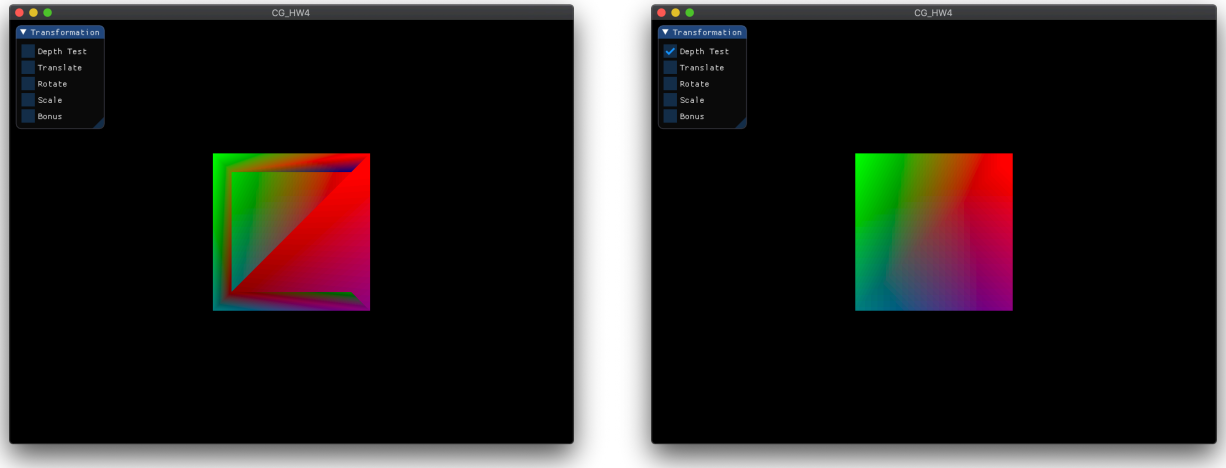
张吉祺 16340286

Basic:

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。



根据**HW2_v0:Basic:1.**的方法定义顶点数组存放点坐标和颜色，并定义顶点索引数组指定顶点顺序用于顶点的重复使用。不同之处是要在三维空间画一个立方体而不是画仅仅一个平面。这里定义的是局部坐标，在之后会变为世界坐标，观察坐标，裁剪坐标，屏幕坐标。这里需要用到几个变换矩阵，最重要的几个分别是模型(Model)、观察(View)、投影(Projection)三个矩阵。

模型矩阵将物体的坐标从局部变换到世界空间，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。

观察矩阵用来将世界坐标变换到观察空间，也称为摄像机空间。

透视投影矩阵起到透视投影效果。

给顶点着色器添加代码：

```
uniform mat4 model;
uniform mat4 view;
uniform mat4 projection;
```

并使其输出位置为：

```
gl_Position = projection * view * model * vec4(aPos, 1.0);
```

使其进行坐标变换。然后要在渲染循环中借助专门为OpenGL量身定做的数学库GLM提供的方法根据需要计算这三个矩阵，例如操作模型矩阵：

```
glm::mat4 model(1.0f);
```

然后查询uniform变量的地址：

```
uint modelLoc = glGetUniformLocation(shaderProgram, "model");
```

然后把矩阵数据发送给着色器：

```
glUniformMatrix4fv(modelLoc, 1, GL_FALSE,
glm::value_ptr(model));
```

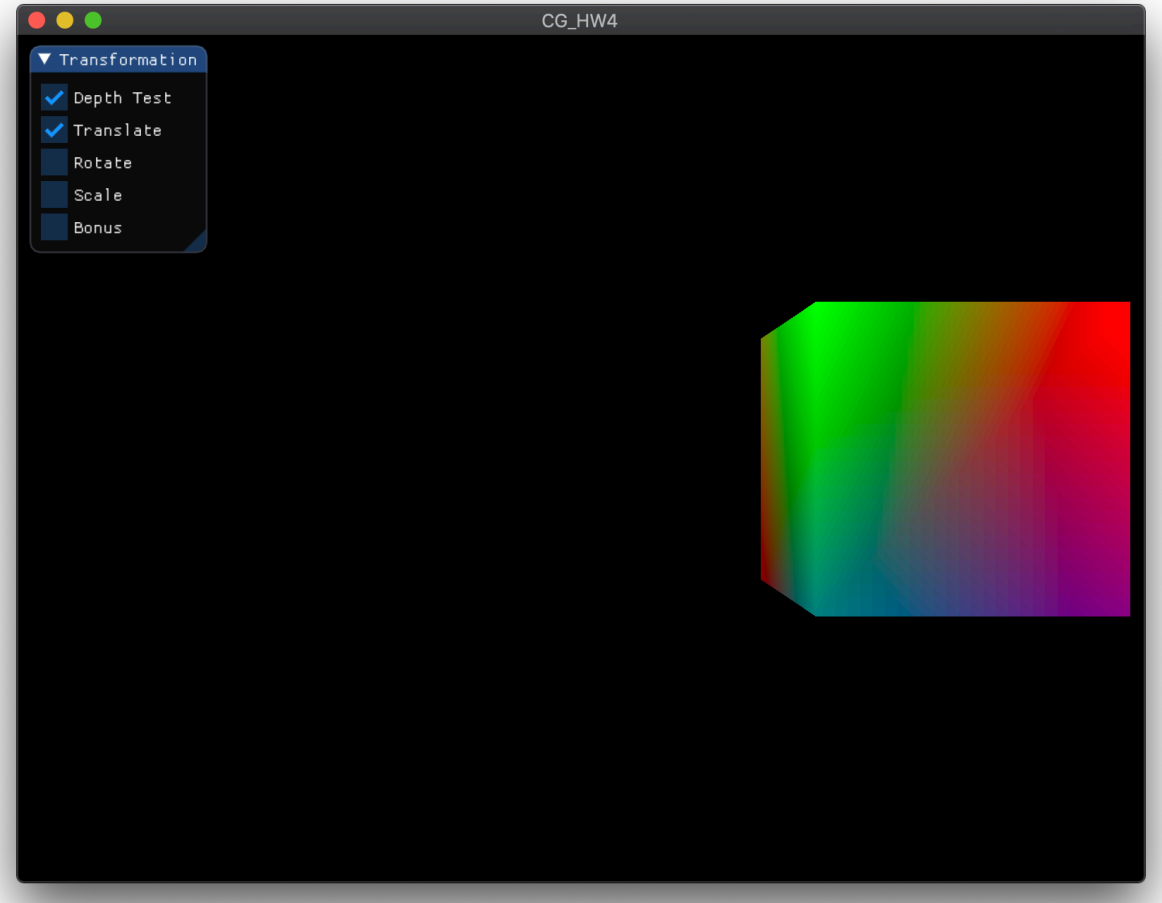
关闭深度测试时，立方体的某些本应被遮挡住的面被绘制在了这个立方体其他面之上，开启深度测试看起来就正常了。

原因是OpenGL是一个三角形一个三角形绘制立方体，会覆盖先绘制的三角形的像素。而开启深度测试，GLFW会为片段生成深度缓冲存储深度信息，当片段想要输出它的颜色时，OpenGL会将它的深度值和深度缓冲进行比较，如果当前的片段在其它片段之后，它将会被丢弃，否则将会覆盖。

这里每次渲染循环之前需要清除深度缓冲：

```
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
```

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

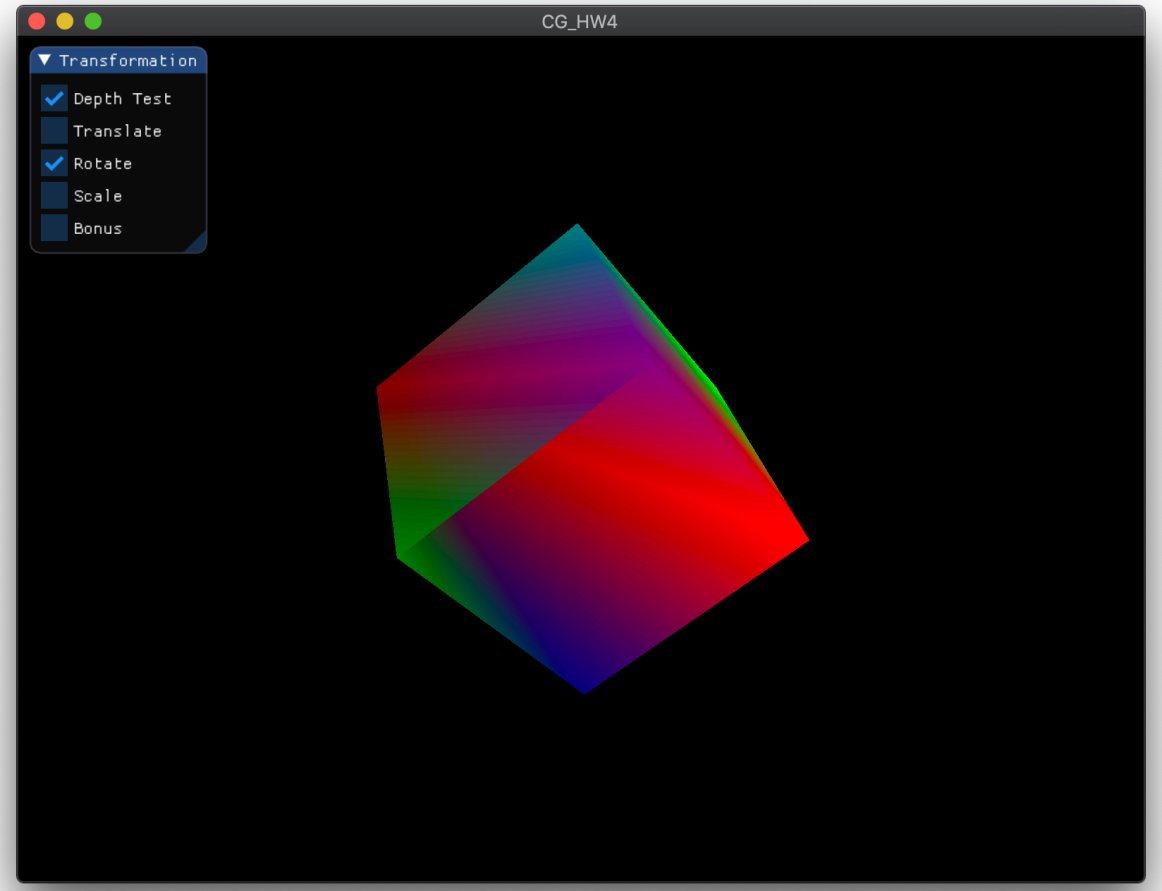


修改**Basic:1.**渲染循环中的模型矩阵：

```
model = glm::translate(model,
glm::vec3(sin((float)glfwGetTime()) * 5.0f, 0.0f, 0.0f));
```

用`glm::translate`方法让其平移，利用sin函数在[-1, 1]摆动的特性让其在x轴上来回移动，并对其乘5扩大到[-5, 5]。

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

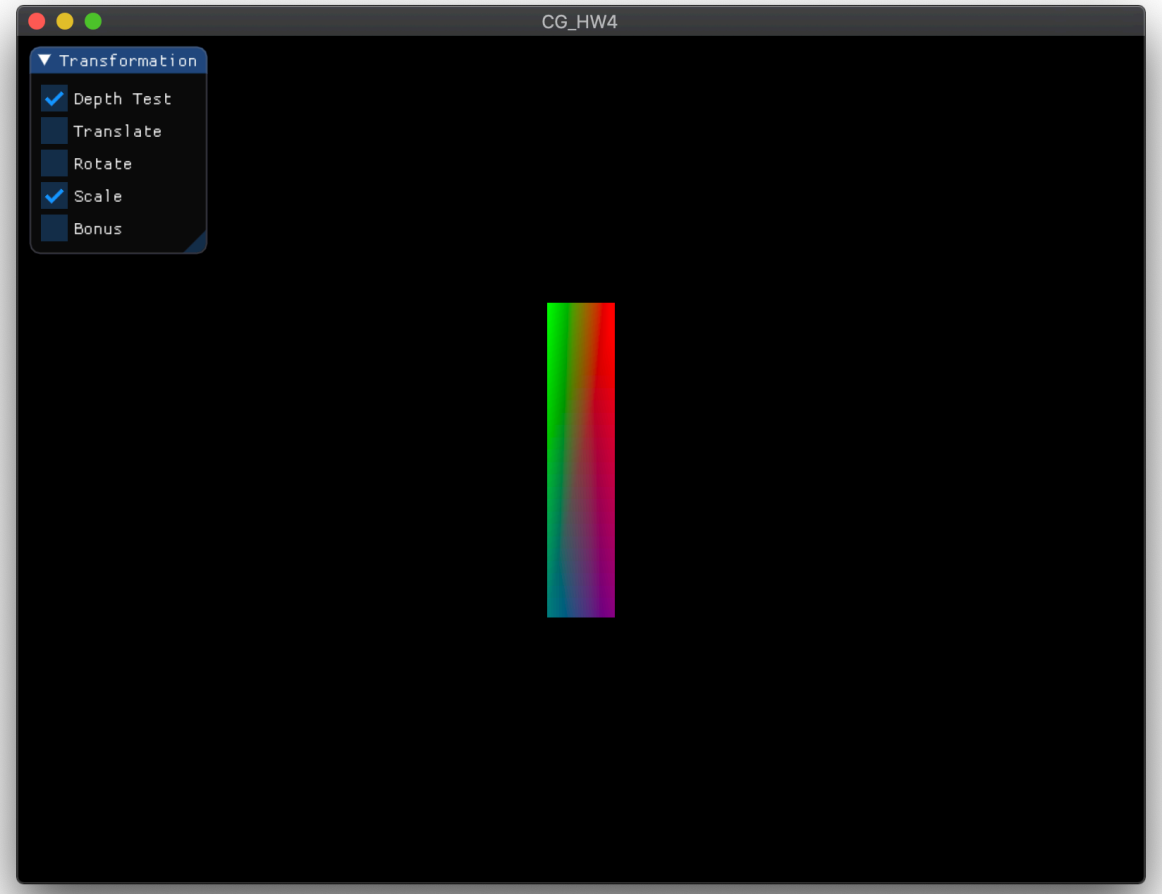


修改**Basic:1.**渲染循环中的模型矩阵：

```
model = glm::rotate(model, (float)glfwGetTime() * glm::radians(50.0f), glm::vec3(1.0f, 0.0f, 1.0f));
```

用`glm::rotate`方法让其旋转，设置旋转轴为(1, 0, 1)让其在x=z轴上旋转。

4. 放缩(Scaling):使画好的cube持续放大缩小。



修改**Basic:1.**渲染循环中的模型矩阵：

```
model = glm::scale(model,
glm::vec3(sin((float)glfwGetTime()) + 1.2f, 1.0f, 1.0f));
```

用`glm::scale`方法让其放缩，利用sin函数在[-1, 1]摆动的特性让其在x轴上来回放缩，并对其加1.2调整到[0.2, 2.2]。

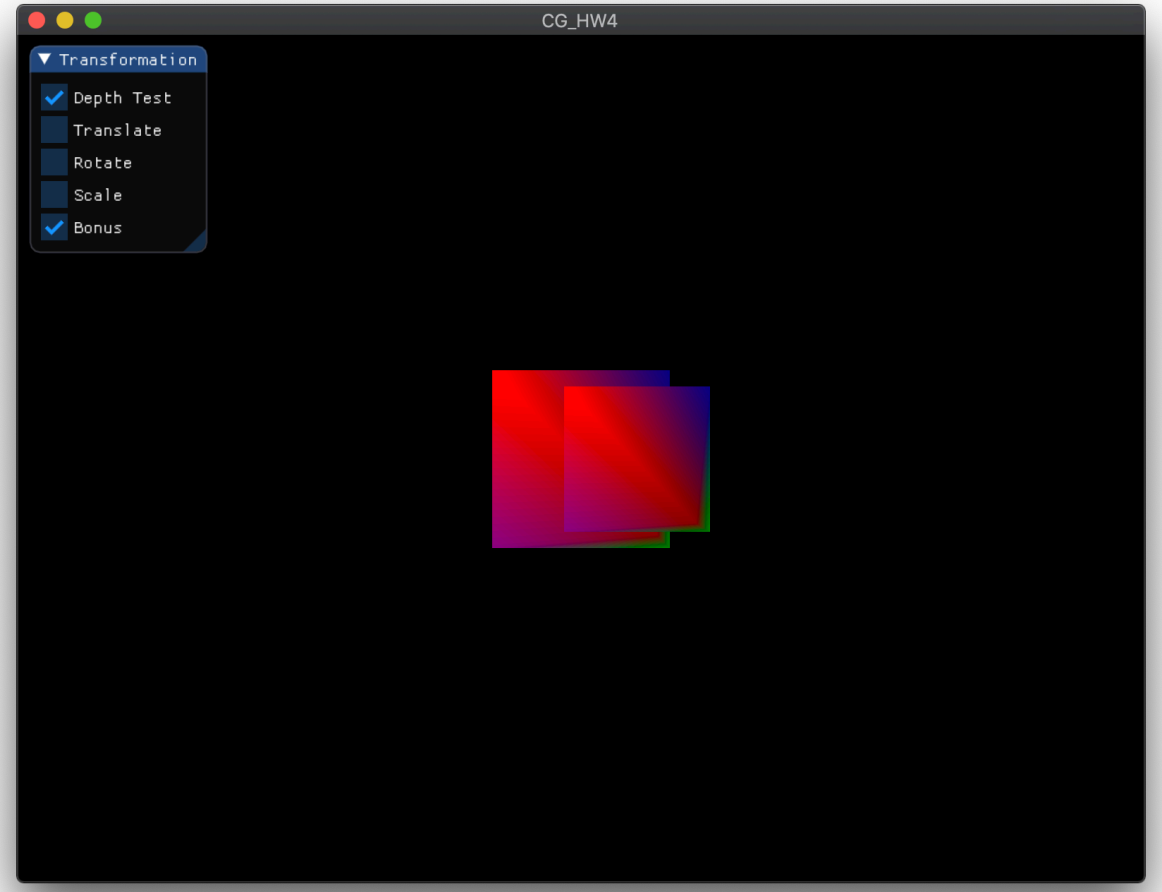
5. 结合Shader谈谈对渲染管线的理解

渲染管线是把三维坐标转变为屏幕的二维像素的处理过程，可以被划分为两个主要部分：第一部分把3D坐标转换为2D坐标，第二部分是把2D坐标转变为实际的有颜色的像素。渲染管线被划分为高度专门化的并行执行的几个阶段，每个阶段将会把前一个阶段的输出作为输入。Shader是GPU上为渲染管线各阶段独立运行的小程序，其中，GPU没有默认的顶点着色器和片段着色器需要定义，其他阶段都有默认的Shader。

张吉祺 16340286

Bonus:

1. 将以上三种变换相结合，打开你们的脑洞，实现有创意的动画。比如:地球绕太阳转等。



利用**Basic:2.-4.**的变换，先用缩放方法计算出模型矩阵，目的是调整“太阳”大小，再对这个模型矩阵用旋转方法计算，设置旋转轴为(0, 1, 0)， 使“太阳”自转，调用 `glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT, 0);` 渲染；再定义一个单位模型矩阵，先用缩放方法计算，缩小“地球”，再用平移方法计算，按x=-cos(time)，z=sin(time)的方式平移，使其在x-z平面上沿圆形轨迹逆时针平移，使“地球”公转，再用旋转方法计算，使“地球”自转，再次调用 `glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT, 0);` 渲染，这里跟“太阳”共用了前面定义好的顶点和索引信息。