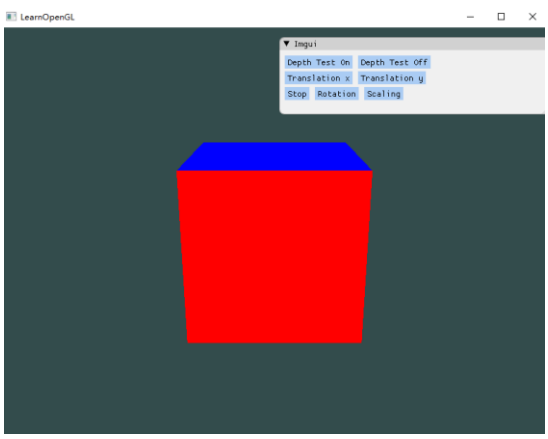
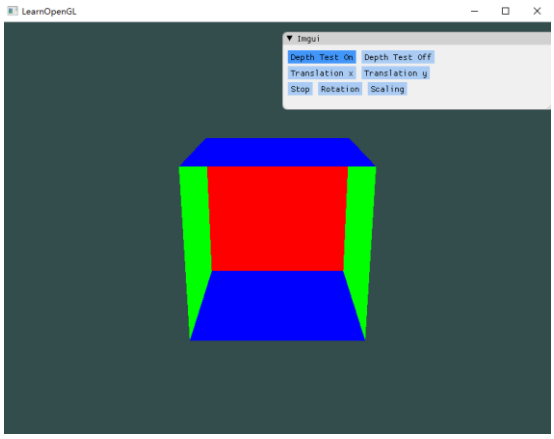
1. 画一个立方体(cube)：边长为 4， 中心位置为(0, 0, 0)。分别启动和关闭深度测试 glEnable(GL_DEPTH_TEST) 、 glDisable(GL_DEPTH_TEST) ，查看区别，并分析原因。

```
glEnable(GL_DEPTH_TEST);
```

```
glDisable(GL_DEPTH_TEST);
```





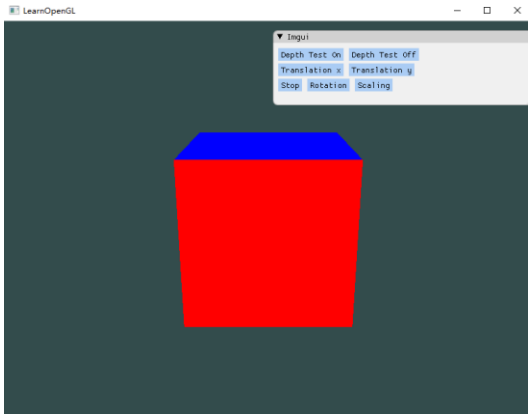
OpenGL 是通过渲染三角形来渲染出立方体的，未开启深度测试的情况下部分三角形渲染顺序不同会导致覆盖。开启深度测试后会按照深度顺序进行渲染。

2. 平移(Translation)：使画好的 cube 沿着水平或垂直方向来回移动。

```
glm::translate(model, (float)sin(glfwGetTime()) * glm::vec3(1.0f, 0.0f, 0.0f));
```

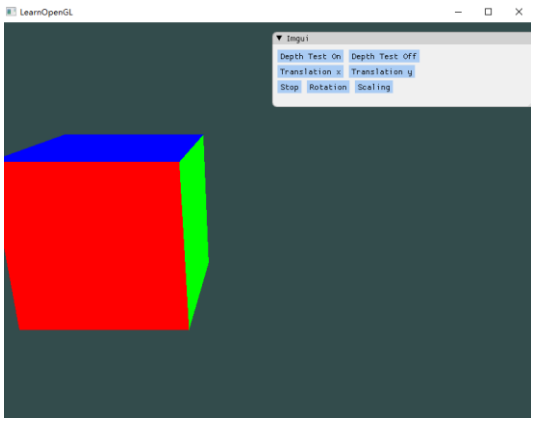
```
glm::translate(model, (float)sin(glfwGetTime()) * glm::vec3(0.0f, 1.0f, 0.0f));
```

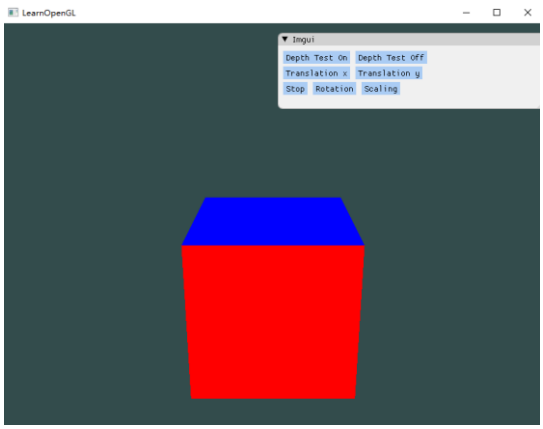
原图



水平移动

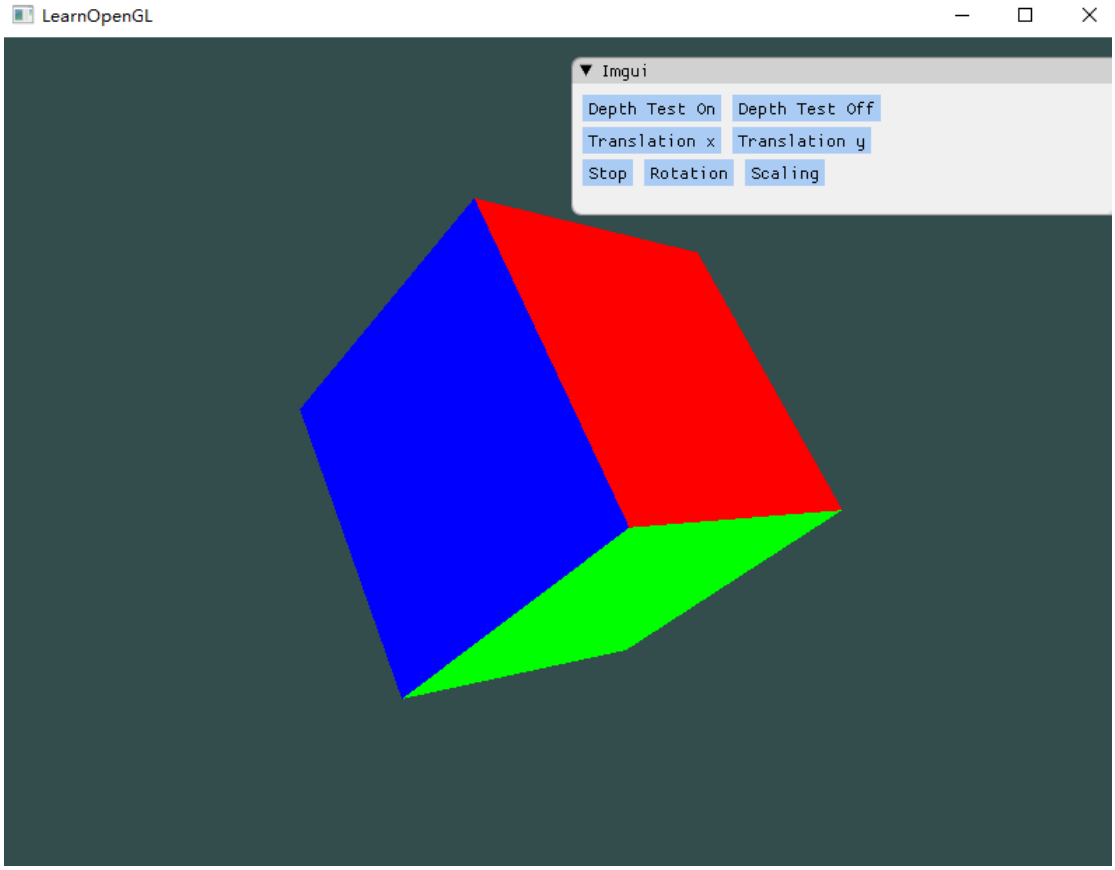
垂直移动





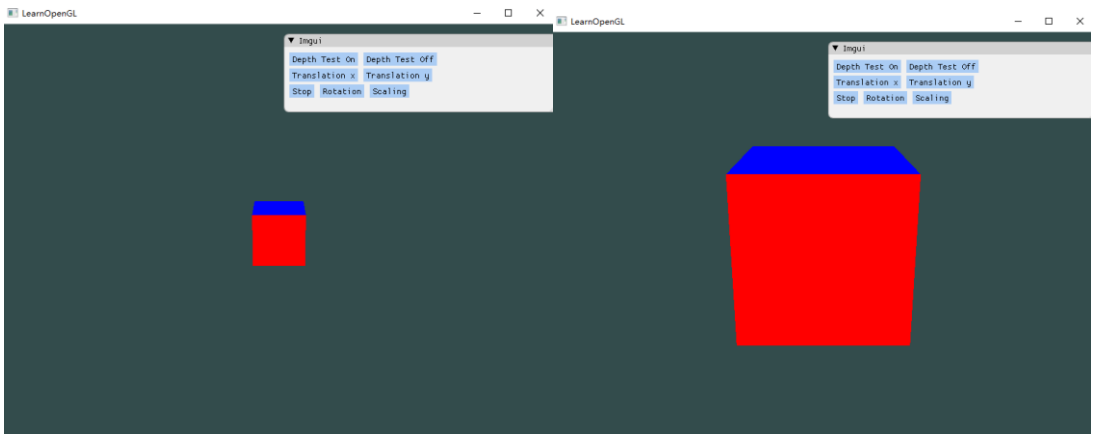
3. 旋转(Rotation)：使画好的 cube 沿着 XoZ 平面的 x=z 轴持续旋转。

```
glm::rotate(model, (float)glfwGetTime(), glm::vec3(1.0f, 0.0f, 1.0f));
```

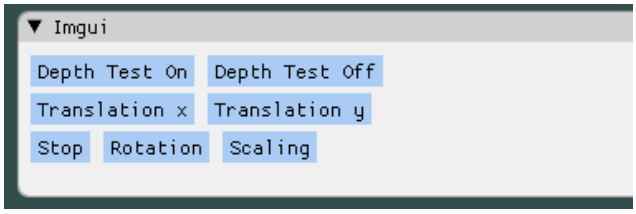


4. 放缩(Scaling)：使画好的 cube 持续放大缩小。

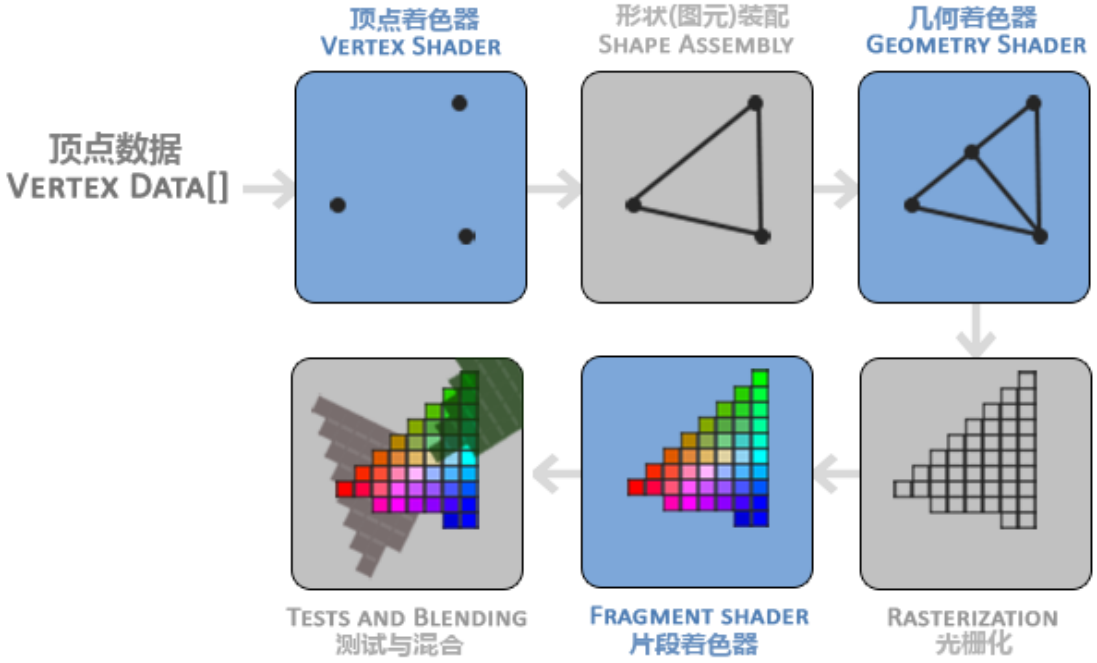
```
glm::scale(model, (float)abs(sin(glfwGetTime())) * glm::vec3(0.5f, 0.5f, 0.5f));
```



5. 在 GUI 里添加菜单栏，可以选择各种变换。



6. 结合 Shader 谈谈对渲染管线的理解



Opengl 的 shader 渲染管线使用一个通道来进行数据渲染。每个步骤只有一个输入及输出。

首先，我们以数组的形式传递 3 个 3D 坐标作为图形渲染管线的输入，用来表示一个三角形，这个数组叫做顶点数据(Vertex Data)；顶点数据是一系列顶点的集合。

图形渲染管线的第一个部分是顶点着色器(Vertex Shader)，它把一个单独的顶点作为输入。顶点着色器主要的目的是把 3D 坐标转为另一种 3D 坐标，同时顶点着色器允许我们对顶点属性进行一些基本处理。

图元装配(Primitive Assembly)阶段将顶点着色器输出的所有顶点作为输入（如果是 GL_POINTS，那么就是一个顶点），并所有的点装配成指定图元的形状。

图元装配阶段的输出会传递给几何着色器(Geometry Shader)。几何着色器把图元形式的一系列顶点的集合作为输入，它可以通过产生新顶点构造出新的（或是其它的）图元来生成其他形状。

几何着色器的输出会被传入光栅化阶段(Rasterization Stage)，这里它会把图元映射为最终屏幕上相应的像素，生成供片段着色器(Fragment Shader)使用的片段(Fragment)。在片段着色器运行之前会执行裁切(Clipping)。裁切会丢弃超出你的视图以外的所有像素，用来提升执行效率。

片段着色器的主要目的是计算一个像素的最终颜色，这也是所有 OpenGL 高级效果产生的地方。通常，片段着色器包含 3D 场景的数据（比如光照、阴影、光的颜色等等），这些数据可以被用来计算最终像素的颜色。

在所有对应颜色值确定以后，最终的对象将会被传到最后一个阶段，叫做 Alpha 测试和混合(Blending)阶段。这个阶段检测片段的对应的深度，用它们来判断这个像素是其它物体的前面还是后面，决定是否应该丢弃。这个阶段也会检查 alpha 值（alpha 值定义了一个物体的透明度）并对物体进行混合(Blend)。