作业5实验报告

实验要求

Basic:

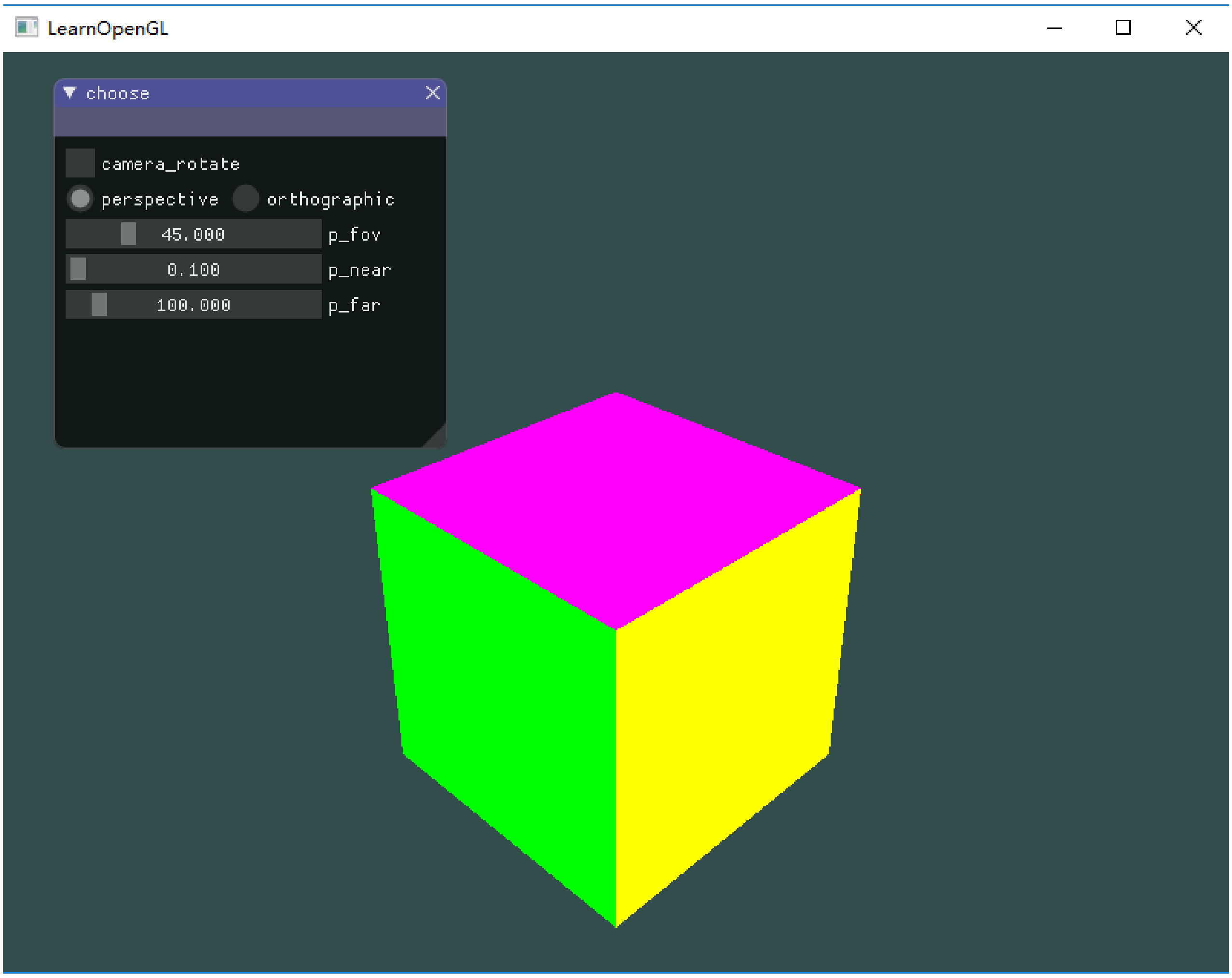
1. 投影(Projection): 把上次作业绘制的cube放置在(-1.5, 0.5, -1.5)位置，要求6个面颜色不一致 正交投影(orthographic projection)：实现正交投影，使用多组(left, right, bottom, top, near, far)参数， 比较结果差异 透视投影(perspective projection)：实现透视投影，使用多组参数，比较结果差异
2. 视角变换(View Changing): 把cube放置在(0, 0, 0)处，做透视投影，使摄像机围绕cube旋转，并且时刻看着cube中心
3. 在GUI里添加菜单栏，可以选择各种功能。 Hint: 使摄像机一直处于一个圆的位置，可以参考以下公式： 原理很容易理解，由于圆的公式 a^2+b^2=1 ，以及有 sin(x)^2+cos(x)^2=1 ，所以能保证摄像机在XoZ平面的 一个圆上。
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）

Bonus:

1. 实现一个camera类，当键盘输入 w,a,s,d ，能够前后左右移动；当移动鼠标，能够视角移动("look around")，即类似FPS(First Person Shooting)的游戏场景

实验截图

1. 投影，cube放在(-1.5, 0.5, -1.5)位置，6个面颜色不一致



LearnOpenGL

choose

□ camera_rotate

● perspective ○ orthographic

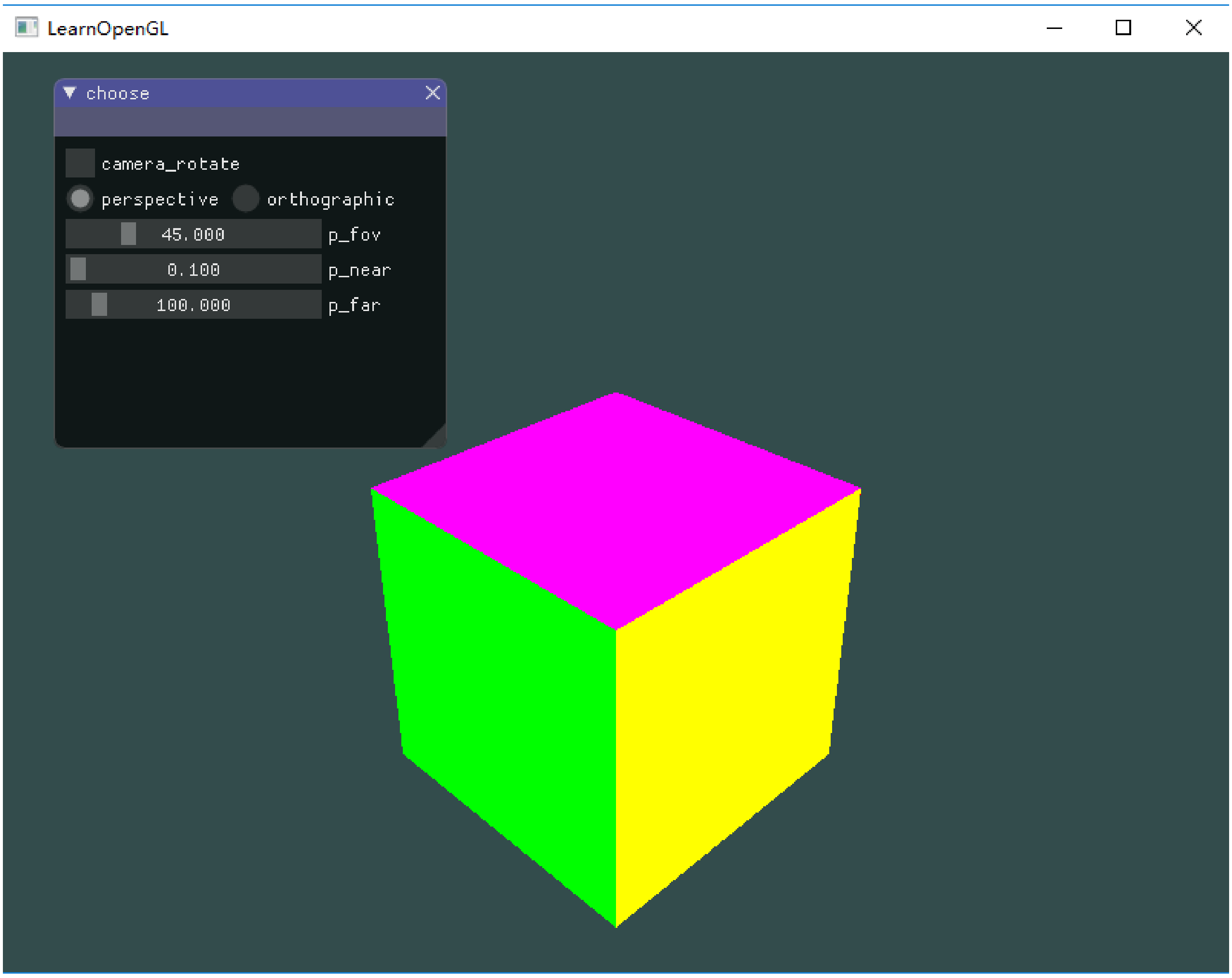
45.000 p_fov

0.100 p_near

100.000 p_far

2. 透视投影，使用多组参数

   初始状态：



LearnOpenGL

choose

□ camera_rotate

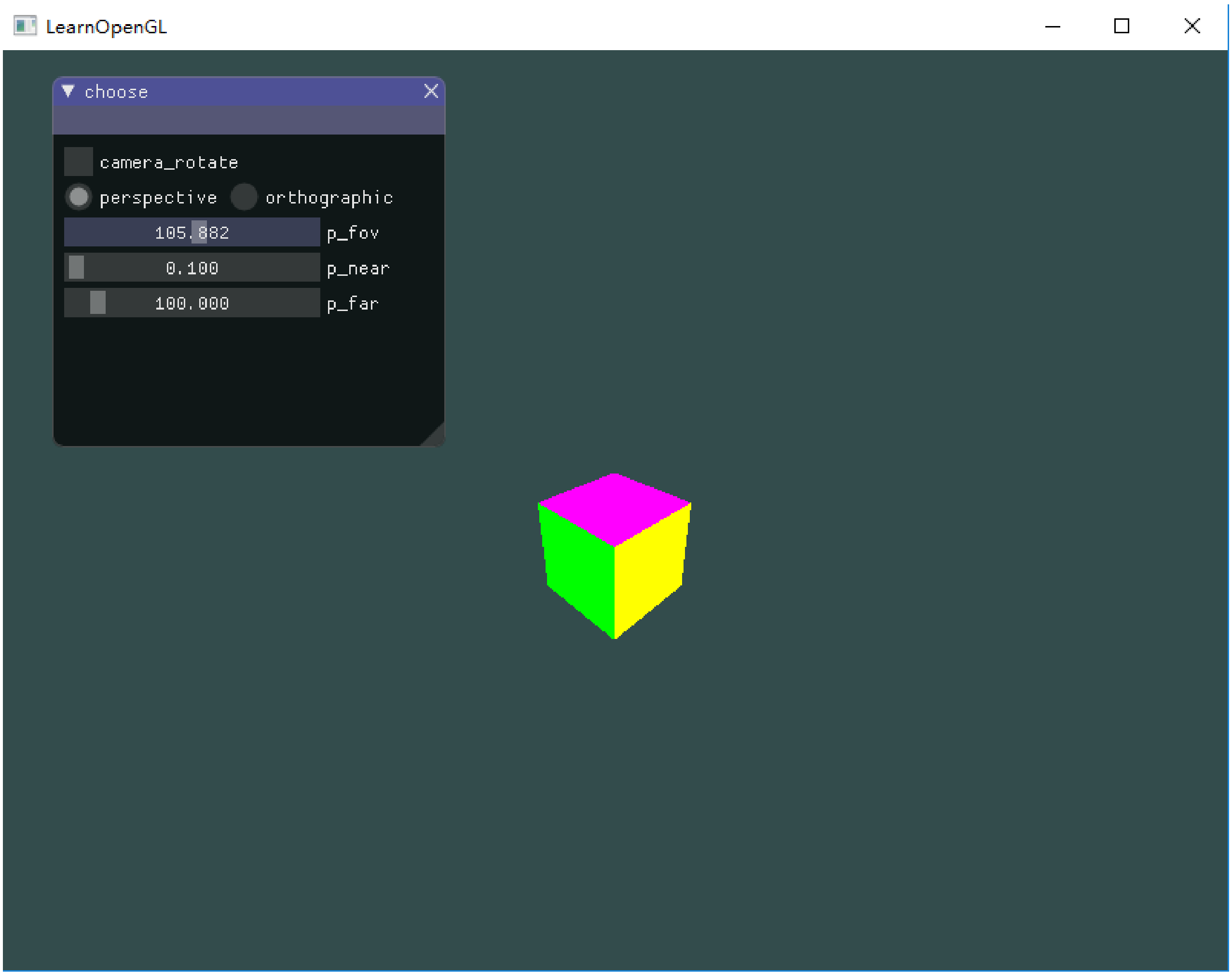
● perspective ○ orthographic

45.000 p_fov

0.100 p_near

100.000 p_far

增大p_fov，fov值增大，平面面积增大，相比下使得cube变小：



□ camera_rotate

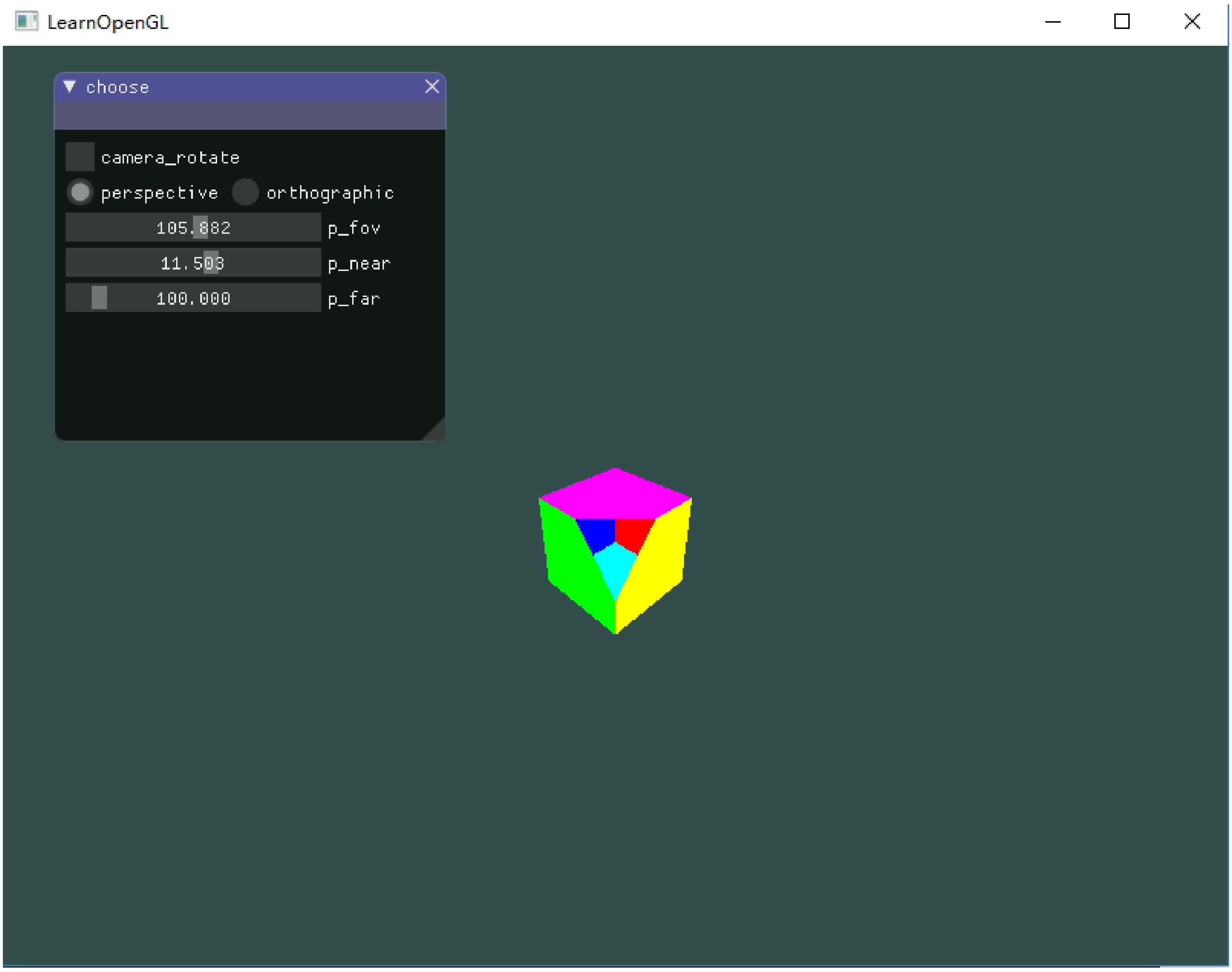
● perspective ○ orthographic

105.882 p_fov

0.100 p_near

100.000 p_far

增大p_near，near增大，near面往后移至超过cube一小部分，所以显示出cube被截掉一个角:



□ camera_rotate

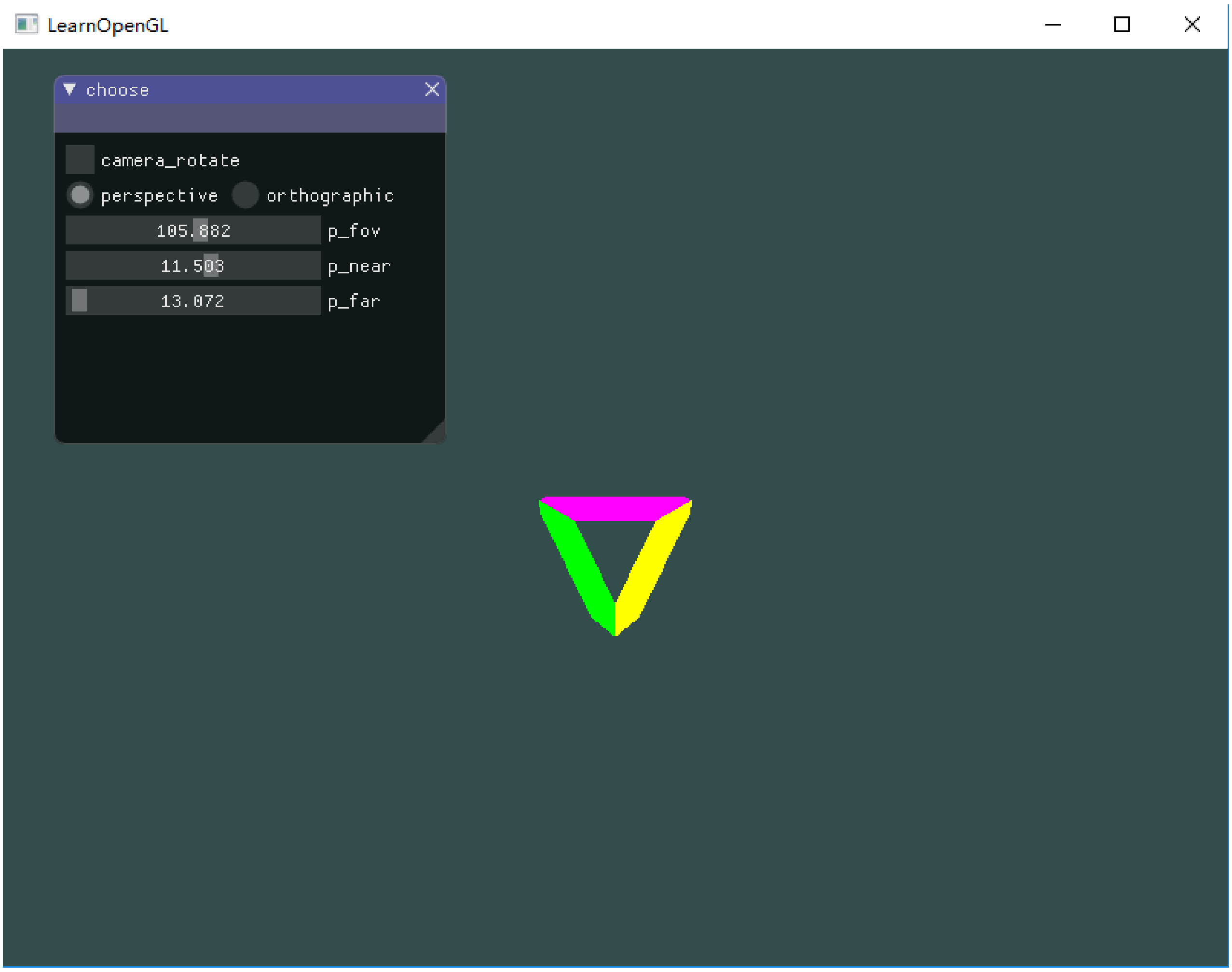
● perspective ○ orthographic

105.882 p_fov

11.503 p_near

100.000 p_far

减小P_far， far减小，远平面往前移至挡住cube一部分，在远平面后的cube不可见，故显示如下:



□ camera_rotate

● perspective ○ orthographic

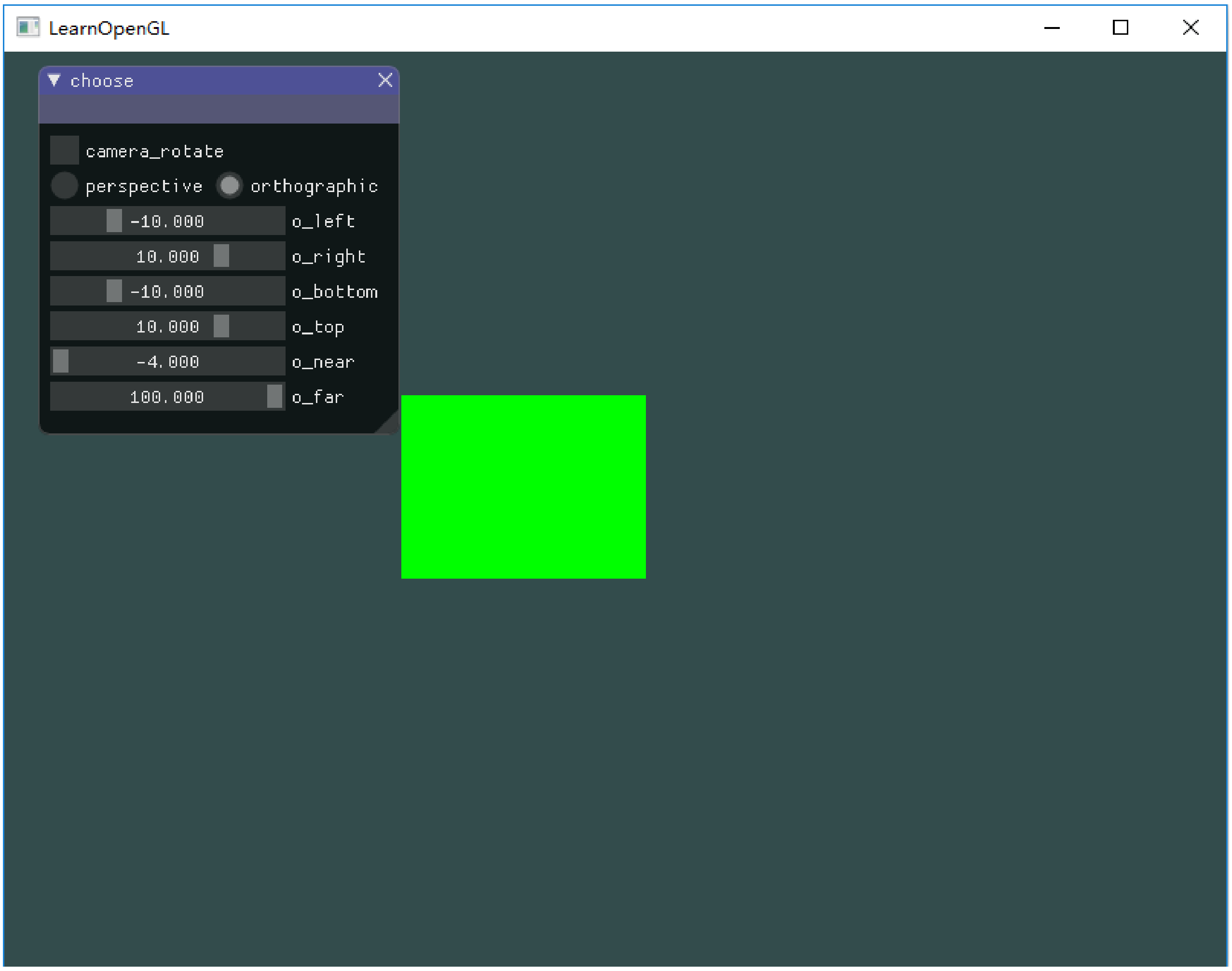
105.882 p_fov

11.503 p_near

13.072 p_far

3. 正交投影，使用多组参数

初始状态：



□ camera_rotate

○ perspective ● orthographic

-10.000 o_left

10.000 o_right

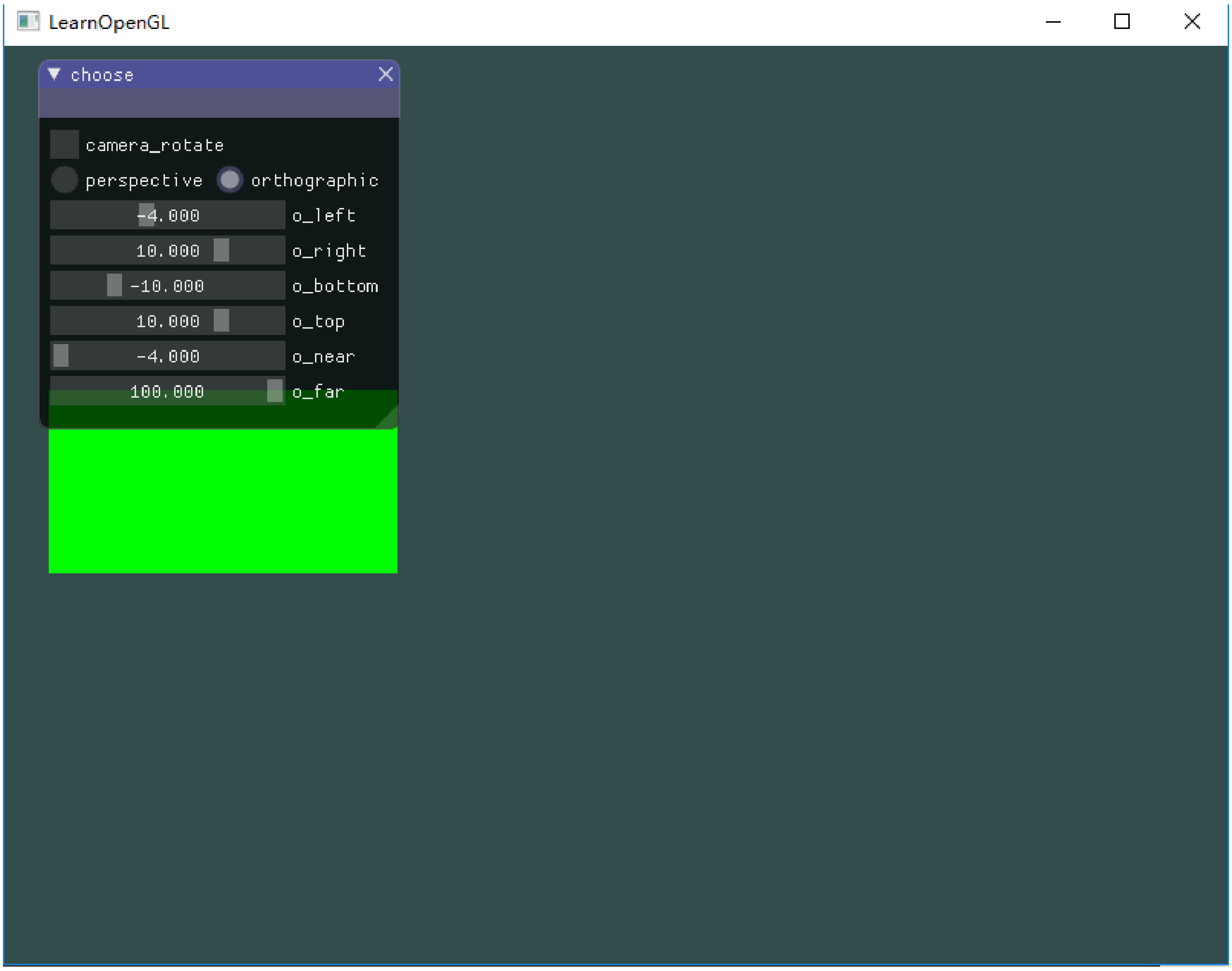
-10.000 o_bottom

10.000 o_top

-4.000 o_near

100.000 o_far

改变o_left，left变小，平面向右缩小，相对的cube向左移动并有所拉伸:

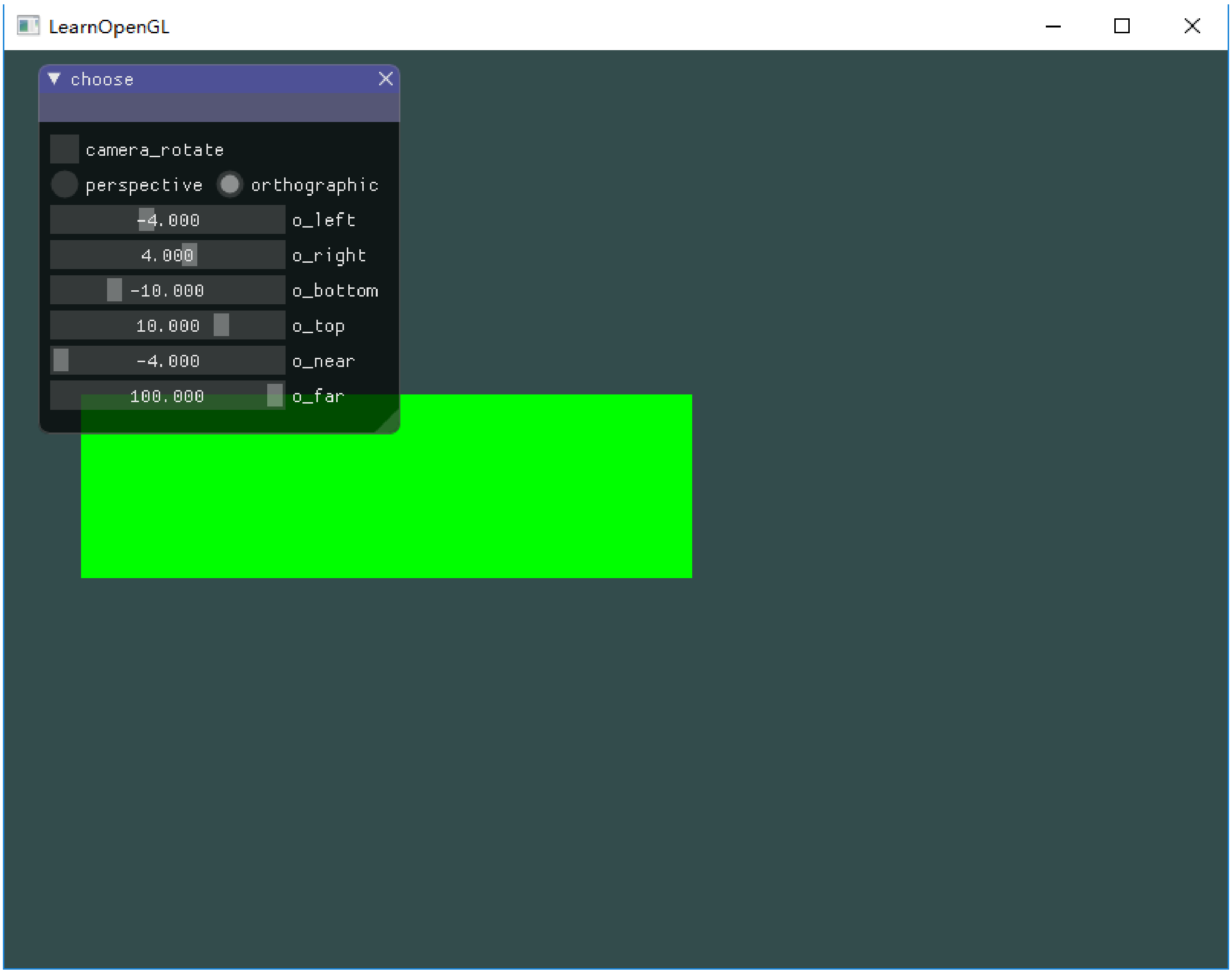


□ camera_rotate

○ perspective ● orthographic

```
-4.000    o_left
10.000    o_right
-10.000   o_bottom
10.000    o_top
-4.000    o_near
100.000   o_far
```

改变o_right，right变小，平面向左缩小，相对的cube向右移动并有所拉伸:



□ camera_rotate

○ perspective ● orthographic

-4.000 o_left

4.000 o_right

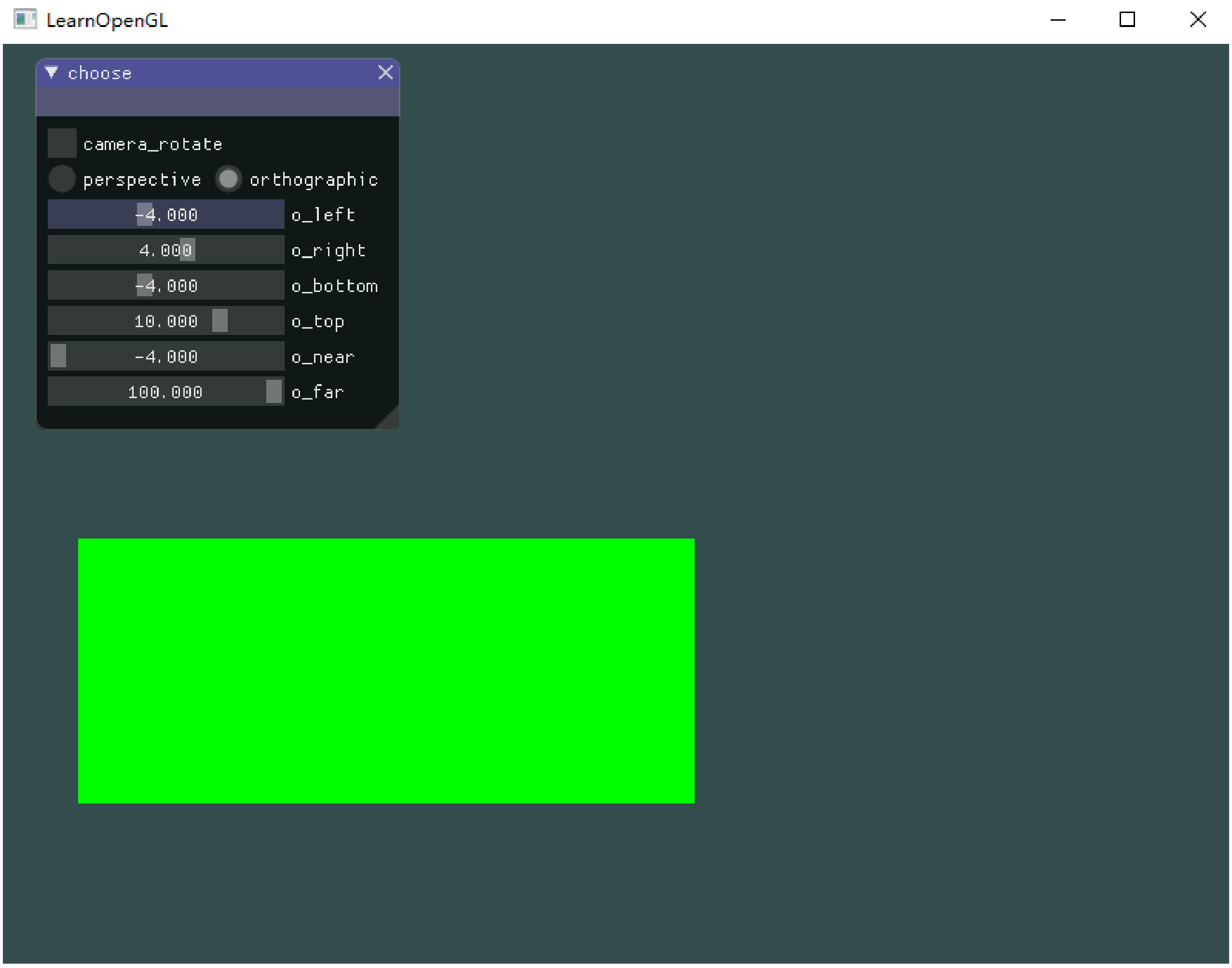
-10.000 o_bottom

10.000 o_top

-4.000 o_near

100.000 o_far

改变o_bottom，bottom变小，平面向上缩小，相对的cube向下移动并有所拉伸:



□ camera_rotate

○ perspective ● orthographic

-4.000 o_left

4.000 o_right

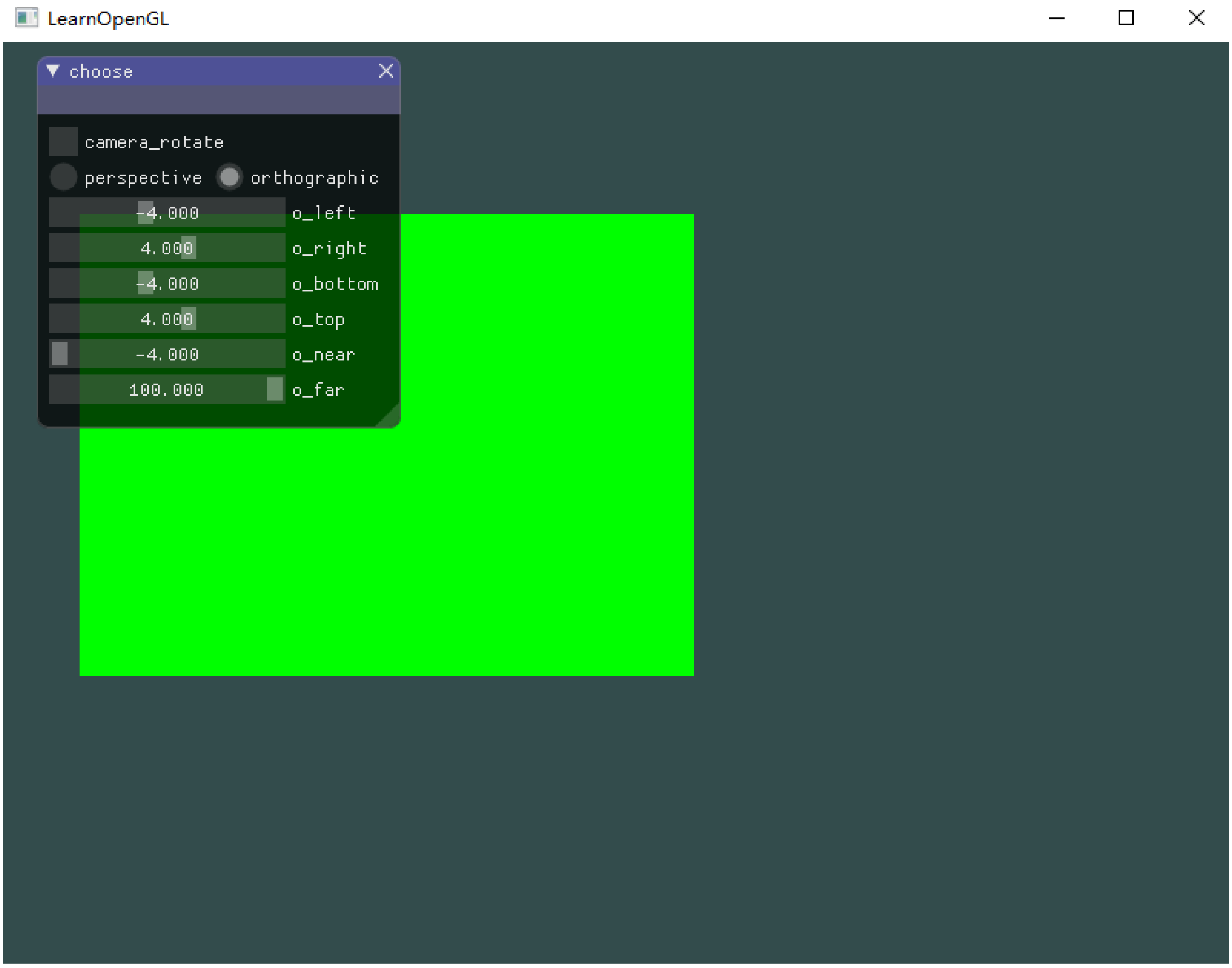
-4.000 o_bottom

10.000 o_top

-4.000 o_near

100.000 o_far

改变o_top，top变小，平面向下缩小，相对的cube向上移动并有所拉伸:



□ camera_rotate

○ perspective ● orthographic

-4.000 o_left

4.000 o_right

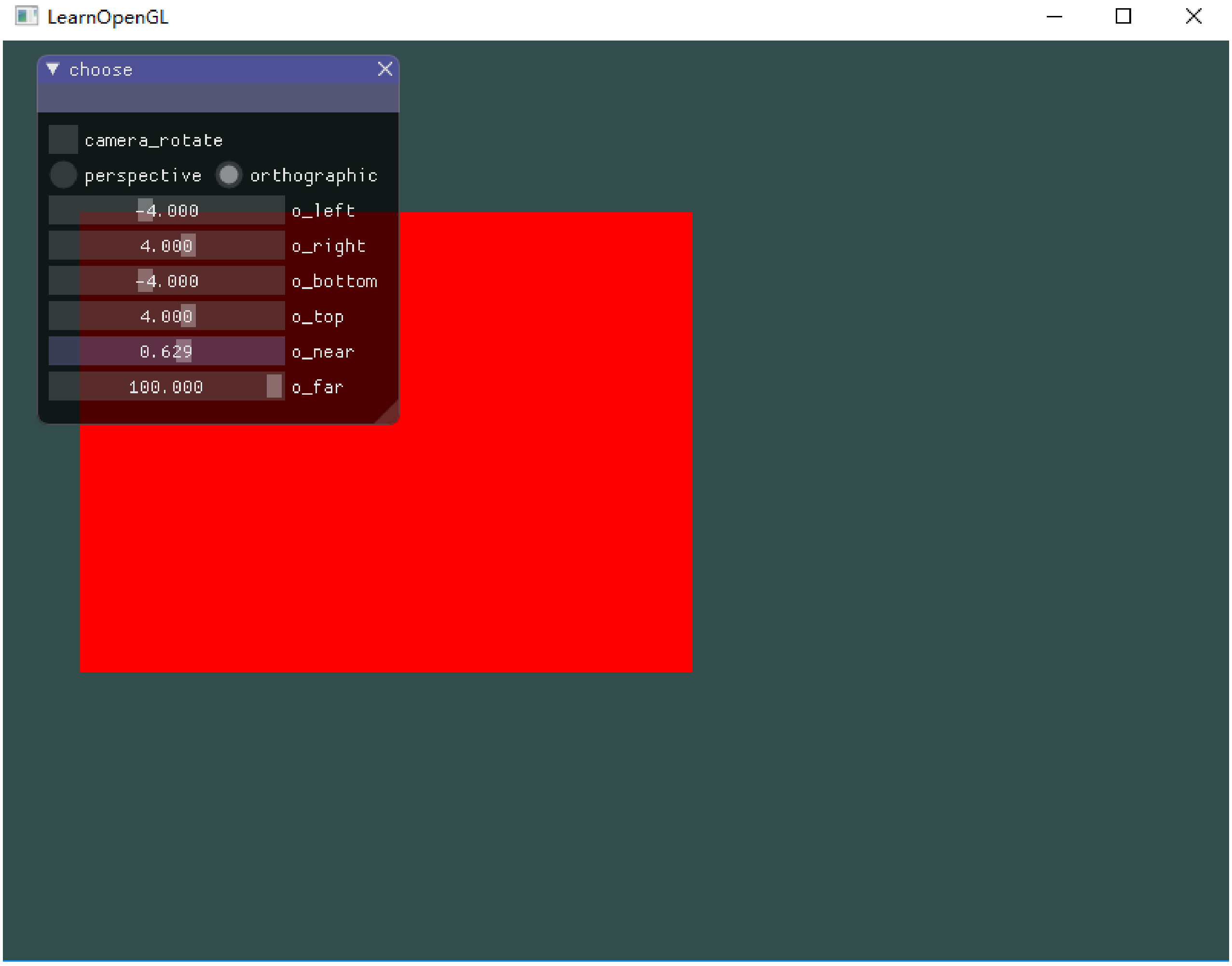
-4.000 o_bottom

4.000 o_top

-4.000 o_near

100.000 o_far

改变o_near，near增大，near平面往后移动，超过cube，故看到了cube的背面:



choose

□ camera_rotate

○ perspective ● orthographic

-4.000 o_left

4.000 o_right

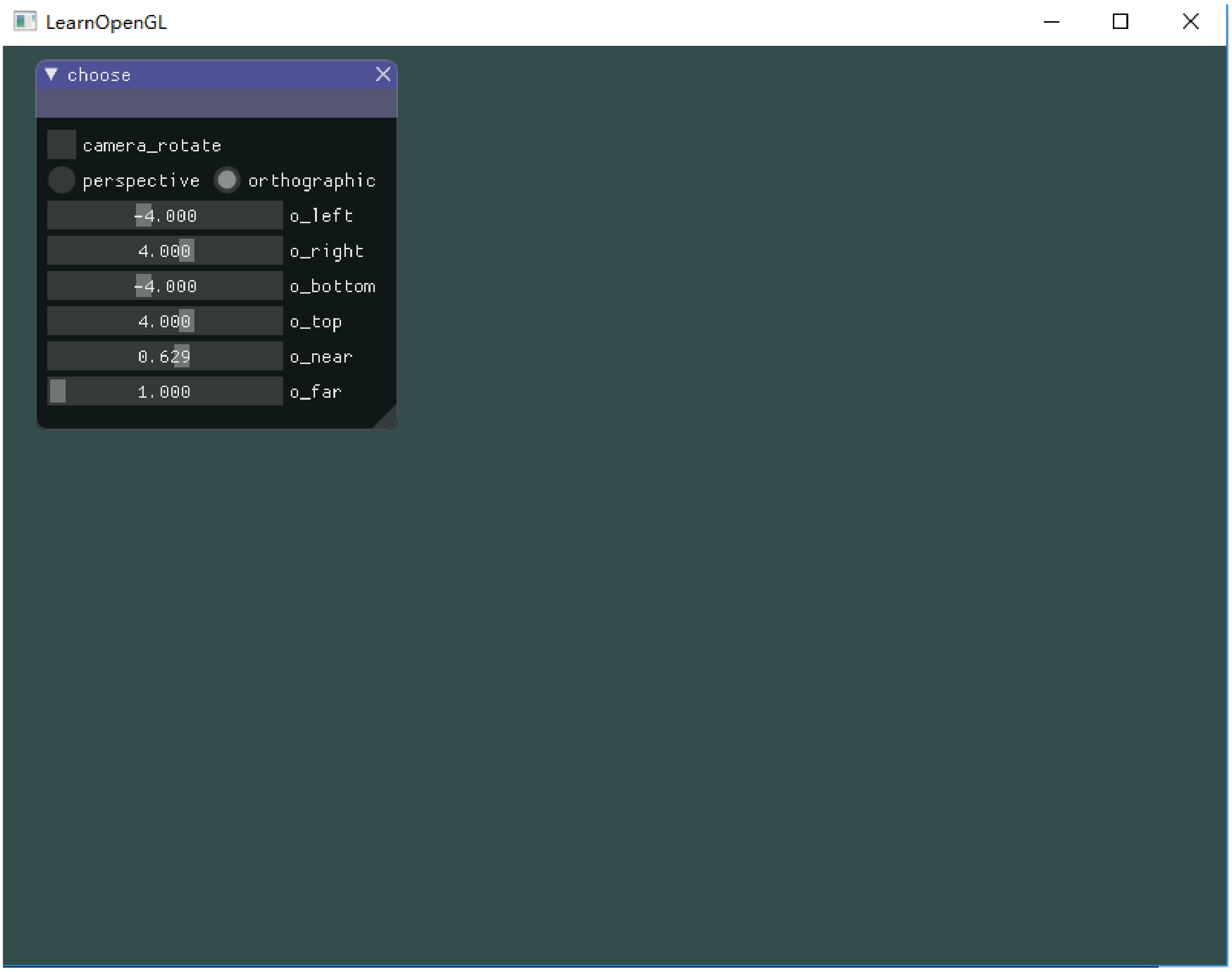
-4.000 o_bottom

4.000 o_top

0.629 o_near

100.000 o_far

改变o_far，far变小，far平面超过cube背面，使得near、far平面都处于cube内部，故看不到东西:



□ camera_rotate

○ perspective ● orthographic

-4.000 o_left

4.000 o_right

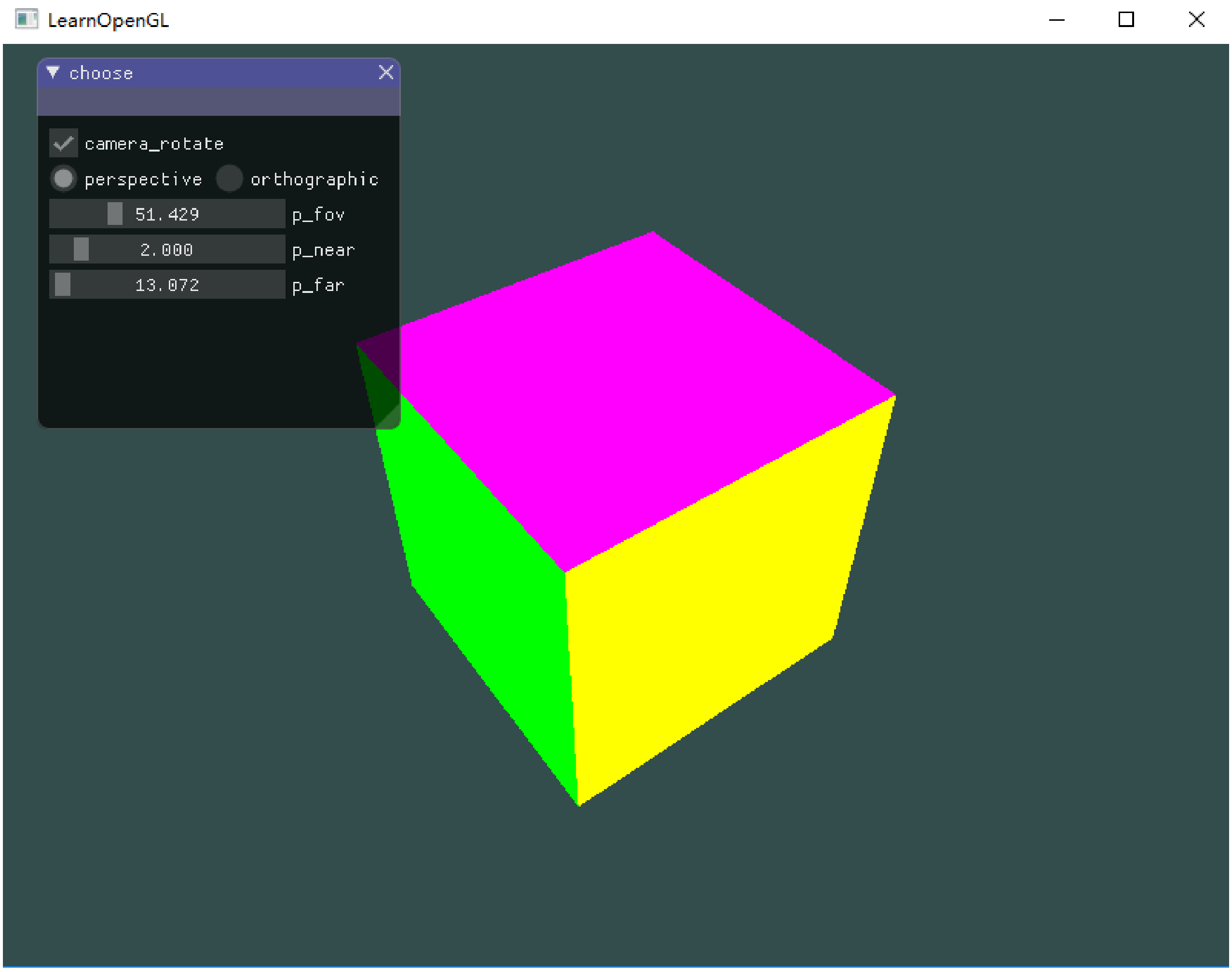
-4.000 o_bottom

4.000 o_top

0.629 o_near

1.000 o_far

4. 视角变换， 摄像机绕cube旋转，并且时刻看着cube中心，具体看gif



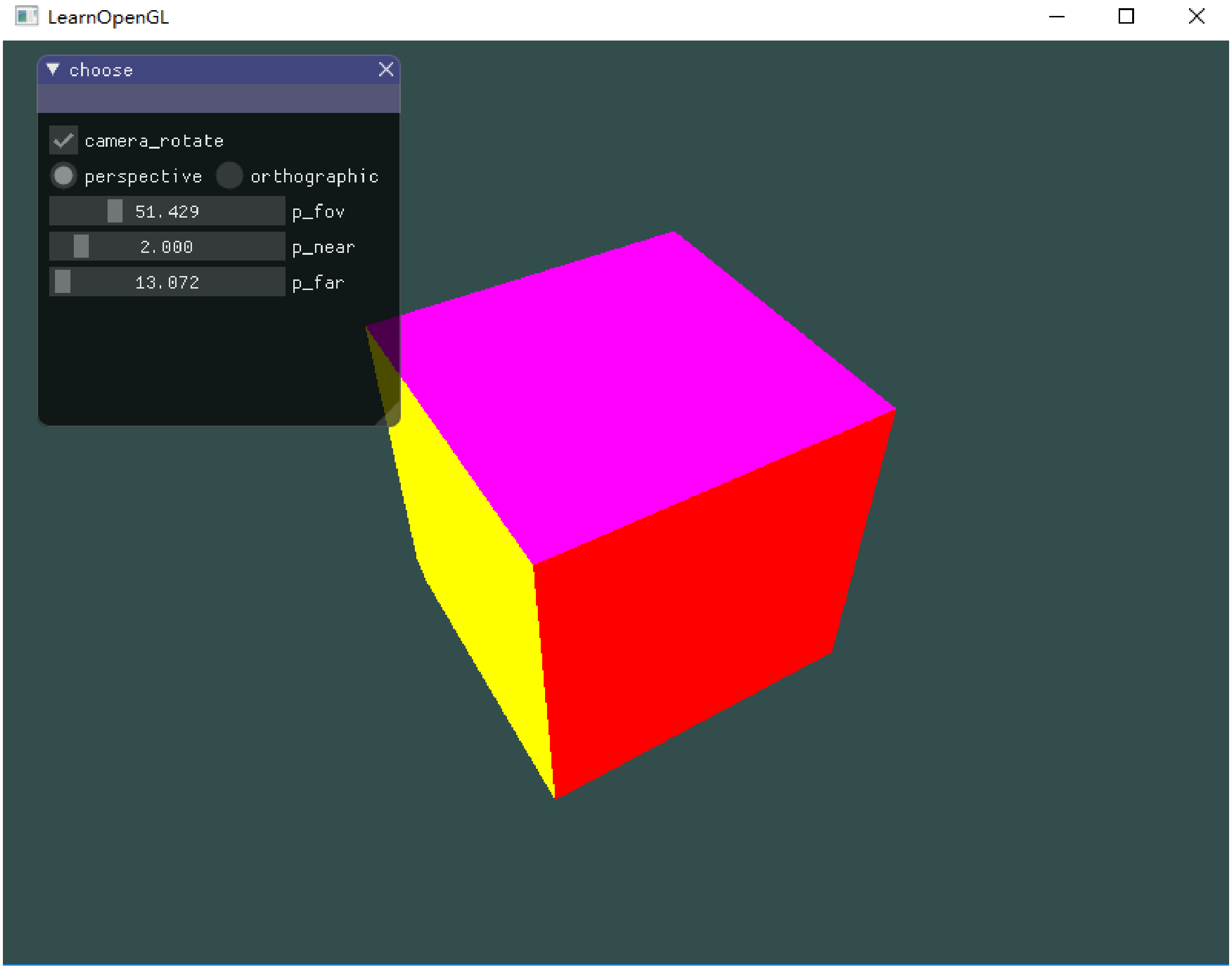
☑ camera_rotate

● perspective ○ orthographic

51.429 p_fov

2.000 p_near

13.072 p_far



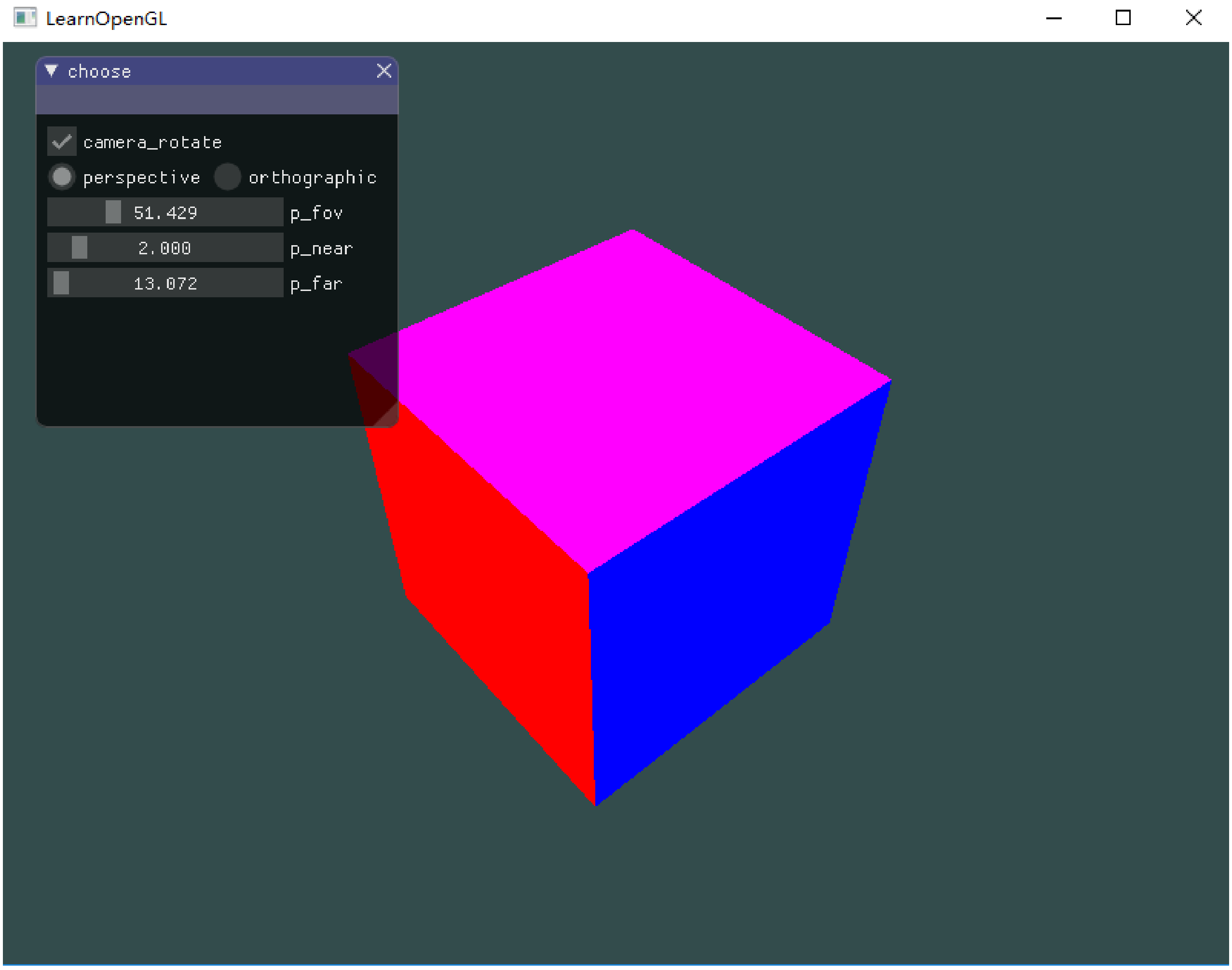
☑ camera_rotate

● perspective ○ orthographic

51.429 p_fov

2.000 p_near

13.072 p_far



choose

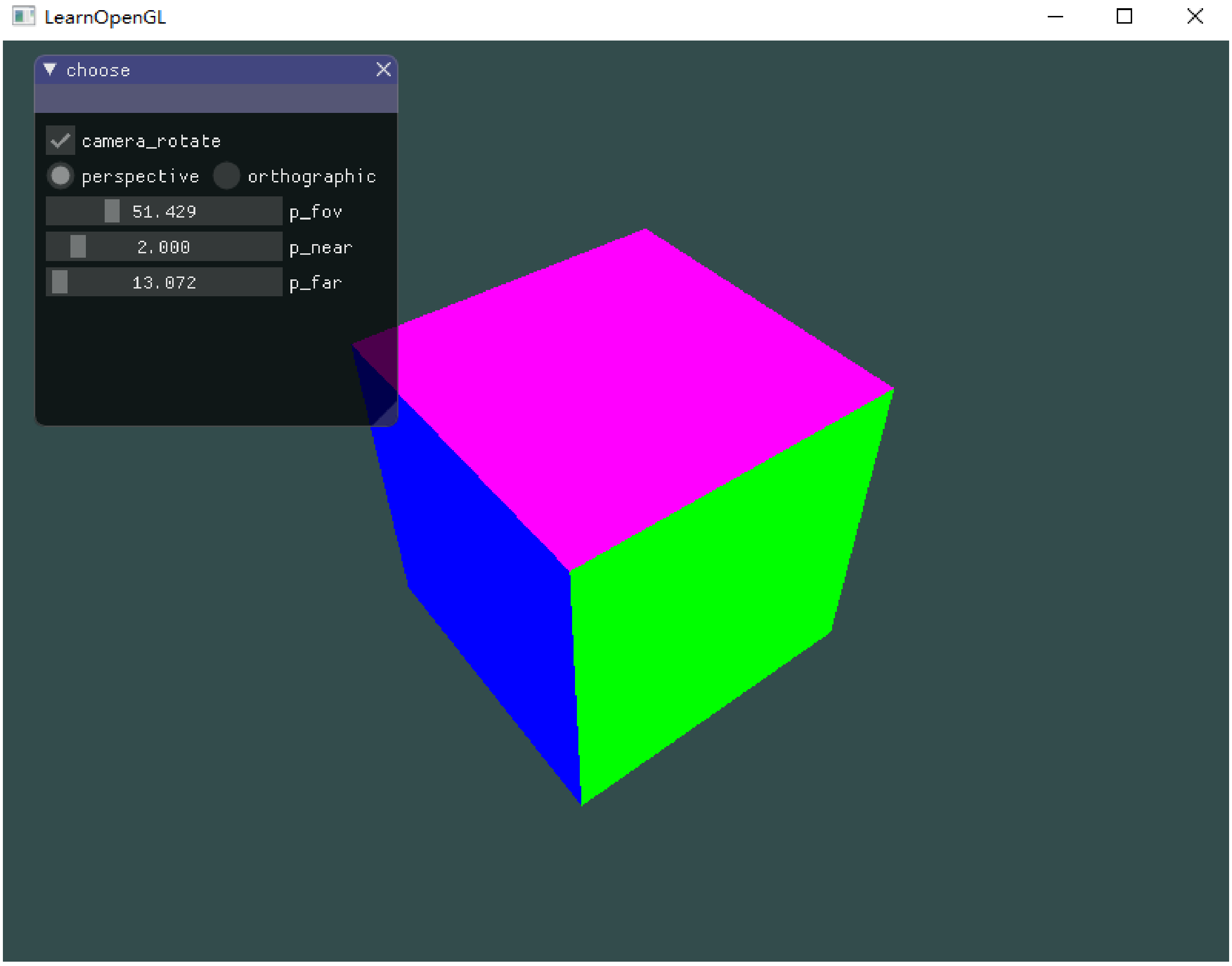
☑ camera_rotate

● perspective ○ orthographic

51.429 p_fov

2.000 p_near

13.072 p_far



☑ camera_rotate

● perspective ○ orthographic

51.429 p_fov

2.000 p_near

13.072 p_far

5. Bonus: 实现camera类，输入w,a,s,d可前后移动，移动鼠标可以移动视角。具体看gif

关键代码

1. cube放在(-1.5, 0.5, -1.5)的两种方式

```
// 通过lookAt函数将cube放在(-1.5, 0.5, -1.5)
view = glm::lookAt(glm::vec3(camera_x, camera_z, 8), glm::vec3(-1.5f, 0.5f, -1.5f),
glm::vec3(0, 1, 0));
// 通过平移函数将其放在(-1.5, 0.5, -1.5)
model = glm::translate(model, glm::vec3(-1.5f, 0.5f, -1.5f));
```

2. 正交投影

```
float o_top = 10.0f;
float o_left = -10.0f;
float o_right = 10.0f;
float o_bottom = -10.0f;
float o_near = -4.0f;
float o_far = 100.0f;
...
projection = glm::ortho(o_left, o_right, o_bottom, o_top, o_near, o_far);
```

3. 透视投影

```
tatic int perspective = 1;
float p_fov = 45.0f;
float p_near = 0.1f;
float p_far = 100.0f;

if (perspective) {
    projection = glm::perspective(glm::radians(p_fov), (float)SCR_WIDTH / (float)SCR_HEIGHT, p_near, p_far);
    ...
}
```

4. 摄像机绕cube旋转

```
if (camera_rotate) {
    camera_x = sin((float)glfwGetTime()) * 8.0f;
    camera_z = cos((float)glfwGetTime()) * 8.0f;
    view = glm::lookAt(glm::vec3(camera_x, 8, camera_z), glm::vec3(0, 0, 0), glm::vec3(0, 1, 0));
}
```

5. camera类

```
#include <glad/glad.h>
#include <GLFW/glfw3.h>
#include "shader.h"
#include "imgui.h"
#include "imgui_impl_glfw.h"
#include "imgui_internal.h"
#include "imgui_impl_opengl3.h"
#include "imconfig.h"
#include <iostream>
#include <stdlib.h>
#include <string>
using namespace std;

class Camera {
    public:
    Camera(glm::vec3 cameraPos, glm::vec3 objPos, glm::vec3 cameraUp) {
        this->cameraPos = cameraPos;
        this->cameraFront = objPos - cameraPos;
        this->cameraUp = cameraUp;
    }

    void moveForward(float const moveSpeed) {
        cameraPos += cameraFront * moveSpeed;
    }
    void moveBack(float const moveSpeed) {
        cameraPos -= cameraFront * moveSpeed;
    }
    void moveRight(float const moveSpeed) {
        cameraPos += glm::normalize(glm::cross(this->cameraFront, this->cameraUp)) *
moveSpeed;
    }
    void moveLeft(float const moveSpeed) {
        cameraPos -= glm::normalize(glm::cross(this->cameraFront, this->cameraUp)) *
moveSpeed;
    }
    void rotate(GLfloat const pitch, GLfloat const yaw) {
        glm::vec3 front = glm::vec3(cos(glm::radians(yaw)) *
cos(glm::radians(pitch)), sin(glm::radians(pitch)), sin(glm::radians(yaw)) *
cos(glm::radians(pitch)));
        cameraFront = glm::normalize(front);
    }

    glm::vec3 getCameraPos() {
        return cameraPos;
    }

    glm::vec3 getCameraFront() {
        return cameraFront;
    }

    glm::vec3 getCameraUp() {
        return cameraUp;
    }

private:

    GLfloat pfov, pratio, pnear, pfar;
    glm::vec3 cameraPos, cameraFront, cameraUp;

};
```

6. 输入w,a,s,d前后移动

```
Camera camera(glm::vec3(4.0f, 4.0f, 4.0f), glm::vec3(0, 0, 0.0f), glm::vec3(0, 1,
0));
// part 1
float deltaTime = 0.0f;
float lastFrame = 0.0f;

void processInput(GLFWwindow *window)
{
    // 计算当前帧与上一帧的时间差
    float currentFrame = glfwGetTime();
    deltaTime = currentFrame - lastFrame;
    lastFrame = currentFrame;
    // 设置移动速度
    float cameraSpeed = 5.0f * deltaTime;

    if (glfwGetKey(window, GLFW_KEY_ESCAPE) == GLFW_PRESS) {
        glfwSetWindowShouldClose(window, true);
    }
    else if (glfwGetKey(window, GLFW_KEY_W) == GLFW_PRESS) {
        camera.moveForward(cameraSpeed);
    }
    else if (glfwGetKey(window, GLFW_KEY_S) == GLFW_PRESS) {
        camera.moveBack(cameraSpeed);
    }
    else if (glfwGetKey(window, GLFW_KEY_A) == GLFW_PRESS) {
        camera.moveLeft(cameraSpeed);
    }
    else if (glfwGetKey(window, GLFW_KEY_D) == GLFW_PRESS) {
        camera.moveRight(cameraSpeed);
    }
}
```

7. 移动鼠标可以移动视角

```
//part 2
bool firstMouse = true;
float yaw = -90.0f;
float pitch = 0.0f;
float lastX = 400.0f;
float lastY = 300.0f;

void mouse_callback(GLFWwindow* window, double mouse_x, double mouse_y) {
    if (firstMouse) {
        lastX = mouse_x;
        lastY = mouse_y;
        firstMouse = false;
    }

    float xOffset = mouse_x - lastX;
    float yOffset = lastY - mouse_y;
    lastX = mouse_x;
    lastY = mouse_y;

    float sensitivity = 0.05f;
    xOffset *= sensitivity;
    yOffset *= sensitivity;

    yaw += xOffset;
    pitch += yOffset;

    if (pitch > 89.0f)
        pitch = 89.0f;
    if (pitch < -89.0f)
        pitch = -89.0f;

    camera.rotate(pitch, yaw);
}

// 设置模式使鼠标光标隐藏
glfwSetInputMode(window, GLFW_CURSOR, GLFW_CURSOR_DISABLED);
// 添加鼠标移动回调
```

```
glfwSetCursorPosCallback(window, mouse_callback);
```

实验思考

1. 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）

   答：合二为一的原因我觉得是增强两者的联系，因为摄像机拍摄的就是物体，设置好要拍摄的物体和摄像机摆放的空间显得更加的自然，而且通过同时设置ModelView的话还可以根据需要进行正交、透视投影的变换。也可以很方便的实现摄像机镜头追随物体的功能。