Shadow Map

1 基本定义

Shadow Map 是一种用于 3D 渲染中的阴影映射技术。它通过将场景中的光源视角下的深度信息存储在一个纹理中，从而在渲染时判断哪些像素被遮挡，进而生成阴影效果。

其工作原理如下：

1. ** 生成深度图 **：首先从光源的视角渲染场景，生成一个深度图（Shadow Map），该图包含了从光源到每个像素的距离信息。

2. ** 渲染场景 **：然后从摄像机的视角渲染场景时，使用深度图来判断每个像素是否被光源遮挡。

2 优化

2.1 depth bias

为了减少阴影图中的伪影（shadow acne），通常会在深度值上添加一个偏移量（depth bias）。这个偏移量可以是一个常数，也可以是根据表面法线和光源方向计算得出的动态值。

2.2 让阴影贴图和视野范围更匹配

2.2.1 Fitting

(a) 潜在阴影接收者：Potential Shadow Receivers（PSR）是指可能会接收阴影的物体或表面。它们通常是场景中与光源有遮挡关系的物体。

(b) 潜在阴影投射者：Potential Shadow Casters（PSC）是指可能会投射阴影的物体或表面。它们通常是场景中与光源有遮挡关系的物体。

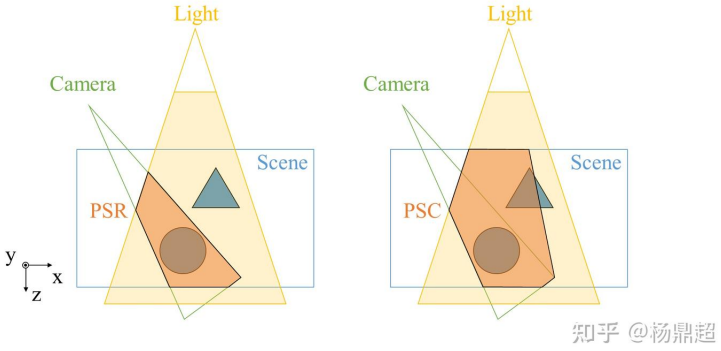


图 1: PSR 和 PSC 的关系示意图

（1）把 Light 视锥体在 xy 方向通过缩放和平移，使得 PSR 和 PSC 的边界与 Light 视锥体的边界对齐，如图2所示。这样可以确保阴影贴图覆盖所有可能的阴影接收者和投射者，从而减少阴影伪影和漏影的情况。

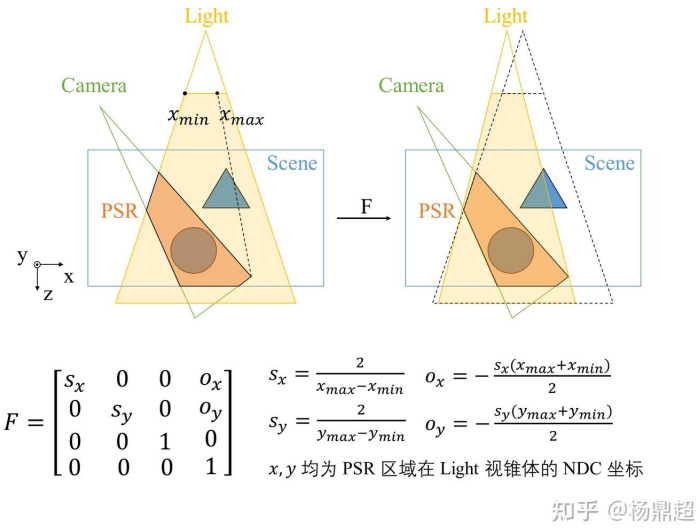


图 2: xy 方向调整

（2）把 Light 视锥体在 z 方向通过压缩，使得 PSR 和 PSC 的边界与 Light 视锥体的边界对齐。

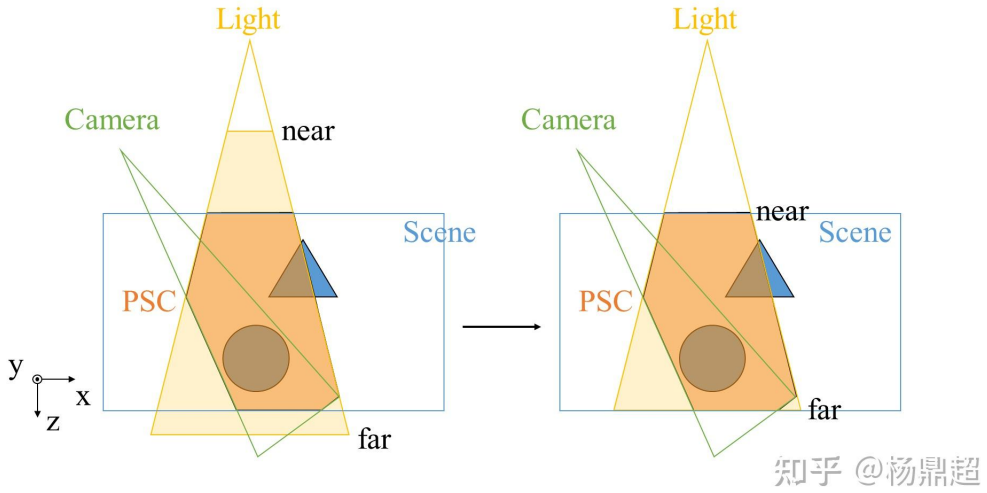


图 3: z 方向调整

2.2.2 Warp

Fitting 方法通过调整视锥体的形状来优化阴影贴图的使用，不能保证阴影贴图像素和实际渲染片元的一一对应。

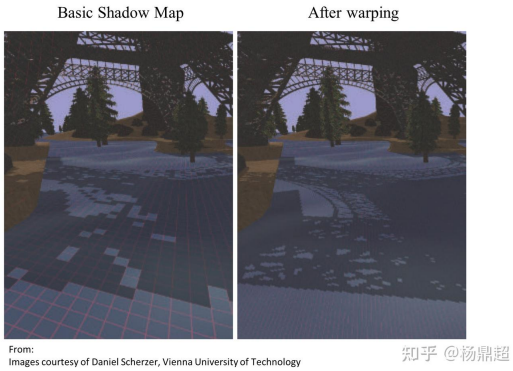


图 4: warp 方法示意图

在图4中，由于近处的阴影贴图的精度需求高于远处，所以左图出现了严重的锯齿现象；右图对变换矩阵做了一定的扭曲（warp），使得近处的精度更高，从而减少了锯齿现象。

2.2.3 Partition

Partition 方法：沿着 z 轴，将阴影贴图分成多个层次（partitions），每个层次使用不同的分辨率和视锥体。这样可以在近处使用高分辨率的阴影贴图，而在远处使用低分辨率的阴影贴图，从而提高渲染效率。

（1）沿 z 轴均匀切分：

$$z_i = z_n + \frac{i}{N}(z_f - z_n), i = 0, 1, ..., N$$

（2）沿 z 轴对数切分：

$$z_i = z_n(\frac{z_f}{z_n})^{\frac{i}{N}}, i = 0, 1, ..., N$$

（3）沿 z 轴混合切分：

$$z_i = \lambda(z_n(\frac{z_f}{z_n})^{\frac{i}{N}}) + (1 - \lambda)(z_n + \frac{i}{N}(z_f - z_n)), i = 0, 1, ..., N$$

2.3 软阴影

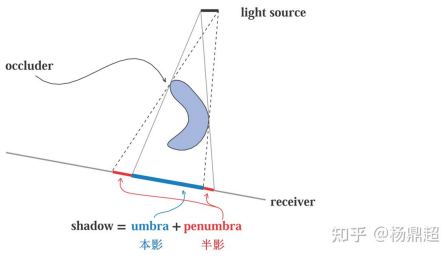


图 5: 软阴影示意图

2.3.1 PCF

PCF（Percentage Closer Filtering）是一种用于生成软阴影的技术。它通过在阴影贴图中采样多个点，并计算这些点的平均值来实现软阴影效果。

（1）Box 滤波

（2）双线性滤波

（3）泊松圆盘滤波

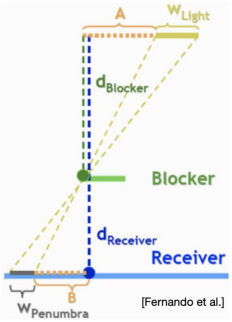
PCF 不能在 shadow map 生成的时候就进行计算，因为：

（a）shadow map 生成的时候只包含深度信息，而 PCF 需要访问多个深度值来计算平均值。

（b）shadow map 是 step 函数，是非线性函数，不能用于 mipmapping。

2.3.2 PCSS

PCSS（Percentage Closer Soft Shadows）是一种改进的 PCF 方法，根据遮挡物与光源和着色点的距离，根据相似三角形，动态计算 PCF 的采样半径，从而实现更自然的软阴影效果。



$$w_{Penumbra} = (d_{Receiver} - d_{Blocker}) \cdot w_{Light} / d_{Blocker}$$

图 6: PCSS 示意图

如图6所示，$\omega_{Penumbra}$ 表示 PCF 采样范围，$d_{Receiver}$ 表示着色点和遮挡物的平均距离，$d_{Blocker}$ 表示光源与遮挡物的平均距离，ω_{Light} 表示面光源的范围。

PCSS 的计算步骤如下：

（1）计算着色点和遮挡物的平均距离 $d_{Receiver}$，以及光源与遮挡物的平均距离 $d_{Blocker}$（多次采样 shadow map 取平均）。

（2）根据 $d_{Receiver}$ 和 $d_{Blocker}$ 计算 PCF 采样范围 $\omega_{Penumbra}$。

（3）根据 PCF 采样范围 $\omega_{Penumbra}$，在阴影贴图中采样多个点，并计算这些点的平均值来生成软阴影。

遮挡物平均距离 $d_{Receiver}$ 计算方法如下：

使用一个从着色点出发的向面光源的视锥，这个视锥会在该光源生成的 shadow map 中圈出一片范围，则这部分范围内的深度值将会用来采样并计算遮挡物平均距离。

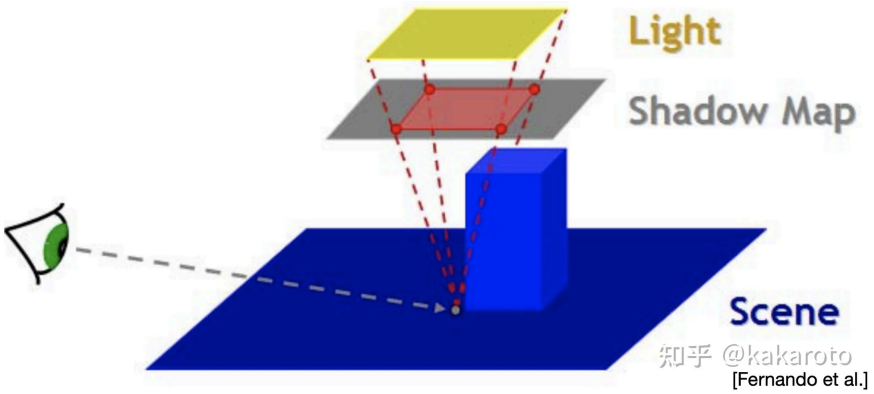


图 7: PCSS2 示意图

3 不同类型的阴影贴图

3.1 Convolution Shadow Map (CSM)

记相机深度为 d，阴影贴图深度为 z，则阴影计算结果可以用傅里叶变换表示：

$$f(d,z) \approx \frac{1}{2} + 2\sum_{k=1}^{M} \frac{1}{c_k} sin[c_k(d-z)]$$

其中 $c_k = \pi(2k-1)$

拆开后：

$$f(d,z) \approx \frac{1}{2} + 2\sum_{k=1}^{M} \frac{1}{c_k} sin(c_k d)cos(c_k z) - 2\sum_{k=1}^{M} \frac{1}{c_k} cos(c_k d)sin(c_k z)$$

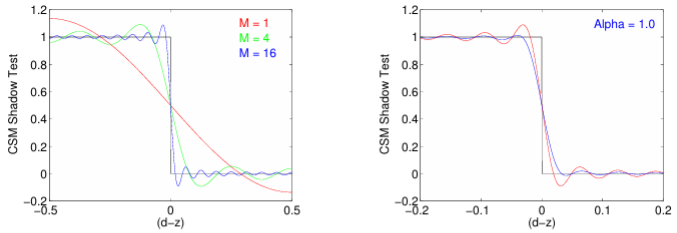
由于 f 是阶跃函数，进行傅里叶展开时，当 M 不够大，会遇到吉布斯现象（振铃现象）。

处理手段有以下三种：

（a）乘以衰减项 $exp(-\alpha(\frac{k}{M})^2)$

（b）平移

（c）缩放



(a) Reconstruction (b) Ringing Suppression

(c) Offset (d) Scaling

图 8: CSM 振铃现象

3.2 Exponential Shadow Map (ESM)

用指数函数 e^{-cx} 来近似 $x = 0$ 处的渐变，实现步骤如下

1. 在生成阴影贴图时，存储 e^{-cz} 而不是 z。
2. 在渲染时，计算 e^{-cd}。
3. $f(d, z) = e^{-c(d-z)} = e^{cz} \times e^{-cd}$
4. $f(d, z) = saturate(f(d, z))$

缺点：

1. c 较小时，会有严重的漏光现象
2. c 较大时，软阴影效果不明显，而且可能会超过浮点数的表示上限

3.3 Variance Shadow Map (VSM)

切比雪夫不等式：设 X 的均值为 μ，方差为 σ^2，则对于任意 $t > \mu$，都有：

$$P(X \geq t) \leq \frac{\sigma^2}{\sigma^2 + (t - \mu)^2}$$

实现步骤如下：

1. shadow map 存储 d 和 d^2

2. 对 shadow map 进行 box 滤波，让深度变化更加平滑，这样纹理中存储的是 $E(d)$ 和 $E(d^2)$

3. 先比较点 p 的深度 t 和阴影贴图的平均深度 $E(d)$，如果 $t < E(d)$，则点 p 在阴影中，返回结果为 1；否则，计算方差 σ^2，将切比雪夫不等式的概率最大值作为结果返回。

$$\sigma^2 = E(d^2) - E(d)^2$$

VSM 也会存在漏光问题（方差较大），可以将阴影值进行重映射，将 [0, 1] 映射到 [min, 1]，缓解漏光问题。

3.4 Cascade Shadow Map (CSM)

CSM 流程如下：

（1）计算各级 SubFrustrum

根据 z 轴的分布，将视锥体分成多个子视锥体（SubFrustrum），每个子视锥体对应一个阴影贴图。

（2）计算各级 SubFrustrum 的 AABB

（a）ndc 空间下视锥体八个顶点已知，利用矩阵逆变换，可以得到视锥体的八个顶点在世界空间中的坐标（可直接计算 AABB）

（b）计算各级 SubFrustrum 的外接球

如图9所示，根据勾股定理有

$$(\frac{a}{2})^2 + x^2 = (\frac{b}{2})^2 + (l - x)^2 = r^2$$

进一步推导得到

$$x = \frac{l}{2} - \frac{a^2 - b^2}{8l}$$

由此可以得到外接圆的圆心和半径，在实际计算中，只需将 a 和 b 看作 SubFrustum 的近平面和远平面的对角线即可。

（3）根据 AABB 计算各级 LightViewProj 矩阵

（a）从外接球出发，沿着平行光的反方向，移动足够长的距离，确定光源方向的观察位置

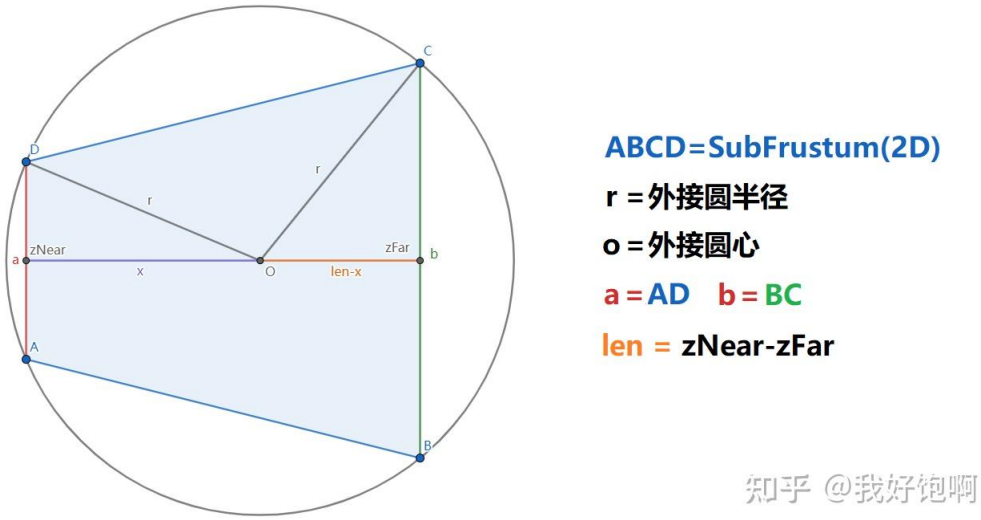


图 9: SubFrustrum 在 2D 视角下的外接圆

（b）根据外接球半径确认正交投影的宽高

4. 渲染各级 ShadowMap

5. 渲染场景时，根据片元深度选择对应级别的 ShadowMap 进行阴影计算

3.5 Screen Space Shadow

屏幕空间阴影，以 Unity 的URP 实现展开。

注：（1）不应用于透明物体

（2）只用于平行光

（3）此处 Unity 的做法是设置单个平行光为主光源

步骤如下：

（1）相机视角生成深度图（Z-Pre），光源视角也生成深度图

（2）重建对应的世界坐标

（3）生成屏幕空间阴影贴图

将对应的世界坐标转换成光源空间坐标，跟光源对应的 shadow map 进行深度比较，生成屏幕空间阴影贴图

注：如果平行光支持 CSM，需根据世界坐标确认处于哪一个 cascade

（4）渲染不透明物体时，对屏幕空间阴影贴图进行采样

优点：

减少因物体重叠导致重复采样 Shadow Map，只对每个像素点计算一次阴影，减少阴影重复计算

缺点：

（1）不支持透明物体

（2）需要额外存储屏幕空间阴影贴图

3.6 Contact Shadow Map

shadow map 为了防止精度问题导致的摩尔纹，通常会加一个 bias，但是会导致物体底部接触面出现漏光的问题，Contact Shadow Map 就是为了缓解这个问题。Contact Shadow Map 也是基于屏幕空间的算法，主体内容是 Ray Marching。

此处以 Unity 的HDRP 实现展开。

3.6.1 算法细节

（1）用 32bit 的 mask 存储最终结果，

（1.1）其中 24bit 表示每个光源是否在阴影中，支持平行光、点光源、聚光灯，最多支持 24 个光源

（1.2）额外的 8bit 用于表示全局的 fade（屏幕边缘处理），对每个光源的 Contact Shaodw 会计算一个局部的 fade，然后和全局的 fade 进行比较，取最大值

（2）控制 Contact Shadow 产生的范围以及衰减效果

（2.1）Min Distance 和 Max Distance，根据像素对应的观察空间的深度，控制 Contact Shadow 的生成范围

（2.1）Fade In Distance 和 Fade Out Distance，控制 Contact Shadow 在 Min Distance 和 Max Distance 区间内的衰减效果，Distance Scale Factor 控制 Contact Shadow 的衰减幅度

3.7 算法步骤

（1）计算 dither：如果开启 TAA，根据帧索引和 InterleavedGradient-Noise 计算 [-0.5, 0.5] 范围内的 dither；如果没有，dither 取-0.5。将 dither 应用于光线起始位置，避免阴影产生 artifacts。

（2）对光线起始位置做一个偏移

```
float3 rayStartWS = positionWS - positionWS * _ContactShadowBias;
```

（3）沿着光线方向进行步进

（3.1）如果是平行光，输入的前进方向为平行光方向的反方向（世界空间）

（3.2）如果是点光源或者聚光灯，根据光源位置和着色点位置计算步进方向

（4）光线步进在 UV 空间 +Z 中进行，需对光线起始位置和光线方向进行变换

举例：如光线起始位置为 float3，其中 x、y 分量为屏幕空间坐标，z 分量为 NDC 空间深度

（5）采样深度并判断遮挡

（5.1）初始化为强制半分辨率采样深度，加速查找到遮挡点，如果采样到遮挡点，不立即退出循环，设置为全分辨率再次采样，判断遮挡

（5.2）计算深度比较阈值，深度差值比较在 (0, 2 * compareThreshold) 内才算有效遮挡，阈值计算公式如下

```
float GetDepthCompareThreshold(float step, float rayStartZ, float rayOrt
{
    return abs(rayOrthoZ - rayStartZ) * _ContactShadowThickness * max(0.0
}

// Here we compute a ray perpendicular to view space.
// This is the ray we use to
//    compute the threshold for rejecting samples.
// This is done this way so that the threshold
//    is less dependent of ray slope.
float4 rayOrthoViewSpace = rayStartCS +
    float4(GetViewToHClipMatrix()[0][2], GetViewToHClipMatrix()[1][2],
        GetViewToHClipMatrix()[2][2], GetViewToHClipMatrix()[3][2])
        * rayLength;
rayOrthoViewSpace = rayOrthoViewSpace / rayOrthoViewSpace.w;
rayStartCS.xyz = rayStartCS.xyz / rayStartCS.w;

```

```
18      float compareThreshold = GetDepthCompareThreshold(step,
19          rayStartCS.z, rayOrthoViewSpace.z);
```

（6）超出屏幕空间处理，如下

```
1       // Off screen masking
2       // We remove the occlusion if the ray is occluded and only if direction
3       float2 vignette
4           = max(6.0f * abs(rayStartCS.xy + rayDirCS.xy * t) - 5.0f, 0.0f);
5       fade = occluded;    // occluded is 0 or 1
6       fade *= saturate(1.0f - dot(vignette, vignette));
```

4 参考

图形学基础 - 阴影 - ShadowMap 及其延伸

Convolution Shadow Map

Exponential Shadow Maps

影子传说——三种 Shadowmap 改进算法的原理与在 Unity 中的实现

Cascade Shadow Map 实现记录

实时渲染丨 Shadow Map：PCF、PCSS、VSM、MSM