

学习笔记_阴影贴图

笔记本: DirectX 12

创建时间: 2022/9/21 8:53

更新时间: 2022/9/23 16:11

作者: handsome小赞

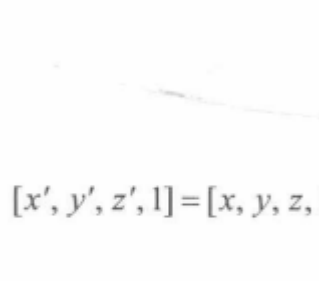
URL: <https://learn.microsoft.com/zh-cn/windows/win32/api/directxmath/nf-directxma...>

- 渲染场景深度

- 以光源的视角渲染场景深度（一种变相的“**渲染到纹理**”技术）
- 以下将考察 ShadowMap 工具类，它存储了以光源为视角而得到的场景深度数据，其简单地封装了渲染阴影会用到的一个深度/模板缓冲区、一个视图、多个视图。
- 用于阴影贴图的那个 深度/模板缓冲区也被称为**阴影图（shadow map）**
- 阴影图的分辨率会直接影响阴影效果，同时影响性能损耗。
- 阴影贴图算法需要执行两个**渲染过程（render pass）**：
 1. 先以光源的视角将场景深度数据渲染至阴影图中（**深度绘制过程 depth pass**）
 2. 以玩家的摄像机视角将场景渲染至后台缓冲区内。为实现阴影算法，将阴影图作为着色器输入

- 正交投影

- 与透视投影一样，我们既希望保留正交投影中的深度信息，又希望采用规格化设备坐标。
- 正交投影矩阵**


$$[x', y', z', l] = [x, y, z, l] \begin{bmatrix} \frac{2}{w} & 0 & 0 & 0 \\ 0 & \frac{2}{h} & 0 & 0 \\ 0 & 0 & \frac{1}{f-n} & 0 \\ 0 & 0 & \frac{n}{n-f} & 1 \end{bmatrix}$$

正交投影矩阵不需要除以z分量的非线性变换，因为整个变换过程都是线性变换。

- 投影纹理坐标

- 投影纹理贴图技术能够将纹理投射到任意形状的几何体上，又因为其原理与投影机的工作方式比较相似，故而得名

- 步骤

1. 将点 p 投影至光源的投影窗口，并将其坐标变换到 NDC 空间
2. 将投影坐标从 NDC 空间变换到纹理空间，以此将它们转换为纹理坐标

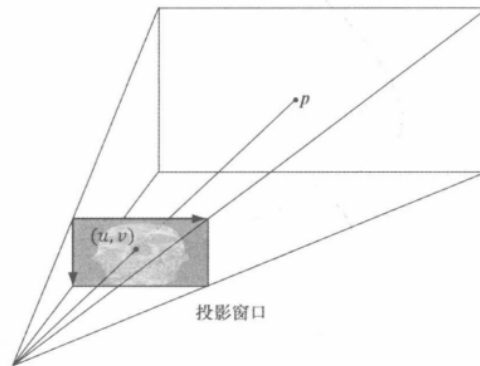


图 20.4 根据光源射向点 p 的投影线，我们便能通过相对于投影窗口中纹理空间的坐标 (u, v) 来确定点 p 在投影窗口中的纹素

- 视锥体之外的点

受寻址模式影响，位于 $[0, 1]$ 坐标之外（视锥体之外的点）的投影坐标将导致奇怪的结果，所以需要进行处理：

1. 使用颜色分量皆为 0 的边框颜色寻址模式
2. 使用聚光灯，聚光灯外的部分不受光照而全黑

- 正交投影

- 使用聚光灯来控制视锥体之外的像素就行不通了
- 因为是正交投影，所以从世界空间到投影空间不需要进行透视除法了

- 阴影贴图

- 算法描述

- 大致思路：从光源的视角将场景深度以“渲染至纹理”的方法绘制到名为阴影图（shadow map）的深度缓冲区中。
为了以光源的视角渲染场景，我们需要定义一个可以将坐标从世界空间变换到光源空间的光源观察矩阵，以及一个用于描述光源在世界空

间中的照射范围的光源投影矩阵。

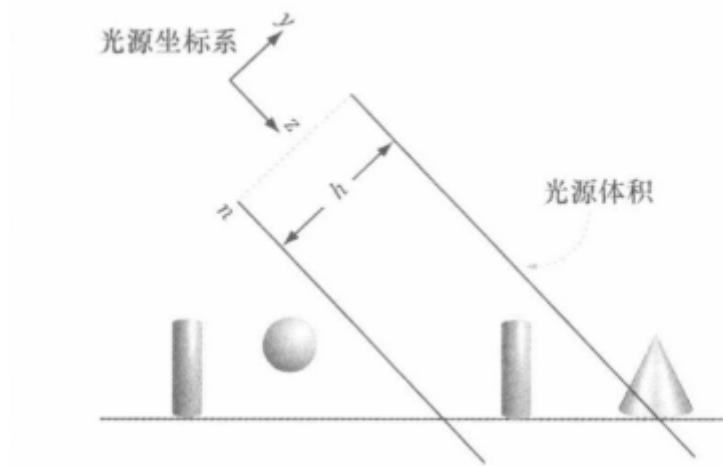


图 20.5 平行光线仅在光源体积中传播，所以只有位于该体积中的场景部分才能被光照射到。

如需令光源照遍整个场景，那么我们就应将光源体积设置为可容纳整个场景的大小

- 完成阴影图的构建后，以“玩家”摄像机视角来渲染场景。针对每一个要被渲染的像素 p ，我们都会以光源的视角来计算其深度，将它记为 $d(p)$ 。同时，在使用投影纹理贴图技术时，我们还要沿着自光源至像素 p 的路径来堆阴影图采样，以获取阴影图所存的深度值 $s(p)$ 。仅当 $d(p) > s(p)$ 像素 p 位于阴影范围之内。

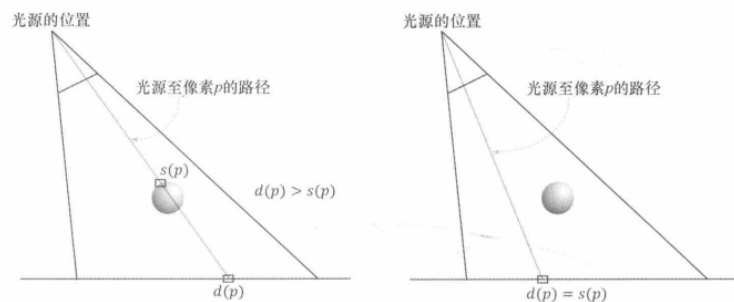


图 20.6 在左图中，从光源的角度上看，像素 p 的深度值为 $d(p)$ 。但是，在同一光线路径上，离光源最近的像素深度值为 $s(p)$ ，且 $d(p) > s(p)$ 。因此我们便能推断出：以光源的视角来看，在像素 p 之前有一物体遮住了它，因此 p 便生活在了阴影之中。右图中，从光源角度来看，像素 p 的深度值为 $d(p)$ ，而且在光源至此点这一条路径上来看，它离光源最近。即 $s(p) = d(p)$ ，由于我们可以推理出 p 会感受到光明的温度

注意 深度值要以 NDC 坐标进行比较，因为阴影图其实是一种深度缓冲区。

• 偏移与走样

- 阴影图存储的是距离光源最近的可视像素深度，但它的分辨率有限，以致每一个阴影图纹素都要表示场景中的一片区域。
- 优化方法：对阴影图存储的深度值进行恒定的偏移量的调整。但是偏移量过大容易造成 peter-panning。且当遇到极大斜率的三角形时，就需要选取更大的偏移量，但同时也更容易导致 peter-panning。

- 当前图像硬件已经支持相关技术，通过名为**斜率缩放偏移**（slope-scaled-bias，也有译作斜率偏移补偿）的光栅化状态属性就可以轻松实现。

```
typedef struct D3D12_RASTERIZER_DESC {
    [...]
    INT         DepthBias;
    FLOAT        DepthBiasClamp;
    FLOAT        SlopeScaledDepthBias;
    [...]
} D3D12_RASTERIZER_DESC;
```

1. DepthBias: 一个固定的应用偏移量。对于格式为 UNORM 的深度缓冲区而言，可参考下列代码注释中该整数值设置方法。
2. DepthBiasClamp: 所允许的最大深度偏移量。以此来设置深度偏移量的上限。不难想象，极其陡峭的倾斜度会导致斜率缩放偏移量过大，继而造成 peter-panning 失真。
3. SlopeScaledDepthBias: 根据多边形的斜率来控制偏移程度的缩放因子。具体配置方法可参见下列代码注释中的公式。

```
// [出自 MSDN]
// 如果当前的深度缓冲区采用 UNORM 格式且与输出合并阶段绑定在一起,或深度缓冲区还没有执行绑定操作,
// 则偏移量的计算过程如下
//
// Bias = (float)DepthBias * r + SlopeScaledDepthBias * MaxDepthSlope;
//
// 这里的 r 是在将深度缓冲区格式转换为 float32 类型后,其深度值可取到的大于 0 的最小可表示的值
// [MSDN 援引部分结束]
//
// 对于一个 24 位的深度缓冲区来说, r = 1 / 2^24
//
// 例如: DepthBias = 100000 ==> 实际的 DepthBias = 100000/2^24 = .006
```

- **注意 在将场景渲染至阴影图时**，便会应用该斜率缩放偏移量。这是由于我们希望以**光源的视角**基于多边形的斜率而进行偏移操作，从而避免阴影失真。

深度偏移发生在光栅化期间（裁剪阶段之后），因此不会对几何体裁剪造成影响。

• 百分比渐近过滤

- 在用投影纹理坐标 (u,v) 对阴影图进行采样时，往往不会命中阴影图中纹素的准确位置，而是通常位于阴影图中的4个纹素之间。此时，应针对采样结果进行双线性插值（采样结果：才采样取得深度值后，与像素深度进行比较取最小值），此方法即 **百分比渐近过滤 (PCF)**

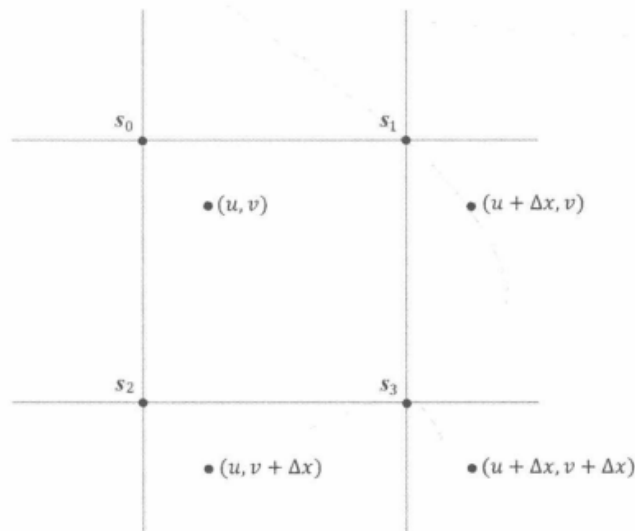


图 20.12 采集 4 个阴影图样本

$$\Delta x = 1 / \text{SHADOW_MAP_SIZE}$$

- 由于现代存储器的带宽与延迟并没有随着 GPU 计算能力的剧增而得到相近程度的巨大改良。所以 PCF 所需的4个纹理样本代价仍较高。但目前，我们已经可以通过调用 `SampleCmpLevelZero` 方法来调取兼容 Direct3D 11+ 版本的图形硬件对 PCF 技术的内部支持。

```
Texture2D gShadowMap : register(t1);
SamplerComparisonState gsamShadow : register(s6);

// 通过除以 w 来完成投影操作
shadowPosH.xyz /= shadowPosH.w;

// 在 NDC 空间中的深度值
float depth = shadowPosH.z;

// 自动执行 4-tap PCF
gShadowMap.SampleCmpLevelZero(gsamShadow,
    shadowPosH.xy, depth).r;
```

- 方法名字中的 `LevelZero` 部分意味着只能在最高的 mipmap 层级中才能执行此函数的相应任务。因为我们仅希望针对阴影贴图进行采样并比较采样结果（而不会为阴影图生成 mipmap 链）。
- 此方法使用的并不是普通的采样器，而是**比较采样器**。这使硬件能够执行阴影图的比较测试，且需要在过滤采样结果前完成。
- 对于 PCF 技术来说，我们需要使用 `D3D12_FILTER_COMPARISON_MIN_MAG_LINEAR_MIP_POINT` 过滤器，并将比较函数设置为 `LESS_EQUAL`（由于对深度进行了偏移，所以也要用到 `LESS` 比较函数）。此方法的前两个参数分别为**比较采样器对象**以及**纹理坐标**，第三个参数是与阴影图样本相比较的数值。在将比较数值设置为 `depth`，并把比较函数设定为 `LESS_EQUAL` 之后，它将执行以下比较操作：

```
float result0 = depth <= s0;
float result1 = depth <= s1;
float result2 = depth <= s2;
float result3 = depth <= s3;
```

接着，在通过硬件线性插值得到最终的计算结果。

比较采样器配置：

```
const CD3DX12_STATIC_SAMPLER_DESC shadow(
    6, // 着色器寄存器(shaderRegister)
    D3D12_FILTER_COMPARISON_MIN_MAG_LINEAR_MIP_POINT, // 过滤器类型(filter)
    D3D12_TEXTURE_ADDRESS_MODE_BORDER, // U 轴所用的寻址模式(addressU)
    D3D12_TEXTURE_ADDRESS_MODE_BORDER, // V 轴所用的寻址模式(addressV)
    D3D12_TEXTURE_ADDRESS_MODE_BORDER, // W 轴所用的寻址模式(addressW)
    0.0f, // mipmap 层级偏移量(mipLODBias)
    16, // 最大各向异性值(maxAnisotropy)
    D3D12_COMPARISON_FUNC_LESS_EQUAL,
    D3D12_STATIC_BORDER_COLOR_OPAQUE_BLACK);
```

- PCF 技术只需在阴影的边缘执行，因为阴影内外两部分并不涉及混合操作，非黑即白。当然，以实际需求为准，在开销和效果之间做出权衡。
- **注意**，实际工程中所用的 PCF 核未必一定是方形的过滤栅格，不少文献指出，随机拾取点也可以作为 PCF 核。

- 构建阴影图

P607 [20.4.4]

- 阴影因子

阴影因子是我们为光照方程新添加的一种范围在 $[0,1]$ 的表量系数。其值为0，表示位于阴影中的点；值为1，表示此点在阴影之外。在进行PCF时，一个点可能会部分处于阴影中，在这种情况下，阴影因子将位于0~1。`CalcShadowFactor`（计算阴影因子）函数的实现在 `Common.hlsl` 文件中。

- 阴影图检测

1. 通过以光源的视角渲染场景来构建阴影图
 2. 在主渲染过程中对阴影图进行采样
 3. 为每个像素 p 计算 $d(p)$ 与 $s(p)$
把 p 变换到光源的 NDC空间便可以求出对应的 $d(p)$ 值，此时像素 p 的 z 坐标即为此点位于光源空间中的归一化深度值（因为阴影图是一种深度缓冲区）
 4. 在光源的视锥体中运用投影纹理贴图技术，投影出场景的阴影图就可以求出值 $s(p)$
 5. 最后利用 `CalcShadowFactor` 函数（位于`Common.hlsl`）便可求得用于光照计算的阴影因子
- 其中，对于阴影图的 $s(p)$ 的采样，利用 `gShadowTransform` 便可以将坐标由世界空间变换到阴影图纹理空间

`gShadowTransform` 即是由 光源的观察矩阵 \times 光源的正交投影矩阵 \times 纹理空间变换矩阵

- 光源的正交投影矩阵 可由

`XMMatrixOrthographicOffCenterLH` 求得

```
XMMATRIX XM_CALLCONV XMMatrixOrthographicOffCenterLH(  
    [in] float ViewLeft,  
    [in] float ViewRight,  
    [in] float ViewBottom,  
    [in] float ViewTop,  
    [in] float NearZ,  
    [in] float FarZ  
) noexcept;
```

- 注意 若不使用正交投影，则 `gShadowTransform`的第二步的矩阵相乘结果还得再进行透视除法，确保坐标位于NDC空间后，再变换到阴影图纹理空间

- 渲染阴影图

若希望将阴影图渲染出来，可以为它创建一个SRV，以便在着色器程序中对它进行采样。

由于阴影图中每个像素存储的都是一个一维数据，所以我们将它渲染

为一幅灰度图像。

注意