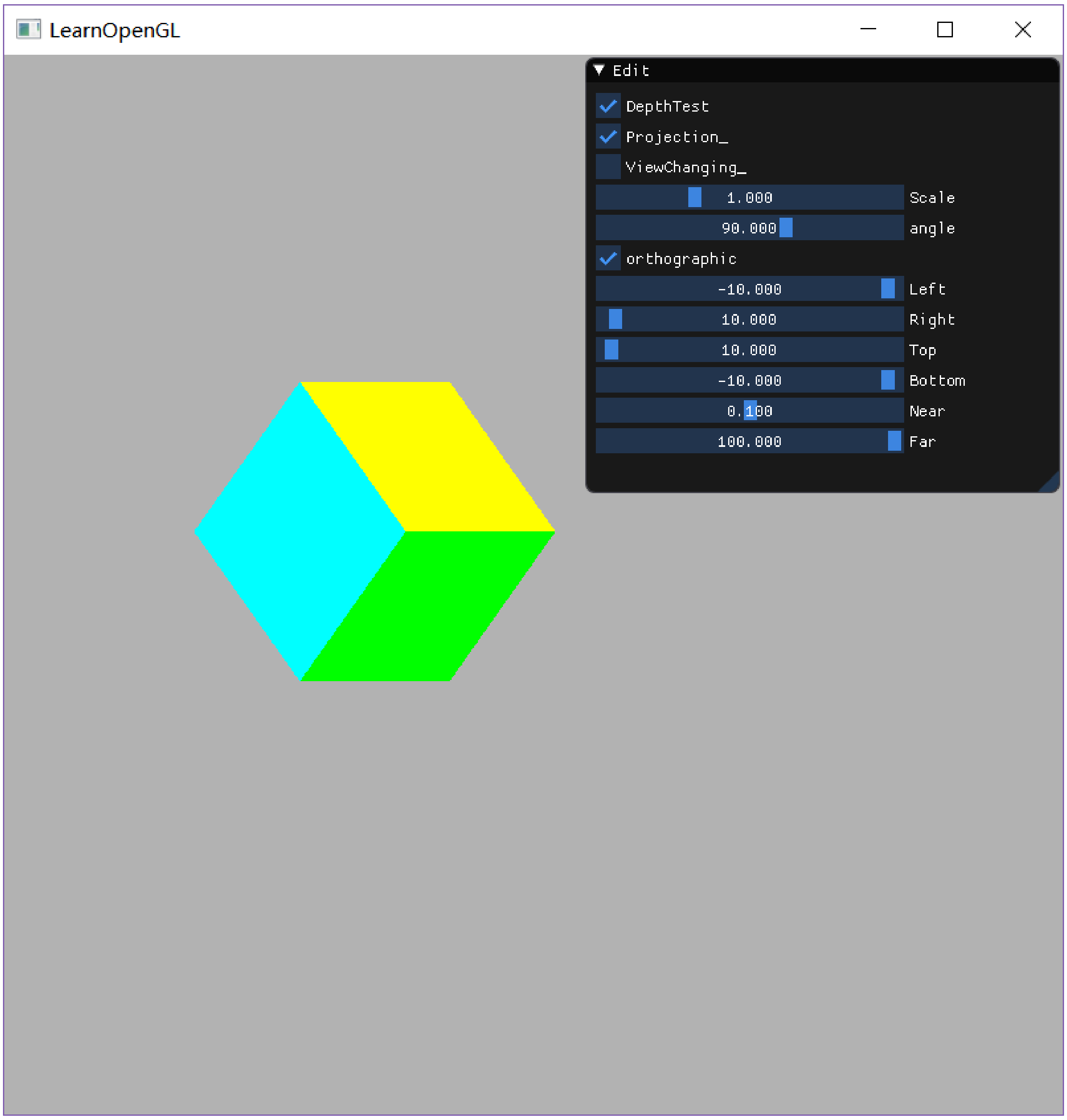
绘制6*2个三角形最终拼接成一个立方体，并将每一面填充一个颜色

```
    //多个三角形最终拼接成一个立方体 (6*2)
    float vertices[] = {
       -2.0f, -2.0f, -2.0f, 1.0f, 0.0f, 0.0f, //triangle 1 : begin
        2.0f, -2.0f, -2.0f, 1.0f, 0.0f, 0.0f,
        2.0f,  2.0f, -2.0f, 1.0f, 0.0f, 0.0f, //triangle 1 : end
        2.0f,  2.0f, -2.0f, 1.0f, 0.0f, 0.0f, // triangle 2 : begin
       -2.0f,  2.0f, -2.0f, 1.0f, 0.0f, 0.0f,
       -2.0f, -2.0f, -2.0f, 1.0f, 0.0f, 0.0f, // triangle 2 : end

       -2.0f, -2.0f,  2.0f, 0.0f, 1.0f, 0.0f,
        2.0f, -2.0f,  2.0f, 0.0f, 1.0f, 0.0f,
        2.0f,  2.0f,  2.0f, 0.0f, 1.0f, 0.0f,
        2.0f,  2.0f,  2.0f, 0.0f, 1.0f, 0.0f,
       -2.0f,  2.0f,  2.0f, 0.0f, 1.0f, 0.0f,
       -2.0f, -2.0f,  2.0f, 0.0f, 1.0f, 0.0f,

       -2.0f,  2.0f,  2.0f, 0.0f, 0.0f, 1.0f,
       -2.0f,  2.0f, -2.0f, 0.0f, 0.0f, 1.0f,
       -2.0f, -2.0f, -2.0f, 0.0f, 0.0f, 1.0f,
        -2.0f, -2.0f, -2.0f, 0.0f, 0.0f, 1.0f,
        -2.0f, -2.0f,  2.0f, 0.0f, 0.0f, 1.0f,
        -2.0f,  2.0f,  2.0f, 0.0f, 0.0f, 1.0f,

         2.0f,  2.0f,  2.0f, 1.0f, 1.0f, 0.0f,
         2.0f,  2.0f, -2.0f, 1.0f, 1.0f, 0.0f,
         2.0f, -2.0f, -2.0f, 1.0f, 1.0f, 0.0f,
         2.0f, -2.0f, -2.0f, 1.0f, 1.0f, 0.0f,
         2.0f, -2.0f,  2.0f, 1.0f, 1.0f, 0.0f,
         2.0f,  2.0f,  2.0f, 1.0f, 1.0f, 0.0f,

        -2.0f, -2.0f, -2.0f, 1.0f, 0.0f, 1.0f,
         2.0f, -2.0f, -2.0f, 1.0f, 0.0f, 1.0f,
         2.0f, -2.0f,  2.0f, 1.0f, 0.0f, 1.0f,
         2.0f, -2.0f,  2.0f, 1.0f, 0.0f, 1.0f,
        -2.0f, -2.0f,  2.0f, 1.0f, 0.0f, 1.0f,
        -2.0f, -2.0f, -2.0f, 1.0f, 0.0f, 1.0f,

        -2.0f,  2.0f, -2.0f, 0.0f, 1.0f, 1.0f,
         2.0f,  2.0f, -2.0f, 0.0f, 1.0f, 1.0f,
         2.0f,  2.0f,  2.0f, 0.0f, 1.0f, 1.0f,
         2.0f,  2.0f,  2.0f, 0.0f, 1.0f, 1.0f,
        -2.0f,  2.0f,  2.0f, 0.0f, 1.0f, 1.0f,
        -2.0f,  2.0f, -2.0f, 0.0f, 1.0f, 1.0f
    };
```

放置在(-1.5, 0.5, -1.5)位置

```
model = translate(model, vec3(-1.5f, 0.5f, -1.5f));
```

截图如下:


LearnOpenGL

Edit

✓ DepthTest

✓ Projection_

ViewChanging_

1.000	Scale
90.000	angle

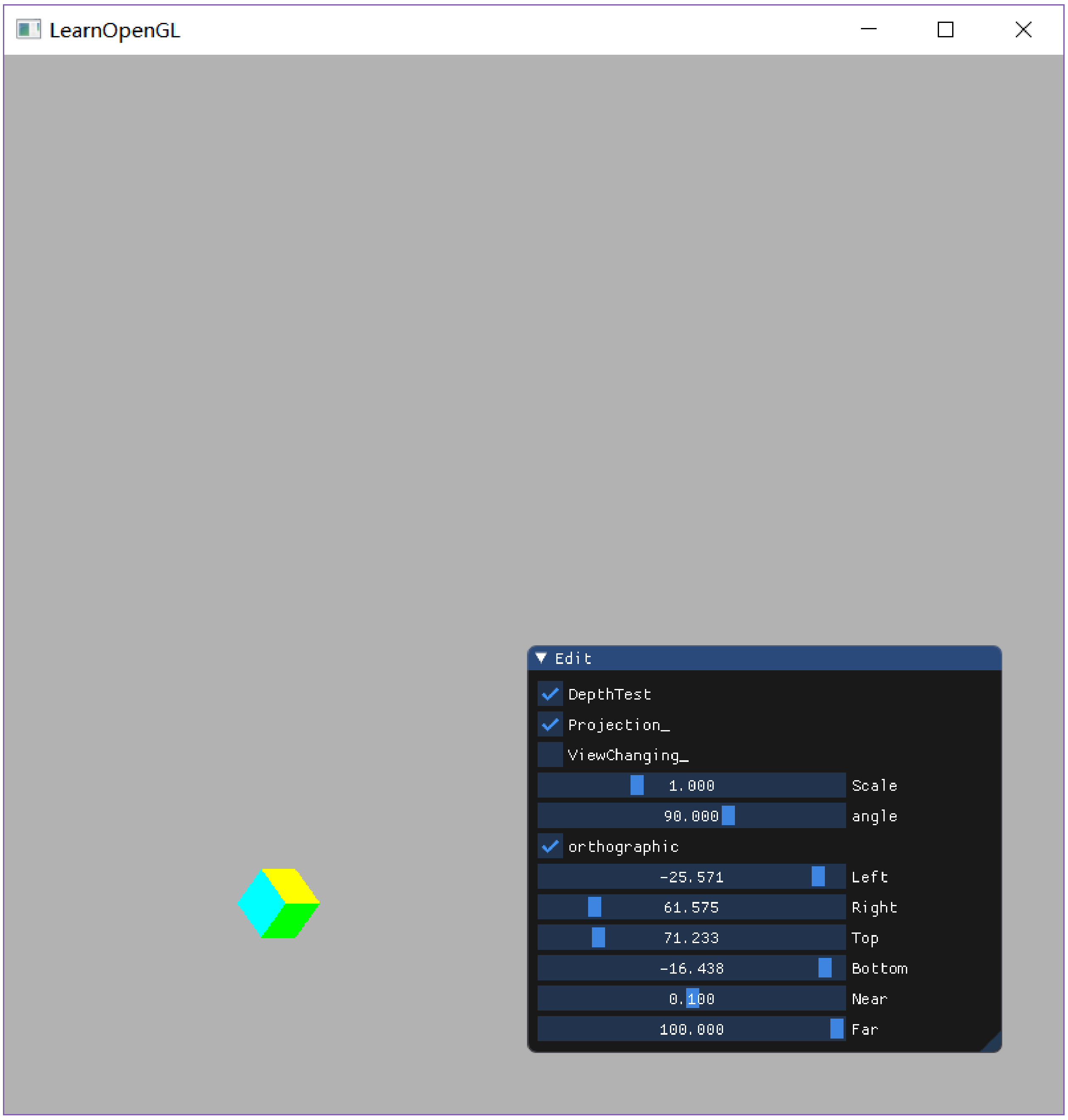
✓ orthographic

-10.000	Left
10.000	Right
10.000	Top
-10.000	Bottom
0.100	Near
100.000	Far

- **正交投影(orthographic projection)：实现正交投影，使用多组(left, right, bottom, top, near, far)参数，比较结果差异**

```
projection = ortho(left, right, bottom, top, nearPos, farPos);
```

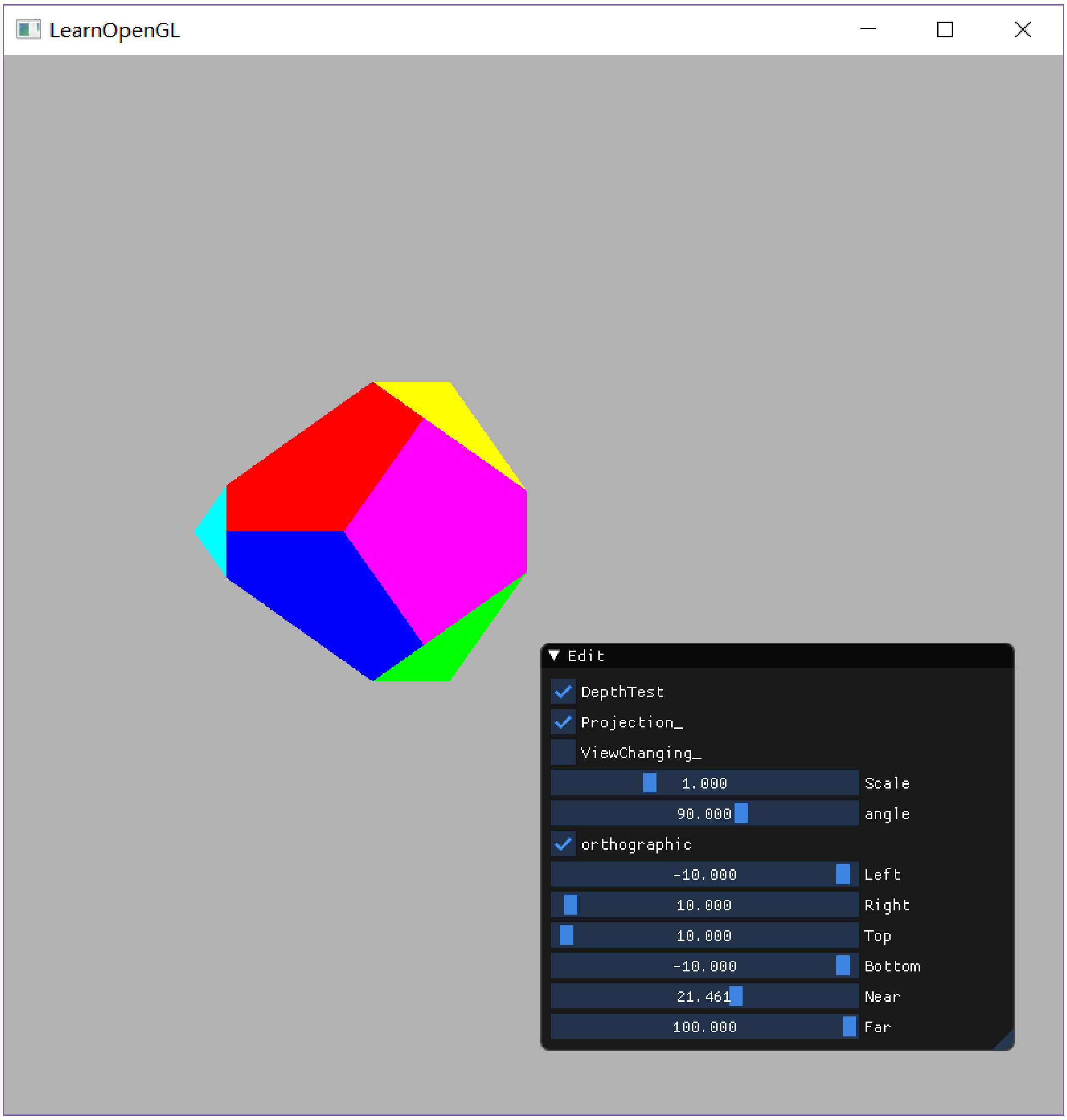
前两个参数指定了平截头体的左右坐标，第三和第四参数指定了平截头体的底部和顶部，通过这四个参数定义了近平面和远平面的大小，调节这四个参数，当投影范围变大时立方体会变小，截图如下：



DepthTest ✓
Projection_ ✓
ViewChanging_
1.000 Scale
90.000 angle
orthographic ✓
-25.571 Left
61.575 Right
71.233 Top
-16.438 Bottom
0.100 Near
100.000 Far

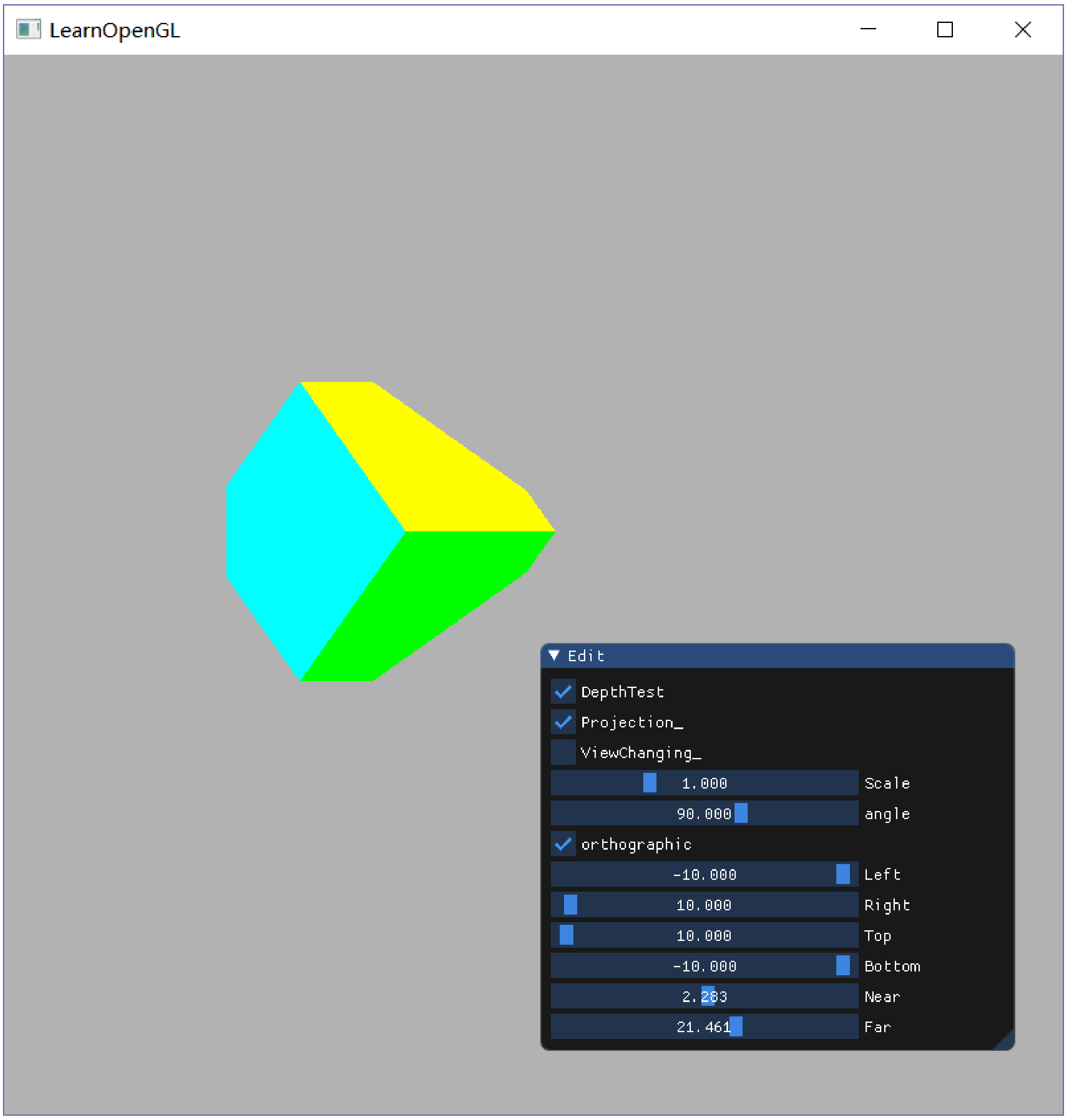
第五和第六个参数则定义了近平面和远平面的距离，调节这两个参数截图如下：

- **改变Near**



DepthTest ✓
Projection_ ✓
ViewChanging_
1.000 Scale
90.000 angle
orthographic ✓
-10.000 Left
10.000 Right
10.000 Top
-10.000 Bottom
21.461 Near
100.000 Far

- **改变Far**



Edit
- [✓] DepthTest
- [✓] Projection_
- [ ] ViewChanging_

1.000	Scale
90.000	angle

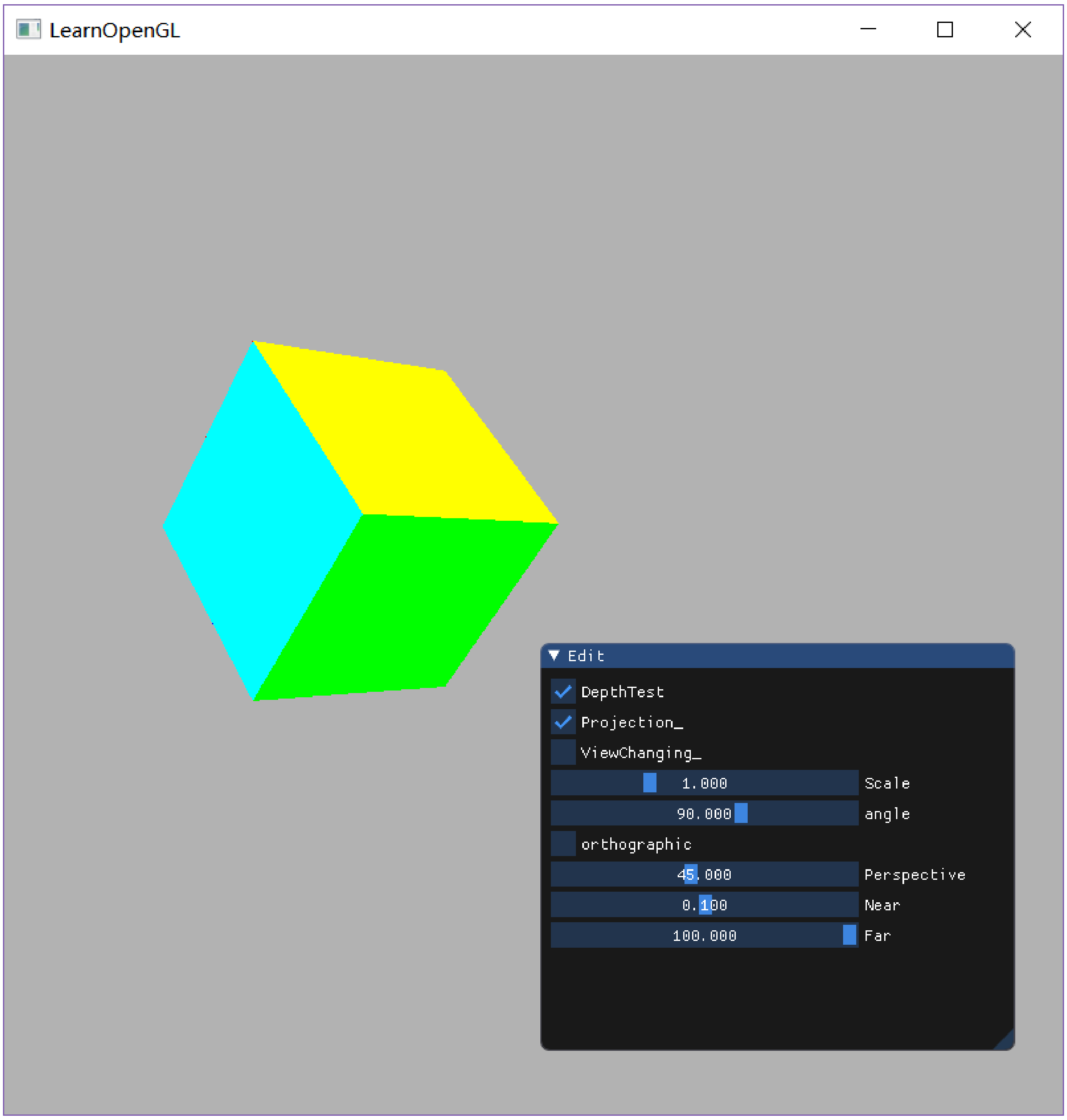
- [✓] orthographic

-10.000	Left
10.000	Right
10.000	Top
-10.000	Bottom
2.283	Near
21.461	Far

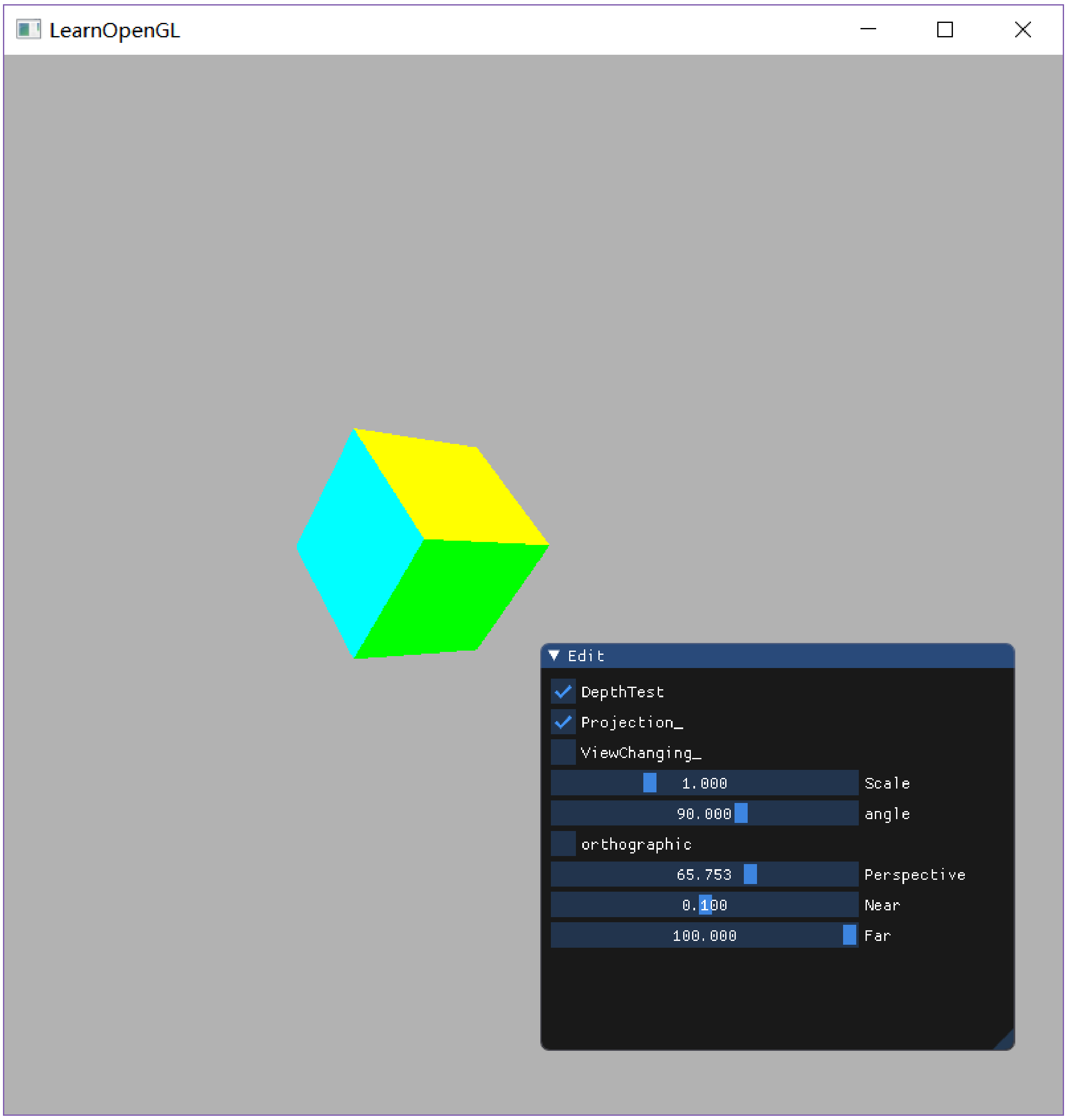
- **透视投影(perspective projection)：实现透视投影，使用多组参数，比较结果差异**

```
projection = glm::perspective(radians(perspective), 800.0f / 800.0f, near_, far_);
```

第一个参数定义了fov的值，它表示的是视野(Field of View)，并且设置了观察空间的大小。它的值通常设置为45.0f，观察效果更加真实。第二个参数设置了宽高比，由视口的宽除以高所得。第三和第四个参数设置了平截头体的近和远平面。我们通常设置近距离为0.1f，而远距离设为100.0f。所有在近平面和远平面内且处于平截头体内的顶点都会被渲染。改变第一个参数的值，可观察到fov角越大立方体越小，截图如下：



DepthTest ✓
Projection_ ✓
ViewChanging_
1.000 Scale
90.000 angle
orthographic
45.000 Perspective
0.100 Near
100.000 Far



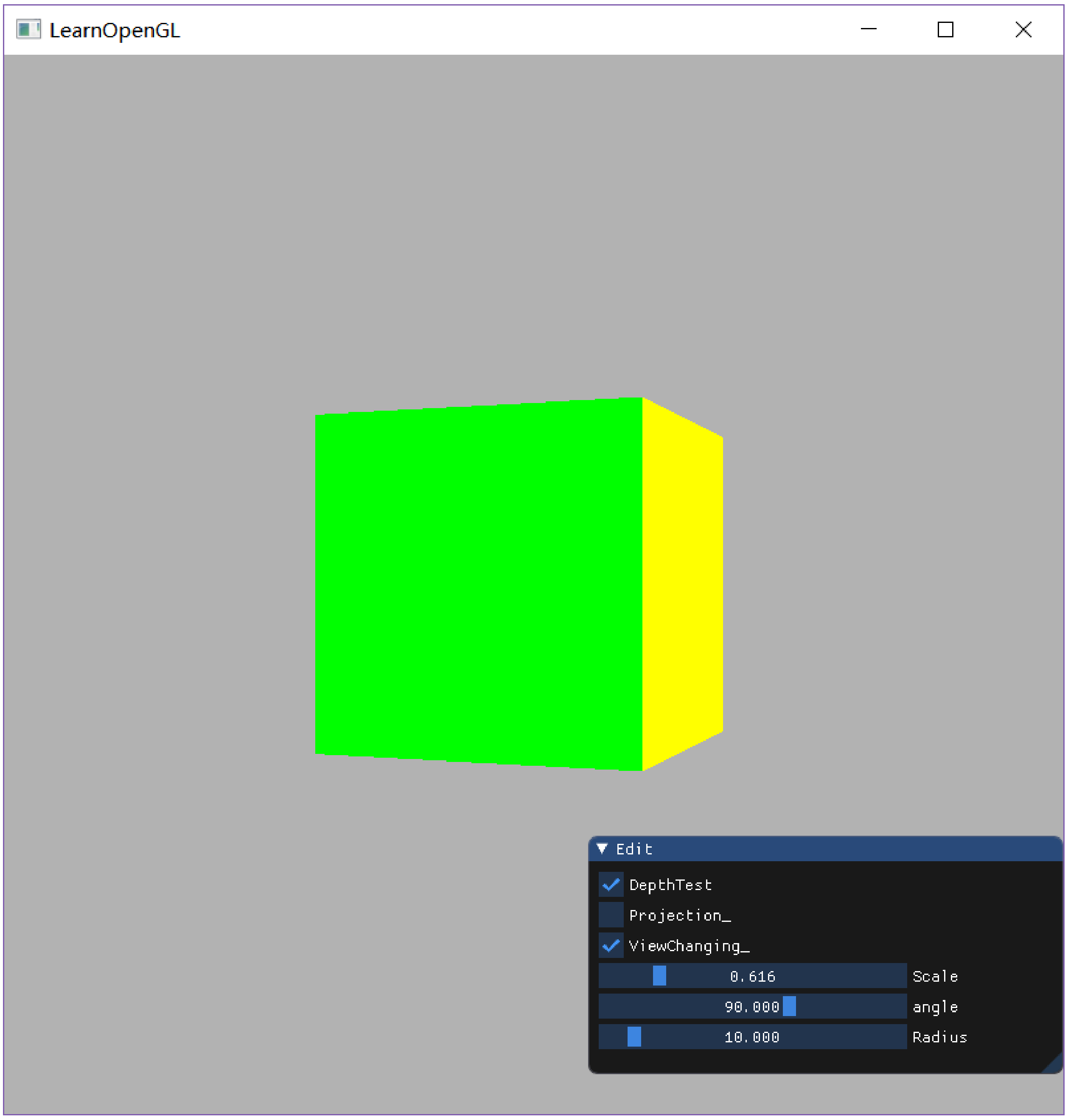
DepthTest ✓
Projection_ ✓
ViewChanging_
1.000 Scale
90.000 angle
orthographic
65.753 Perspective
0.100 Near
100.000 Far

2.视角变换(View Changing):

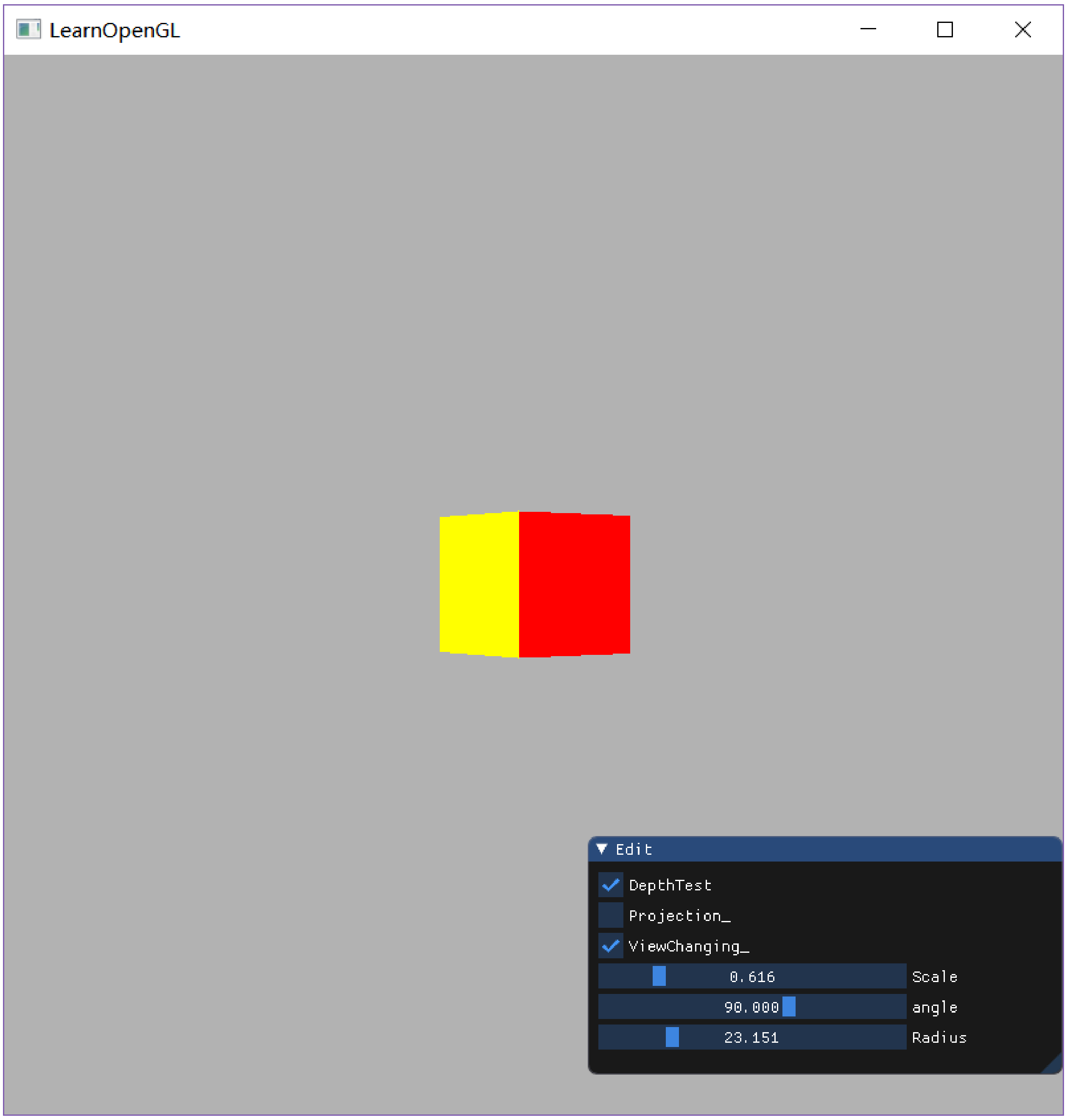
- **把cube放置在(0, 0, 0)处，做透视投影，使摄像机围绕cube旋转，并且时刻看着cube中心**

  对第一个参数（摄像机的位置）进行修改。利用三角函数的相关知识来给每一帧创建一个x和z坐标，这个坐标代表所期望的摄像机圆形轨道上的一点，即当前帧对应的摄像机位置。

```
model = translate(model, vec3(0.0f, 0.0f, 0.0f));
model = glm::scale(model, vec3(scale, scale, scale));

float cameraX = sin(glfwGetTime()) * radius;
float cameraZ = cos(glfwGetTime()) * radius;
view = lookAt(vec3(cameraX, 0.0f, cameraZ),
    vec3(0.0f, 0.0f, 0.0f),
    vec3(0.0f, 1.0f, 0.0f));
projection = glm::perspective(radians(45.0f), 800.0f / 800.0f, 0.1f, 100.0f);
```



DepthTest ✓
Projection_
ViewChanging_ ✓
0.616 Scale
90.000 angle
10.000 Radius

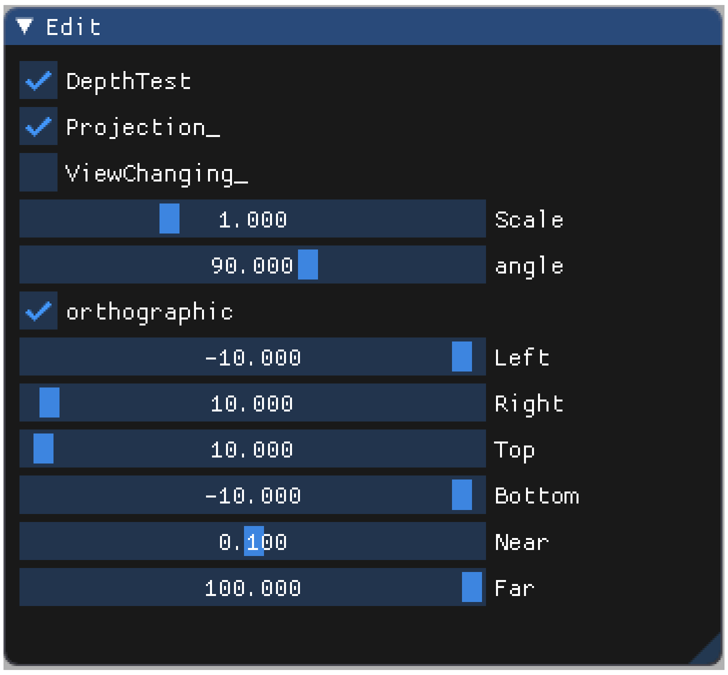


3.在GUI里添加菜单栏，可以选择各种功能。 Hint: 使摄像机一直处于一个圆的位置

```
Begin("Edit");
Checkbox("DepthTest", &depth);
Checkbox("Projection_", &Projection_);
Checkbox("ViewChanging_", &ViewChanging_);
SliderFloat("Scale", &scale, 0.1f, 3.0f);
SliderFloat("angle", &angle, -360.0f, 360.0f);

if (ViewChanging_) {
    SliderFloat("Radius", &radius, 1.0f, 100.0f);
}
else {
    Checkbox("orthographic", &orthographic);

    if (orthographic) {

        SliderFloat("Left", &left, -400.0f, 0.0f);
```

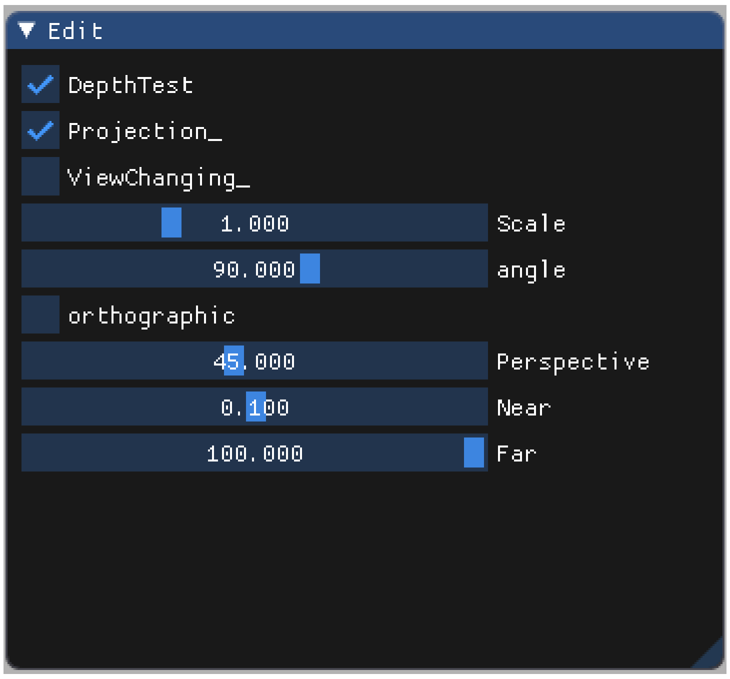
```
            SliderFloat("Right", &right, -5.0f, 400.0f);
            SliderFloat("Top", &top, 0.0f, 400.0f);
            SliderFloat("Bottom", &bottom, -400.0f, 0.0f);
            SliderFloat("Near", &near_, -100.0f, 100.0f);
            SliderFloat("Far", &far_, -100.0f, 100.0f);
        }
        else {
            SliderFloat("Perspective", &perspective, 0.0f, 100.0f);
            SliderFloat("Near", &near_, -100.0f, 100.0f);
            SliderFloat("Far", &far_, -100.0f, 100.0f);
        }
    }

    End();
```

- **orthographic projection:**



```
▼ Edit
[✓] DepthTest
[✓] Projection_
[ ] ViewChanging_
1.000      Scale
90.000     angle
[✓] orthographic
-10.000    Left
10.000     Right
10.000     Top
-10.000    Bottom
0.100      Near
100.000    Far
```

- **perspective projection:**


Edit

✓ DepthTest

✓ Projection_

ViewChanging_

1.000 Scale

90.000 angle

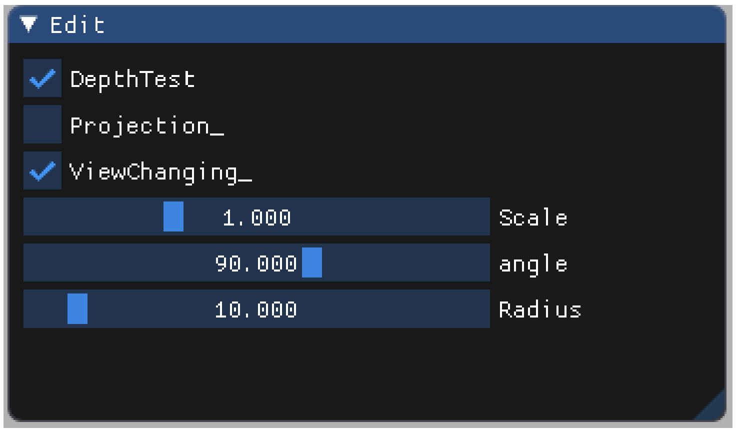
orthographic

45.000 Perspective

0.100 Near

100.000 Far

- **View Changing:**


Edit

✓ DepthTest

Projection_

✓ ViewChanging_

1.000 Scale

90.000 angle

10.000 Radius

4.在现实生活中，我们一般将摄像机摆放的空间View matrix和被拍摄的物体摆设的空间Model matrix分开，但是在OpenGL中却将两个合二为一设为ModelView matrix，通过上面的作业启发，你认为是为什么呢？在报告中写入。（Hints：你可能有不止一个摄像机）

坐标变换矩阵栈(ModelView)：用来存储一系列的变换矩阵，栈顶就是当前坐标的变换矩阵，进入OpenGL管道的每个坐标(齐次坐标)都会先乘上这个矩阵，结果才是对应点在场景中的世界坐标。

OpenGL中没有单独的摄像机矩阵，为了模拟摄像机的变换，要将场景中所有的物体向相反的方向移动，ModelView是一个单独的矩阵堆栈，节省空间，使得计算量减少，并且一个场景中可能有不止一台摄像机，多个视角的结合也不会造成视觉混乱。