《计算机图形学》实验 5 实验报告

学生	刘沅昊	学号	15331220
学院	数据科学与计算机学院	年级专业	16 级软件工程 (数字媒体技术)

实验内容：

Homework

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：

```
camPosX=sin(clock()/1000.0)*Radius;
camPosZ=cos(clock()/1000.0)*Radius;
```

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

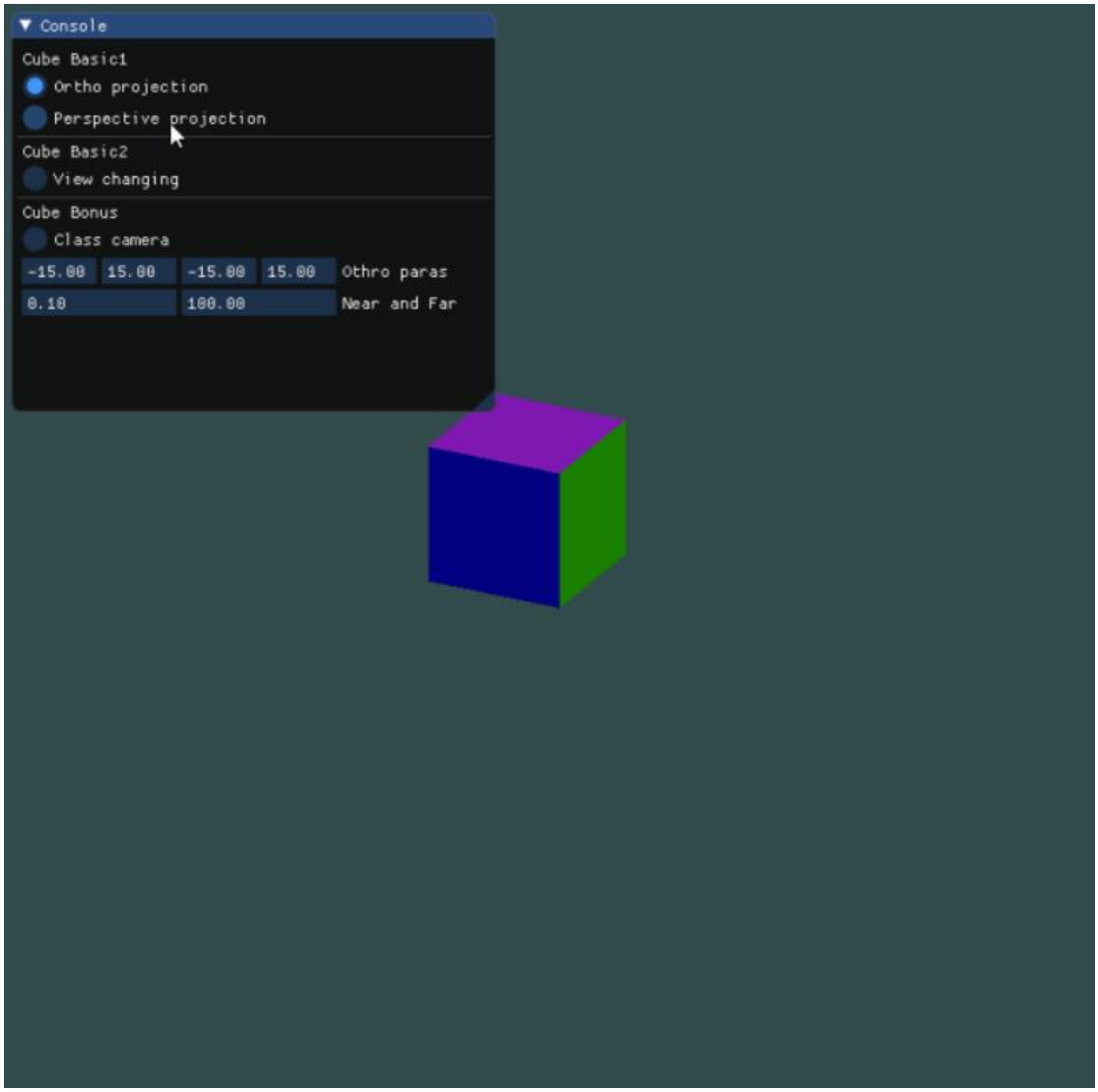
实验结果：

Basic：

1. 投影（Projection）

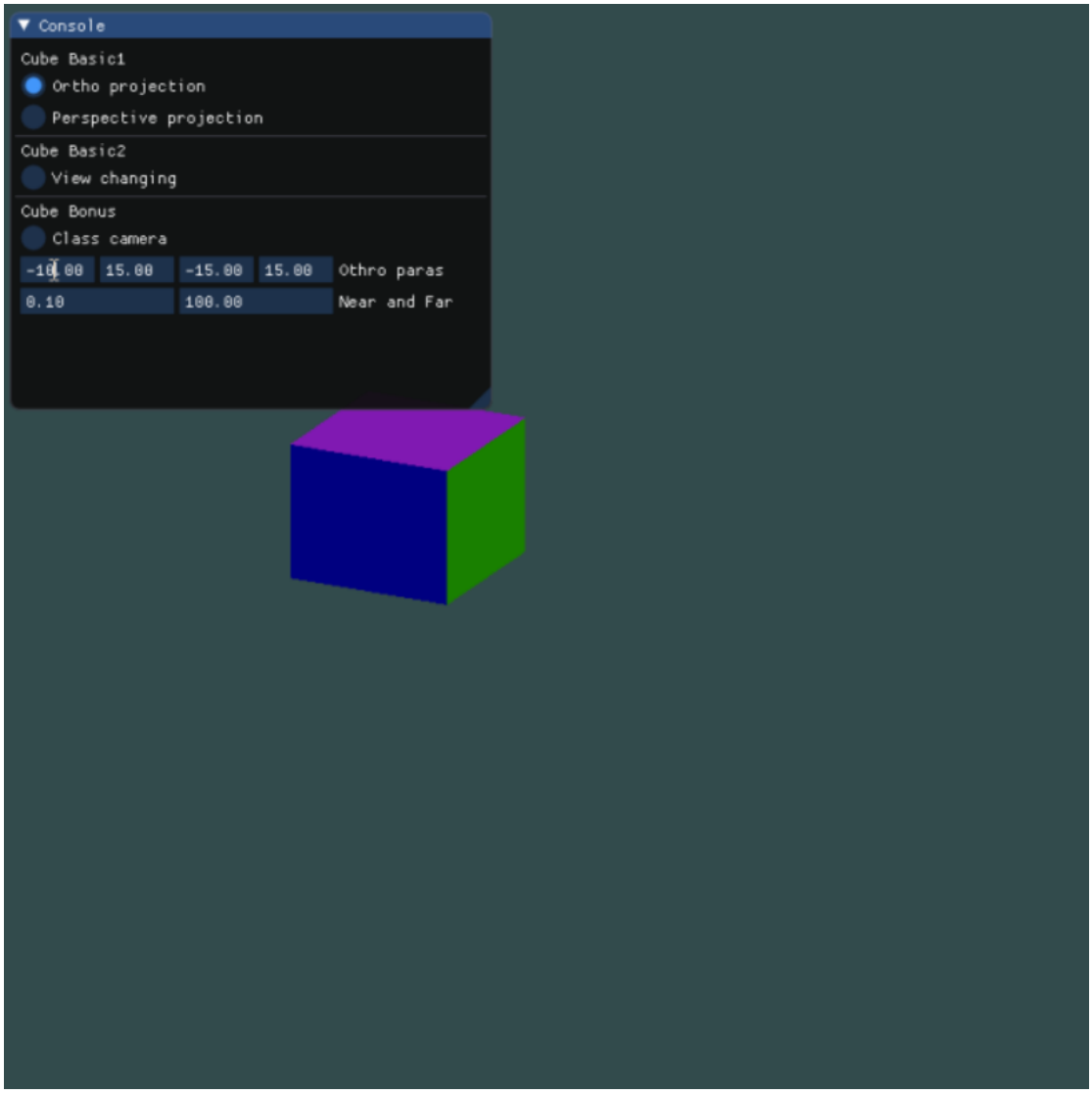
```
237             model = glm::translate(model, glm::vec3(-1.5f, 0.5f, -1.5f));
238             my_shader.setMat4("model", glm::value_ptr(model));
239             view = glm::lookAt(
240                 glm::vec3(10.0f, 10.0f, 20.0f),
241                 glm::vec3(0.0f, 0.0f, 0.0f),
242                 glm::vec3(0.0f, 1.0f, 0.0f)
243             );
```

图表 1 Cube 放在(-1.5,0.5,-1.5)的位置



图表 2 正交投影

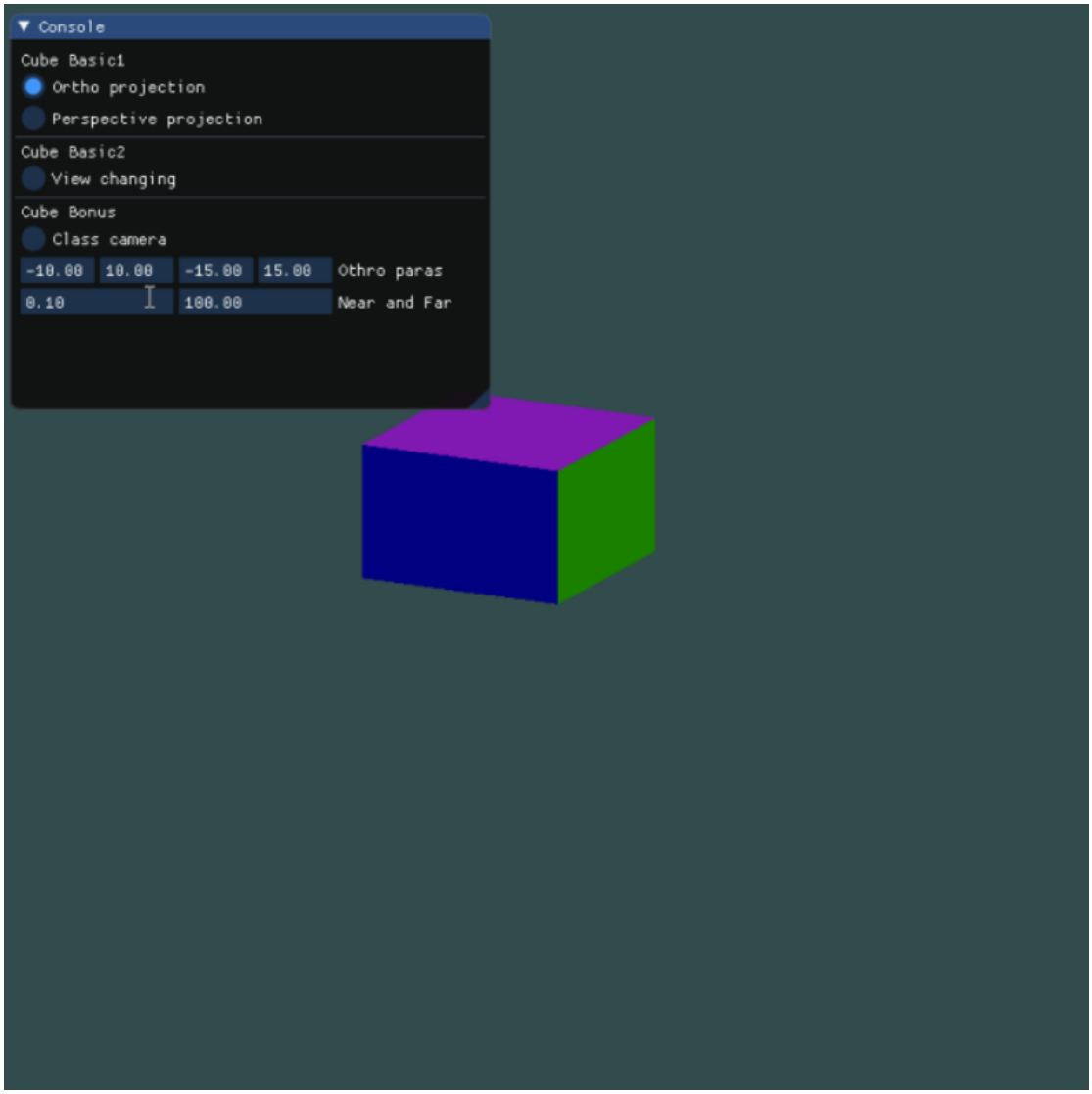
此时参数(left,right,bottom,top,near,far)：(-15.00,15.00,-15.00,15.00,0.10,100.00)



图表 3 正交投影，修改参数 left

此时参数(left,right,bottom,top,near,far):(-10.00,15.00,-15.00,15.00,0.10,100.00)

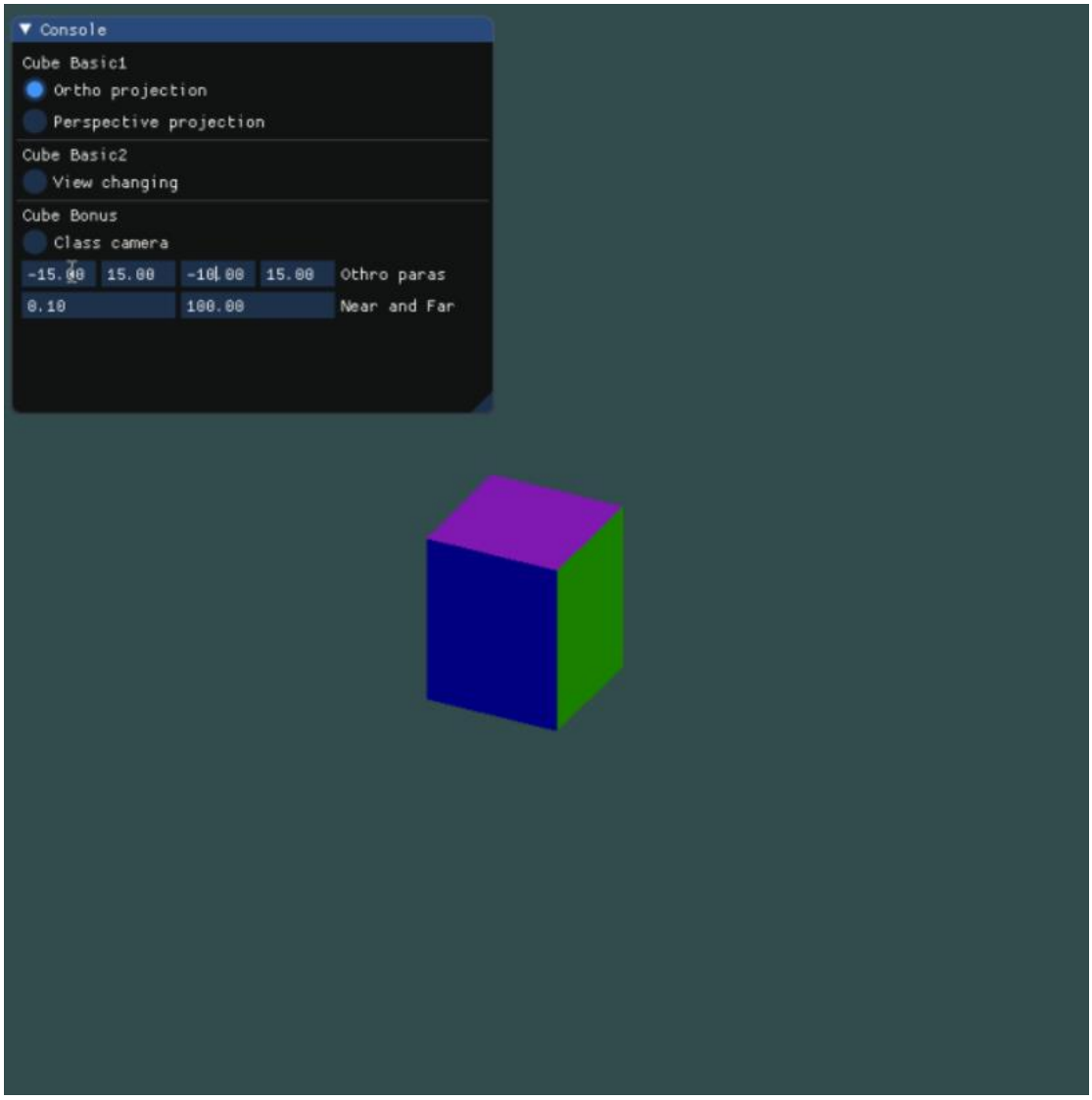
left 参数变小，立方体向左拉伸。



图表 4 正交投影，修改参数 left,right

此时参数(left,right,bottom,top,near,far):(-10.00,10.00,-15.00,15.00,0.10,100.00)

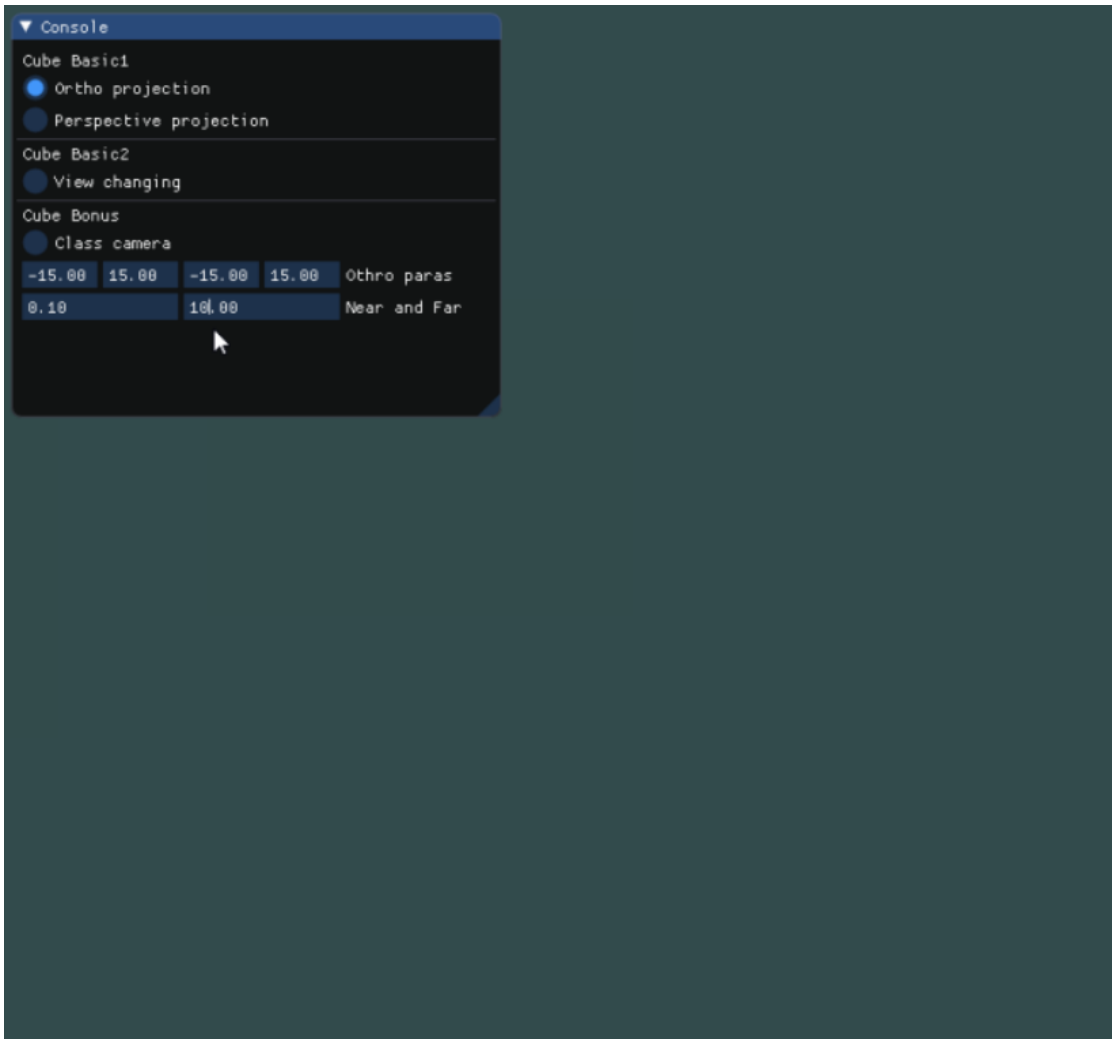
left,right 参数变小，立方体向左右拉伸。



图表 5 正交投影，修改参数 bottom

此时参数(left,right,bottom,top,near,far):(-15.00,15.00,-10.00,15.00,0.10,100.00)

bottom 参数变小，立方体向下拉伸。

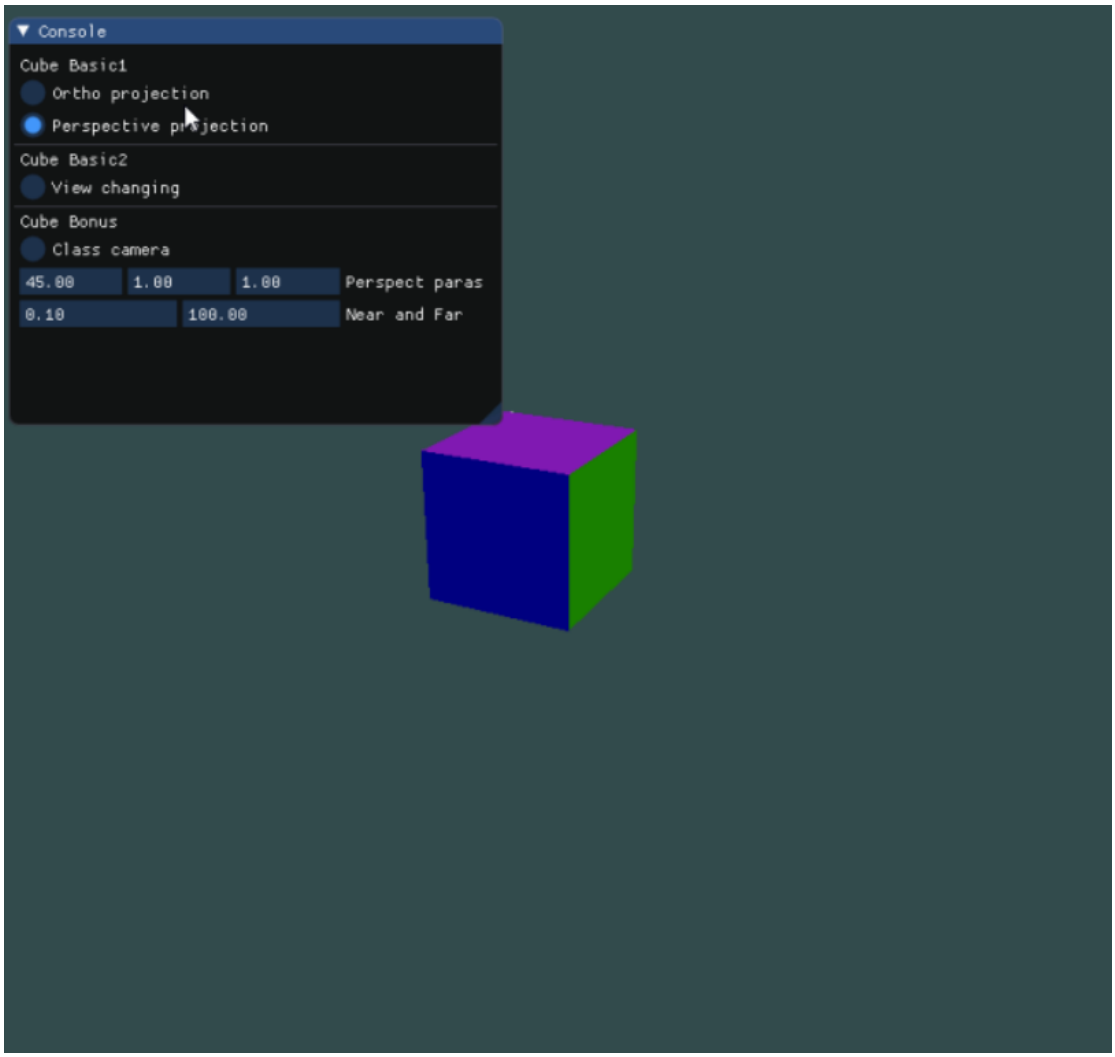


图表 6 正交投影，修改参数 far

此时参数(left,right,bottom,top,near,far):(-15.00,15.00,-15.00,15.00,0.10,10.00)far 参数变小，因为摄像机的位置在(10.0,10.0,20.0)，摄像机只拍摄距离 0.1 到 10.0 内的物体，所以立方体没有被拍摄。

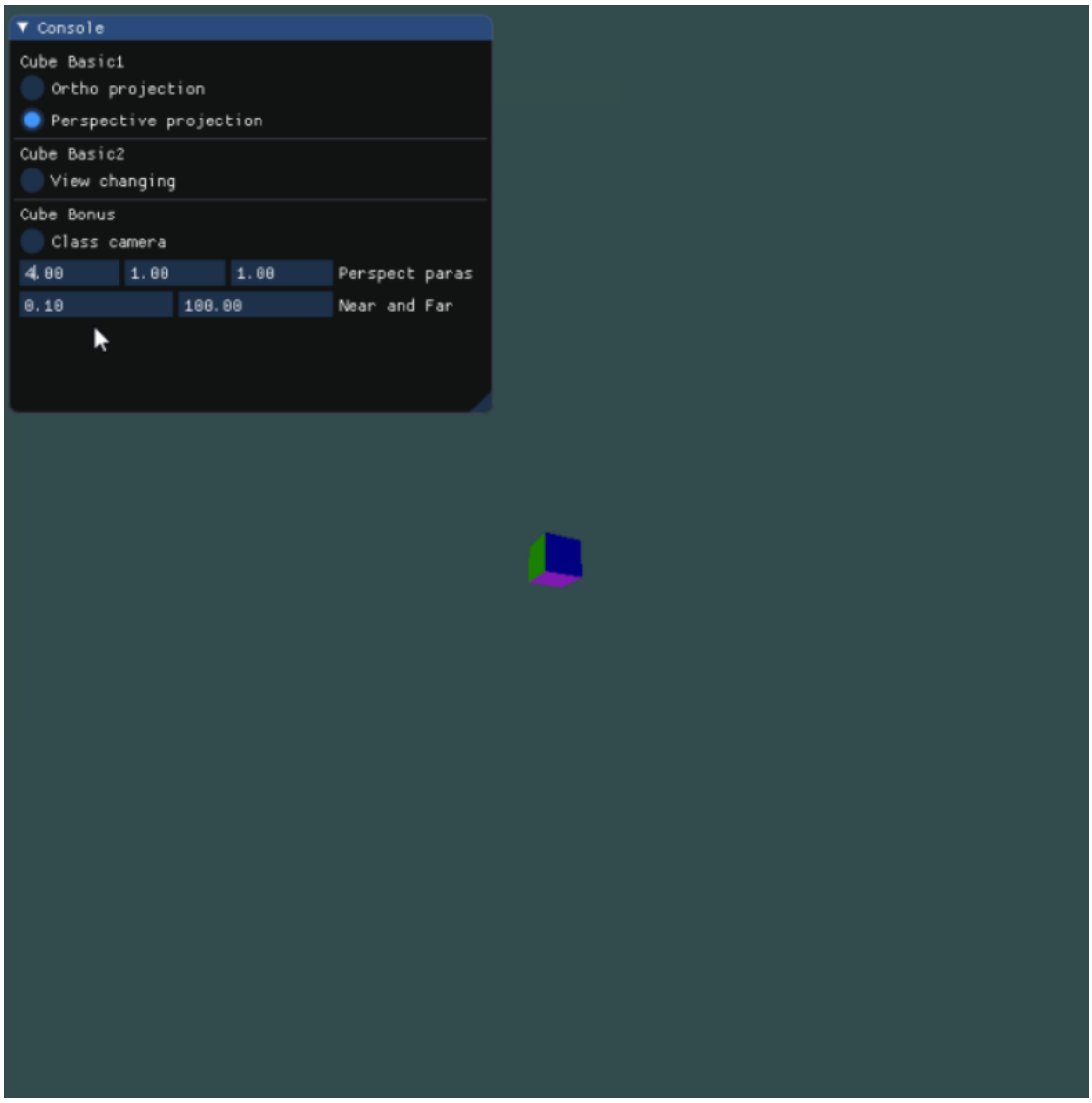
```
237             model = glm::translate(model, glm::vec3(-1.5f, 0.5f, -1.5f));
238             my_shader.setMat4("model", glm::value_ptr(model));
239             view = glm::lookAt(
240                 glm::vec3(10.0f, 10.0f, 20.0f),
241                 glm::vec3(0.0f, 0.0f, 0.0f),
242                 glm::vec3(0.0f, 1.0f, 0.0f)
243             );
```

图表 7 摄像机位置



图表 8 透视投影

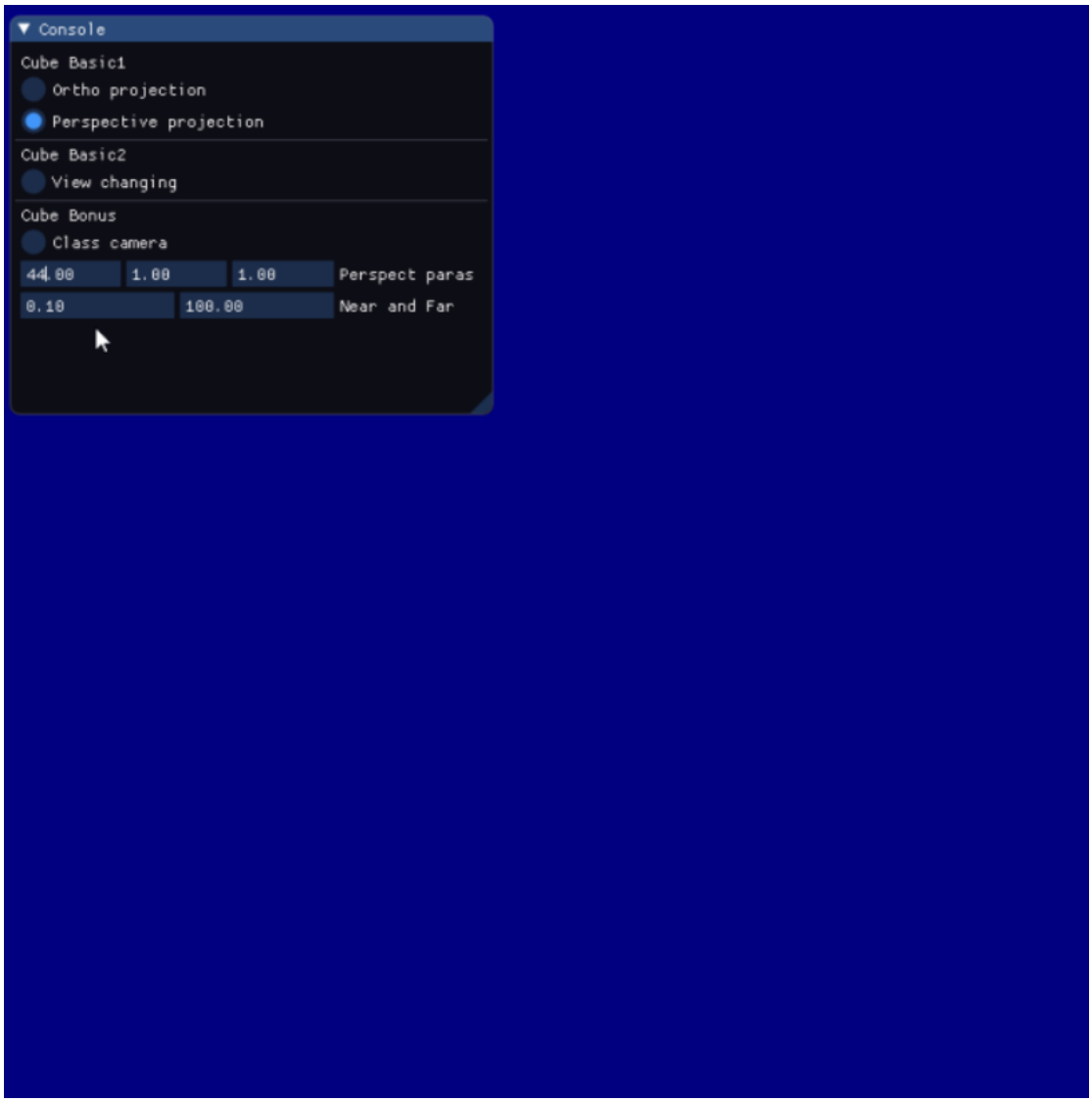
此时参数(fov,width/length,near,far)：（45.00,1.00/1.00,0.10,100.00）



图表 9 透视投影，修改参数 fov

此时参数(fov,width/length,near,far)：（4.00,1.00/1.00,0.10,100.00）

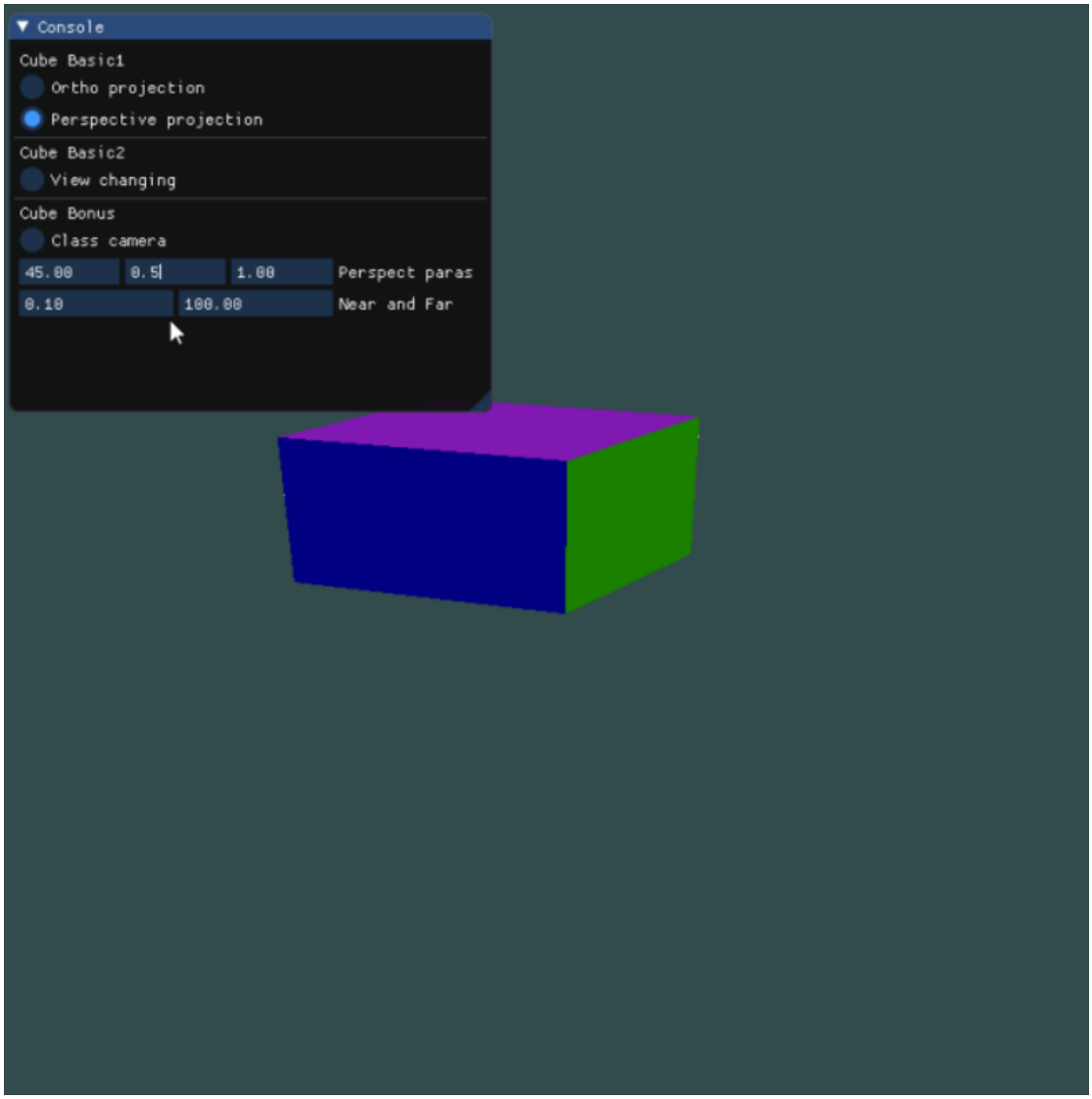
此时物体变小，又倒立了。



图表 10 透视投影，修改参数 fov

此时参数(fov,width/length,near,far)：（44.00,1.00/1.00,0.10,100.00）

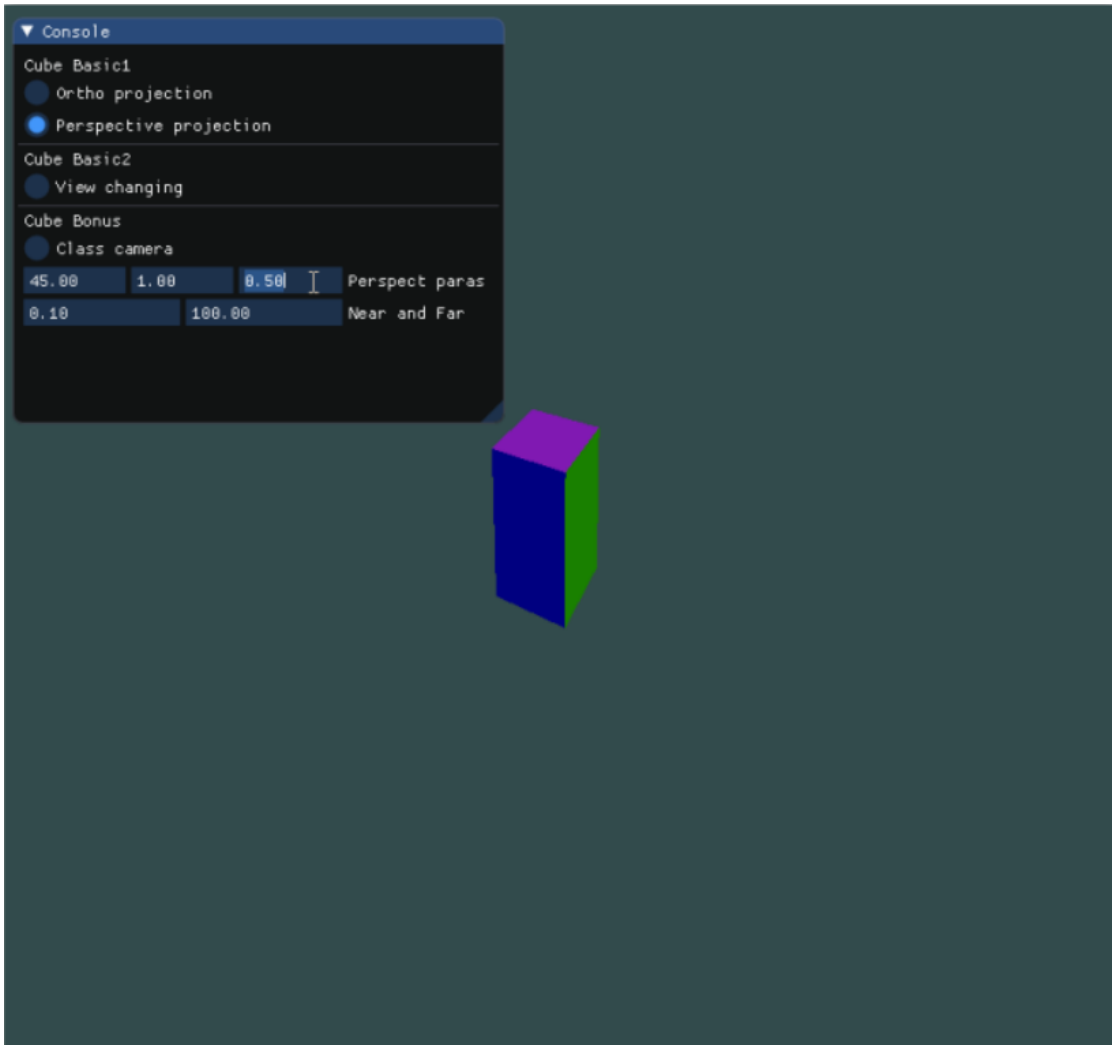
物体被放大了。



图表 11 透视投影，修改参数 width

此时参数(fov,width/length,near,far)：（44.00,0.50/1.00,0.10,100.00）

width 值变小，width/length 比值也变小，表现为物体左右拉伸。



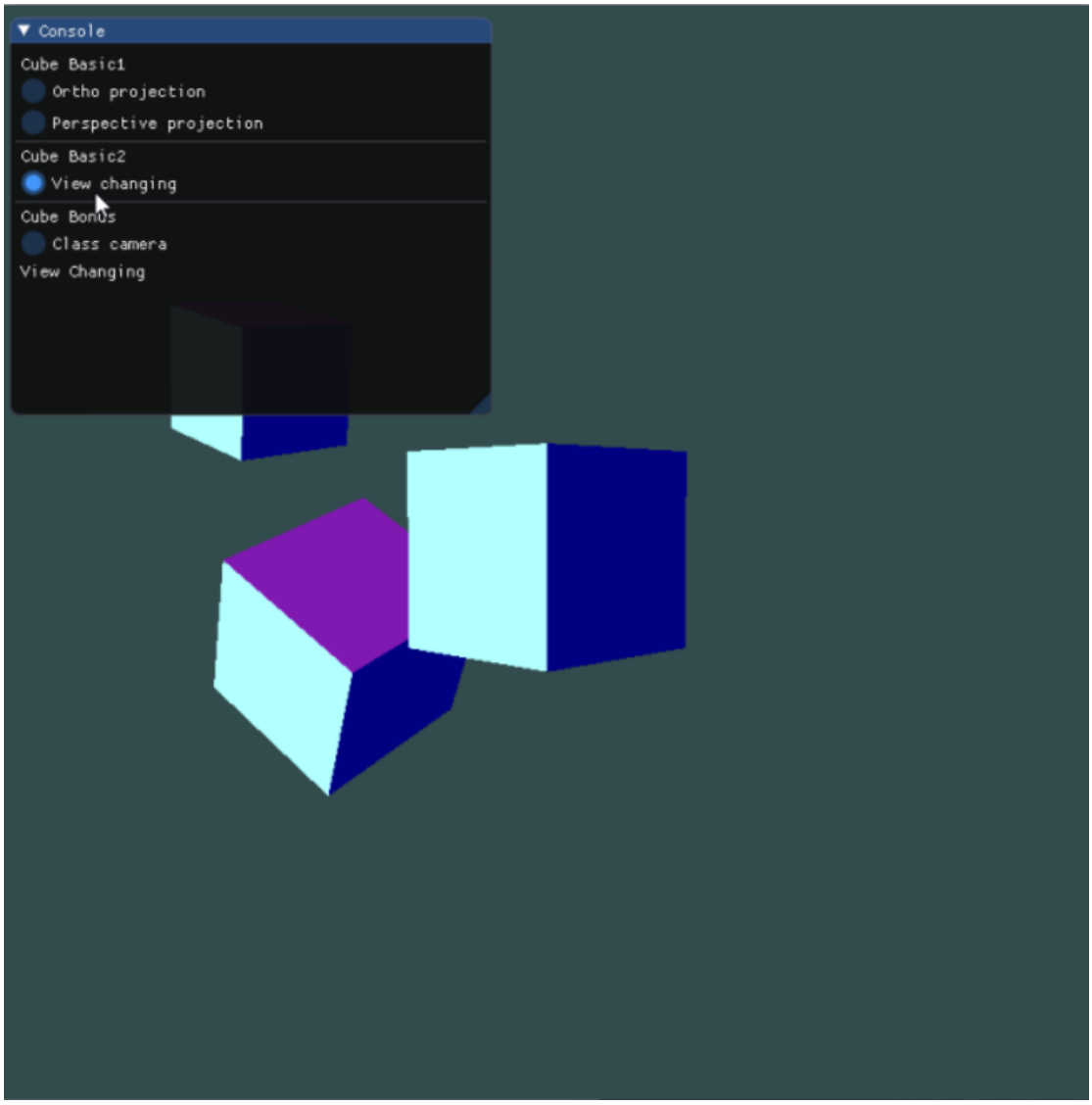
图表 12 透视投影，修改参数 length

此时参数(fov, width/length, near, far)：（44.00, 1.00/0.50, 0.10, 100.00）

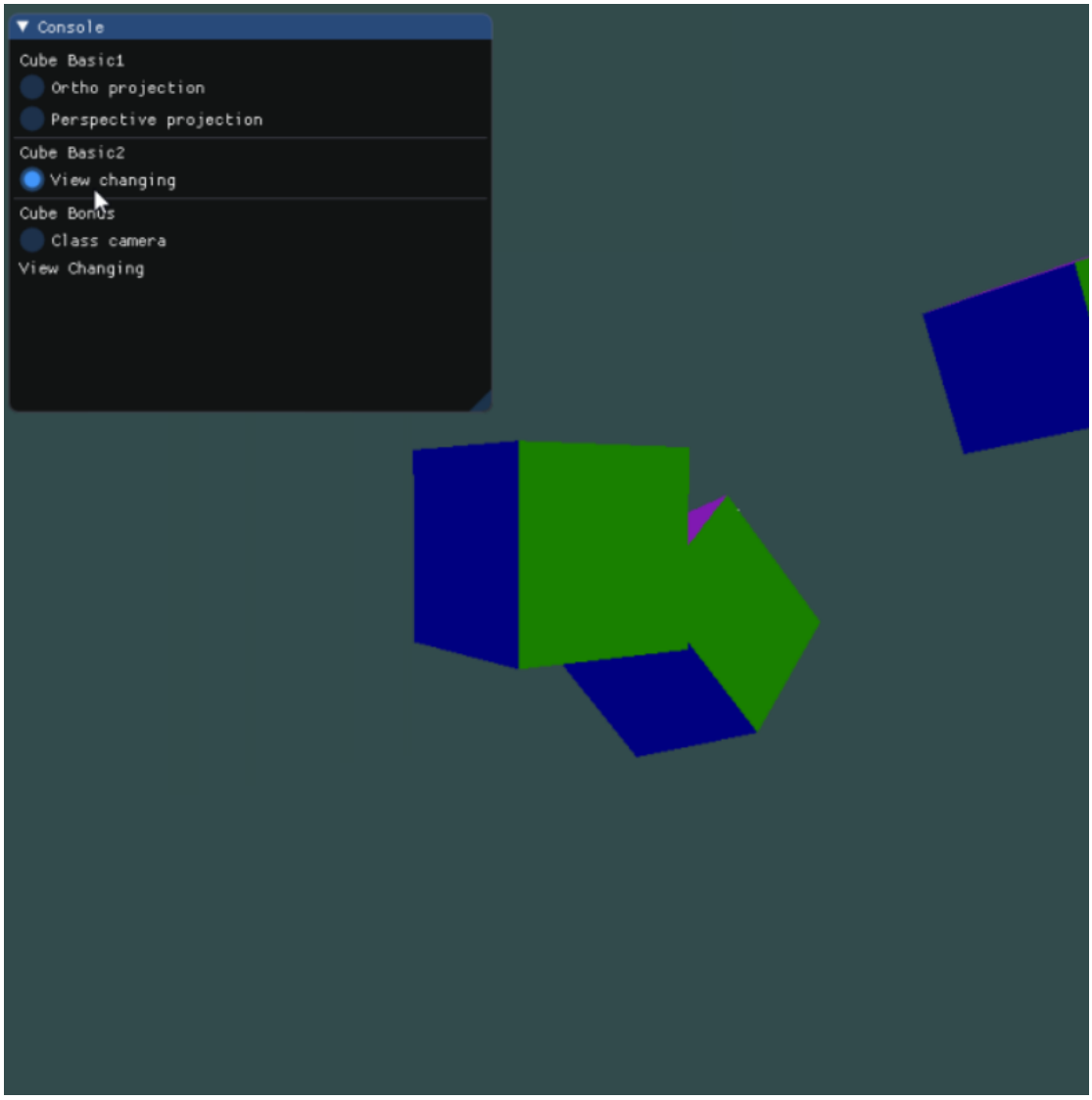
length 值变小，width/length 比值变大，表现为物体上下拉伸。

总结：调整 left，right，bottom，top 参数时，cube 会在水平或垂直方向上进行伸缩或平移，调整 fov 参数会影响视野大小（表现为 cube 的大小变化以及倒立成像），而调整 near，far 参数时会影响 cube 可视部分

2. 视角变换（View Changing）



图表 13 视角变换


Console
Cube Basic1
Ortho projection
Perspective projection
Cube Basic2
View changing
Cube Bonus
Class camera
View Changing

图表 14 视角变换

3. 添加 GUI

```
187         {
188             ImGui::Begin("Console");
189 
190             ImGui::Text("Cube Basic1");
191             ImGui::RadioButton("Ortho projection", &choose, 0);
192             ImGui::RadioButton("Perspective projection", &choose, 1);
193             ImGui::Separator();
194             ImGui::Text("Cube Basic2");
195             ImGui::RadioButton("View changing", &choose, 2);
196             ImGui::Separator();
197             ImGui::Text("Cube Bonus");
198             ImGui::RadioButton("Class camera", &choose, 3);
199 
200             switch (choose)
201             {
202             case 0:
203                 ImGui::InputFloat4("Othro paras", ortho, 2);
204                 ImGui::InputFloat2("Near and Far", near_far, 2);
205                 break;
206             case 1:
207                 ImGui::InputFloat3("Perspect paras", perspect, 2);
208                 ImGui::InputFloat2("Near and Far", near_far, 2);
209                 break;
210             case 2:
211                 ImGui::Text("View Changing");
212                 break;
213             case 3:
214                 isFirstInFPS = true;
215                 break;
216             default:
217                 break;
218             }
219 
220             ImGui::End();
221         }
```

图表 15 添加 GUI

4. Model 矩阵是一种变换矩阵，它能通过对物体进行位移、缩放、旋转来将它置于它本应该在的位置或朝向。观察空间就是从摄像机的视角所观察到的空间。而这通常是由一系列的位移和旋转的组合来完成，平移/旋转场景从而使得特定的对象被变换到摄像机的前方。将两个合二为一的 ModelView 矩阵事实上是利用改变摄像机的观察位置、方向和距离来变相实现 Model 矩阵的作用。当有多个摄像机的时候我们就不用每次成像用 Model 矩阵改变物体的位置，通过给每个摄像头设置 ModelView 矩阵就能实现物体的位移、缩放、旋转，减少了运算量。

Bonus：

```cpp
// ---------------------------
// 实现一些输入控制,需要一个窗口以及一个按键作为输入
// ---------------------------
void processInput(GLFWwindow *window)
{
    // 检查用户是否按下了返回键(Esc), 退出程序
    if (glfwGetKey(window, GLFW_KEY_ESCAPE) == GLFW_PRESS)
        glfwSetWindowShouldClose(window, true);
    // W 控制前进
    if (glfwGetKey(window, GLFW_KEY_W) == GLFW_PRESS)
        camera.ProcessKeyboard(FORWARD, deltaTime);
    // S 控制后退
    if (glfwGetKey(window, GLFW_KEY_S) == GLFW_PRESS)
        camera.ProcessKeyboard(BACKWARD, deltaTime);
    // A 控制左移
    if (glfwGetKey(window, GLFW_KEY_A) == GLFW_PRESS)
        camera.ProcessKeyboard(LEFT, deltaTime);
    // D 控制右移
    if (glfwGetKey(window, GLFW_KEY_D) == GLFW_PRESS)
        camera.ProcessKeyboard(RIGHT, deltaTime);
    // 空格 控制上升
    if (glfwGetKey(window, GLFW_KEY_SPACE) == GLFW_PRESS)
        camera.ProcessKeyboard(UP, deltaTime);
    // 左Ctrl 控制下降
    if (glfwGetKey(window, GLFW_KEY_LEFT_CONTROL) == GLFW_PRESS)
        camera.ProcessKeyboard(DOWN, deltaTime);
    // C 改变模式
    if (glfwGetKey(window, GLFW_KEY_C) == GLFW_PRESS) {
        glfwSetInputMode(window, GLFW_CURSOR, GLFW_CURSOR_NORMAL);
        isFirstInFPS = false;
        choose = 0;
    }
}

```

图表 16 源文件中的键盘事件函数

```cpp
// glfw: whenever the mouse moves, this callback is called
// -------------------------------------------------------
void mouse_callback(GLFWwindow* window, double xpos, double ypos)
{
    if (firstMouse)
    {
        lastX = xpos;
        lastY = ypos;
        firstMouse = false;
    }

    float xoffset = xpos - lastX;
    float yoffset = lastY - ypos; // reversed since y-coordinates go from bottom to top

    lastX = xpos;
    lastY = ypos;

    camera.ProcessMouseMovement(xoffset, yoffset);
}

// glfw: whenever the mouse scroll wheel scrolls, this callback is called
// ----------------------------------------------------------------------
void scroll_callback(GLFWwindow* window, double xoffset, double yoffset)
{
    camera.ProcessMouseScroll(yoffset);
}
```

图表 17 源文件中的鼠标事件函数，鼠标移动和滚轮滑动

```
72      // Processes input received from any keyboard-like input system. Acce
73      void ProcessKeyboard(Camera_Movement direction, float deltaTime)
74      {
75          float velocity = MovementSpeed * deltaTime;
76          if (direction == FORWARD)
77              Position += Front * velocity;
78          if (direction == BACKWARD)
79              Position -= Front * velocity;
80          if (direction == LEFT)
81              Position -= Right * velocity;
82          if (direction == RIGHT)
83              Position += Right * velocity;
84          if (direction == UP)
85              Position += Up * velocity;
86          if (direction == DOWN)
87              Position -= Up * velocity;
88      }
```

图表 18 Camera 类根据键盘输入能上下左右移动

```
90      // Processes input received from a mouse input system. Expects the offset value in both the x and y direction.
91      void ProcessMouseMovement(float xoffset, float yoffset, GLboolean constrainPitch = true)
92      {
93          xoffset *= MouseSensitivity;
94          yoffset *= MouseSensitivity;
95
96          Yaw += xoffset;
97          Pitch += yoffset;
98
99          // Make sure that when pitch is out of bounds, screen doesn't get flipped
100         if (constrainPitch)
101         {
102             if (Pitch > 89.0f)
103                 Pitch = 89.0f;
104             if (Pitch < -89.0f)
105                 Pitch = -89.0f;
106         }
107
108         // Update Front, Right and Up Vectors using the updated Euler angles
109         updateCameraVectors();
110     }
111
112     // Processes input received from a mouse scroll-wheel event. Only requires input on the vertical wheel-axis
113     void ProcessMouseScroll(float yoffset)
114     {
115         if (Zoom >= 1.0f && Zoom <= 45.0f)
116             Zoom -= yoffset;
117         if (Zoom <= 1.0f)
118             Zoom = 1.0f;
119         if (Zoom >= 45.0f)
120             Zoom = 45.0f;
121     }
```

图表 19 Camera 类根据鼠标移动和滚轮滑动控制视野