计算机图形学

叶茂林 2024年3月28日星期四

**OpenGL渲染管线**

CPU将准备好的**顶点数据**传给GPU，GPU执行着色器程序

**顶点着色器**会对顶点数据进行坐标变换，将模型坐标变换到世界坐标、观察坐标、裁剪坐标

然后进行**图元装配**，对顶点数据进行重新组装成几何图形，同时进行裁剪、面剔除

再经过**几何着色器**重构顶点数据：生成新顶点、删除图元

到光栅化阶段OpenGL**光栅器**将图元转化为像素片段

然后到**片元着色器**进行渲染，加上光照、纹理、阴影，计算出最终的颜色

最后**逐片元**经过alpha测试、模板测试、深度测试处理透明物体和覆盖后写入帧缓冲区

**SSAA**

最早使用超采样抗锯齿，就是使用更高的分辨率，对更多的采样点进行着色

**MSAA**

多重采样是对一个像素点对应多个采样点，计算有多少个采样点在三角形内部，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，包括微表面模型、能量守恒和BRDF

微表面：微观尺度上的平面是由一堆微小镜面组成，它的粗糙度取决于微小镜面的排列整齐度，越粗糙反射光线就会越分散，越光滑反射光线就会越集中

能量守恒：出射光线的能量不能超过入射光线的能量，渲染方程

BRDF双向反射分布函数：描述对于某种材质，对于照射到它的光线它是如何反射的，包括微表面的法线分布函数、几何函数和菲涅尔方程（描述折射率）

**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**

模型坐标系、世界坐标系、摄像机坐标系、裁剪坐标系、标准设备坐标系、屏幕坐标系

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。

**渲染方程的物理意义**

从某个观察视角观察某一点，这点的渲染效果等于它自身的发光加上其他光照射到它的反射光