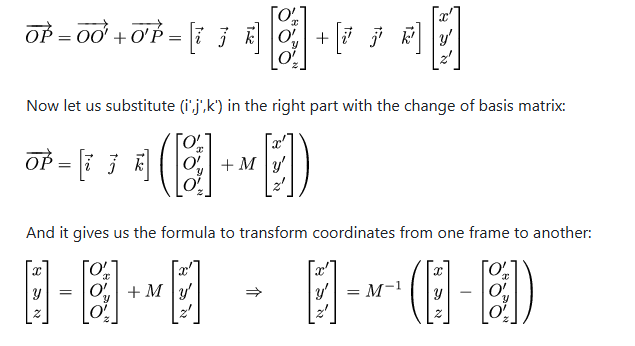
C++软光栅渲染

1. 使用布雷斯汉姆算法实现线段绘制以优化性能，主题思想是提前乘除法，以近似减少浮点数运算。如果x增加1那么y会增加y1-y0/x1-x0，但是此时我们不立刻进行增加而是让这个值进行累积当这个值大于0.5那么y+1。
2. 判断点是否在一个三角形内部的方法有两种，首先非图形学领域中常用的矢量叉积法，直接判断向量叉积的符号是否相同如果相同那么视为在三角形内，其次是现代渲染管线中使用的重心坐标法，计算目标点的重心坐标，如果重心坐标的值均大于0那么则在三角形内部，这个方法在整个过程中只计算了一次叉乘，效率更高的同时还能得到用于插值的参数。
3. 光栅化常见的方法有两种，一种是优于CPU的扫描线，另一种是利于GPU并行计算的包围盒。
4. 背面剔除，如果光线和法向量的乘积为负数那么表明光线来源于模型的背面此时我们可以剔除这些不可见的三角形以优化性能
5. Z-buffer的作用在于判断像素的覆盖关系，其值是通过三角形顶点插值来进行计算的。其中会记录所有坐标的当前距离相机的距离。
6. 纹理也是通过重心坐标插值来进行计算的。注意重心坐标计算的结果
7. 缩放和切变都是线性变换直接通过矩阵乘法来获得，旋转也是线性变换，任意的旋转都可以通过三次切变来实现，平移通过在线性变换处理完成后加上一个矩阵来实现。但是这样的做法并不优雅如果我们希望将所有的变换都放到一个一个矩阵中进行呢？引入齐次坐标在原本的坐标向量中增加一个1。齐次坐标除了用来统一计算方式外
8. 世界空间移动到观察空间：在OpenGL的渲染的过程中相机的位置通常是不会发生改变的,但是我们需要在相机的视角下进行光照裁剪投影之类的操作如果不进行移动那么后续的每一次计算都需要考虑摄像机的视角问题，所以我们可以直接将模型移动到摄像机空间以优化后续计算。

如何确定摄像机坐标：将相机与原点的连线作为v轴，取世界坐标的vector.up，与其叉乘得到r轴，在将v轴与r轴叉乘得到u轴。

通过计算可知实际上每一个点的移动都只需要乘轴移动的逆矩阵。



1. 三种最经典的着色模型

平面着色：整个图元都填充相同的颜色，效率最高但是效果最烂

高洛德着色：片元的所有数据都插值进行计算

冯氏着色：只插值计算片元的法向量，其他数据都通过计算得出，效果最好但是效率比较低。

10.反射向量的计算公式：入射光线 - 2\*（入射光线 \* 法向量）\*法向量

11.法线贴图用于采样得到片元具体的法线以便于实现计算每一个片元的实际的光照等信息，只有在冯氏着色中会使用，前面的两种着色方式都不需要使用片元的法向量，只需要在顶点处使用法向量来计算颜色，然后在片元处进行插值即可。