渲染管线学习

光栅化前的准备（MVP变换）

思路：首先通过透视投影，将顶点投影到屏幕上。然后通过循环所有像素测试它们是否位于生产的2D三角形内。

像素点计算优化：计算2D平面中哪些像素需要计算的时候，计算三角形AABB包围盒，判断包围盒中像素是否在三角形内，再进行颜色计算。

解决重叠问题：使用深度缓冲-z缓冲区，z缓冲区记录了点到相机距离

光栅化所需要的数据：图像缓冲区（二维颜色数组），深度缓冲区（二维浮点数组），三角形顶点->投影后的数据。三者一起进行光栅化处理后得到光栅化结果。

相机空间->栅格空间

相机空间->屏幕空间

$$P_{\text{screen}}.x = \frac{\text{near} \times P_{\text{camera}}.x}{-P_{\text{camera}}.z}$$
$$P_{\text{screen}}.y = \frac{\text{near} \times P_{\text{camera}}.y}{-P_{\text{camera}}.z}$$
$$P_{\text{screen}}.z = -P_{\text{camera}}.z$$

屏幕空间->NDC空间（r，l，t，b分别代表屏幕空间的上下左右的坐标）

$$-1 < \frac{2x}{(r-l)} - \frac{r+l}{(r-l)} < 1$$
$$-1 < \frac{2y}{(t-b)} - \frac{t+b}{(t-b)} < 1$$

NDC空间->栅格空间：将 NDC 空间中的 x 和 y 坐标重新映射到 [0,1] 范围，然后将结果数字分别乘以图像宽度和高度（请注意，在栅格空间中，y轴方向与NDC空间相比是反转的）

渲染管线学习

光栅化

确定像素点在三角形内

边缘计算法（需要三次叉乘）：顺时针连接三角形，得到三条边向量，分别与该向量起始点出发到判断点的向量进行二维叉积，得到方向面积，三个方向面积都为正数则该点是在三角形内

重心坐标计算法：设点p为判断点，则从三角形某个顶点出发，到其余两个顶点以及p点构成三个向量，可以使用三角形边向量来表示从出发点到某个判断点的向量，根据平行四边形的比例可得到公式。最后根据该公式得到u，v值，根据值来判断是否在三角形内。

```
P = A +  u * (C - A) + v * (B - A)
```

```
u >= 0
v >= 0
u + v <= 1
```

三角形重心计算

我们了解到：

$$\lambda_0 + \lambda_1 + \lambda_2 = 1.$$

此外，我们知道三角形表面上的任何值都可以使用以下方程计算：

$$Z = \lambda_0 \cdot Z0 + \lambda_1 \cdot Z1 + \lambda_2 \cdot Z2.$$

在本例中，我们插值的值是 Z，它可以代表我们选择的任何值，例如摄像机空间中三角形顶点的 z 坐标。我们可以将第一个方程重写为：

$$\lambda_0 = 1 - \lambda_1 - \lambda_2.$$

将此方程代入计算 Z 的方程并简化产量：

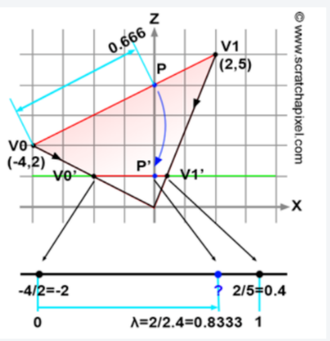
$$Z = Z0 + \lambda_1(Z1 - Z0) + \lambda_2(Z2 - Z0).$$

条款$Z1 - Z0$和$Z2 - Z0$通常可以预先计算，这将 Z 的计算简化为两次加法和两次乘法。这种优化值得一提，因为 GPU 使用了它，并且经常出于这个原因对其进行讨论。

三角形边重叠

三角形边处于顶部水平，或者是左边的边且斜率为正，则认为重叠点为该边上的点

三角形深度插值



$$\lambda = \frac{P'x - V0'.x}{V1'.x - V0'.x} = \frac{0 - (-2)}{0.4 - (-2)} = \frac{2}{2.4} = 0.833.$$

$$\frac{1}{P.z} = \frac{1}{V0.z} \cdot (1 - \lambda) + \frac{1}{V1.z} \cdot \lambda.$$