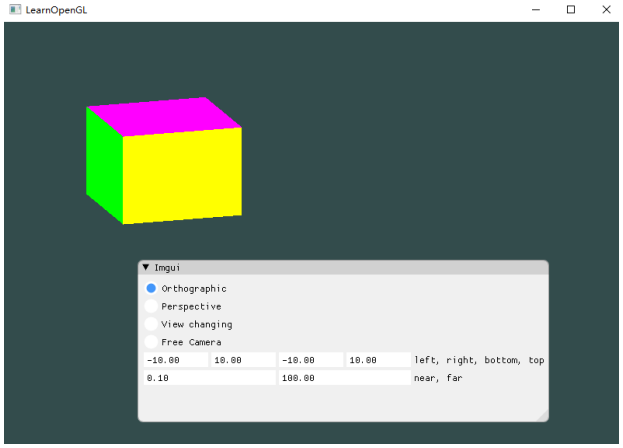
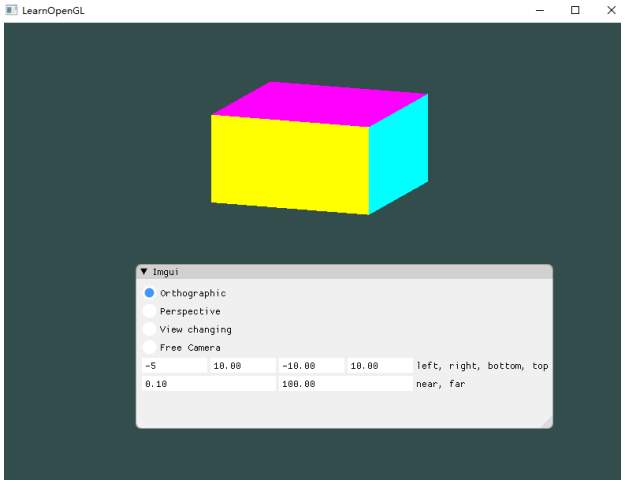
1. 投影(Projection): 把上次作业绘制的 cube 放置在(-1.5, 0.5, -1.5)位置，要求 6 个面颜色不一致



left=10

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

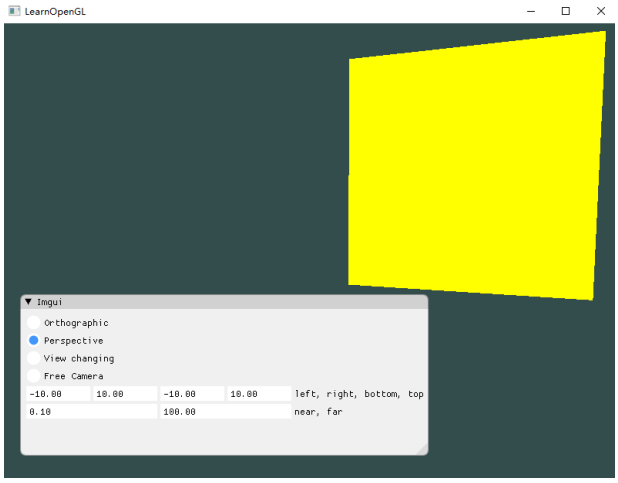
projection = glm::ortho(ortho[0], ortho[1], ortho[2], ortho[3], distance[0], distance[1]);



left=5

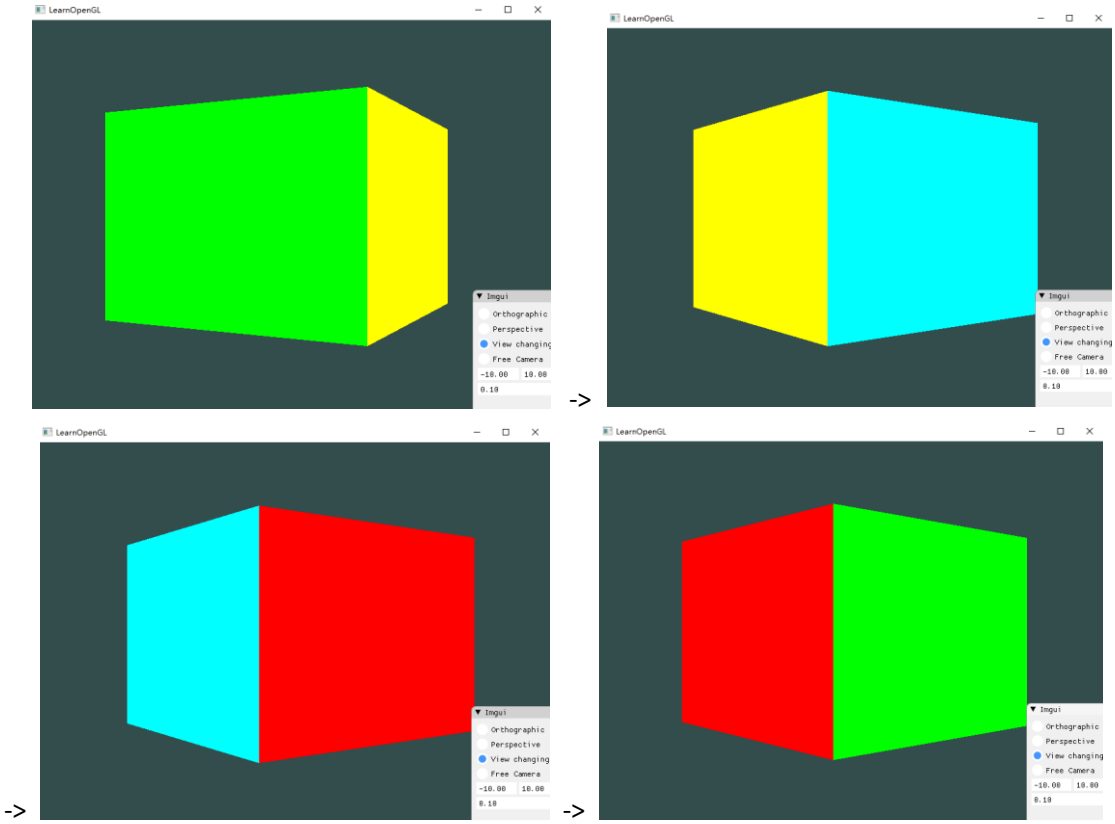
透视投影(perspective projection)：实现透视投影，使用多组参数，比较结果差异

projection = glm::perspective(glm::radians(camera.Zoom), (float)SCR_WIDTH / (float)SCR_HEIGHT, distance[0], distance[1]);

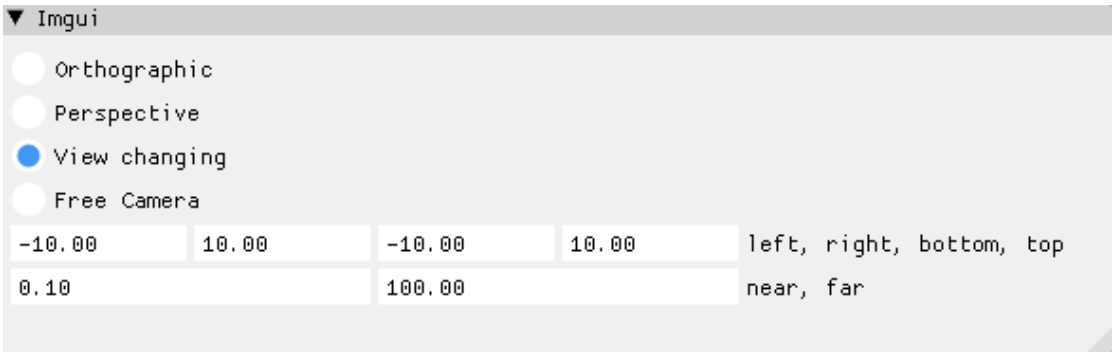


2. 视角变换(View Changing): 把 cube 放置在(0, 0, 0)处，做透视投影，使摄像机围绕 cube 旋转，并且时刻看着 cube 中心。

view = glm::lookAt(glm::vec3(camPosX, 0.0f, camPosZ), glm::vec3(0.0f, 0.0f, 0.0f), glm::vec3(0.0f, 1.0f, 0.0f));



3. 在 GUI 里添加菜单栏，可以选择各种功能。



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？

对于每个顶点，分别做 model 和 view 变换加起来进行了两次矩阵运算。计算次数为 2n 而合并过后只需要做 n+1 次矩阵运算，节省了大量的运算资源。

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