启动和关闭深度测试的区别及原因

区别：启动深度测试后会将位于背面的不可见的部分画面丢弃；而关闭深度测试后，背面的图像可能会绘制在正面的图像上面。

原因：OpenGL是一个三角形一个三角形地来绘制我们的立方体的，所以即便之前那里有东西它也会覆盖之前的像素。开启深度测试后，GLFW会自动生成一个深度缓冲。深度值存储在每个片段里面（作为片段的**z**值），当片段想要输出它的颜色时，OpenGL会将它的深度值和z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。

结合Shader谈谈对渲染管线的理解

在OpenGL中，任何事物都在3D空间中，而屏幕和窗口却是2D像素数组，这导致OpenGL的大部分工作都是关于把3D坐标转变为适应你屏幕的2D像素。3D坐标转为2D坐标的处理过程是由OpenGL的图形渲染管线管的。

图形渲染管线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。

图形渲染管线可以被划分为几个阶段，每个阶段将会把前一个阶段的输出作为输入。所有这些阶段都是高度专门化的，并且很容易并行执行。正是由于它们具有并行执行的特性，当今大多数显卡都有成千上万的小处理核心，它们在GPU上为每一个渲染管线阶段运行各自的小程序，从而在图形渲染管线中快速处理你的数据。这些小程序叫做着色器(Shader)。

实现思路

1. 绘制立方体：

   要想渲染一个立方体，一共需要36个顶点（6个面 x 每个面有2个三角形组成 x 每个三角形有3个顶点），将传入VBO的数据修改为这36个顶点即可。为了使立方体的6个面便于区分，用不同的颜色绘制6个面，通过Uniform将颜色传递给片段着色器：

   ```
   float colorX[6] = { 1.0f, 0.0f, 0.0f, 1.0f, 1.0f, 0.0f };
   float colorY[6] = { 0.0f, 1.0f, 0.0f, 1.0f, 0.0f, 1.0f };
   float colorZ[6] = { 0.0f, 0.0f, 1.0f, 0.0f, 1.0f, 1.0f };
   for (int i = 0; i < 6; i++) {
       loc = glGetUniformLocation(shaderProgram, "ourColor");
       glUniform4f(loc, colorX[i], colorY[i], colorZ[i], 1.0f);
       glDrawArrays(GL_TRIANGLES, 6 * i, 6);
   }
   ```

2. 坐标变换

   通过Uniform，将model、 view、 projection 3个矩阵传递给顶点着色器。其中model矩阵将局部空间转换为世界空间（对物体进行缩放、位移和旋转）；view矩阵将世界空间转换为观察空间（我们想要在场景里面稍微往后移动，以使得物体变成可见的，而将摄像机向后移动，和将整个场景向前移动是一样的。因此view矩阵将场景沿着z轴负方向移动一段距离）；projection 矩阵使用透视投影将观察空间转换为裁剪空间。在顶点着色器中，将原坐标和这3个矩阵从右向左进行矩阵乘法运算即可。

3. GUI

将缩放、位移、旋转等参数的变量绑定至GUI中，实现手动进行各种变换。给GUI添加一个Checkbox，控制是否开启深度测试。使用glfwGetTime()函数获取随时间而改变的数值，控制立方体自动平移、旋转和缩放。