GL—》画点—》染色

准备材料（点，颜色等）

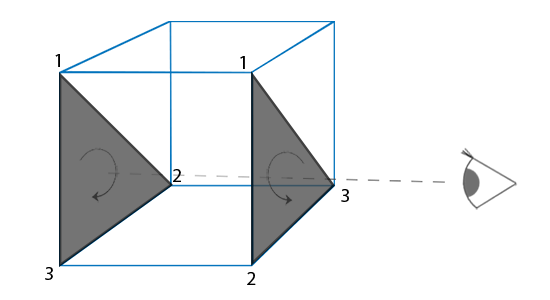
生成矩阵VAO(画点) 生成VBO EBO 绑定矩阵VAO 绑定缓冲VBO 绑定缓冲数据 绑定缓冲EBO 绑定缓冲数据 获取矩阵数据，解绑，绑定VAO 画点 删除 释放内存

光照:呈现色彩（ax f ,by f, cz f）=光源（a f, b f,c f）\*被照射物体的颜色（x f,y f,z f）

要想让混合在多个物体上工作，我们需要最先绘制最远的物体，最后绘制最近的物体：

1. 先绘制所有不透明的物体。
2. 对所有透明的物体排序。
3. 按顺序绘制所有透明的物体

环绕顺序：眼见即为逆时针



glEnable(GL\_CULL\_FACE); 面剔除开关

glCullFace函数有三个可用的选项：

* GL\_BACK：只剔除背向面。
* GL\_FRONT：只剔除正向面。
* GL\_FRONT\_AND\_BACK：剔除正向面和背向面。

glFrontFace(GL\_CCW);逆时针 glFrontFace(GL\_CW);顺时针

渲染缓冲对象能为帧缓冲对象提供一些优化。通常的规则是，如果不需要从一个缓冲中采样数据，那么对这个缓冲使用渲染缓冲对象会是明智的选择。如果需要从缓冲中采样颜色或深度值等数据，那么应该选择纹理附件。性能方面它不会产生非常大的影响的。

使用光栅化的方法得知[纹理坐标](https://www.zhihu.com/search?q=%E7%BA%B9%E7%90%86%E5%9D%90%E6%A0%87&search_source=Entity&hybrid_search_source=Entity&hybrid_search_extra={"sourceType":"answer","sourceId":799333394}" \t "https://www.zhihu.com/question/_blank)和顶点色：

1. Edge walking，走cpu，将网格线的三角形分为两个平底梯形；再按行迭代根据左右两边端点算出左右线段和水平线，进行插值计算出两边交点纹理坐标；再按像素迭代，按上步的左右端点和水平线进行按点插值
2. Edge equation,走gpu，按三角形的外接矩形进行两层for循环迭代每个像素点；参考三角以及重心点坐标来插值，计算量大但可并行处理，计算单元可独立渲染

glm::mat4 view = glm::mat4(glm::mat3(camera.GetViewMatrix()));

通过取4x4矩阵左上角的3x3矩阵来移除变换矩阵的位移部分，来移除任何的位移，但保留旋转变换（天空盒操作）