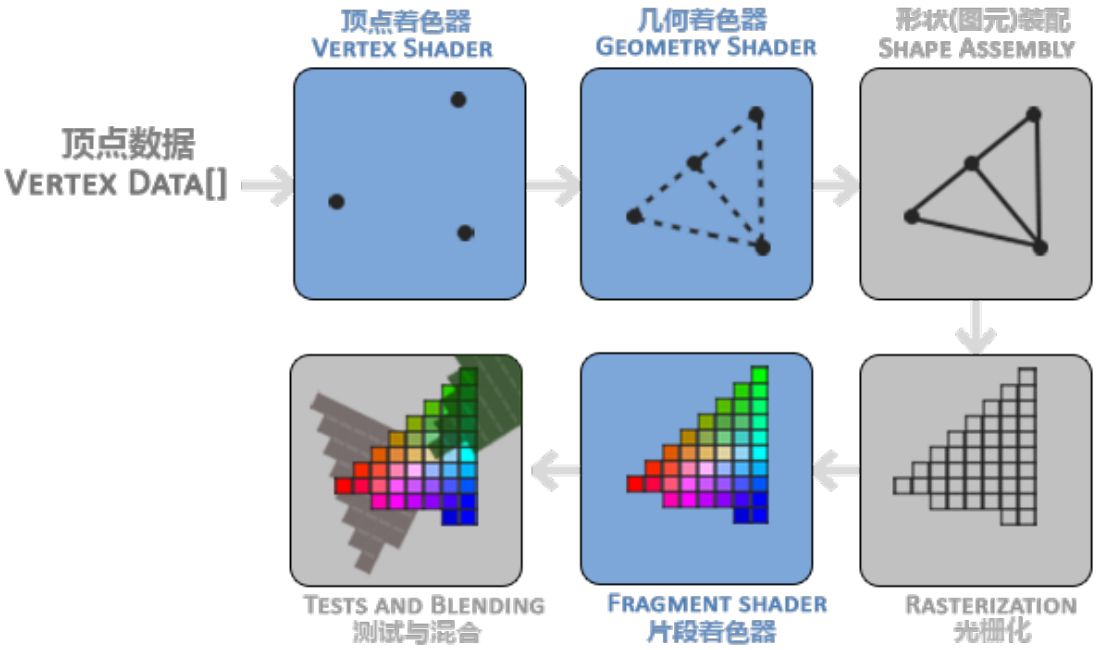
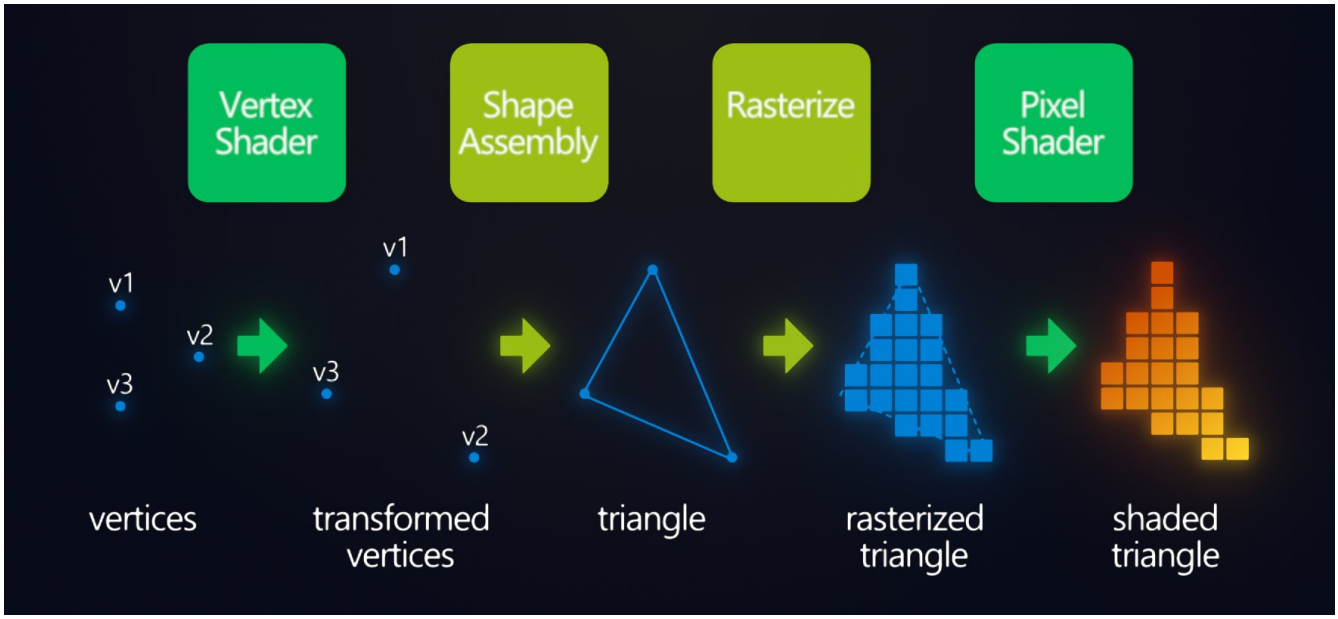
在OpenGL中，所有的事物都在3D空间，屏幕和窗口都是2D像素数组

图形渲染管线（Graphics Pipeline）主要包括两个部分：

第一部分把你的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





着色器

着色器是仅用于硬件管道中一个阶段的微型程序

着色器程序则是包含多个链接在一起的着色器的 GPU 程序

1、加载顶点着色器文件和片段着色器文件，并分别存储在单独的 C 字符串中

2、调用glCreateShader两次；针对 1 个顶点和 1 个片段着色器索引

3、调用glShaderSource从字符串中复制上述每一项的代码

4、对两个着色器索引调用glCompileShader

5、调用glCreateProgram创建新程序的索引

6、调用glAttachShader两次，将两个着色器索引附加到程序

7、调用glLinkProgram

8、调用glGetUniformLocation获取名为“ inputColour ”的变量的唯一位置

9、在调用之前，先调用glUseProgram切换到着色器...

10、glUniform4f(location, r,g,b,a)为片段着色器分配初始颜色（例如 glUniform4f(colour_loc, 1.0f, 0.0f, 0.0f, 1.0f)表示红色）

https://antongerdelan.net/opengl/shaders.html

顶点着色器

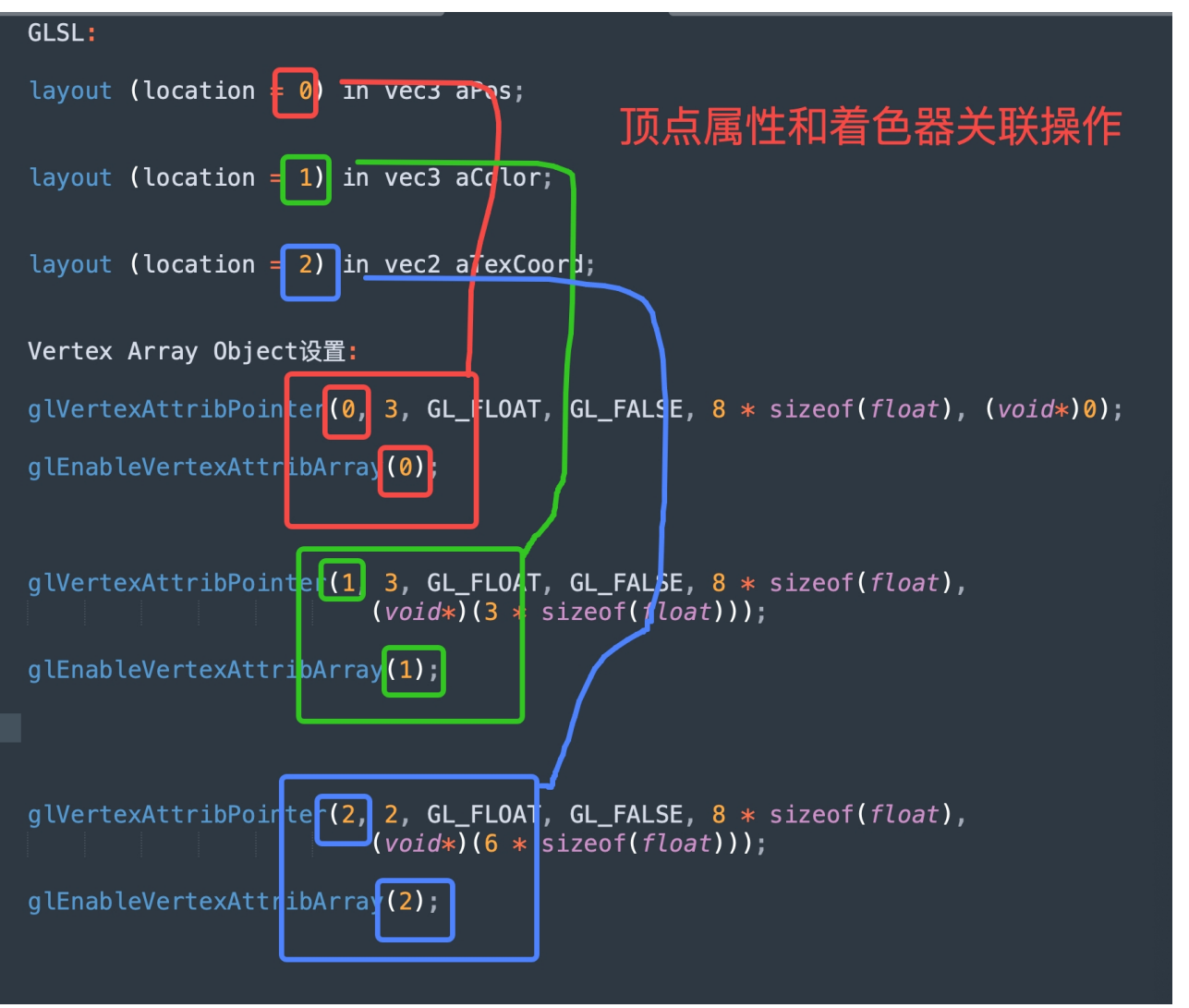
一个顶点(Vertex)是一个3D坐标的数据的集合。而顶点数据是用顶点属性(Vertex

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

Vertex Data Buffer

```
生成 绑定 加载数据
glGenBuffers(1, &VBO);
glBindBuffer(GL_ARRAY_BUFFER, VBO);
glBufferData(GL_ARRAY_BUFFER, sizeof(vertices), vertices, GL_STATIC_DRAW);
```

每个顶点属性从一个VBO管理的内存中获得它的数据，而具体是从哪个VBO（程序中可以有多个VBO）获取则是通过在调用glVertexAttribPointer时绑定到GL_ARRAY_BUFFER的VBO决定的。由于在调用glVertexAttribPointer之前绑定的是先前定义的VBO对象，顶点属性0现在会链接到它的顶点数据。

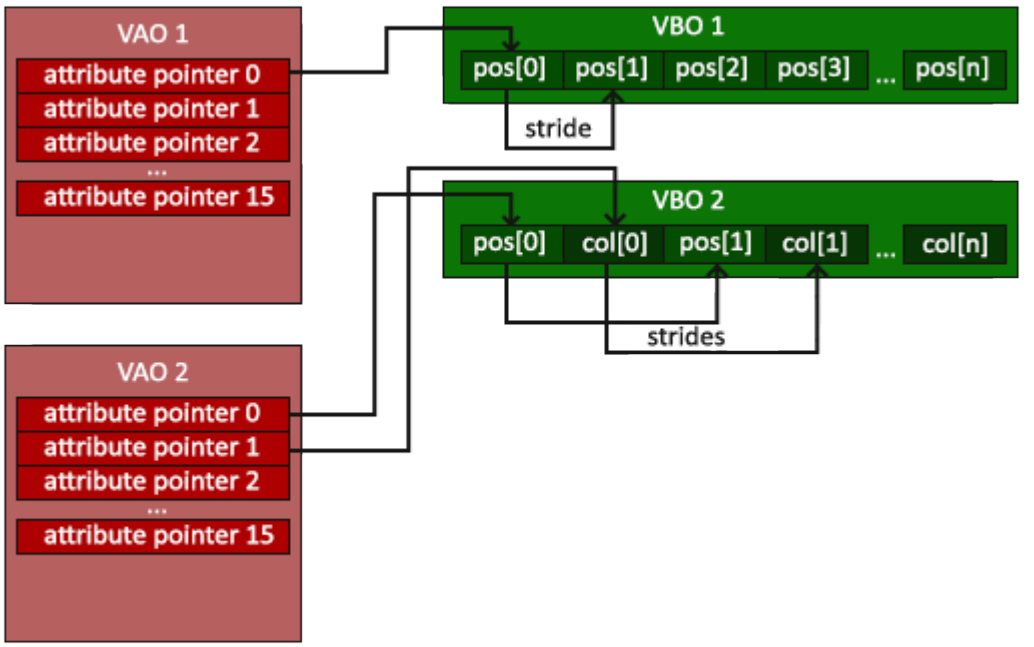


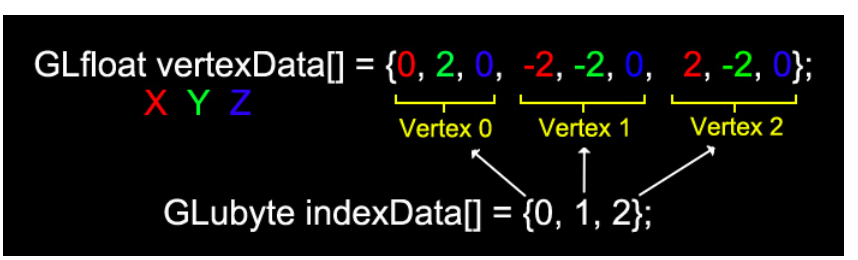
glVertexAttribPointer详解

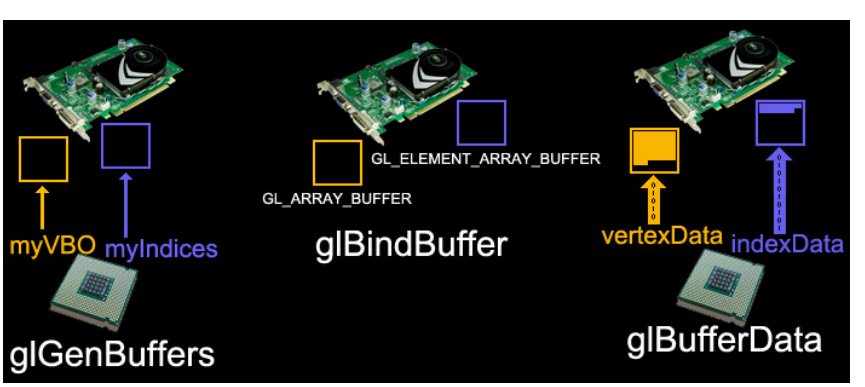
第三个参数指定数据的类型，这里是GL_FLOAT(GLSL中vec*都是由浮点数值组成的)

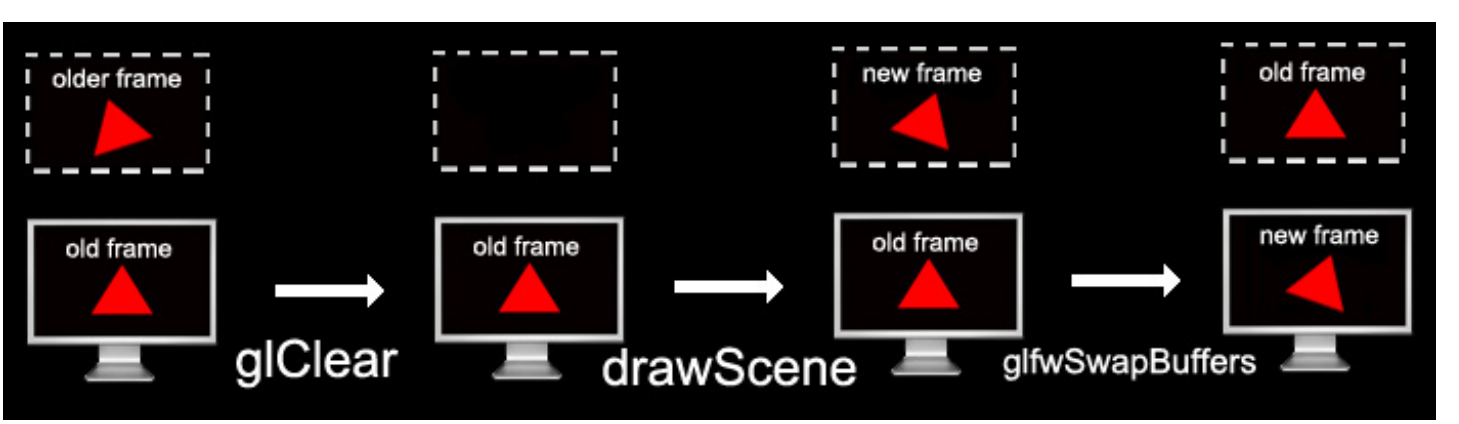
数据被标准化(Normalize)。如果我们设置为GL_TRUE，所有数据都会被映射到0（对于有符号型signed数据是-1）到1之间。我们把它设置为GL_FALSE

Vertex Array Object

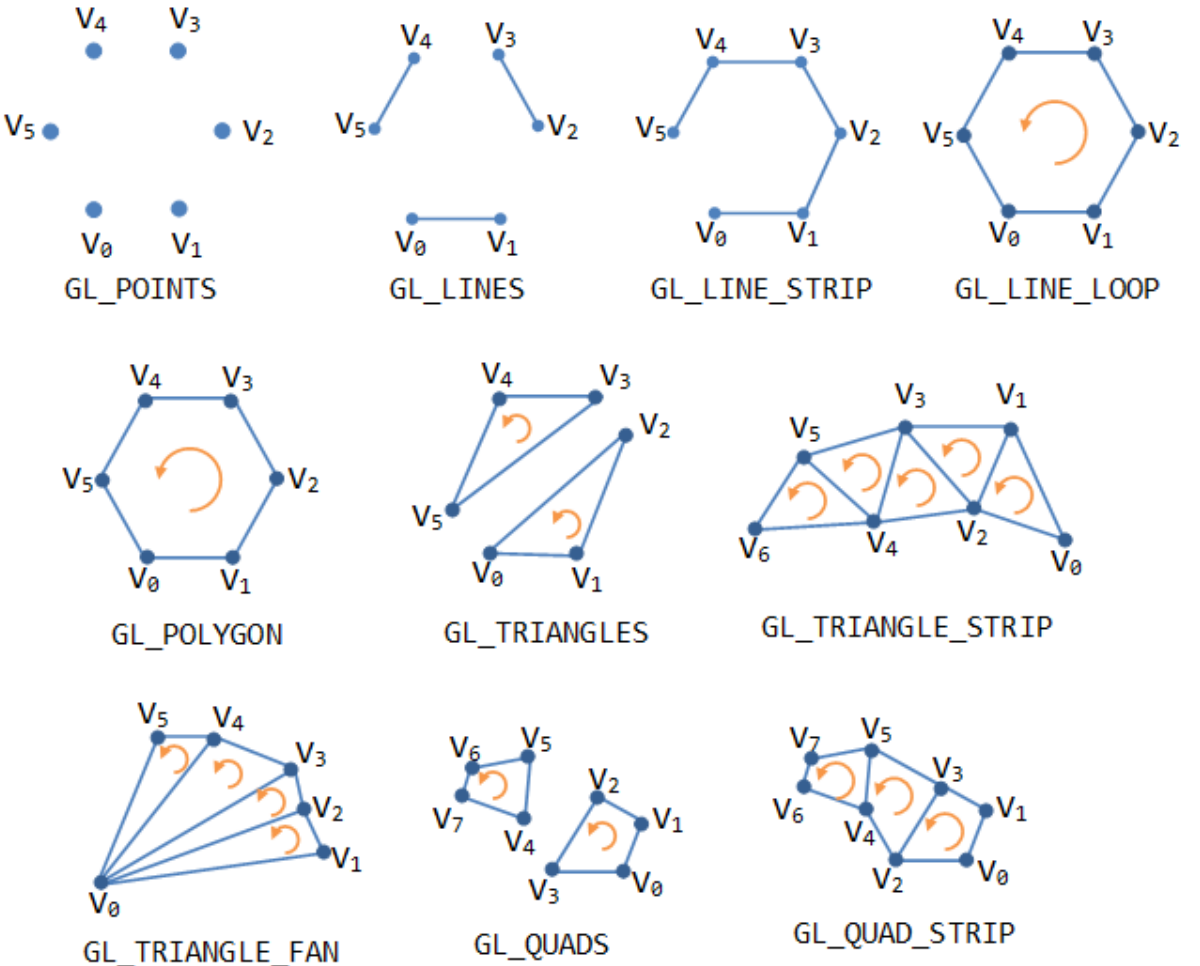








图元装配

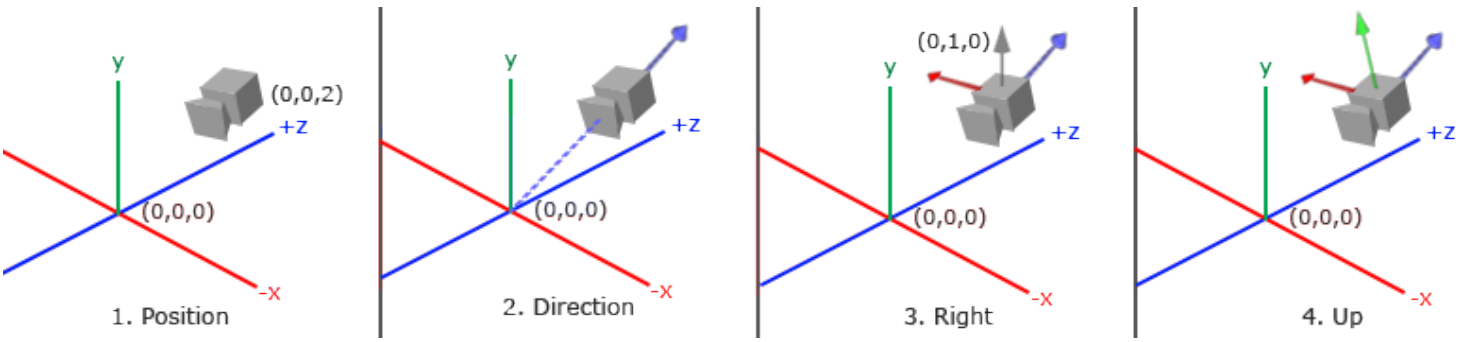


OpenGL Primitives

坐标变换

Camera

要定义一个摄像机，我们需要它在世界空间中的位置、观察的方向、一个指向它右侧的向量以及一个指向它上方的向量



不要忘记正z轴是从屏幕指向你的，如果我们希望摄像机向后移动，我们就沿着z轴的正方向移动。

1、摄像机位置

摄像机位置简单来说就是世界空间中一个指向摄像机位置的向量

```
glm::vec3 cameraPos = glm::vec3(0.0f, 0.0f, 3.0f);
```

2、摄像机方向

让摄像机指向场景原点：(0, 0, 0)

用场景原点向量减去摄像机位置向量的结果就是摄像机的指向向量

```
//摄像机指向场景原点
glm::vec3 cameraTarget = glm::vec3(0.0f, 0.0f, 0.0f);
glm::vec3 cameraDirection = glm::normalize(cameraPos - cameraTarget);
```

方向向量(Direction Vector)并不是最好的名字，因为它实际上指向从它到目标向量的相反方向

3、右轴

我们需要的另一个向量是一个右向量(Right Vector)，它代表摄像机空间的x轴的正方向。为获取右向量我们需要先使用一个小技巧：先定义一个上向量(Up Vector)

```
glm::vec3 up = glm::vec3(0.0f, 1.0f, 0.0f);
glm::vec3 cameraRight = glm::normalize(glm::cross(up, cameraDirection));
```

4、上轴

在我们已经有了x轴向量和z轴向量，获取一个指向摄像机的正y轴向量就相对简单了：我们把右向量和方向向量进行叉乘

glm::vec3 cameraUp = glm::cross(cameraDirection, cameraRight);

LookAt

$$LookAt = \begin{bmatrix} R_x & R_y & R_z & 0 \\ U_x & U_y & U_z & 0 \\ D_x & D_y & D_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 & -P_x \\ 0 & 1 & 0 & -P_y \\ 0 & 0 & 1 & -P_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

其中R是右向量，
U是上向量，
D是方向向量，
P是相机的位置向量。
请注意，位置向量是反转的，因为我们最终想要将世界平移到与移动方向相反的方向

```
//创建一个观察矩阵
glm::mat4 view;
view = glm::lookAt(glm::vec3(0.0f, 0.0f, 3.0f), //摄像机位置
           glm::vec3(0.0f, 0.0f, 0.0f), //目标位置
           glm::vec3(0.0f, 1.0f, 0.0f));//世界空间的上向量
```

光照

冯氏光照模型(Phong Lighting

冯氏光照模型的主要结构由3个分量组成：环境(Ambient)、漫反射(Diffuse)和镜面(Specular)光照