

## 学习笔记\_环境光遮蔽

笔记本: DirectX 12

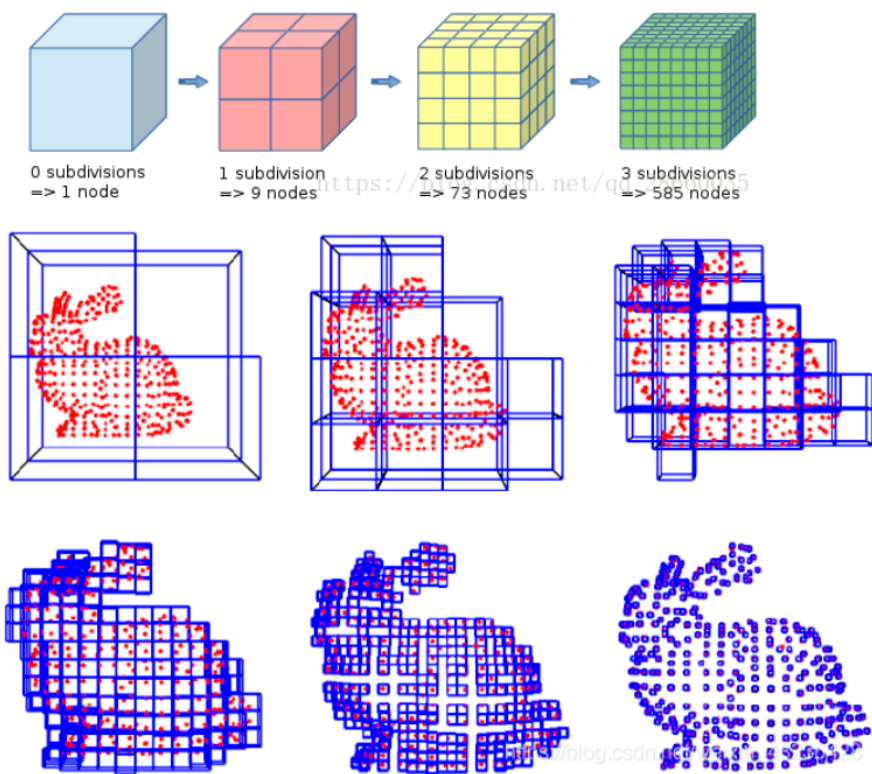
创建时间: 2022/9/27 10:57

更新时间: 2022/10/3 15:14

作者: handsome小赞

- 通过投射光线实现环境光遮蔽

- 具体做法:** 我们随机投射出一些光线, 使它们穿过以点  $p$  为中心的半球, 并检测这些光线与网格(遮挡物)相交的情况。如果投射了  $N$  条光线, 而其中的  $h$  条与网格相交, 则点  $p$  所对应的遮蔽率为  $\text{occlusion} = h/N \in [0,1]$ 。当然, 仅当光线与网格体的交点  $q$  到点  $p$  的距离小于某个阈值  $d$  时才会视作遮挡。
- 注意** 可以使用八叉树对光线检测进行提速。



仅适合预处理

- 屏幕空间环境光遮蔽(SSAO)

SSAO 其所采用的策略:

屏幕空间环境光遮蔽 (Screen Space Ambient Occlusion, SSAO) 技术所采用的策略是, 在渲染每一帧画面的过程中, 将场景观察空间中的法线绘制到一个全屏渲染目标 (full screen render target), 并把场景深度绘制到一个普通的深度/模板缓冲区。接下来, 仅用上述观察空间法线渲染目标和深度/模板缓冲区作为输入, 在每个像素处估算出相应的环境光遮蔽数据。只要得到了存有每个像素处环境光遮蔽数据的纹理, 我们就以这一纹理中的 SSAO 信息来为每个像素调整环境光项, 再像往常那样将处理后的场景绘制到后台缓冲区中。

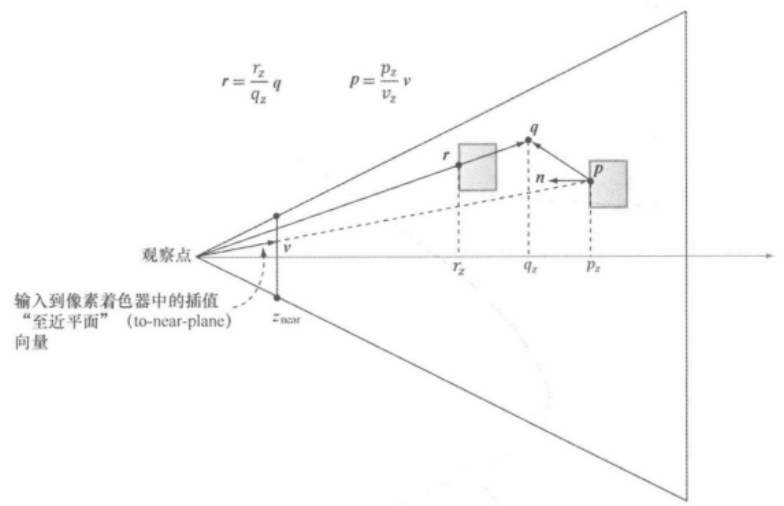
- **法线与深度值的渲染过程**

首先，我们要把场景中各物体的观察空间法线向量渲染到与屏幕发小相同，格式为 DXGI\_FORMAT\_R16G16B16A16\_FLOAT 的纹理图内，同时还要绑定置有场景深度的普通深度/模板缓冲区。

利用像素着色器输出观察空间中的法向量。另外，因为这一次采用了浮点渲染目标，因此向其中写入任何浮点数据都是合理的。

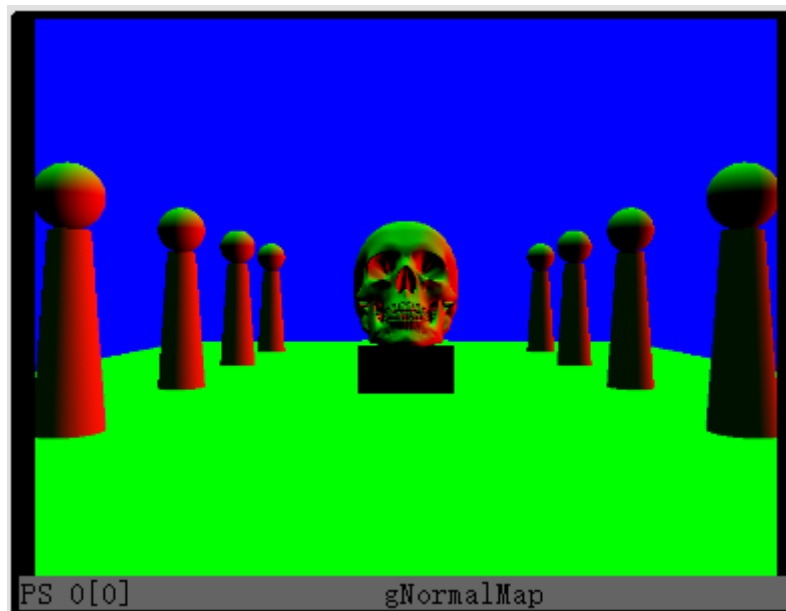
- **环境光遮蔽的渲染过程**

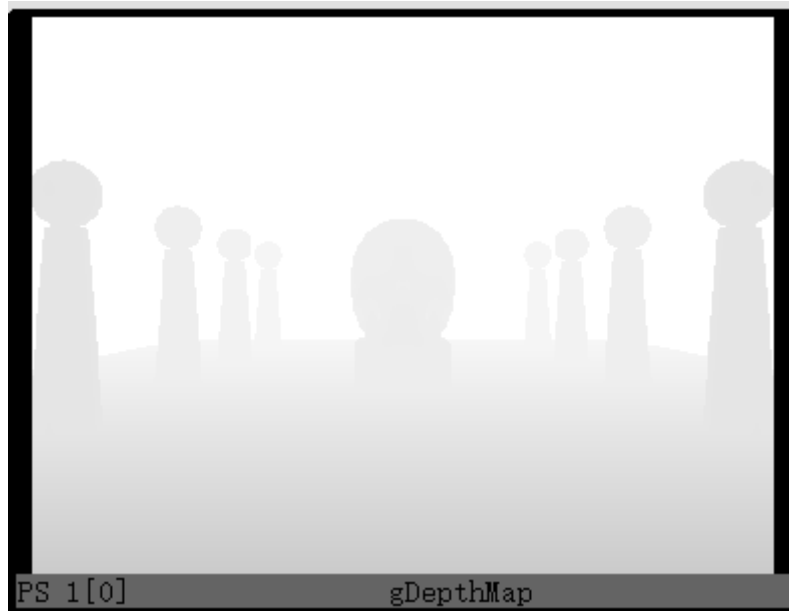
1. 布置好观察空间法线以及场景深度之后，我们就禁用深度缓冲区（期间不在需要对它进行改动）。为每个像素调用 SSAO 像素着色器。
2. 按照 法线图 和 深度图 的宽高的一半来渲染 SSAO 图
3. 重新构建点 p：点 p 为待处理像素，点 q 以点 p 为中心的半球内的随机点（随机点数自定，例程为 14 个，根据平均值法求得遮蔽率），点 r 则是观察点 到点 q 这一路径上最近的可见点。如果  $|z(p) - z(r)|$  足够小，且  $r - p$  与  $n$  之间的夹角小于  $90^\circ$ ，那么 r 将计入点 p 的遮蔽值。



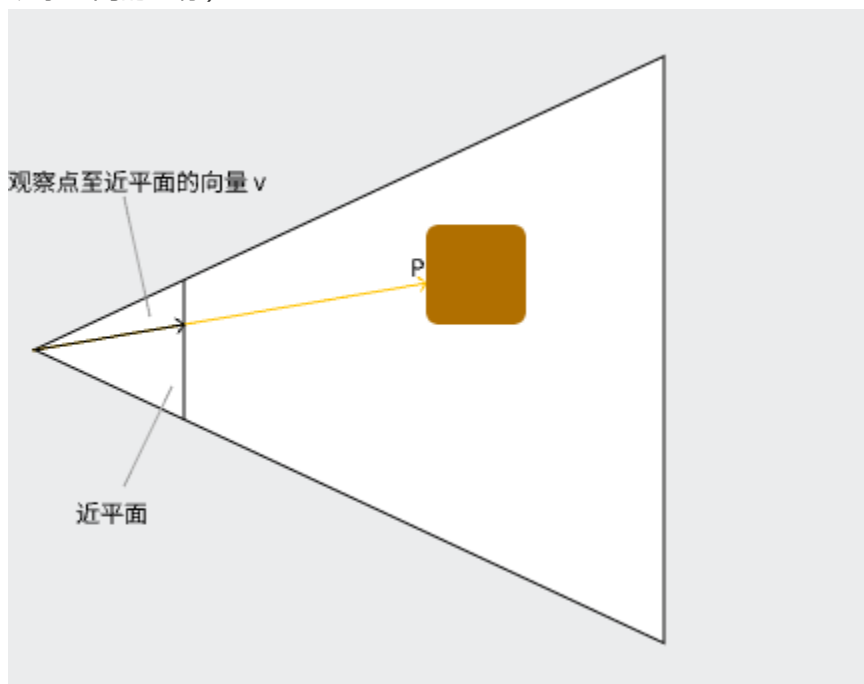
- **重新构建待处理点在观察空间中的位置**

- 此时着色器只拥有场景的法线贴图、NDC坐标的深度贴图





- 因此，此时并没有场景的各个像素坐标信息，但是我们可以根据场景内该像素的深度信息和照射到该像素点的光线照射的向量信息求得其像素点坐标。（因为需要利用该坐标信息来进行测试，所以需要其在观察空间的坐标）



如图，利用该射线的表达式  $P = tV$  即可求得  $P$  的坐标，而  $t = P/V \Rightarrow t = P_z / V_z$ ，所以只需得到  $V$  在观察空间的 depth 即可

- 为了节约成本，我们可以借助硬件的自动插值来求得近平面上的每个像素的观察空间坐标。（因为观察点在观察空间的 原点，所以这里求得的“**近平面上的每个像素的观察空间坐标**”的非齐次的值就是  $V$  向量的值）
  - 那么，需要先在 **VS** 内创建一个全屏的四边形（因为是利用硬件来内插值，所以要遵守**图元的拓扑**，因为事先在 pipelineState 设定了是 Triangle List，所以需要 **6个**顶点数据来构建两个三角

面来组成全屏的矩形)。将其转换到 NDC空间。然后将这个四边形从NDC空间转换到观察空间。

**注意** 因为后面需要对深度贴图进行采样，所以肯定也需要获得一个**内插的纹理坐标**。所以，在此之前才需要先在纹理空间创建那个**6个顶点**的四边形，又因为需要的是一个**全屏**的四边形，所以将它转换到NDC空间并铺满。当然，我们最后需要的是 近平面上的每个像素的观察空间坐标，所以还要从NDC空间转换到观察空间

2. 现在已经获得了“V向量”，那么剩下的都是在 PS 内完成了。  
由前面推导的式子  $P = tV = V \cdot (P_z / V_z)$  可知，要想得到 P 在观察空间中的坐标，还需要 **观察空间的  $P_z$  (NDC空间的 $P_z'$  是 像素的深度信息，在深度贴图内)**，这里就可以利用之前保留的 **纹理坐标** (现在已经经过插值了) 来采样获取。置于从 NDC 空间 到观察空间，因为不是**矩阵**结构了，所以不能用 观察矩阵的**逆矩阵**来求取，但是观察矩阵可以得到转换的方法

$$[x, y, z, 1] \begin{bmatrix} 1 & 0 & 0 & 0 \\ r \tan\left(\frac{\alpha}{2}\right) & 0 & 0 & 0 \\ 0 & \frac{1}{\tan\left(\frac{\alpha}{2}\right)} & 0 & 0 \\ 0 & 0 & A & 1 \\ 0 & 0 & B & 0 \end{bmatrix} \quad (5.2)$$

$$= \left[ \frac{x}{r \tan\left(\frac{\alpha}{2}\right)}, \frac{y}{\tan\left(\frac{\alpha}{2}\right)}, Az + B, z \right]$$

在顶点与投影矩阵相乘之后（即线性部分），我们还要通过将每个坐标分别除以  $w = z$ （即非线性部分）来完成整个变换过程：

$$\left[ \frac{x}{r \tan\left(\frac{\alpha}{2}\right)}, \frac{y}{\tan\left(\frac{\alpha}{2}\right)}, Az + B, z \right] \xrightarrow{\text{除以} w} \left[ \frac{x}{r z \tan\left(\frac{\alpha}{2}\right)}, \frac{y}{z \tan\left(\frac{\alpha}{2}\right)}, A + \frac{B}{z}, 1 \right] \quad (5.3)$$

NDC空间的  $P_z' = A + B/z$ （进行了透视除法才会在NDC空间），那么观察空间的  $z = B / (P_z' - A)$

3. 最终， $P = tV = V \cdot (P_z / V_z)$ ，即可得到 重构的待处理点在观察空间的坐标