本科生实验报告

实验课程:_____计算机图形学_____

实验名称:_____assignment 0_____

专业名称:_____计算机科学与技术_____

学生姓名:_____李骏豪_____

学生学号:_____21307359_____

实验地点:_____

实验成绩:_____

报告时间:_____

Task 1、什么是 OpenGL？OpenGL 与计算机图形学的关系是什么？

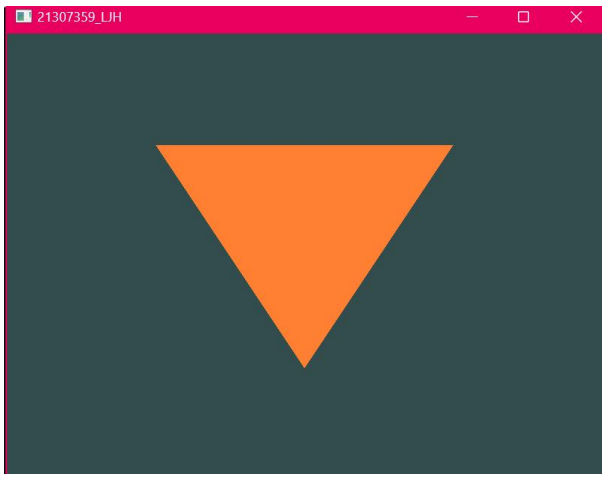
OpenGL（Open Graphics Library）是一个跨平台的图形库，用于开发 2D 和 3D 图形应用程序。它是一种底层的 API，通常与高级图形管道框架（例如 Unity 和 Unreal Engine）一起使用。

计算机图形学是一个广泛的领域，涉及各种技术和算法，用于从数据中创建和操作图形。OpenGL 是计算机图形学的一个重要组成部分，它提供了一个标准的方式来呈现 3D 场景，是当前最为流行的用于实现硬件加速的 3D 图形库之一。利用 OpenGL，图形程序员可以使用这个 API 去创建各种视觉效果，例如线框模式、纹理、照明、阴影等。通过 OpenGL，程序员可以直接控制图形硬件，并极其方便地实现非常高效的图形渲染。同时，OpenGL 的跨平台性使得程序员能够在不同的操作系统中开发运行相同的图形应用程序。

Task 2、完成了着色器章节之后，请修改顶点着色器让三角形上下颠倒。

如下图修改顶点即可：

```
float vertices[] = {
    0.5f,  0.5f, 0.0f,  // top right
    0.5f, -0.5f, 0.0f,  // bottom right
   -0.5f, -0.5f, 0.0f,  // bottom left
   -0.5f,  0.5f, 0.0f,   // top left
    0.0f,  0.5f, 0.0f,   //top middle
    0.0f, -0.5f, 0.0f   //bottom middle
}; //不同顶点的位置
unsigned int indices[] = {  // note that we start from 0! 1,2,3对应的是vertices中的顶点!
   //0, 1, 3,   // first Triangle
   //1, 2, 3    // second Triangle
   //4, 2, 1    //正的三角形
   0, 3, 5    //倒着的三角形
```

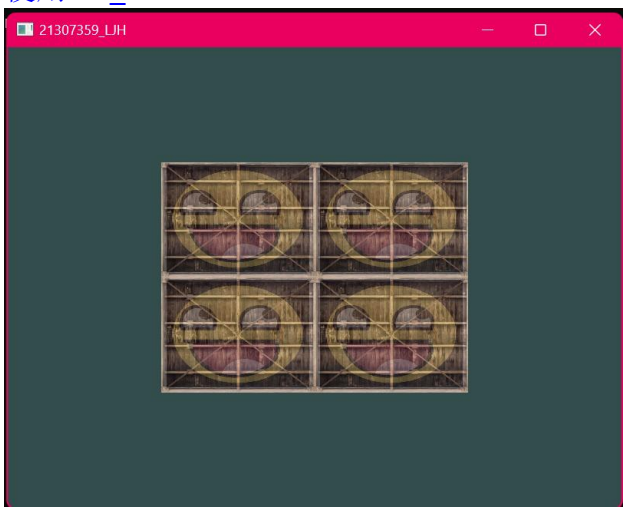


Task 3、完成了纹理章节之后，尝试用不同的纹理环绕方式，设定一个从 0.0f 到 2.0f 范围内的纹理坐标。试试看能不能在箱子的角落放置 4 个笑脸。简述原因并贴上结果。

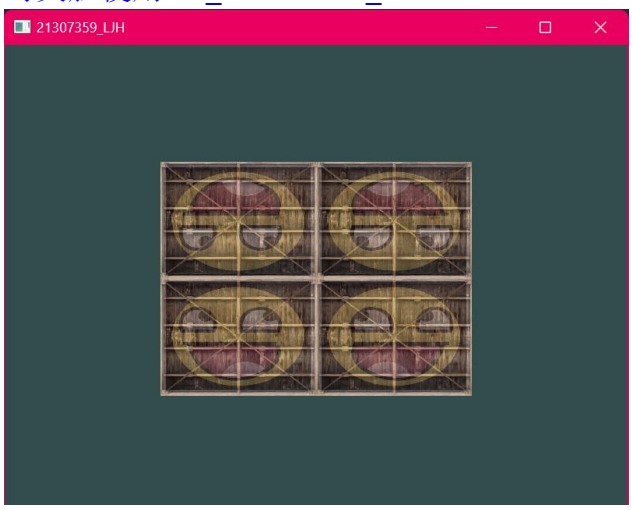
环绕方式	描述
GL_REPEAT	对纹理的默认行为。重复纹理图像。
GL_MIRRORED_REPEAT	和GL_REPEAT一样，但每次重复图片是镜像放置的。
GL_CLAMP_TO_EDGE	纹理坐标会被约束在0到1之间，超出的部分会重复纹理坐标的边缘，产生一种边缘被拉伸的效果。
GL_CLAMP_TO_BORDER	超出的坐标为用户指定的边缘颜色。

```
float vertices[] = {
    // positions          // colors           // texture coords(change to 2.0f)
     0.5f,  0.5f, 0.0f,   1.0f, 0.0f, 0.0f,   2.0f, 2.0f, // top right
     0.5f, -0.5f, 0.0f,   0.0f, 1.0f, 0.0f,   2.0f, 0.0f, // bottom right
    -0.5f, -0.5f, 0.0f,   0.0f, 0.0f, 1.0f,   0.0f, 0.0f, // bottom left
    -0.5f,  0.5f, 0.0f,   1.0f, 1.0f, 0.0f,   0.0f, 2.0f  // top left
};
```

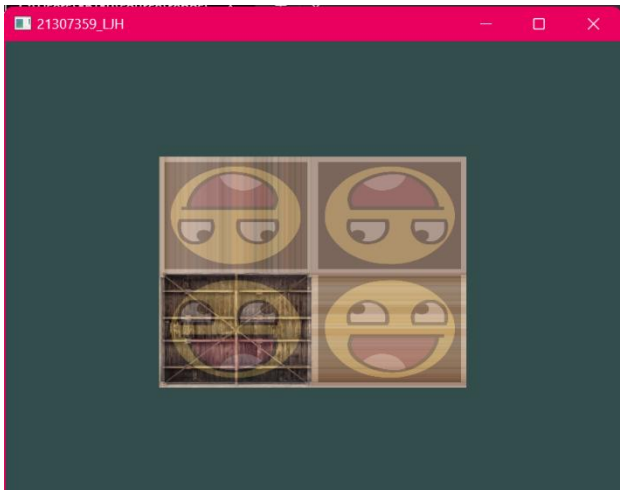
使用 GL_REPEAT:



对笑脸使用 GL_MIRRORED_REPEAT:



再对木箱使用 GL_CLAMP_TO_EDGE：

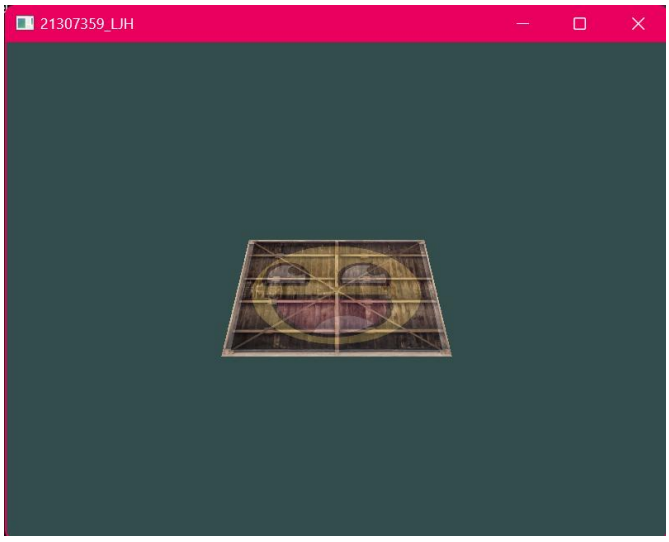


Task 4、完成了坐标系统章节之后，对 GLM 的 projection 函数中的 FoV 和 aspect-ratio 参数进行实验。看能否搞懂它们是如何影响透视平截头体的。

```
glm::mat4 proj = glm::perspective(glm::radians(45.0f), (float)width/(float)height, 0.1f, 100.0f);
```

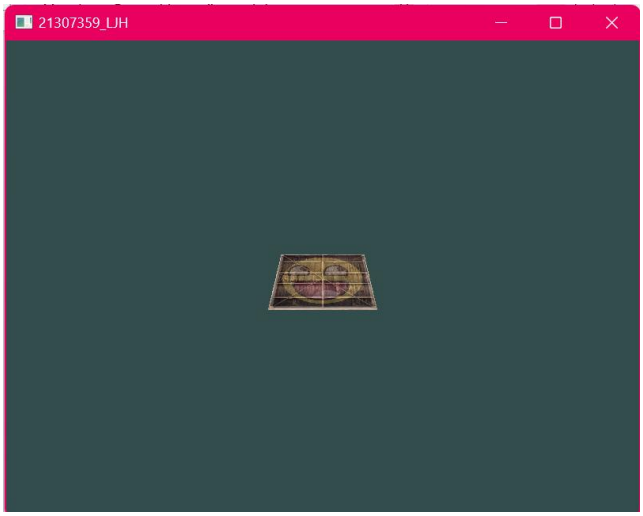
它的第一个参数定义了fov的值，它表示的是视野(Field of View)，并且设置了观察空间的大小。如果想要一个真实的观察效果，它的值通常设置为45.0f，但想要一个末日风格的结果你可以将其设置一个更大的值。第二个参数设置了宽高比，由视口的宽除以高所得。第三和第四个参数设置了平截头体的**近**和**远**平面。我们通常设置近距离为0.1f，而远距离设为100.0f。所有在近平面和远平面内且处于平截头体内的顶点都会被渲染。

正常的变化结果如下：

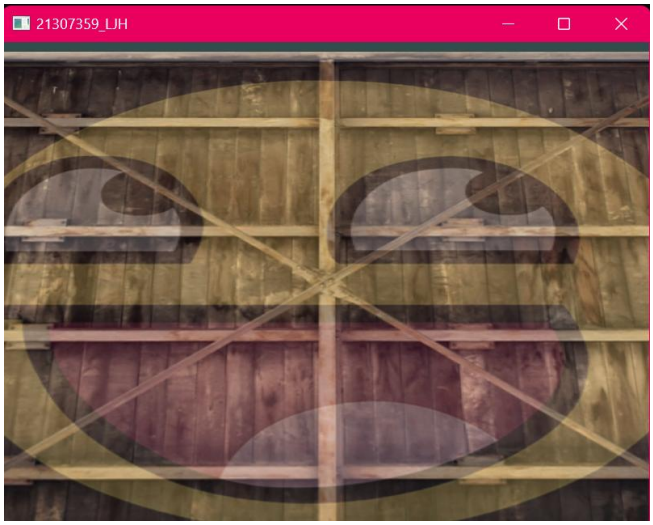


```
// create transformations
glm::mat4 model = glm::mat4(1.0f); // make sure to initialize matrix to identity matrix first,  模型矩阵，变成世界坐标
glm::mat4 view = glm::mat4(1.0f); //观察矩阵，变成观察坐标
glm::mat4 projection = glm::mat4(1.0f); //投影矩阵，变成裁剪坐标
model = glm::rotate(model, glm::radians(-55.0f), glm::vec3(1.0f, 0.0f, 0.0f)); //旋转和平移
view = glm::translate(view, glm::vec3(0.0f, 0.0f, -3.0f)); //// 注意，我们将矩阵向我们要进行移动场景的反方向(-z轴)移动。
projection = glm::perspective(glm::radians(-45.0f), (float)SCR_WIDTH / (float)SCR_HEIGHT, 0.1f, 100.0f);
```

将 FOV 修改为 80.0f 后：



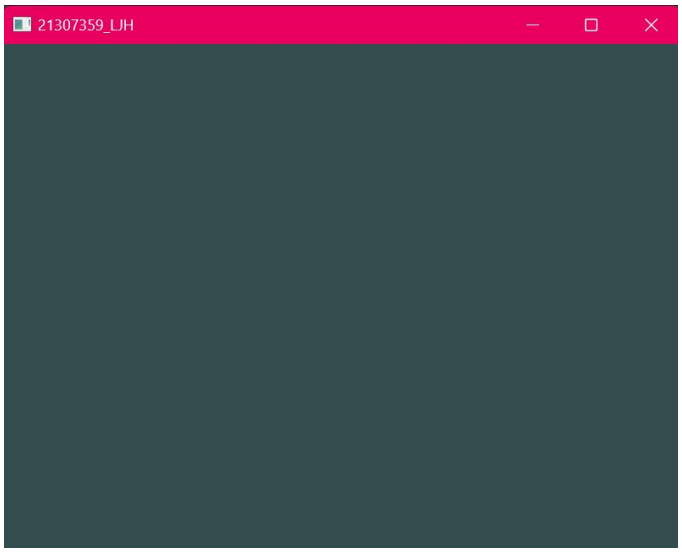
而当 FOV 改成 10.0f 后：



可见其影响着我们视野的宽阔程度。

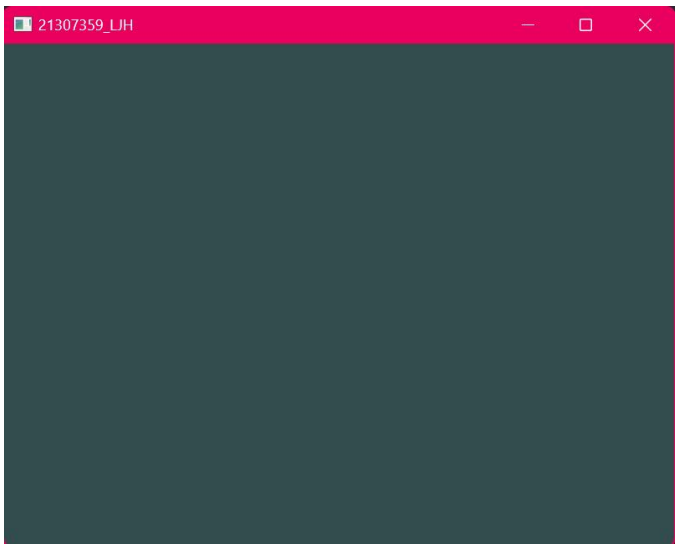
而当将把透视矩阵的 near 值设置太大时（如 10.0f），OpenGL 会将靠近摄像机的坐标（在 0.0f 和 10.0f 之间）都裁剪掉，这会导致（在太过靠近一个物体的时候）视线直接穿过去。

于是什么都看不见（毕竟我们只有一个箱子）：

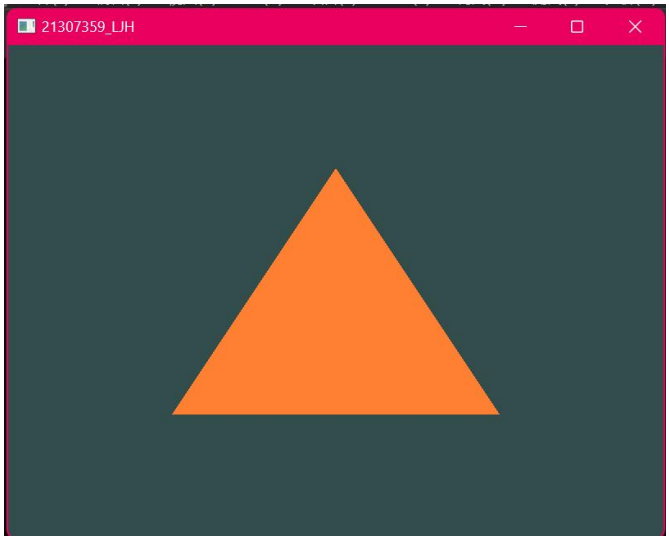


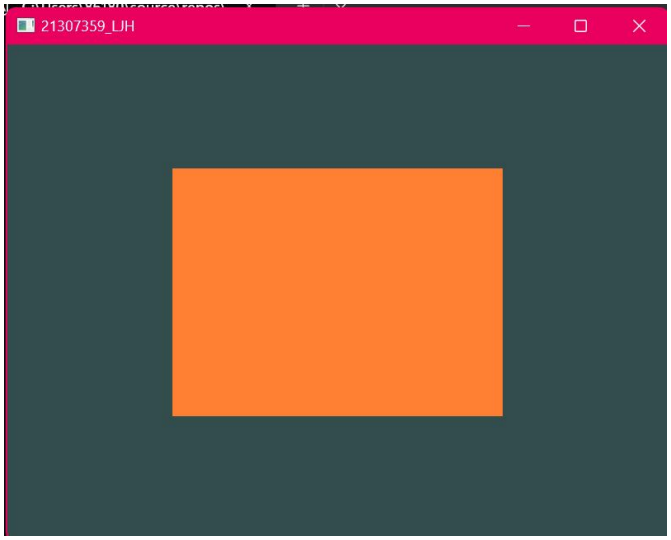
Task 5、请按照顺序将跟着教程实现的运行结果贴出来，要求将运行出来的窗口的标题改成自己的学号。(Tip: glfwCreateWindow 函数)

（1） 你好，窗口

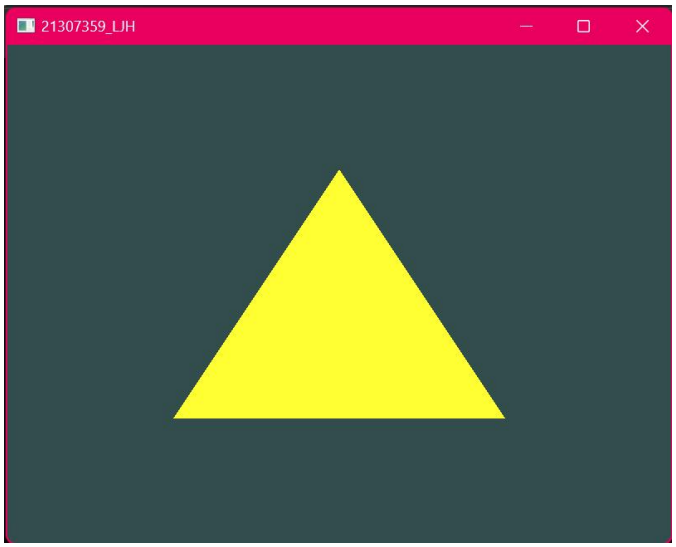


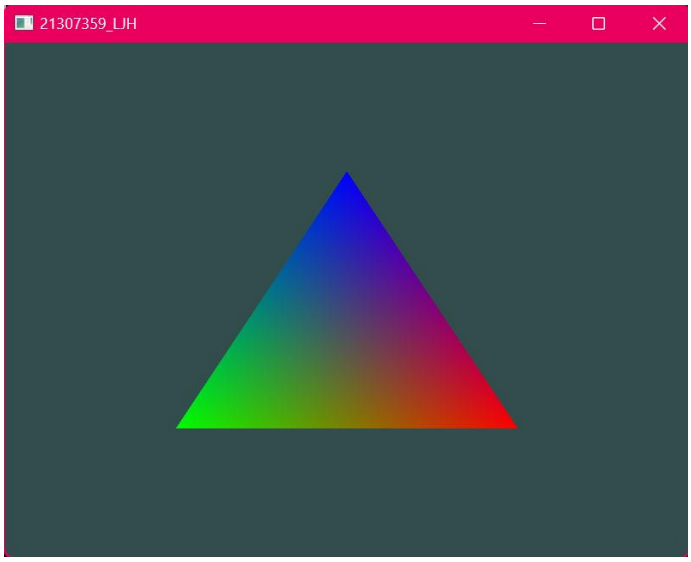
（2） 你好，三角形



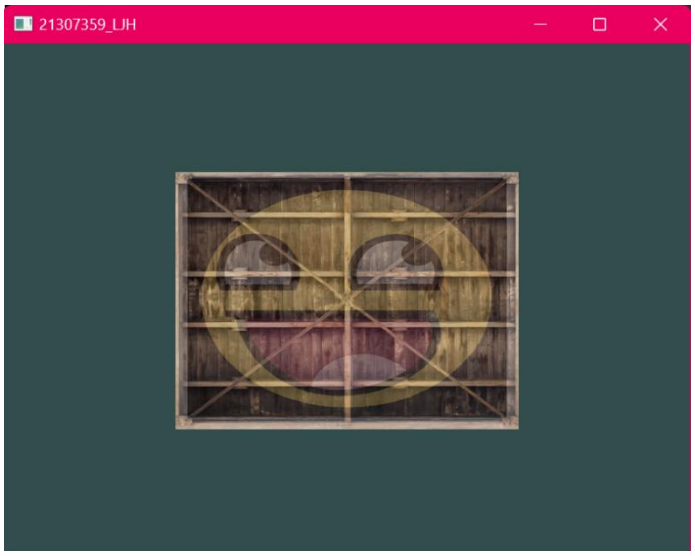


（3） 着色器（尝试了红橙黄颜色的变化，而不是原来的绿色）

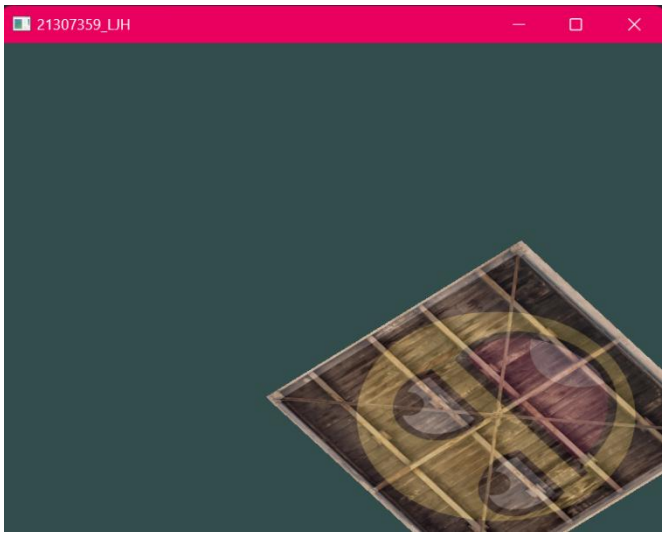




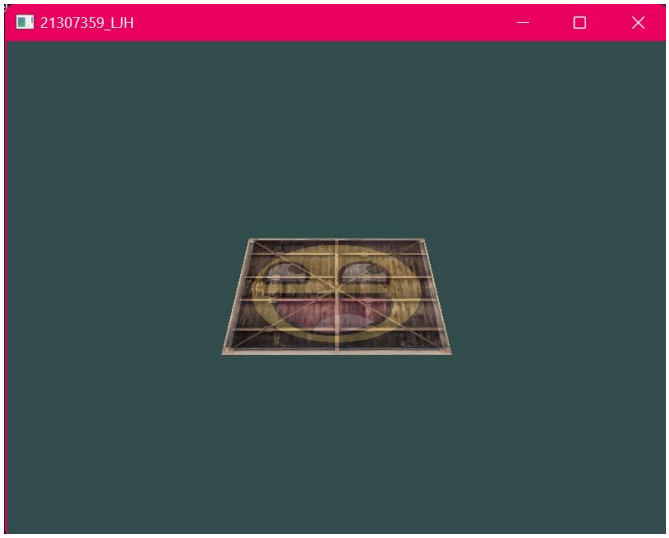
（4） 纹理

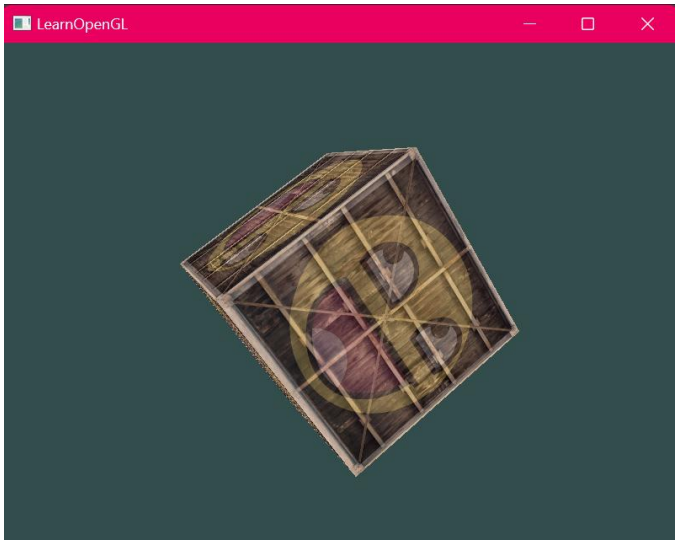


（5） 变换



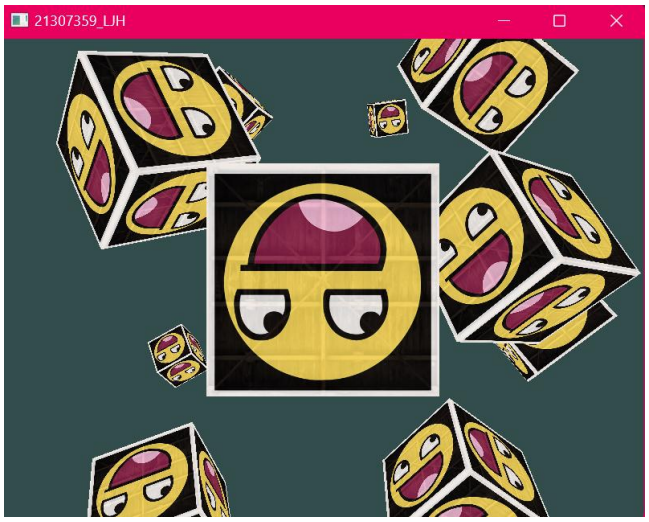
（6） 坐标系统



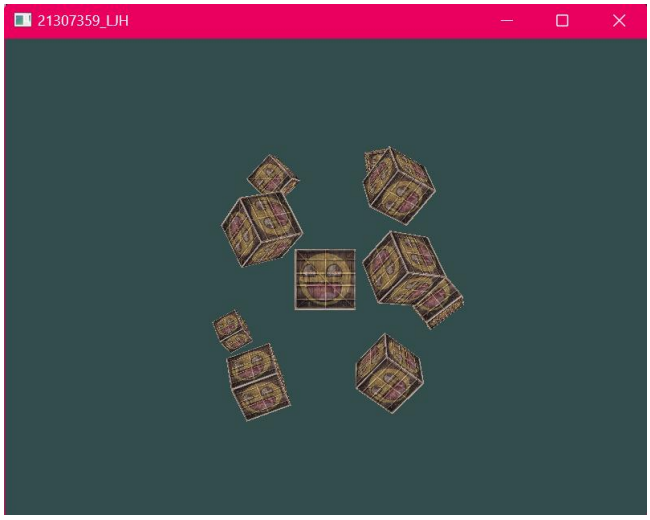


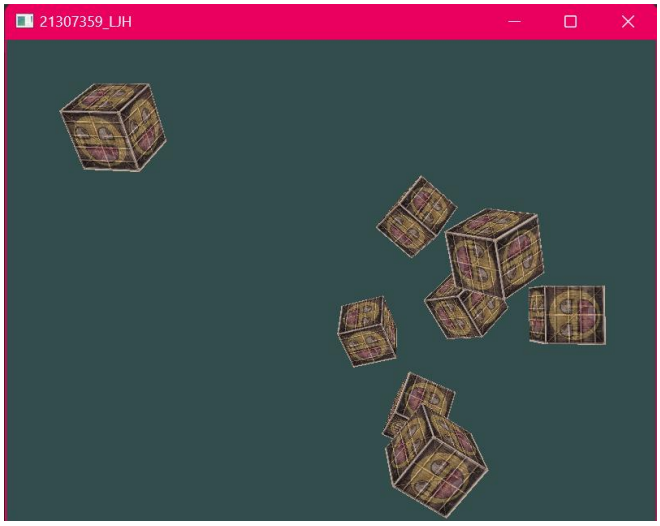
立方体的图形布局已经定义好了，所以当渲染更多物体的时候我们不需要改变我们的缓冲数组和属性数组，我们唯一需要做的只是改变每个对象的模型矩阵来将立方体变换到世界坐标系中。

（btw，要是调整 fs 文件中的 FragColor = mix(texture(texture1, TexCoord), texture(texture2, TexCoord), 0.2);将最后一个表示纹理混合权重的参数改为 0.8，则会得到下面的丑陋结果）



（7） 摄像机





至此，入门教程已经全部完成！