# 一、优化习惯与技巧

## 矩阵计算

•逆矩阵的计算非常耗费资源，应该避免使用GPU计算逆矩阵，或者说是在着色器中计算，而应该在主程序中使用CPU计算，然后通过uniform传给着色器；这种方式类似计算模型视点等矩阵，然后传送给着色器的方法

**计算多边形法向量（Martin Newell[Newell79]）**

p284

正多边形

**圆和圆弧**

1.三种画圆的算法

圆心和半径

圆心和圆上一点

圆上三点

**判断平面多边形的凸性**

1.算法：如果一个多边形所有相邻边向量（首尾相接）的叉积都指向平面内侧或者都指向平面外侧，则该多边形是凸多边形

2.实施：需要n次叉积并判断这些结果的z分量是否有相同的符号

**求两个线段的交点**

AB(t)=A+**b**t

CD(u)=C+**d**u

A+**b**t=C+**d**u => **b**t=**c**+**d**u 为二元线性方程组，可解

**测试多边形的简单性**

定义：除了相邻边在顶点相交外没有边相交

算法：穷举法或[Moret91]和[Preparata85]

**求过三点的圆**

算法：求三点构成的三角形的中垂线交点，即为外接圆圆心

圆心表达式：

半径表达式：

**直线和平面求交及裁剪**

1.直线击中（相交）直线或平面

条件：射线R(t)=A+**c**t；直线或平面

方程：

解：

2.用任意凸多边形或凸多面体裁剪直线（Cyrus-Beck裁剪算法）

条件：直线R(t)=A+**c**t

算法：

1）初始化参数t的区间，若是线段则[0,1]，若是直线则选择非常小和非常大的数作为上下界，上下界的值赋给[tin,tout]

2）对每一条多边形的边界直线，计算边界和被裁剪直线的相交时间thit，根据边界法向量和被裁直线方向向量的点积确定是入射还是出射；如果是射入，则令tin=max(tin,thit)，如果是射出则令tout=min(tout,thit)，若入射时间大于出射时间说明不相交则立刻终止检测

3）如果不相交返回0值，相交返回1值

3.用矩形或立方体裁剪直线段（Cohen-Sutherland算法）

135

4.其他算法

1）Sutherland-Hodgman算法：用任意凸多边形裁剪任意多边形

2）Weiler-Atherton算法：用任意多边形裁剪任意多边形

**平面与平面相交**

条件：第一个平面C+**a**s+**b**t；第二个平面

解：s=E+Ft => R(t)=C+**a**(E+Ft)+**b**t