游戏中的光照与阴影

MILO YIP

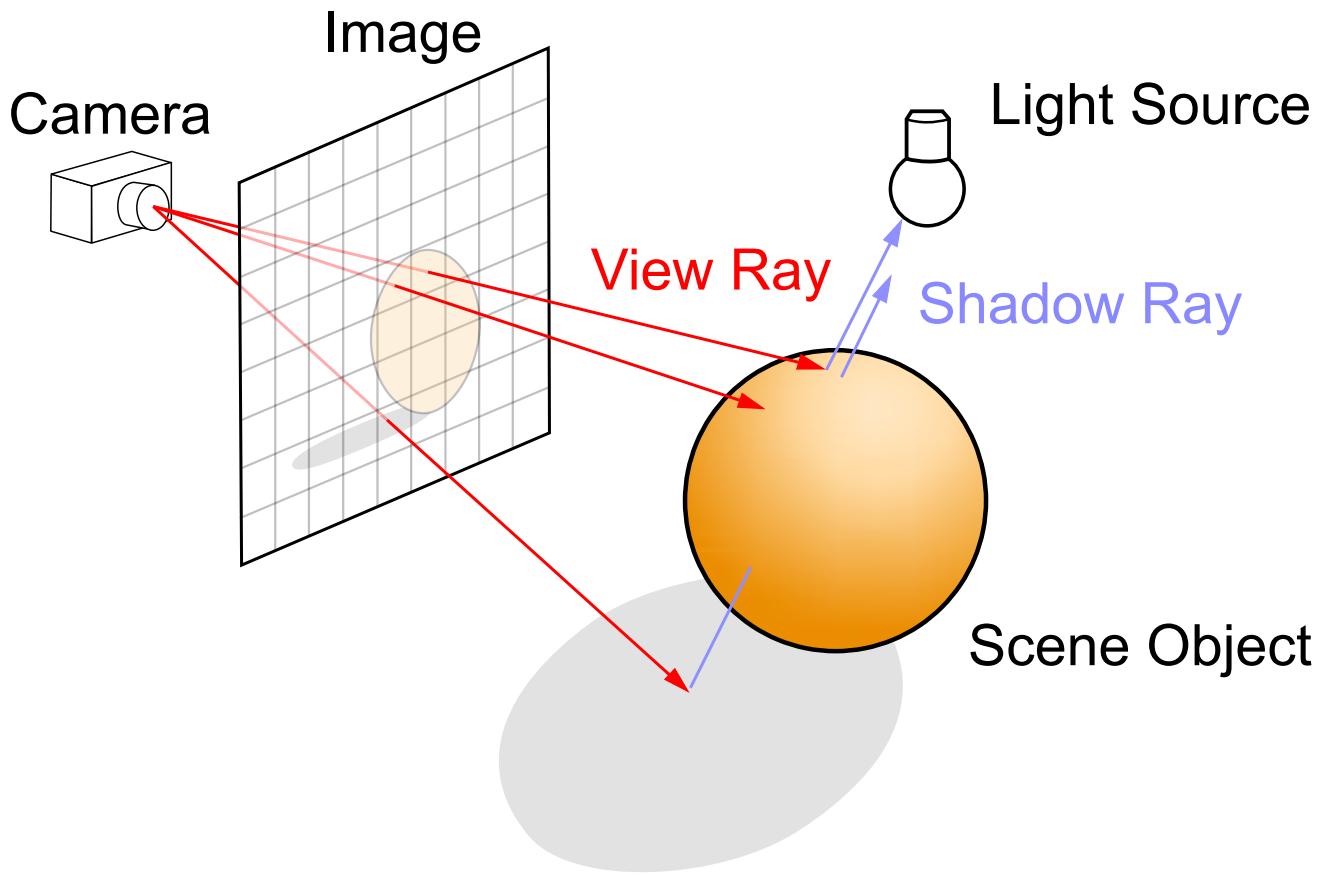
2016/3/17

大纲

1. 简介
2. 光源
3. 阴影

1. 简介

重温渲染概念



渲染方程

$$L_o(\mathbf{x}, \mathbf{v}) = L_e(\mathbf{x}, \mathbf{v}) + \int_{\Omega} f(\mathbf{x}, \mathbf{l}, \mathbf{v}) L_i(\mathbf{x}, \mathbf{l})(\mathbf{n} \cdot \mathbf{l}) \mathrm{dl}$$

- $f(\mathbf{x}, \mathbf{l}, \mathbf{v})$ 为在 $\mathbf{x}$ 处的双向反射分布函数（BRDF），代表材质反射光的特性
- $L_i(\mathbf{x}, \mathbf{l})$ 为入射光的辐射率
- $L_e(\mathbf{x}, \mathbf{v})$ 为自发光的辐射率
- $L_o(\mathbf{x}, \mathbf{v})$ 为出射光的幅射率
- 《材质着色原理与实践》已讲了 f
- 本课程主要讲 L_i

叠加原理

- 光是波，具叠加特性（superposition）：

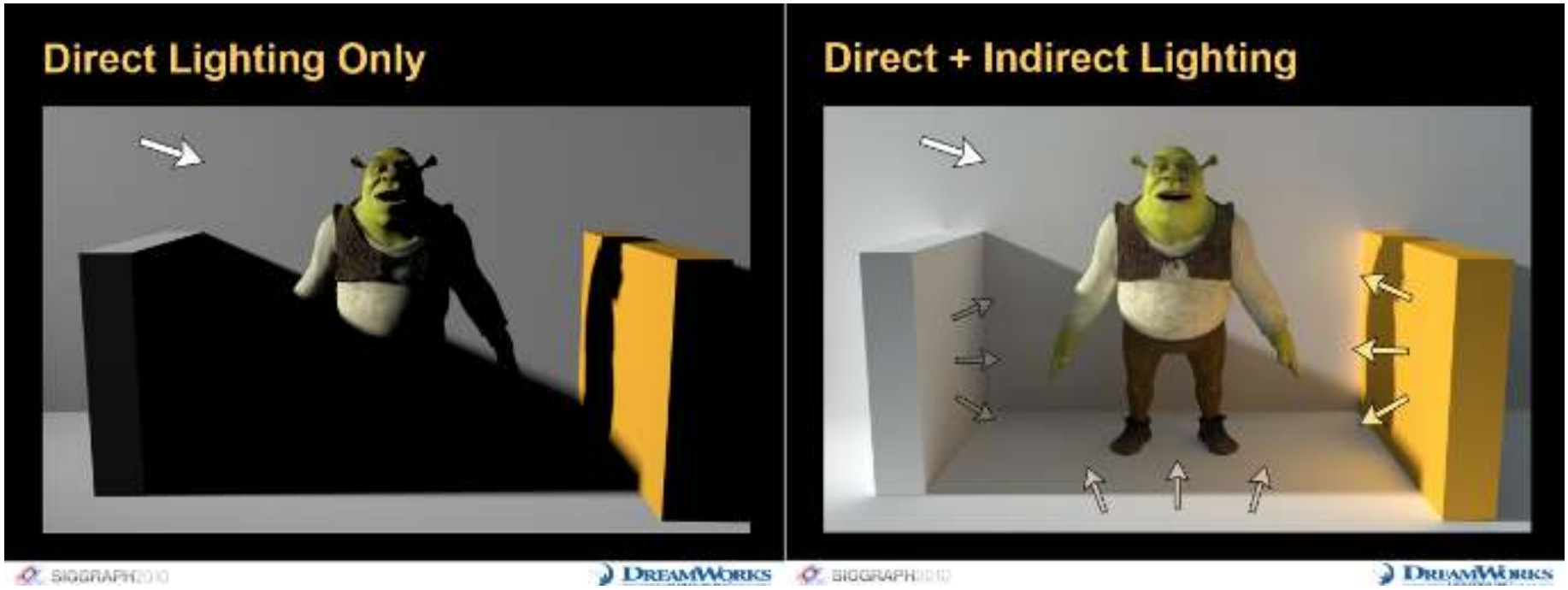
$$F(x_1 + x_2) = F(x_1) + F(x_2)$$

$$F(ax) = aF(x)$$

- 在不考虑光的干涉（interference）现像时，可以独立计算每个光源的影响，把结果相加

「阳光下的泡沫　是彩色的」 G.E.M.《泡沫》

局部光照 vs 全局光照



- 局部光照（local illumination）的入射光 L_i 是直接光源
- 全局光照（global illumination）的入射光 L_i 可来自直接光源或其他间接反射光

局部光照的渲染方程

- 渲染方程原来是对半球方向积分：

$$L_o(\mathbf{x}, \mathbf{v}) = L_e(\mathbf{x}, \mathbf{v}) + \int_{\Omega} f(\mathbf{x}, \mathbf{l}, \mathbf{v}) L_i(\mathbf{x}, \mathbf{l})(\mathbf{n} \cdot \mathbf{l}) \mathrm{dl}$$

- 局部光照只考虑光源作为入射光，渲染方程可简化为：

$$L_o(\mathbf{x}, \mathbf{v}) = L_e(\mathbf{x}, \mathbf{v}) + \sum_{\text{for each light with } \mathbf{l}, L_i} f(\mathbf{x}, \mathbf{l}, \mathbf{v}) L_i(x, \mathbf{l})(\mathbf{n} \cdot$$

实时光照渲染管道

1. 前向着色（forward shading）
2. 延迟渲染（deferred shading）
3. 延迟光照（deferred lighting）
4. 分块前向 / 延迟着色（tiled forward/deferred shading）

前向着色

- 逐个光源和其受影响的几何物体组合渲染：

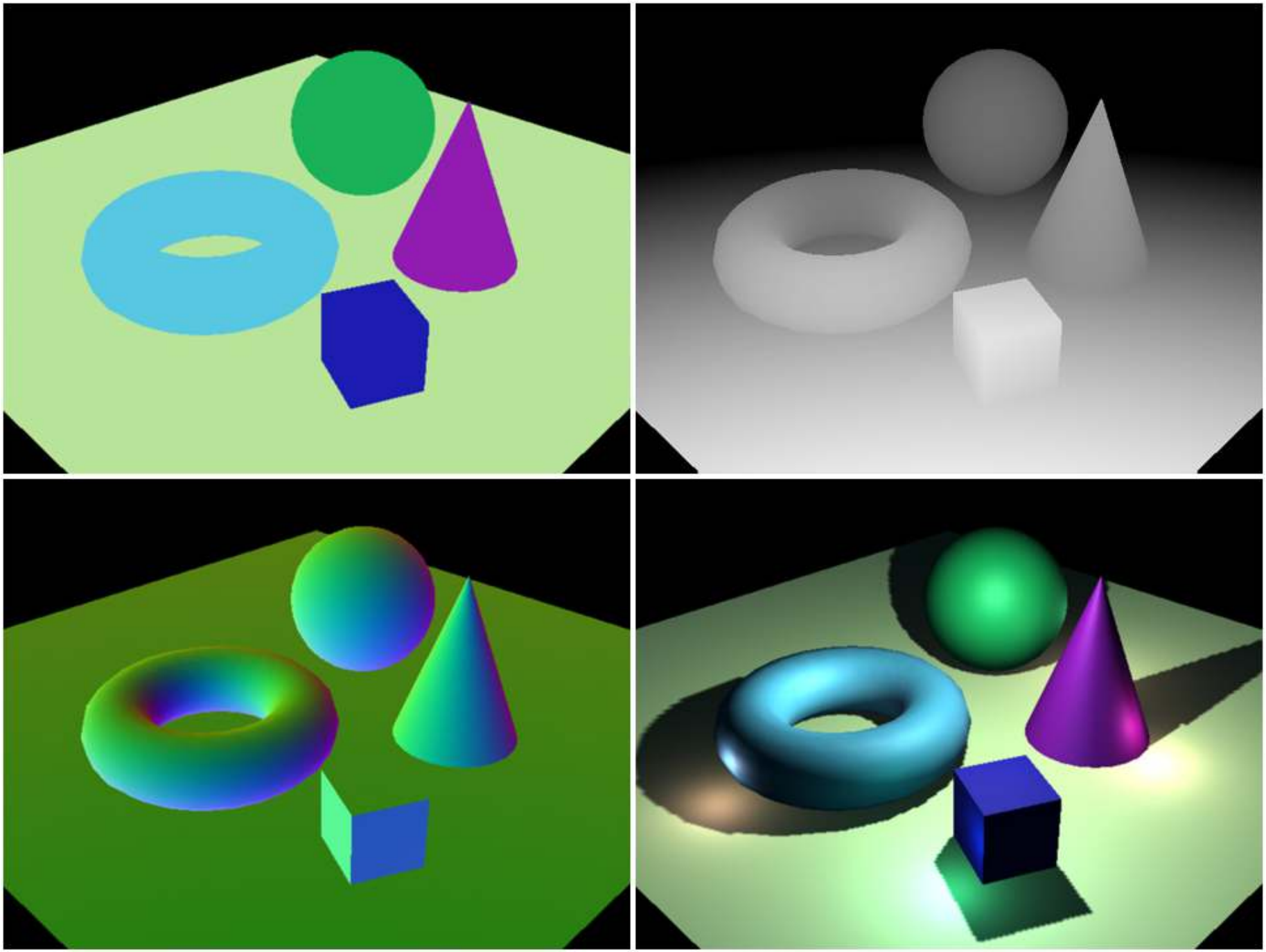
```
For each geometry G
  For each light L
    Rasterize G, shade with L
    Accumulate result to color buffer
```

- 可使用 alpha 混合的叠加模式
- 若有 m 个物体，n 个光源，时间复杂度为$O(mn)$

延迟渲染

```
For each geometry G
  Rasterize depth, normal, diffuse, ... of G to G-buffer

For each light L
  Rasterize L, shade with G-buffer
  Accumulate result to color buffer
```

- 时间复杂度降为$O(m+n)$，可渲染大量光源
- 但是有存取 G-buffer 的开销
- 材质的 BRDF 必须固定，越多参数，G-buffer 越大
- 不能使用 MSAA
- 不能渲染半透明物体（因 G-buffer 只存储一层信息）



延迟光照

- 分解着色方程，例如 Phong 的漫反射项在多光源下：

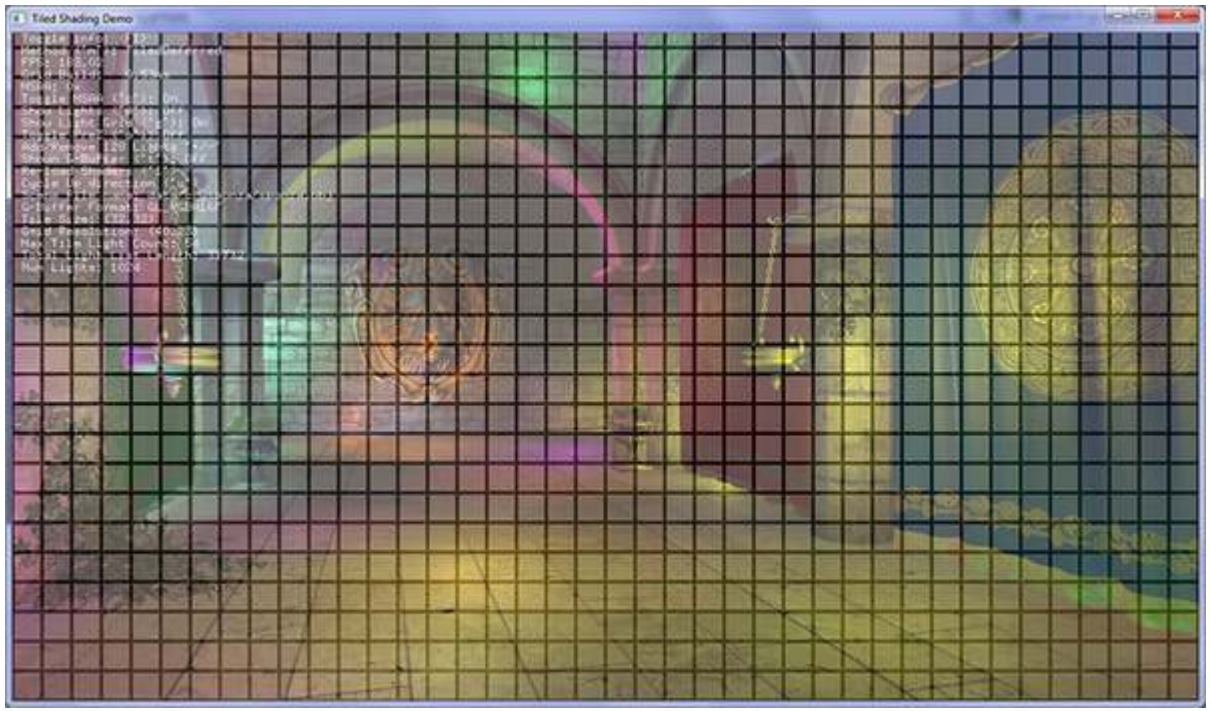
$$\sum_i L_d^i = \sum_i K_D \otimes E_L^i(\mathbf{n} \cdot \mathbf{l}^i) = K_D \otimes \sum_i E_L^i(\mathbf{n} \cdot \mathbf{l}^i)$$

```
For each geometry G
  Rasterize depth, normal of G to G-buffer

For each light L
  Rasterize L, compute light factors with G-buffer
  Accumulate result to light buffer

For each geometry G
  Rasterize G with G-buffer and light buffer
```

- 仍然是 $O(m+n)$，但减少了 G-Buffer 的大小及带频

分块前向 / 延迟着色

- 先收集每个 tile（如 16×16 像素）受哪些光源影响
- 着色时根据像素位置读取光源列表



SIGGRAPH Asia 2014 见闻之
《高效地使用大量光源于实时着色》课程

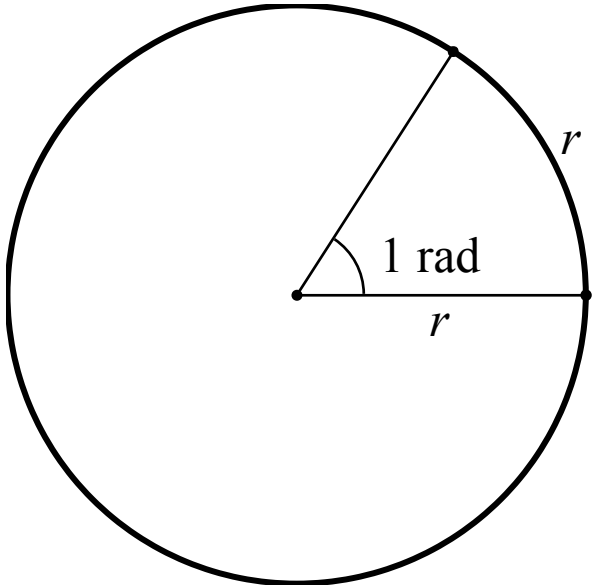
2. 光源

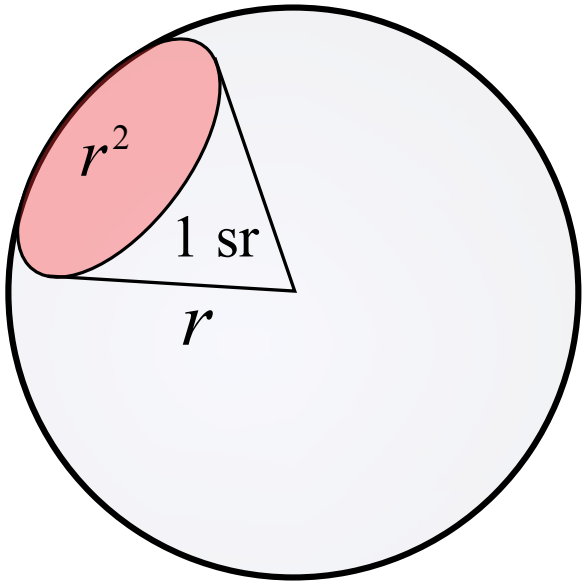
辐射度量学

物理量		意义	国际单位
幅射能 radiant energy	Q	能量	J
幅射通量 radiant flux	Φ	功率	W
輻照度 irradiance	E	每单位面积的幅射通量	$\mathrm{W} \cdot \mathrm{m}^{-2}$
輻射率 radiance	L	每单位立体角、每单位面积的幅射通量	$\mathrm{W} \cdot \mathrm{sr}^{-1} \cdot \mathrm{m}^{-2}$

立体角

- 立体角（solid angle）是平面角的三维版本
- 平面角的单位是弧度（radian），单位 rad
- 立体角的单位是球面度（steradian），单位 sr

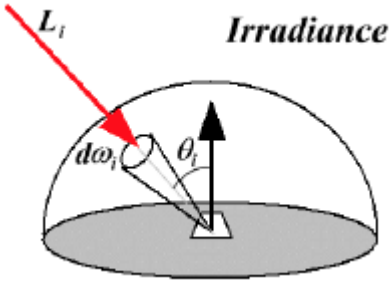




輻照度与輻射率

- 辐照度 E 是单位平面上所有方向的幅射通量 $E = \dfrac{\partial \Phi}{\partial A}$
- 輻射率 L 是单位平面上某方向、每单球面度的幅射通量

$$L = \frac{\partial^2 \Phi}{\cos\theta_i \partial A \partial \omega}$$



$$E = \int_{\Omega} L_i \cos\theta_i d\omega_i$$

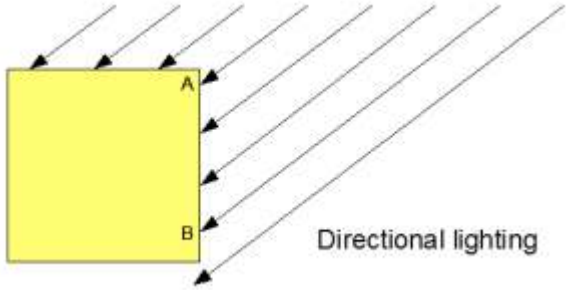
环境光

- 环境光（ambient light）是实时光照的 hack
- 问题：在局部光照中，没被直接光照的地方会完全黑色
- 解决方法：任何位置也给与一个环境光源
- 传统的 Phong 着色中，环境光项：

$$I = K_a I_a$$

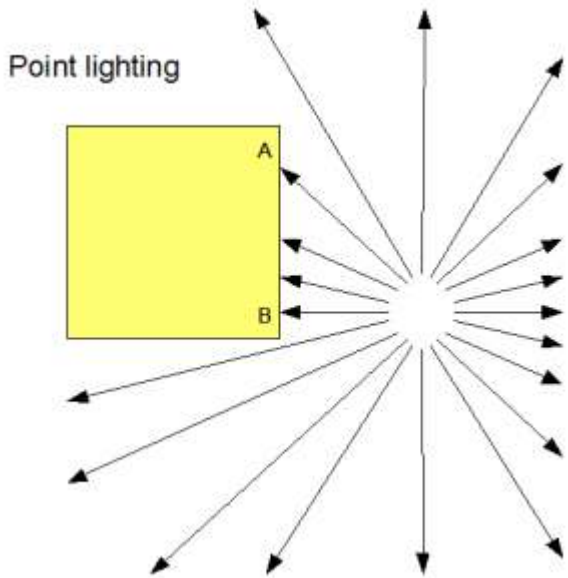
  - K_a 是材质的环境光反射量
  - I_a 是全局的环境光强度
  - 注意：基于物理着色时并不使用这种方式

方向光



- 方向光（directional light）在任何位置的方向都相同
- 可模拟遥远的光源，如太阳光
- 方向光的固定方向为 $\mathbf{l}$，固定幅照度为 E_L

点光



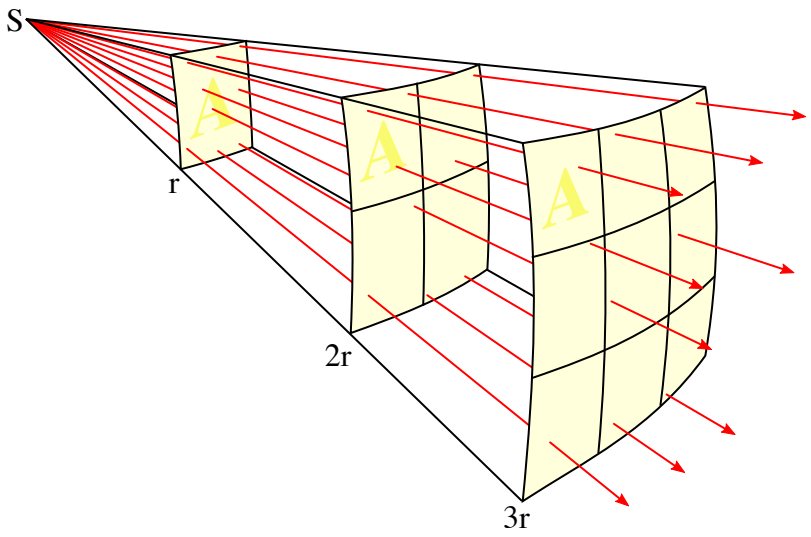
- 点光（point/omni light）是从一点 $\mathbf{x}_p$ 发散的光源
- 对于位置 $\mathbf{x}$，点光的方向 $\mathbf{l}$ 为

$$\mathbf{l} = \frac{\mathbf{x}_p - \mathbf{x}}{\|\mathbf{x}_p - \mathbf{x}\|}$$

距离衰减

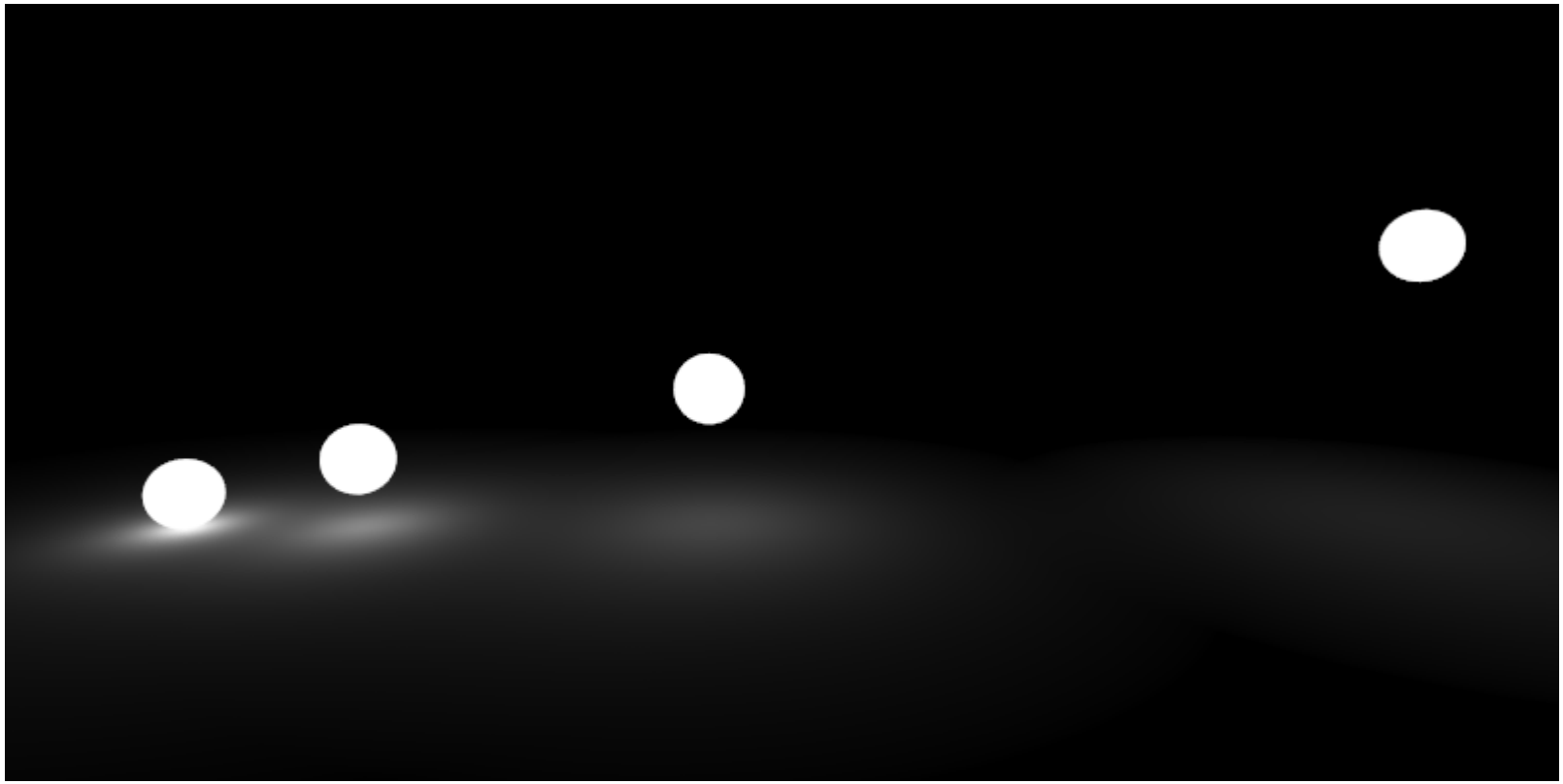
- 设点光源的距离为 $r = \|\mathbf{x}_p - \mathbf{x}\|$
- 基于平方反比定律，点光的幅照度按距离衰减（attenuation）

$$E_L(\mathbf{x}) = \frac{I_L}{r^2}$$

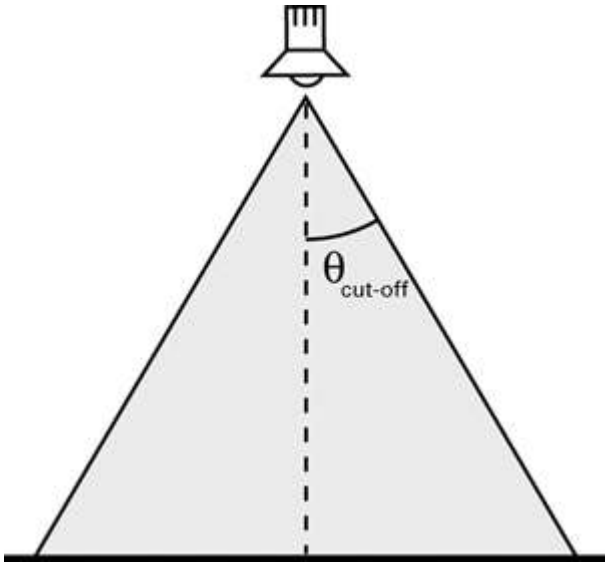


距离范围

- 平方反比定律有个问题：光源影响范围是无穷大的
- 这在物理上是正确的（能看到几十亿光年的天体）
- 但不能剔除光源
- 可用其他衰减函数，使 $r_{\max}$ 以外的幅照度为零，例



聚光



- 聚光（spot light）是点光再加上角度限制

简单聚光

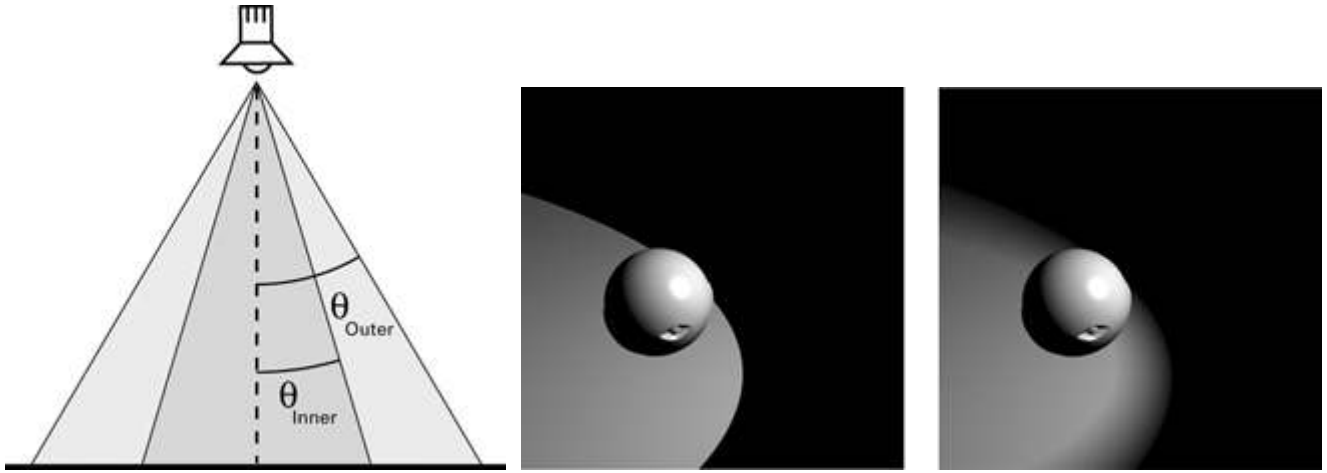
- 和点光相同地计算 $\mathbf{l}$ 和 r
- 给定聚光的方向 $\mathbf{d}$，幅射半角为 $\theta_{\text{cut-off}}$

$$s(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{l} \cdot \mathbf{d} < \cos\theta_{\text{cut-off}} \\ 0 & \text{otherwise} \end{cases}$$

- 那么，其幅照度为

$$E_L(\mathbf{x}) = \frac{I_L}{r^2} \cdot s(\mathbf{x})$$

柔和聚光



- 可使用两个角度去计算$s(\mathbf{x})$，产生边缘的过渡
- 但可以更有弹性……

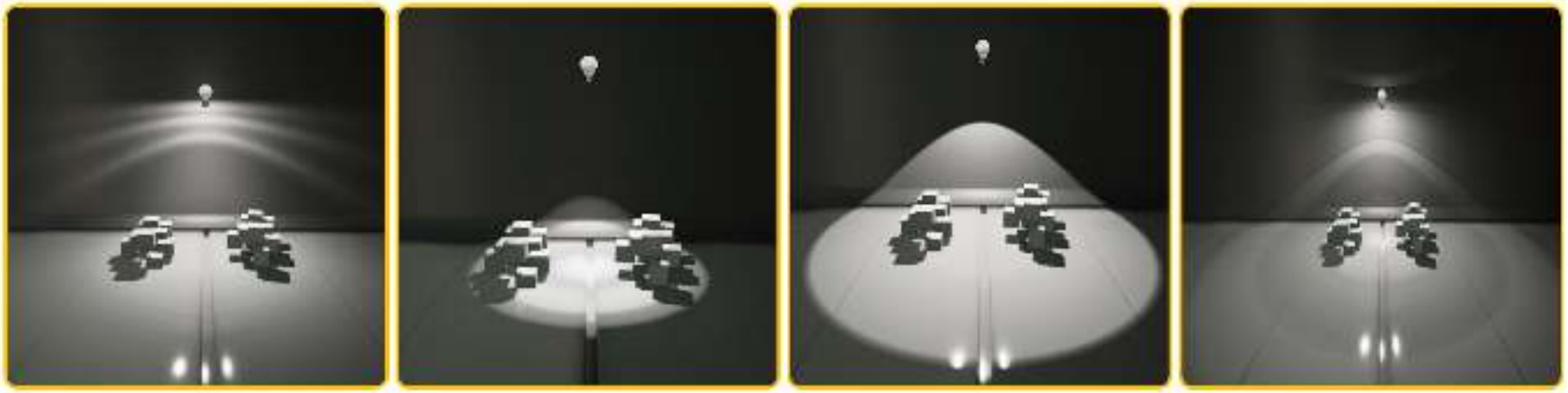
投影贴图

投影贴图实现

- 投影贴图（projective texture, cookie/gobo texture）通过纹理投影的光源
- 需设置一个类似摄像头（camera）的投影器（projector）
- 把世界坐标 $\mathbf{x}$ 转换至投影器的齐次裁剪空间 $[-1, 1]^3$
- 对纹理采样，获得 E_L 或 I_L
- 点光源可使用 cube 纹理

IES PROFILE

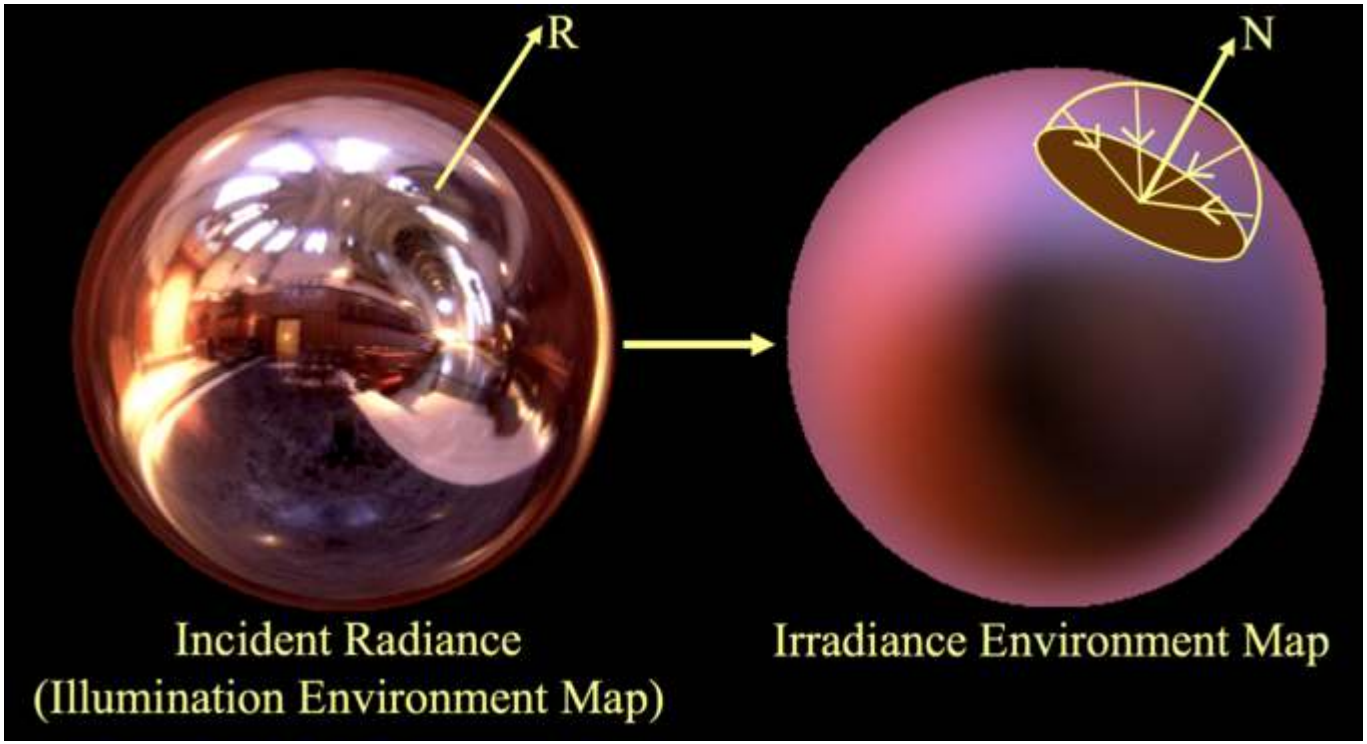
- UE4 支持美国照明工程学位（IES）的灯具数据格式
- 类似于投影贴图，但只需要一维纹理



再探环境光

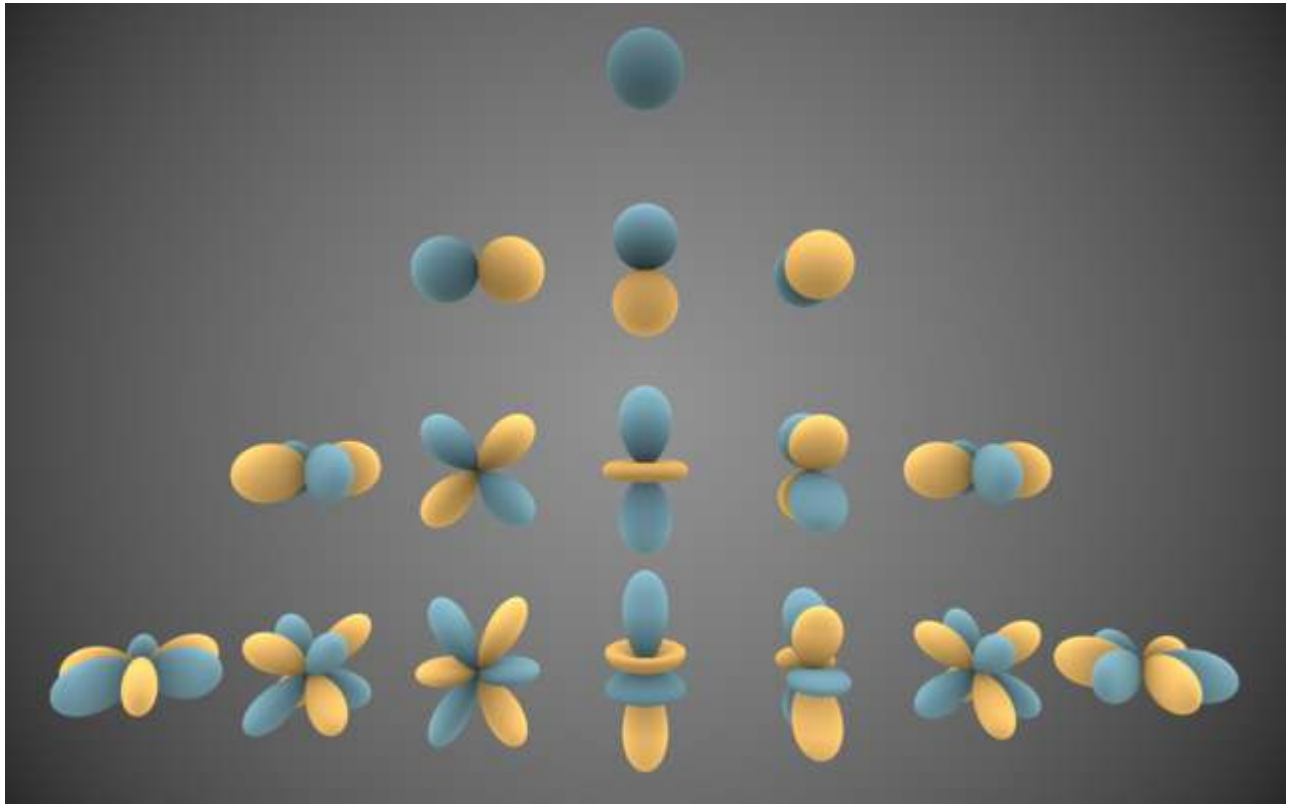
- 真实的环境有许多光源 / 反光体（如天空、地面）
- 局部光源不能满足

幅照度环境贴图



$$E = \int_{\Omega} L_i \cos\theta_i d\omega_i$$

球谐函数



- 球谐函数（spherical harmonics, SH）类似 FFT，把空间域转换成频域，但使用球坐标

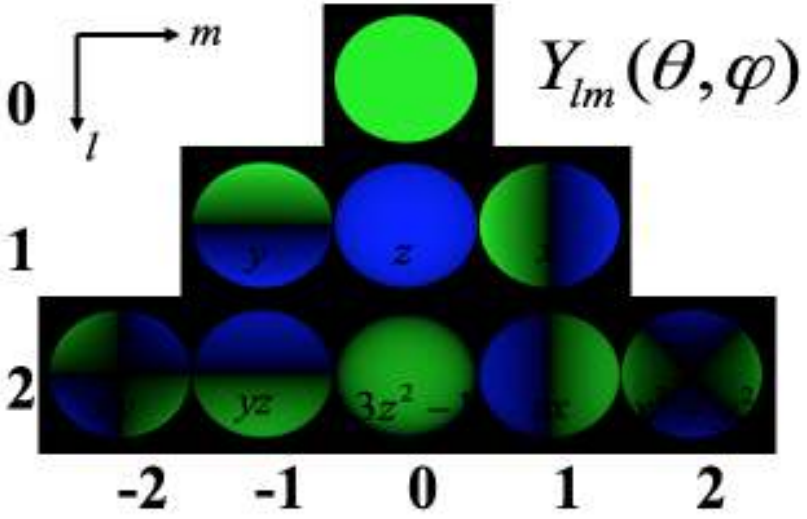
SH 幅照度环境贴图

Exact image

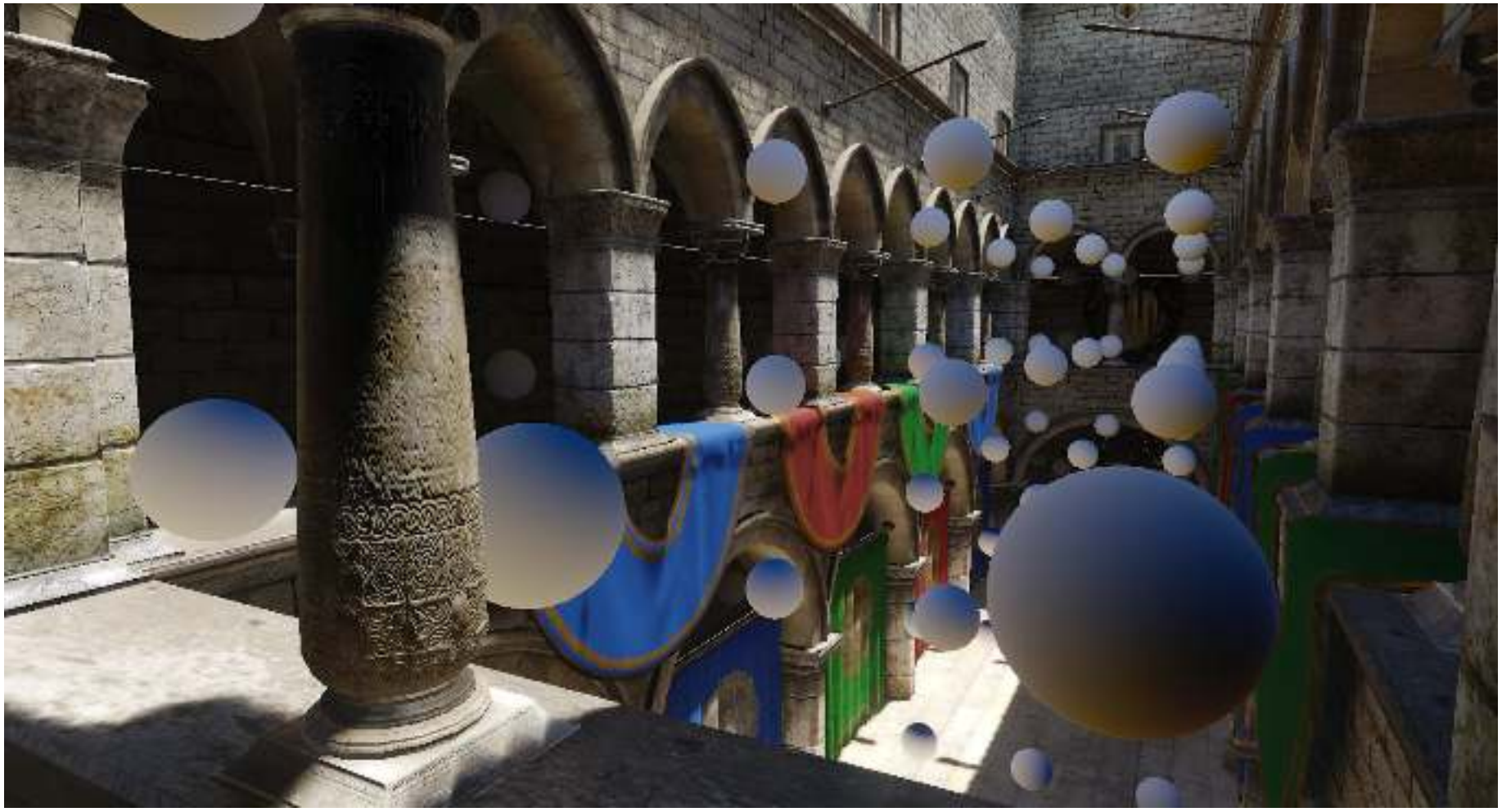
Order 2
9 terms

RMS Error = 1%

For any illumination, average error < 3% [Basri Jacobs 01]



幅照度体积



3. 阴影

什么是阴影？

"0 to 9" by Kumi Yamashita

阴影的相关定义

- 若点 $\mathbf{x}$ 不能看见光源，则它位于阴影之中
- 换句话说，阴影是光源的可见性函数

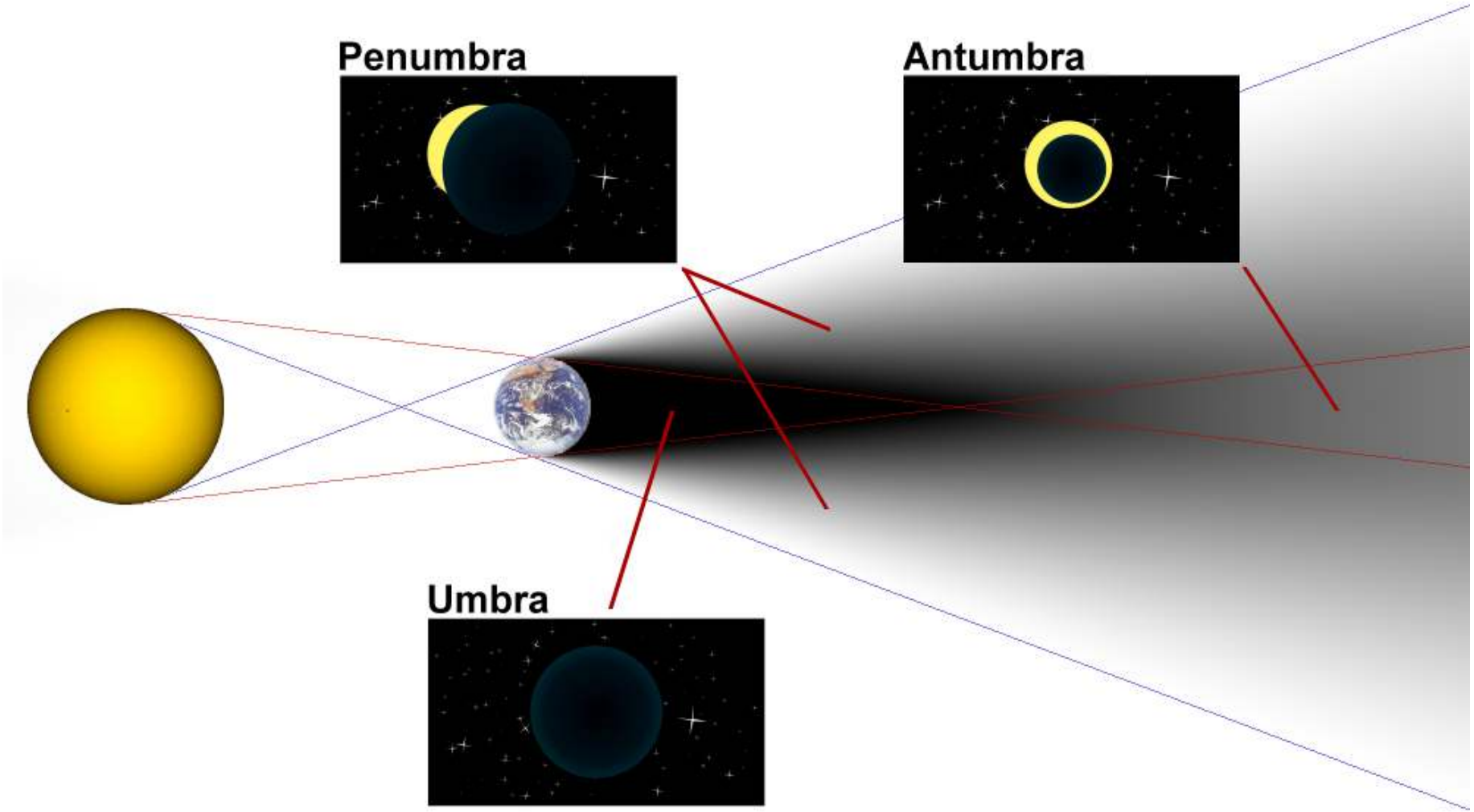
$$V_L : \mathbf{x} \rightarrow [0, 1]$$

- 为了减少运算，通常只会设置部分物件具阴影功能：
  - 在光源上设置是否使用阴影功能
  - 能遮挡光源的物体为阴影投射体（shadow caster）
  - 能接受阴影的物体为阴影接收体（shadow receiver）

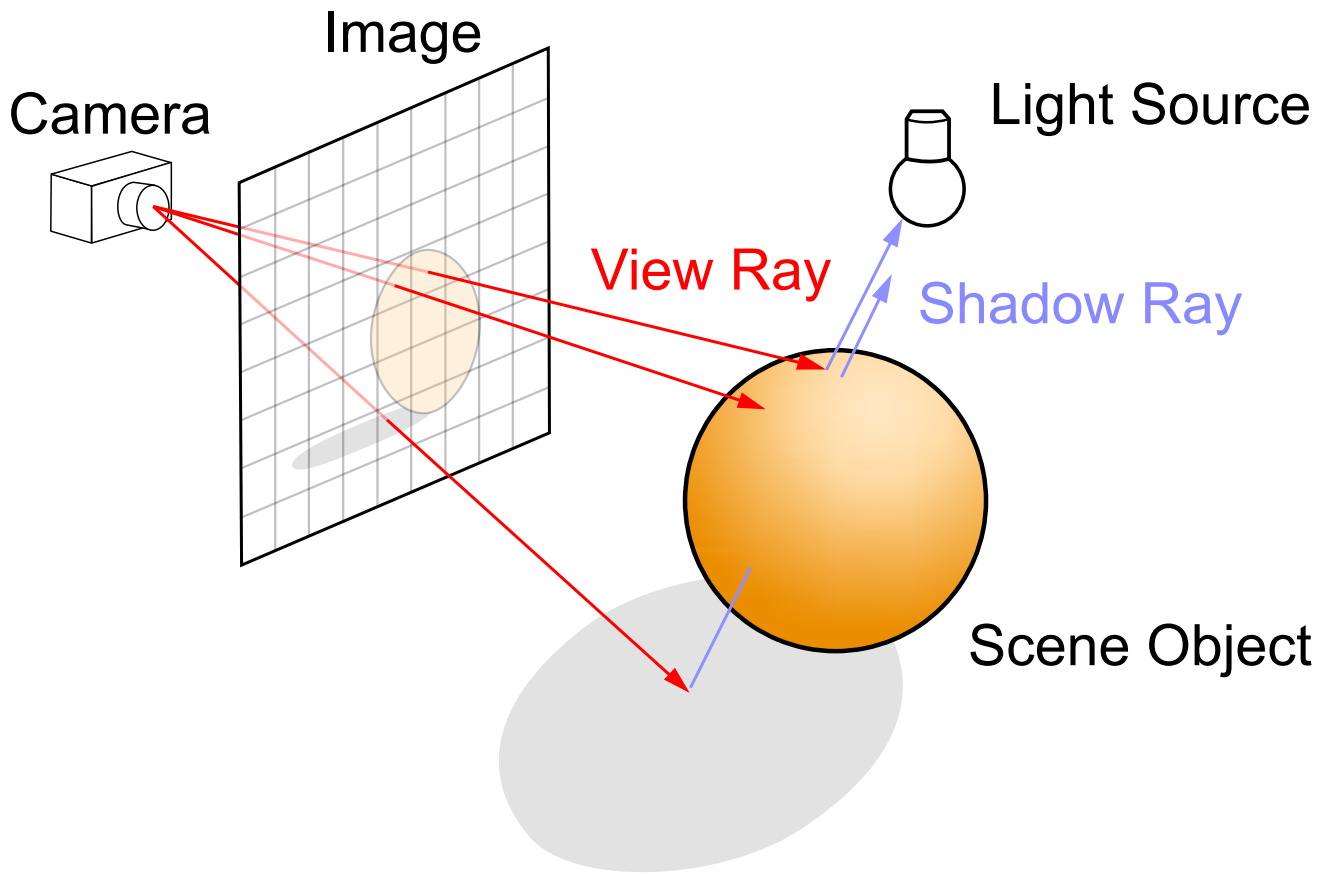
硬阴影与软阴影

- 完全平行的方向光或无穷小的点光会产生硬阴影
- 具面积的光源产生软阴影（soft shadow）

本影 / 半影 / 偽本影



光线追踪阴影



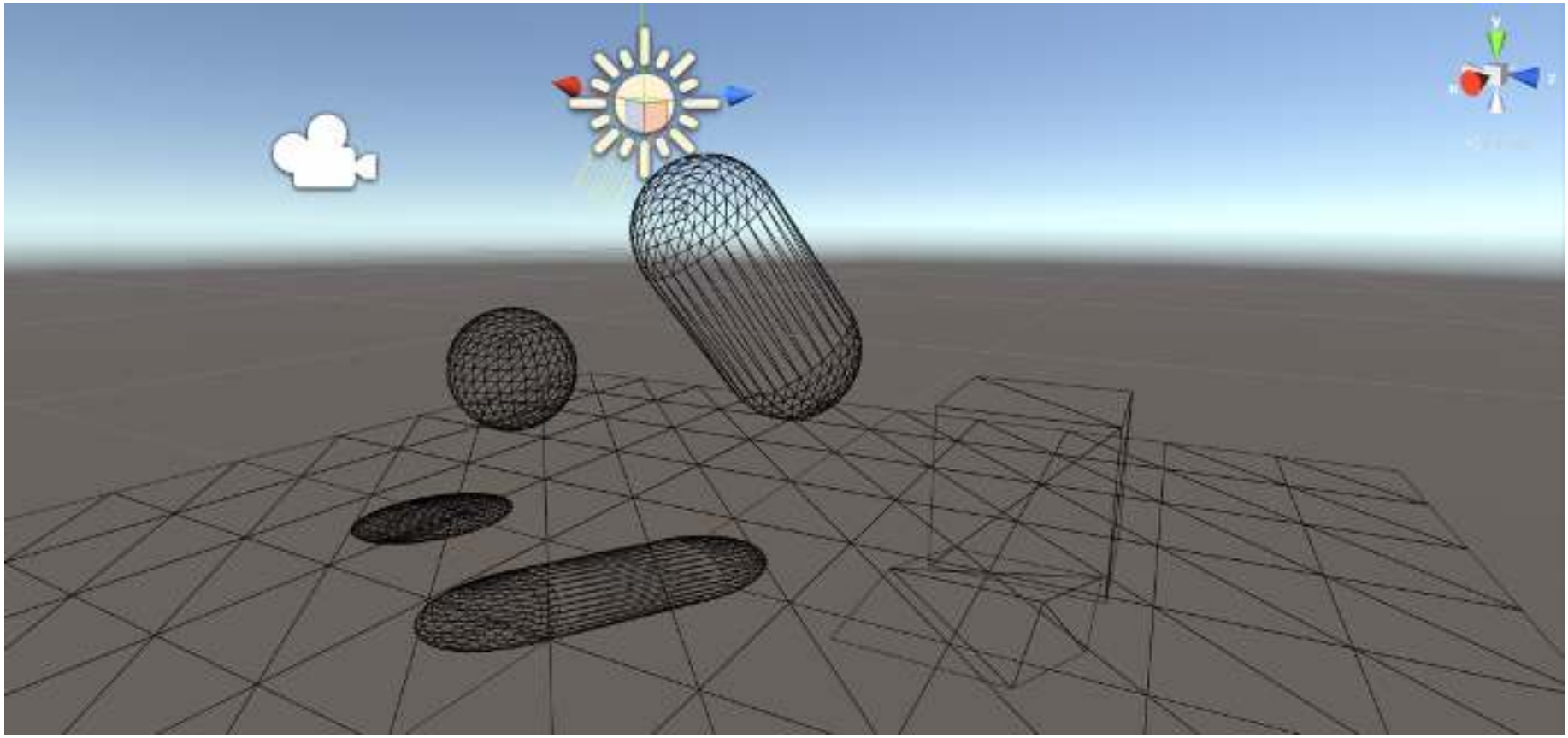
- 使用阴影射线来评估 $V_L(\mathbf{x})$

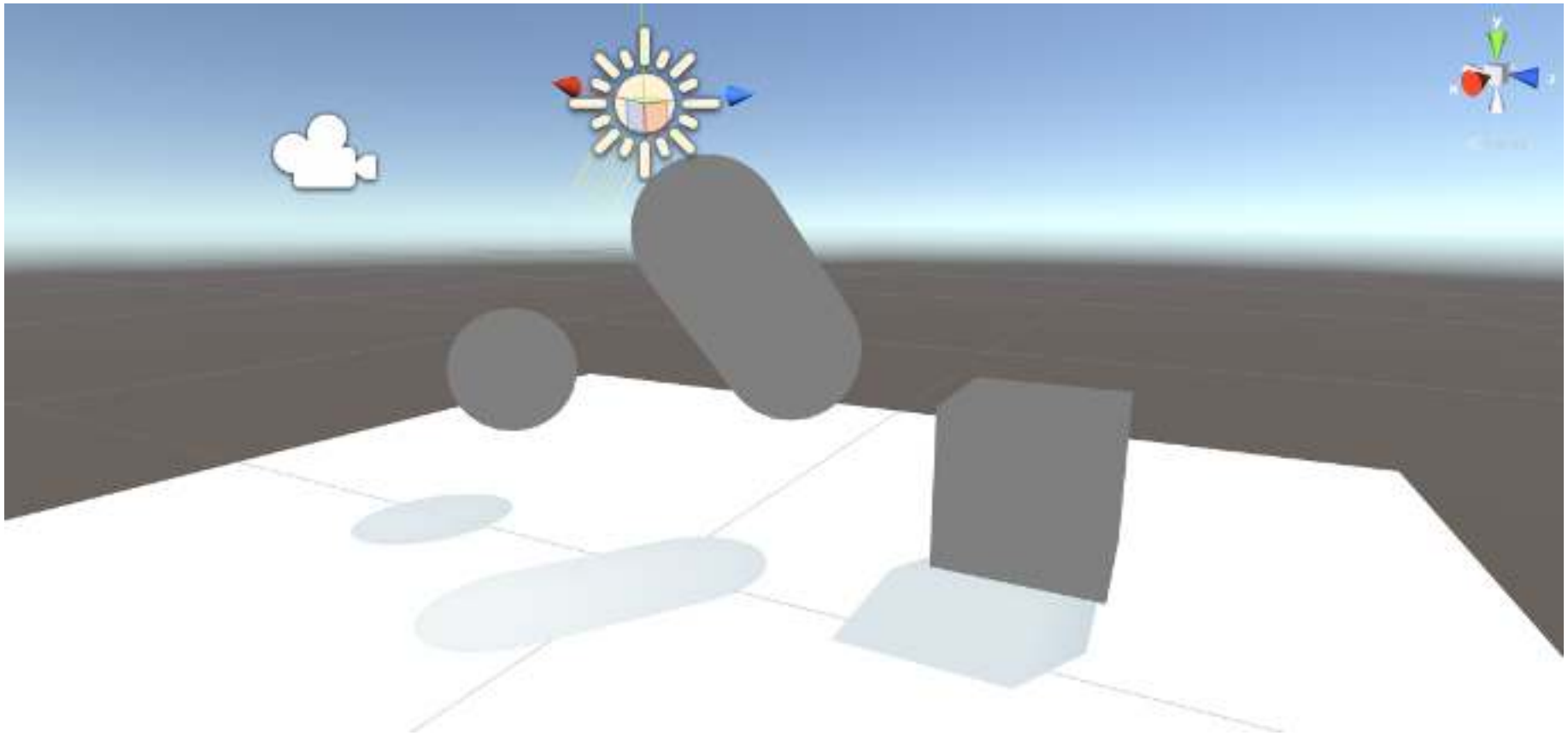
实时阴影的基本方法

- 对于每个支持阴影的光源：
  1. 从光源和阴影投射体建立可见性函数 $V_L(\mathbf{x})$
  2. 渲染阴影接收体时，以 $V_L(\mathbf{x})$ 判断是否受光源影响
- 不同的实时阴影技术差异在于怎么计算及存储 $V_L(\mathbf{x})$

平面阴影

- 若阴影接收体是平面，可把阴影投射体投射至该平面
- 平面阴影（planar shadow）一般是硬阴影





Unity Planar Shadow 例子

平面阴影的渲染

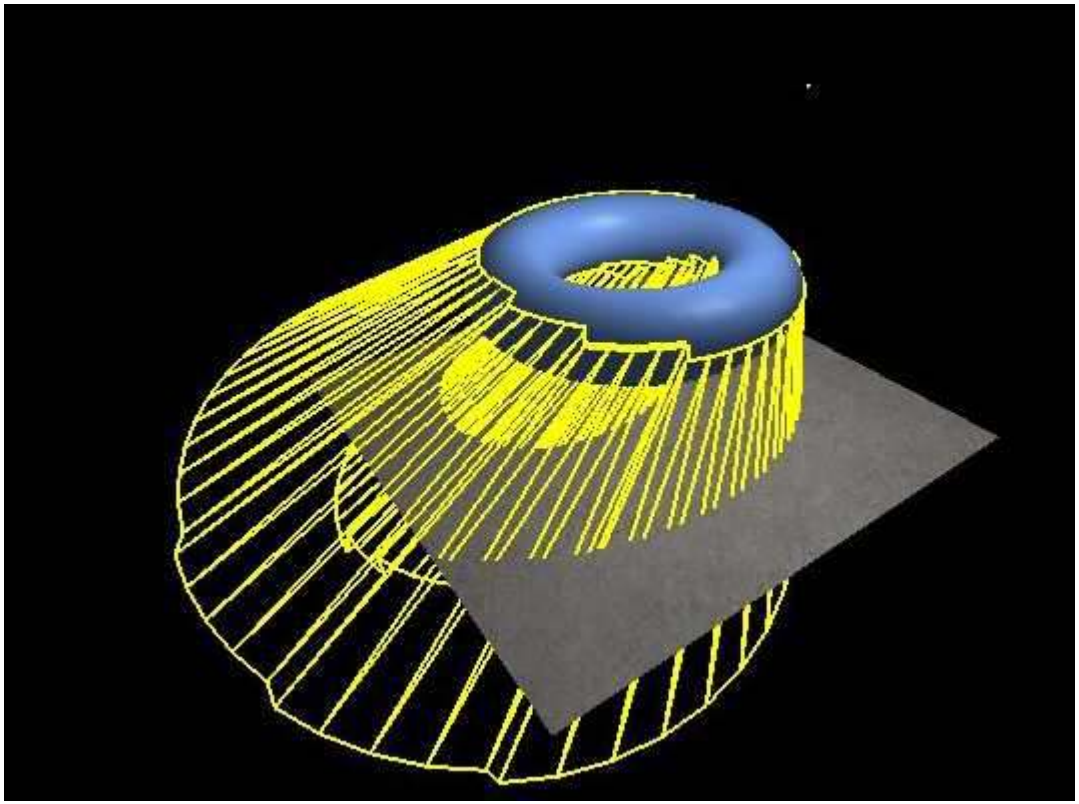
- 正确实现方式：
  1. 渲染阴影接收体的深度、环境光和自发光
  2. 对每个光源，把阴影投射体投射至指定平面
  3. 以 stencil 逐像素记录有没有阴影（即逐像素的 $V_L(\mathbf{x})$，而 $\mathbf{x}$ 必须在平面上）
  4. 重新着色接收阴影体，用 stencil 判断是否叠加光照结果
- Hack 方式：
  1. 正常渲染场景
  2. 对每个阴影投射体投射至指定平面，以 stencil 记录
  3. 渲染全屏长方形，把阴影部分以 alpha 混合加深

阴影体积

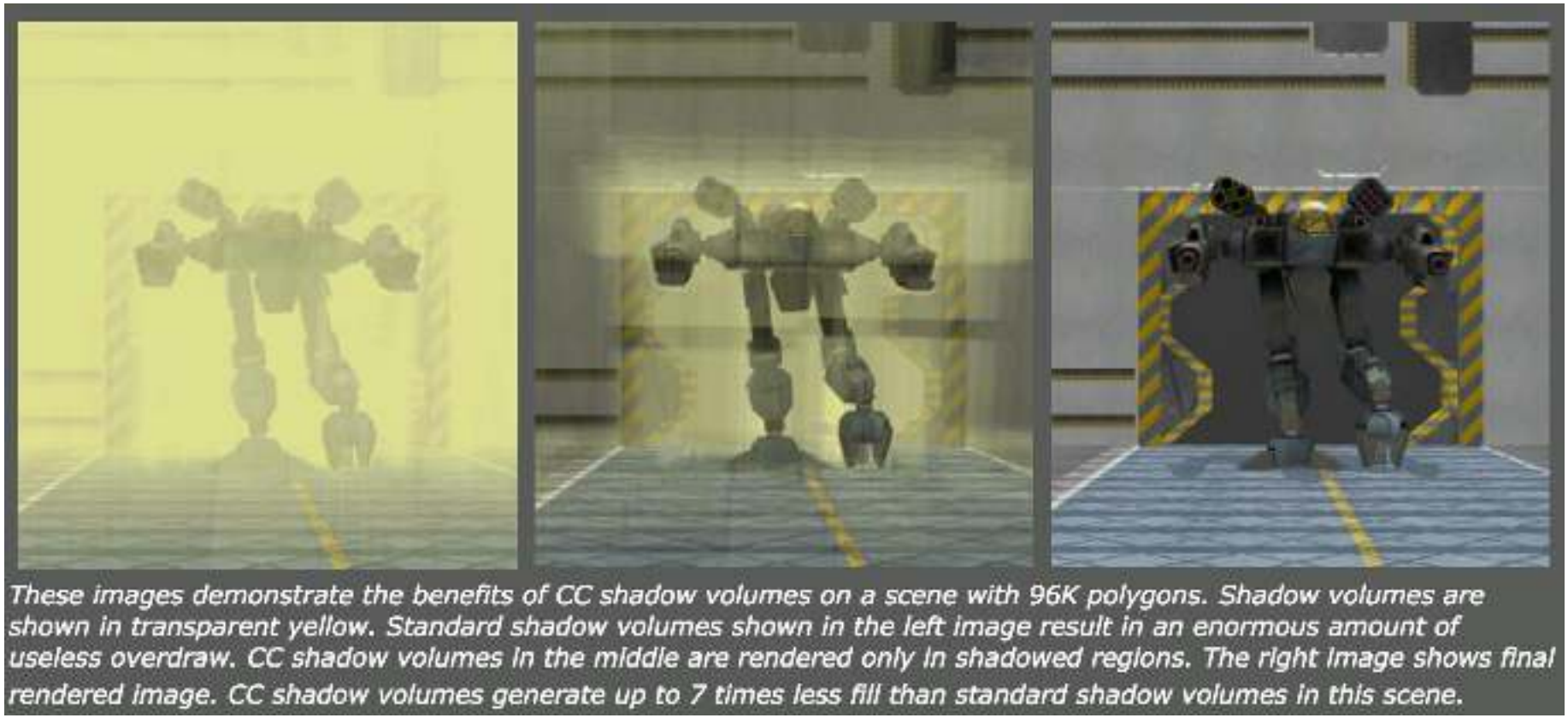
- 阴影体积（shadow volume）由 Crow（1977）发明
- 后来 Heidmann（1991）使用 stencil 实现
- 之后因《毁灭战士3》（2004）而闻名

阴影体积原理

1. 从光源方向，找出阴影投射体的轮廓边（silhouette）
2. 从光源方向，把轮廓边拉伸出（extrude）阴影体积
3. 分别渲染阴影体积的正面 / 反面部分，加 / 减 stencil 值
4. 非零的 stencil 值表示在阴影之中，用来渲染该光源



阴影体积的没落

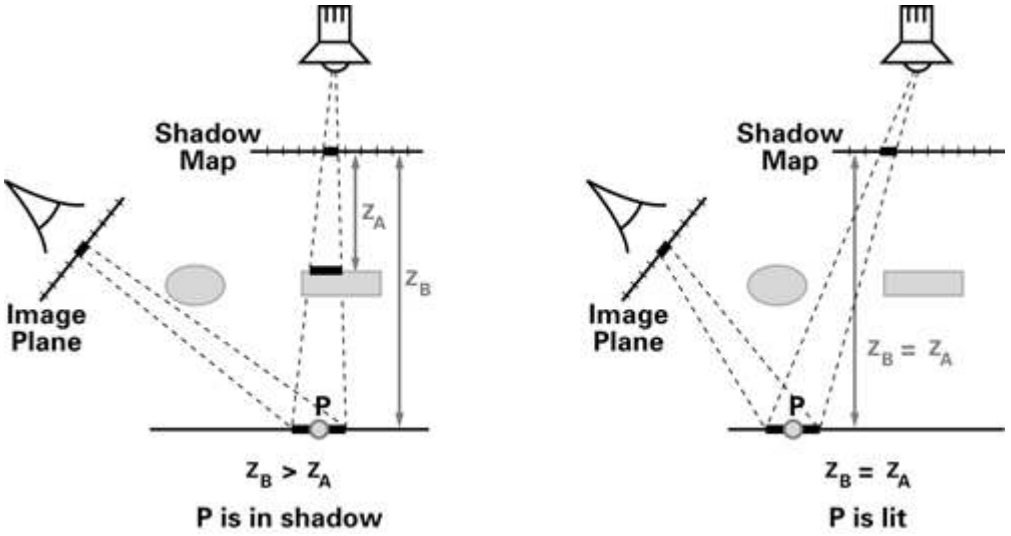


These images demonstrate the benefits of CC shadow volumes on a scene with 96K polygons. Shadow volumes are shown in transparent yellow. Standard shadow volumes shown in the left image result in an enormous amount of useless overdraw. CC shadow volumes in the middle are rendered only in shadowed regions. The right image shows final rendered image. CC shadow volumes generate up to 7 times less fill than standard shadow volumes in this scene.

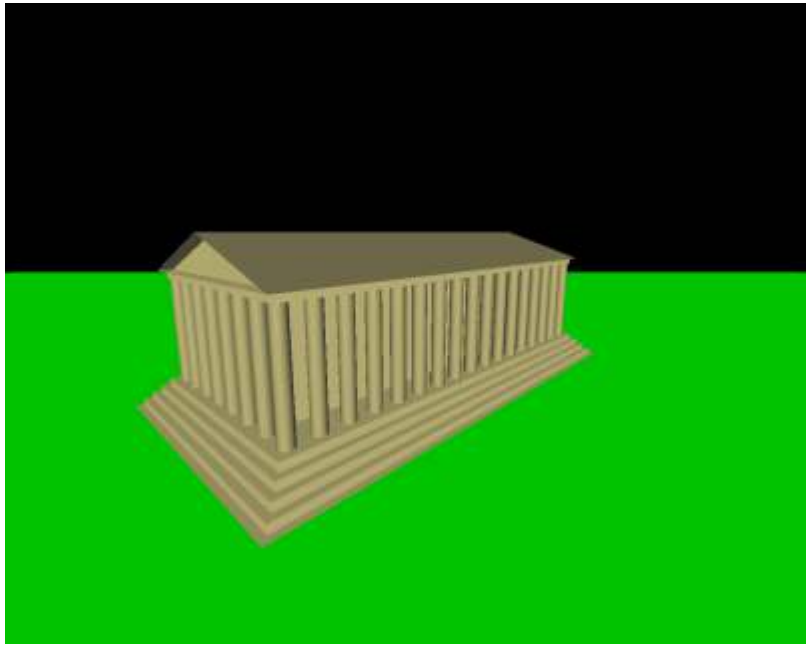
- 阴影体积 overdraw 太严重，即使优化后仍然非常高
- 1080p+分辨率难以承受，也限制阴影投射体的复杂度

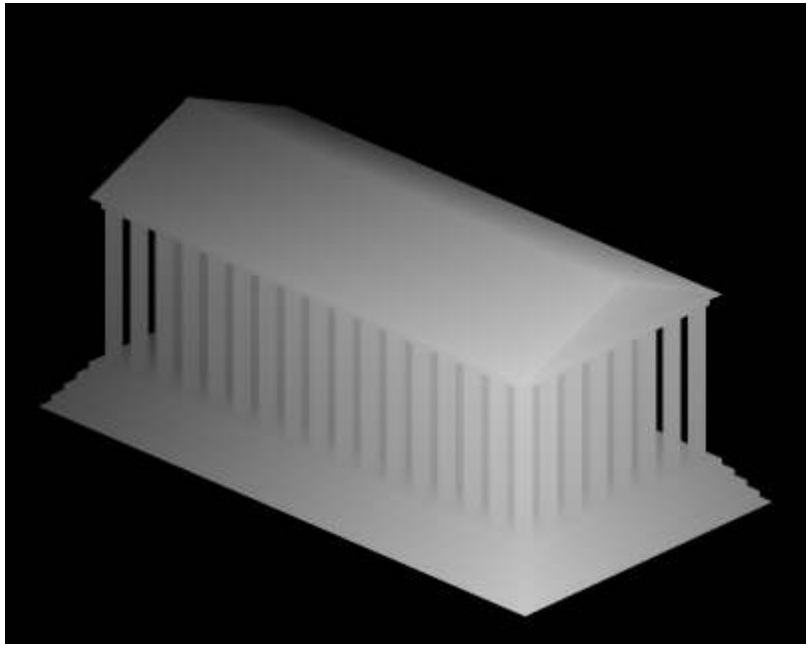
阴影贴图

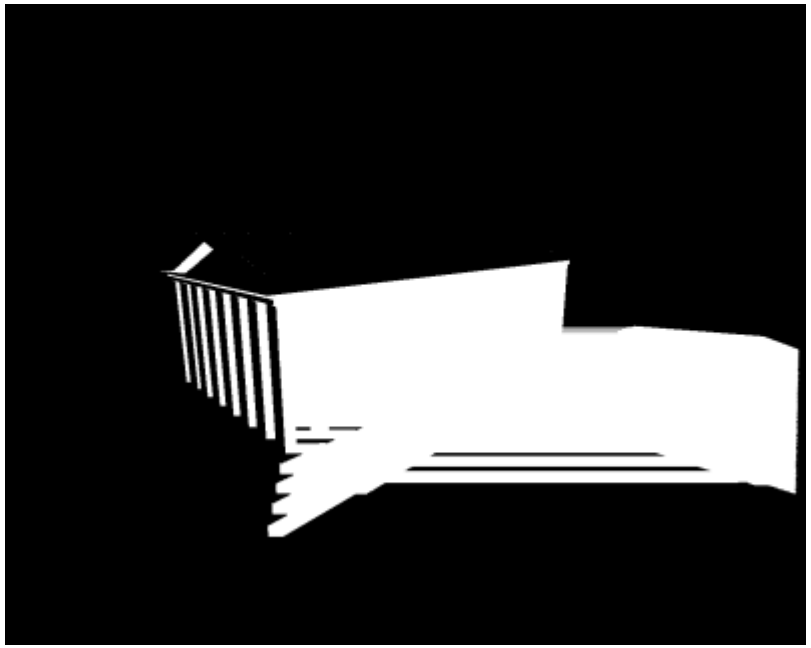
- 阴影贴图（shadow map）是现时的标准阴影算法群
- 先从光源方向渲染阴影投射物的**深度贴图**（阴影贴图）

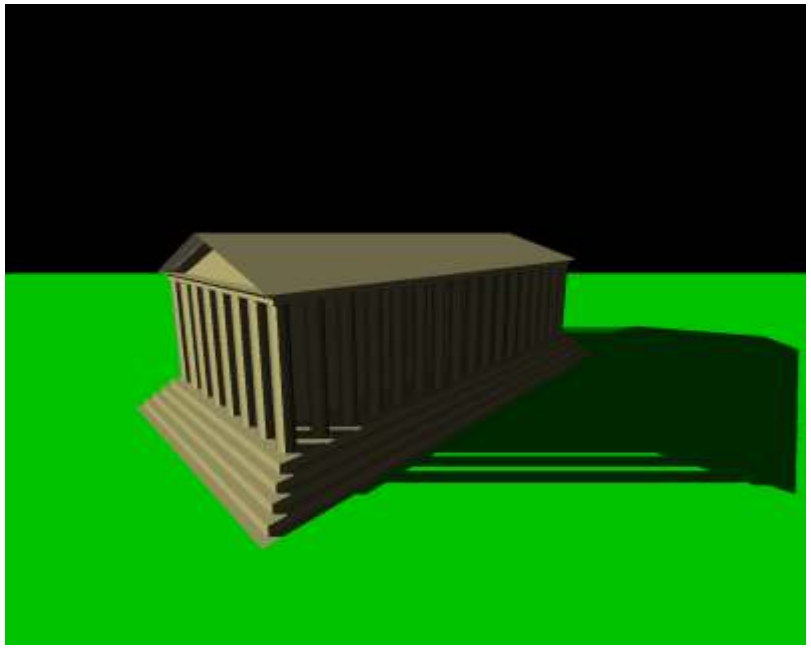


- 渲染阴影接收体时，把 **x** 转换至光源的坐标系，采样光源至 **x** 方向的最近距离
- 若该距离比 **x** 的距离近，即在阴影中



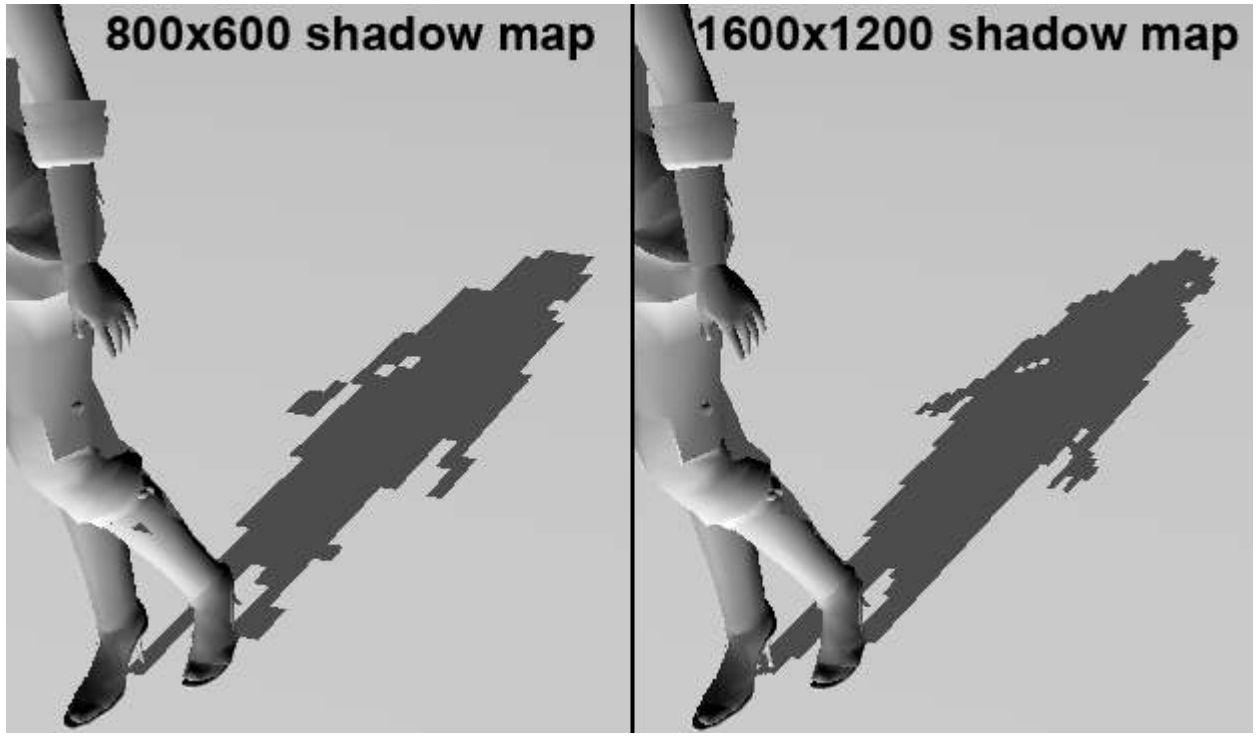




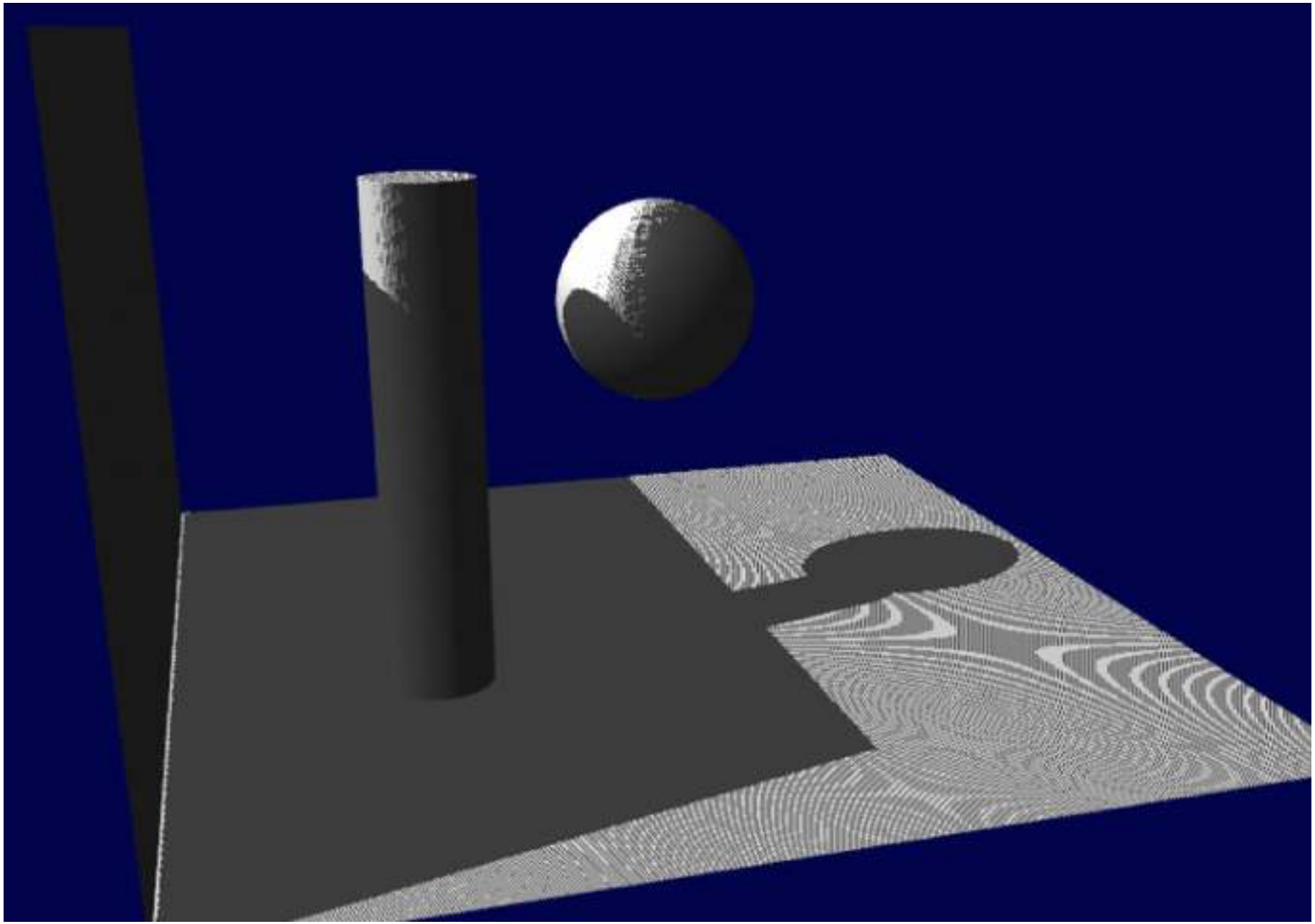


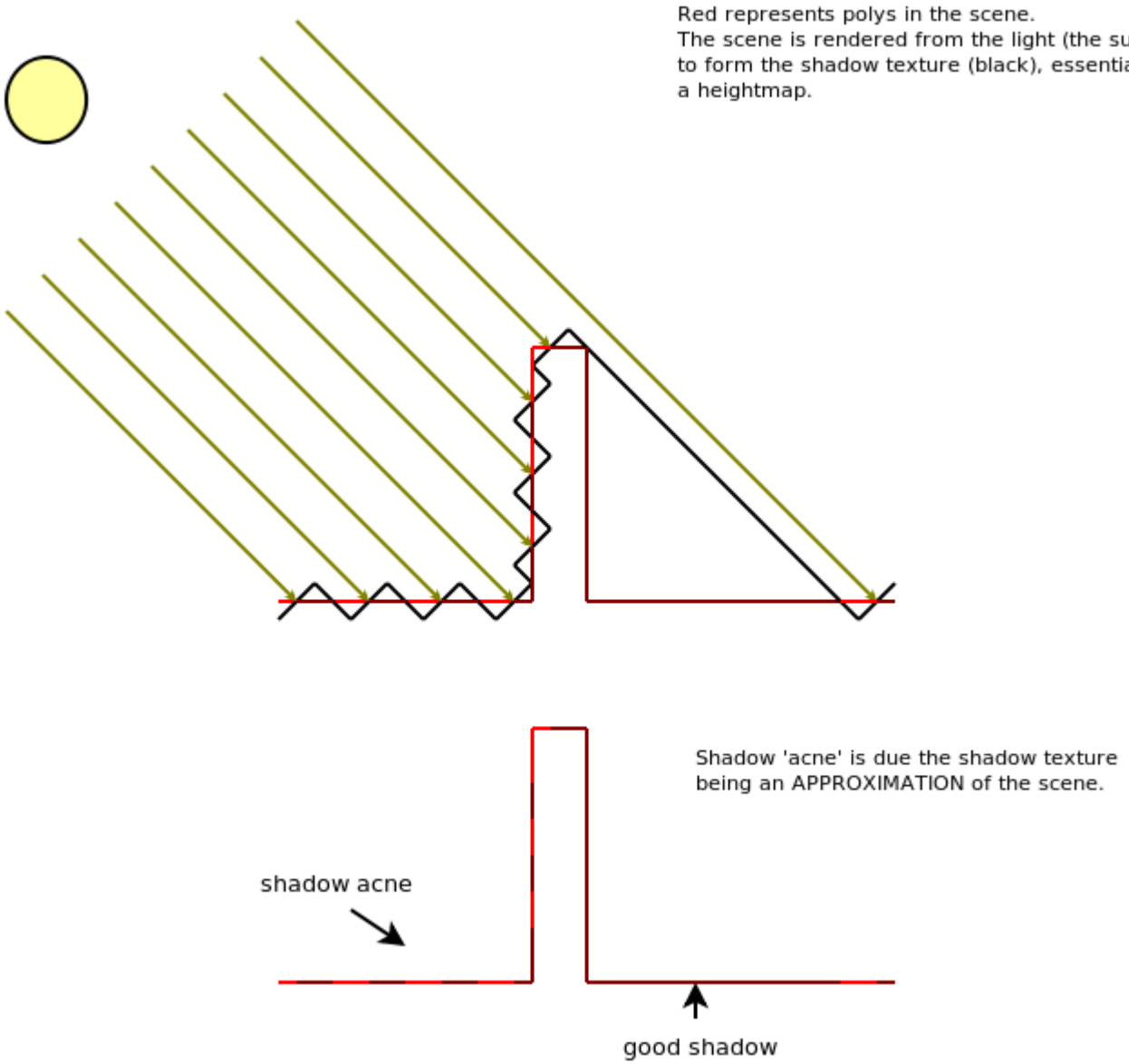
阴影贴图优点

- 容易实现，适合硬件性能好
- 可见性函数 $V_L(\mathbf{x})$ 存储在贴图中，和屏幕分辨率无关，可调整精度

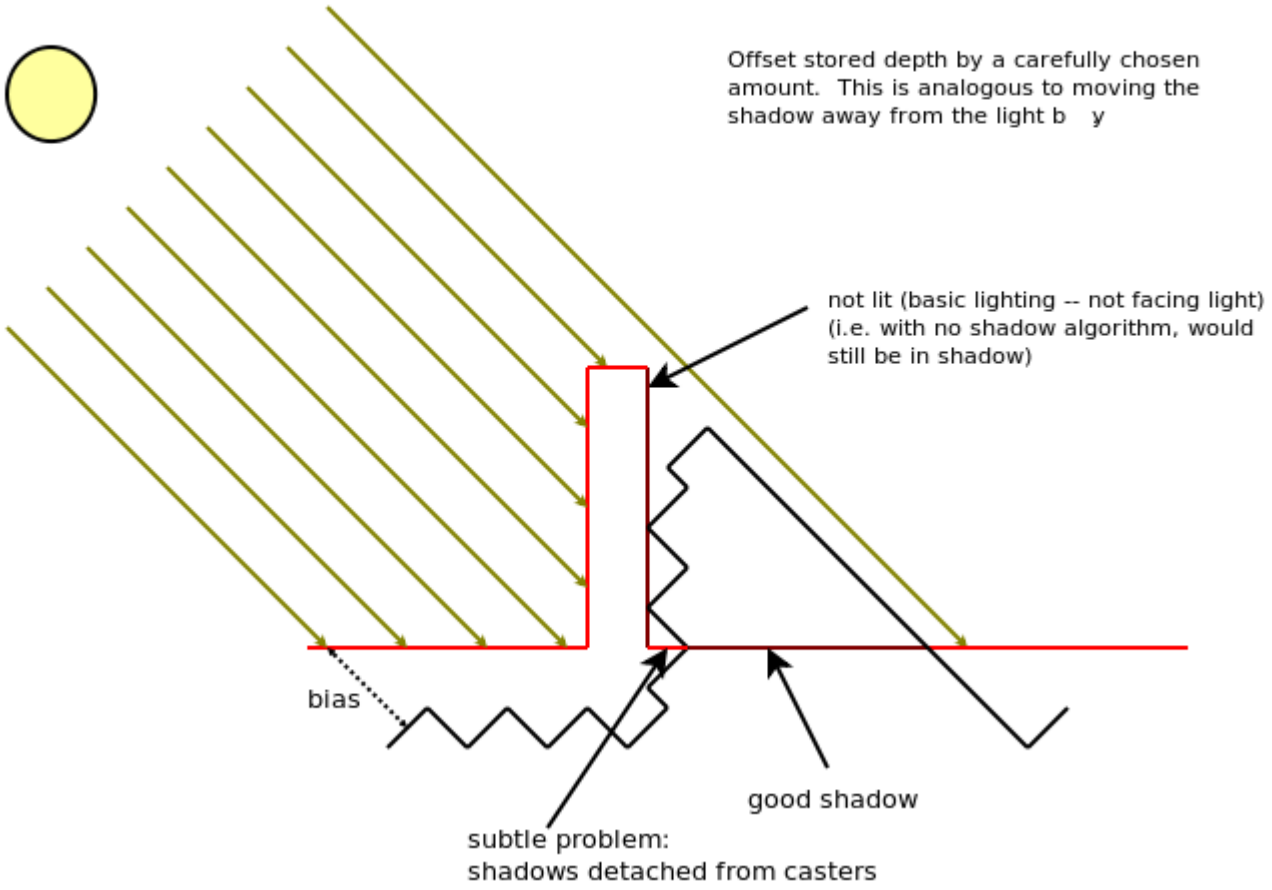


阴影贴图的问题

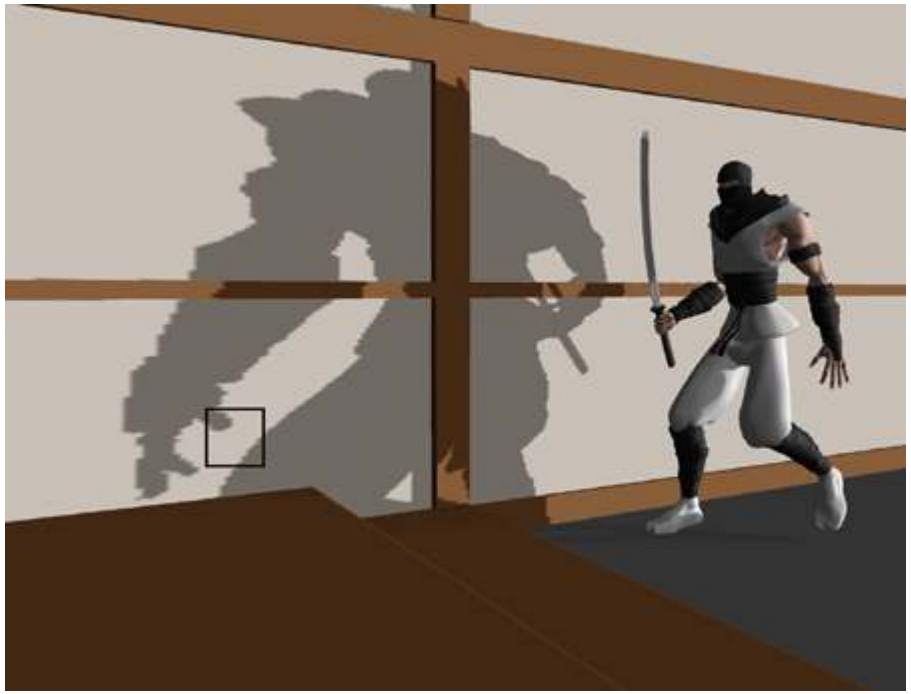



Red represents polys in the scene.
The scene is rendered from the light (the sun)
to form the shadow texture (black), essentially
a heightmap.
Shadow 'acne' is due the shadow texture
being an APPROXIMATION of the scene.
shadow acne
good shadow

FIX: Bias


Offset stored depth by a carefully chosen amount. This is analogous to moving the shadow away from the light b y
not lit (basic lighting -- not facing light) (i.e. with no shadow algorithm, would still be in shadow)
bias
good shadow
subtle problem:
shadows detached from casters

阴影锯齿



- 把深度贴图模糊化不能解决问题
- 问题在于 $V_L(\mathbf{x}) = \{0, 1\}$ 只有两个值

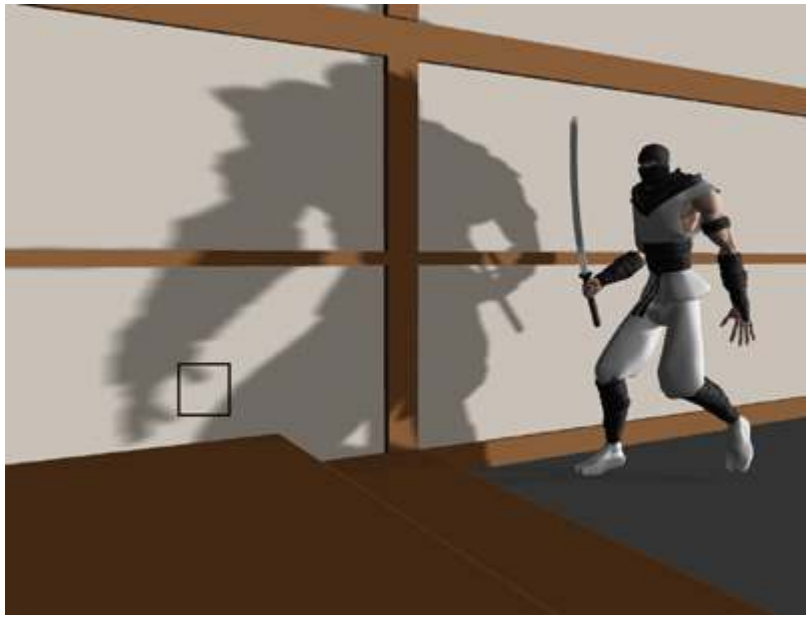
百分比渐近过滤

- 百分比渐近过滤（percentage closer filtering, PCF）是最简单方法
- 在采样点相邻位置采样，计算平均值

0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	1	1	1
0	0	0	0	0	1	1	1	1
0	0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1	1
0	0	0	0	1	1	1	1	1
1	1	1	1	1	1	1	1	1

- Bunnell et al, Shadow Map Antialiasing, GPU Gems, 2004.

PCF品质和性能成反比



(a) (b) (c)

方差阴影贴图

- 深度贴图每像素只记录一个深度，不能过滤
- 方差阴影贴图（Variance shadow map, VSM）为每像素计算深度的**分布**，就可过滤
- 相同品质下，性能比 PCF 好
- Donnelly et al, Variance shadow maps, Proceedings of the 2006 symposium on Interactive 3D graphics and games. ACM, 2006.

方差阴影贴图过程

1. 渲染深度贴图 x，及其平方贴图 x^2
2. 用图像核 $p(x)$ （一般用高斯）过滤 x 和 x^2：

$$M_1 = E(x) = \int_{-\infty}^{\infty} xp(x)dx, \quad M_2 = E(x^2) = \int_{-\infty}^{\infty} x^2 p(x$$

3. 逐像素计算光照时，计算均值及方差

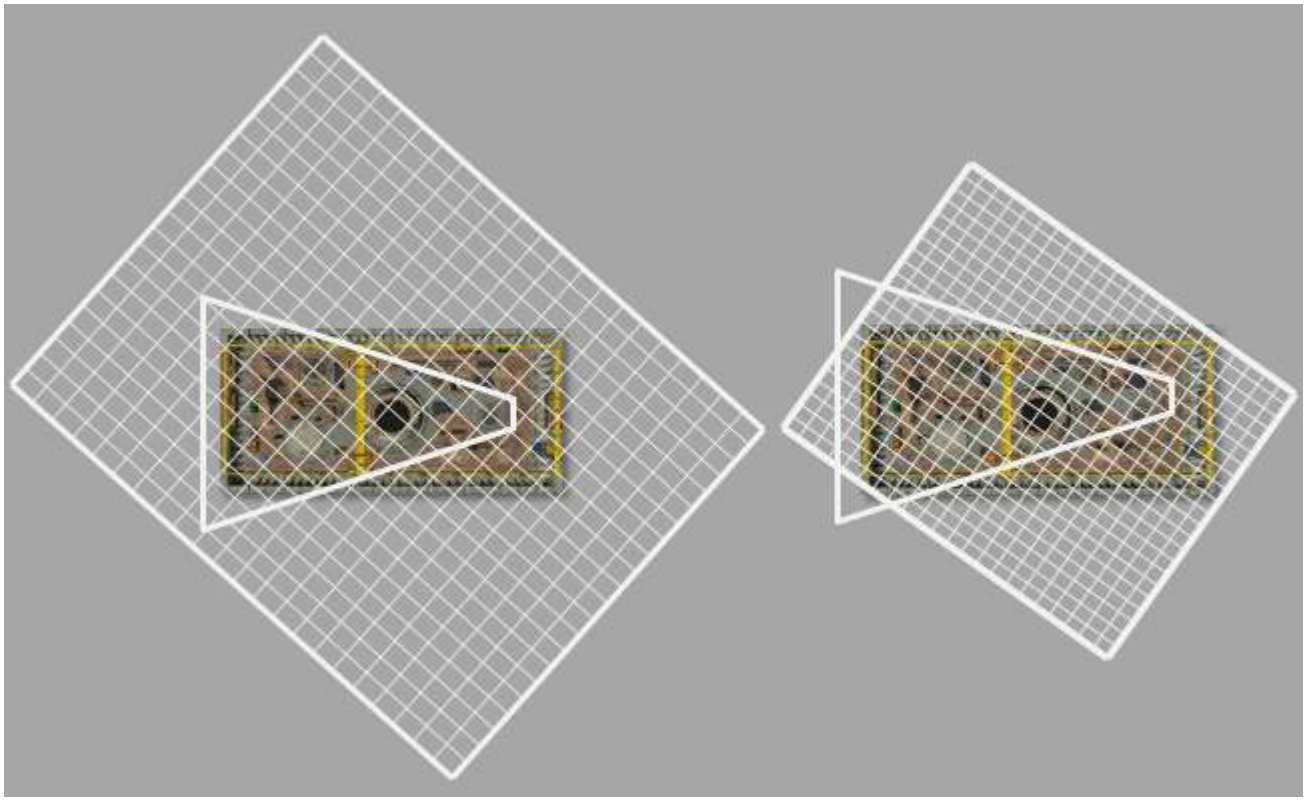
$$\mu = M_1, \quad \sigma = M_2 - M_1^2$$

4. 若像素的深度 $t > \mu$，用**切比雪夫不等式**求遮挡的百分比上界

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

方差阴影贴图效果

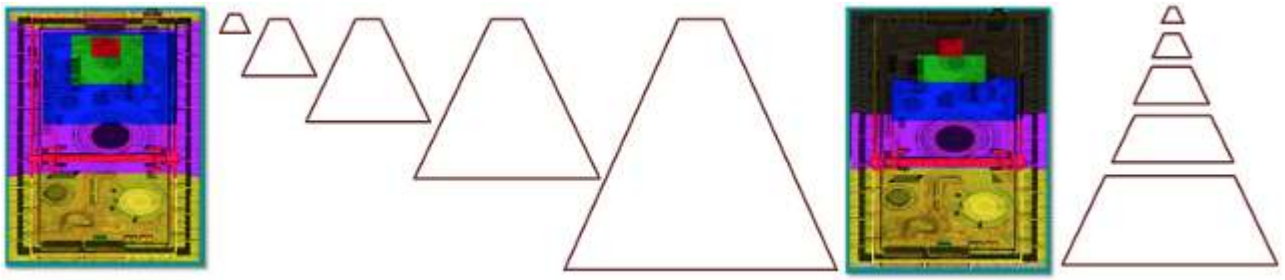
方向光的投影范围问题



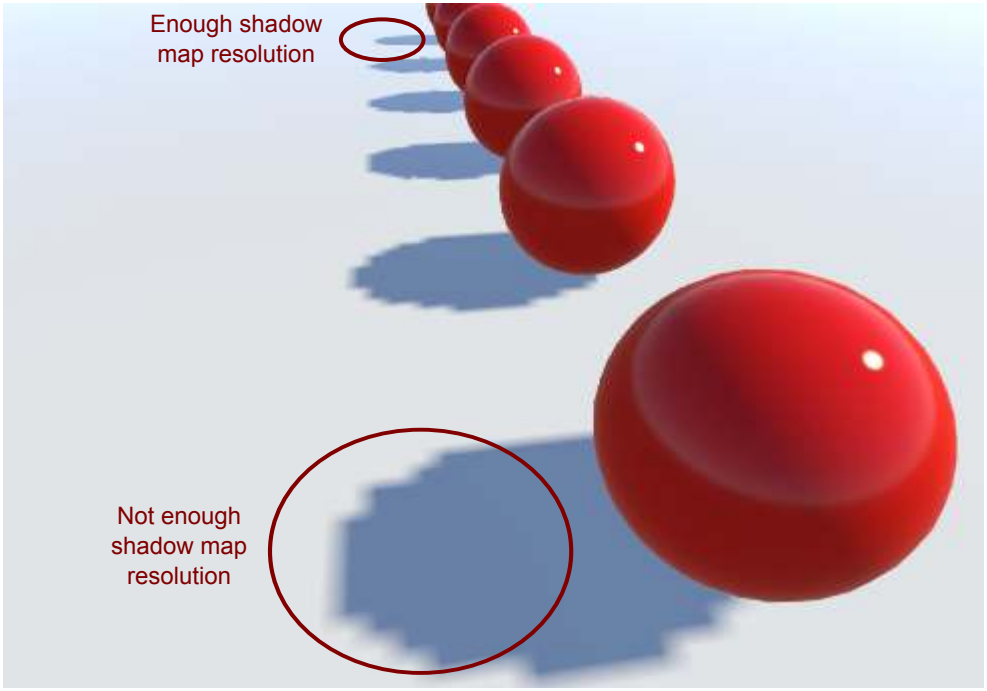
- 可以紧贴摄像头的平截头体及场景
- 但近景的深度贴图精度太低，远景又太高，怎样解决？

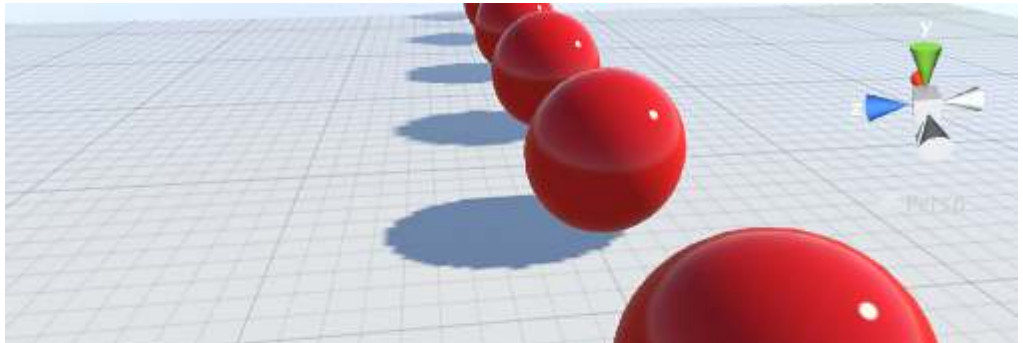
级联阴影贴图

- 一张阴影贴图无法满足，渲染多张深度贴图！
- 问题是如何设置多个投影范围



- Zhang et al, "Parallel-split shadow maps for large-scale virtual environments", VRCAI, 2006.
- Dimitrov et al, "Cascaded shadow maps", NVIDIA, 2007.


Enough shadow
map resolution
Not enough
shadow map
resolution



实时阴影技术

技术类型	例子
基本	SSM
分割	PSSM、CSM
扭曲	LiSPSM、TSM、PSM、CSSM
过滤	PCF、ESM、 CSM、 VSM、 SAVSM、 SMSR
软阴影	PCSS、 SSSS、 FIV
其他	ASM、 AVSM、 CSSM、 DASM、DPSM、DSM、FSM LPSM、MDSM、RTW、RMSM、SDSM、SPPSM、SSSM

参考

- Akenine-Möller, Tomas, Eric Haines, and Naty Hoffman. Real-time rendering. Chapter 7.1-7.4, 9.1, CRC Press, 2008.
- Eisemann, Elmar, et al. Real-time shadows. CRC Press, 2011.