计算机图形学

叶茂林 2024年3月28日星期四

**渲染管线**

渲染管线就是一堆图形数据经过一系列处理最终渲染在屏幕上。

首先是顶点处理，图形数据通过MVP变换得到投影到二维平面的坐标信息，然后按照几何信息顶点可以变成片元信息，例如三角形，接下来就是三角形的光栅化，即处理片元，哪些点需要被显示，像素点应该如何着色，着色完就可以送给帧缓冲显示了

基本上是顶点着色和片元着色，即几何阶段和光栅化阶段，这些是GPU执行的，在这之前还有一个应用程序阶段，由CPU准备好GPU渲染所需要的光源和模型，然后向GPU下达渲染指令

**MSAA**

将一个像素继续划分为多个小部分，计算这个像素在三角形内部的占比，以这个占比乘以原来的像素值作为新的像素值。

**深度测试**

考虑覆盖，对每个像素记录一个深度值，只留下深度浅的像素值。

**法线贴图、凹凸贴图和位移贴图**

法线贴图是用纹理存储了每个像素的法线，而凹凸贴图存储了每个像素的相对高度信息，通过计算相邻点的相对高度来算出这个点的切线，然后算出法线，而位移贴图是在凹凸贴图计算出来的法线上直接移动点的位置，因为移动的是三角形顶点的位置，所以要求模型精度足够精细，或者进行曲面细分

**应用法线贴图的场景**

模型精度低，不增加几何复杂度，实时渲染，节省资源

**多级渐远纹理Mimap**

解决纹理缩小的问题，就是多个纹理单元映射到同一个像素的处理方法，在原始纹理基础上预先生成多级尺寸逐渐减小的纹理图像，然后根据物体的观察距离的大小选择合适层级的纹理

**阴影贴图shadow mapping**

光栅化实现的阴影，阴影存在是因为我们可以看到但是光源看不到，类似于深度缓存Z-Buffer的算法，先从光源出发记录每个点的深度信息，再从摄像机出发，对于可以看到的点，再次计算这个点到光源的距离，如果这个距离大于第一次记录的深度信息，说明这个点到光源之间有物体挡住了，这里就应该有阴影。

**Alpha测试**

根据物体的透明度来决定是否渲染

**PBR**

基于物理的渲染（Physically Based Rendering，PBR），模拟真实世界中光线的物理行为，包括能量守恒、光衰减、环境光遮蔽，微表面模型（表面粗糙度、法线分布函数）、BRDF（双向反射分布函数）、渲染方程

**判断射线和三角形的交点**

1. 先求平面交点，然后判断点在不在三角形内
2. 解重心坐标，判断重心坐标是否有效

**坐标系**

顶点着色器的输出在MVP变换后的裁剪空间，片元着色器的输入在屏幕空间，裁剪空间经过透视除法到NDC，再经过视口变换到屏幕空间。