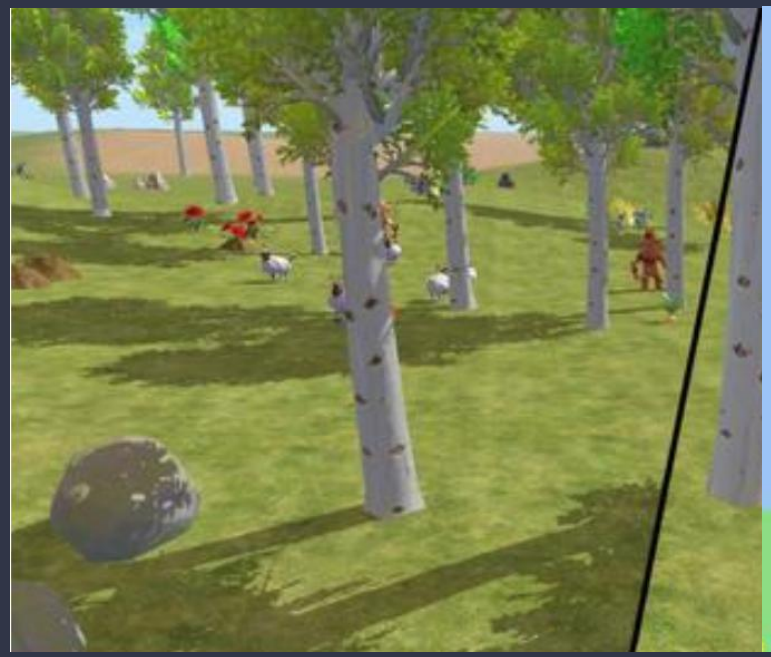
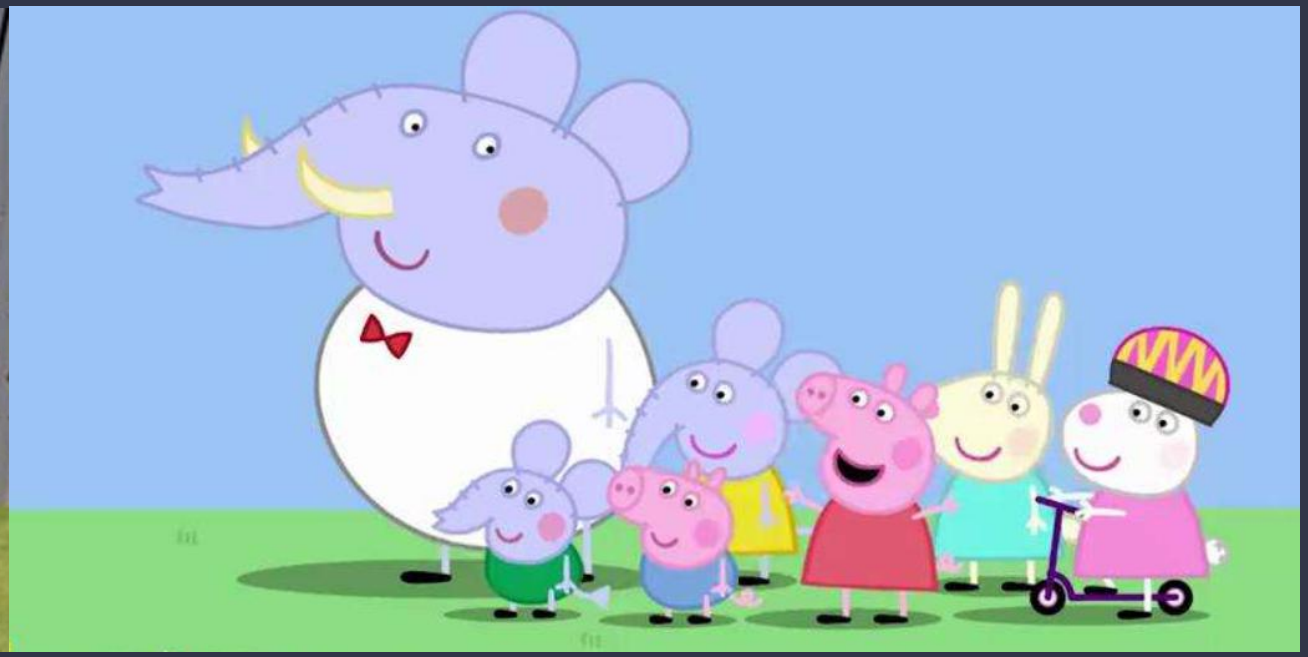
实时阴影技术概述

2020.12

什么是阴影

- 光在传播过程中，在物体后面光不能达到的地方就产生影
- 阴影的价值，大幅度增加渲染效果的真实感
- 阴影本身不一定是真实的！



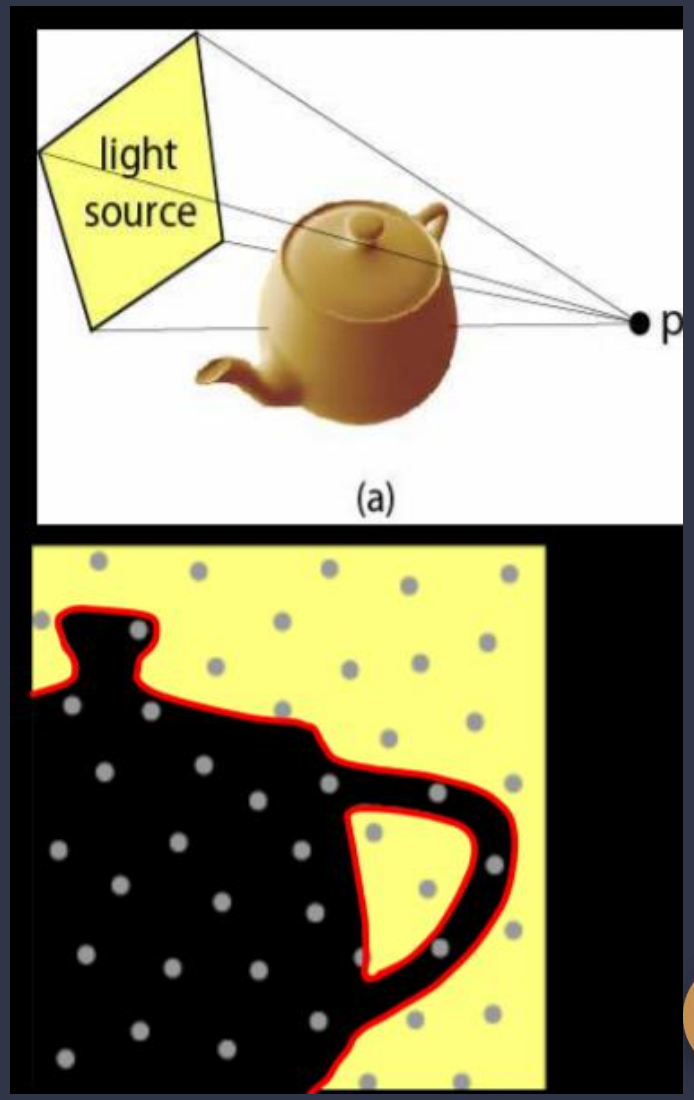


阴影实现的两种思路

- 几何法：Shadow Volume
  - 构建阴影体SV
  - ->渲染场景Depth->关闭DepthWrite->渲染SV正面，Pass则模板加1 ->渲染SV背面，Pass则模板减1->开启DepthWrite->带模板（≠0）渲染场景
  - （卡马克反转）渲染SV背面，Fail则模板加1->渲染SV正面，Fail则模板减1
  - 优点：没有走样问题
  - 缺点：场景复杂时处理困难，无法处理纹理阴影
- 图像法：Shadow Map

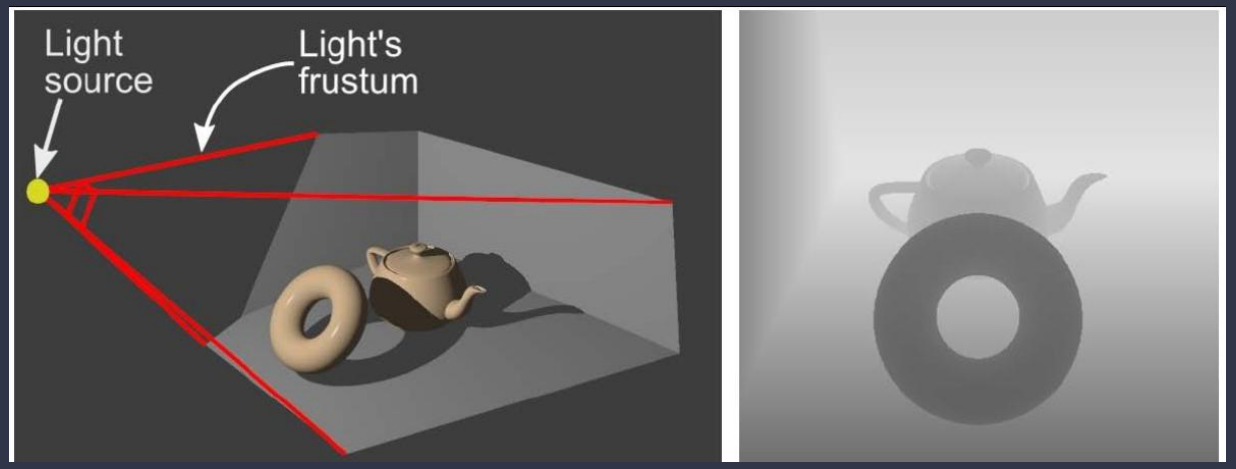
一分钟理解阴影图

- PASS1：以光源的位置为视点，渲染场景
  - 记录渲染结果到一张深度纹理中
- PASS2：正常渲染场景，同时将深度纹理投影到场景里
  - 在某个片元上，如果深度值<片元与p的距离，则为阴影
  - 方法简单，适应性强
- 我就是太阳！
- 一切问题都来自于图像的不连续性！



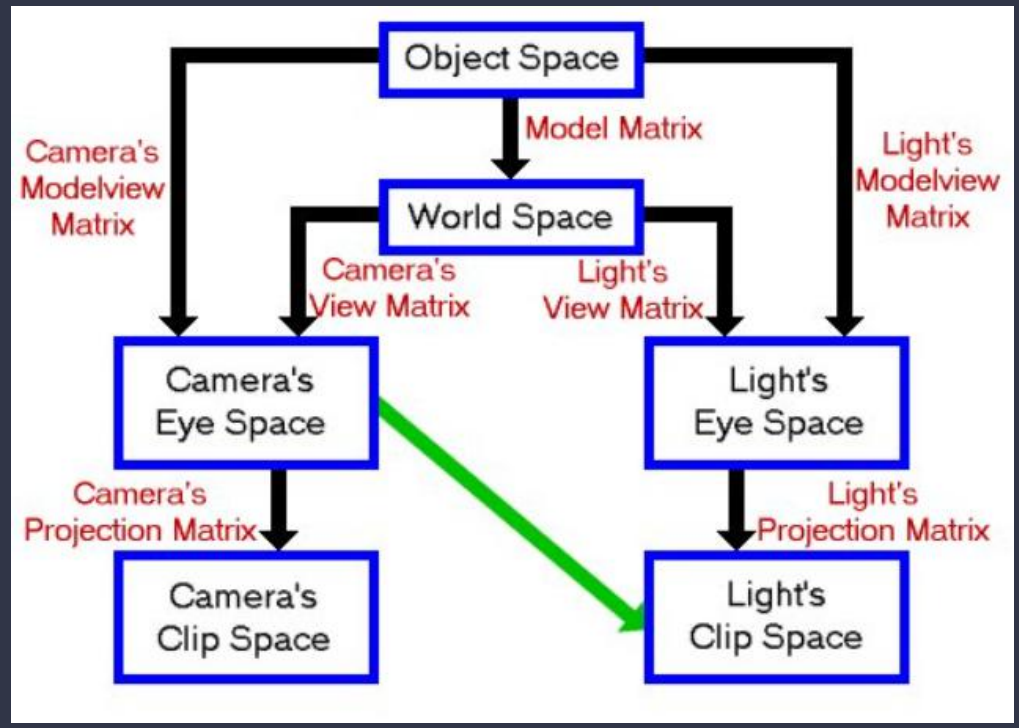
基于OpenGL的实现流程

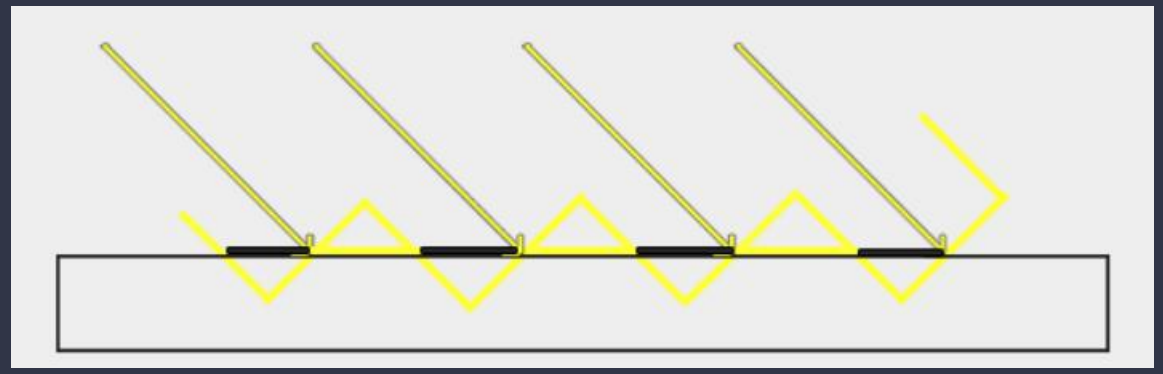
- 创建阴影纹理
  - glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_COMPARE_MODE, GL_COMPARE_R_TO_TEXTURE);
  - glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_COMPARE_FUNC, GL_LEQUAL);
  - glTexParameteri(GL_TEXTURE_2D, GL_DEPTH_TEXTURE_MODE, GL_INTENSITY);
  - glTexImage2D(GL_TEXTURE_2D, 0, GL_DEPTH_COMPONENT24, sizeX, sizeY, 0, GL_DEPTH_COMPONENT, GL_UNSIGNED_BYTE, NULL);



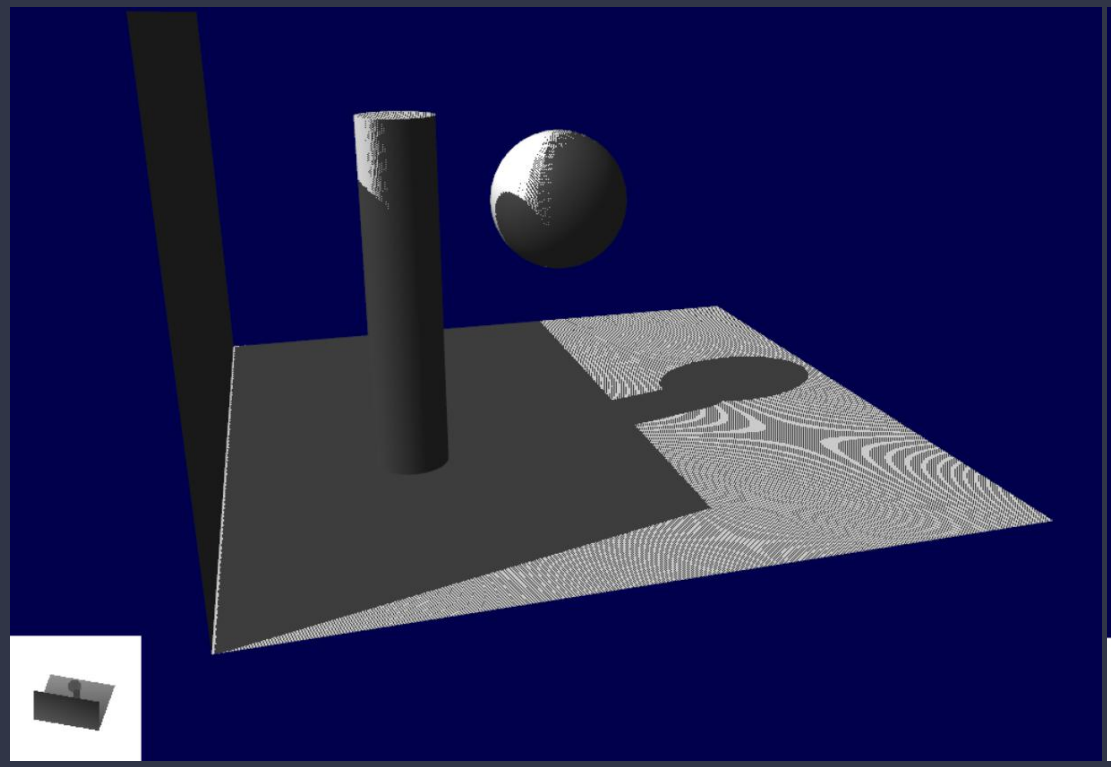
基于OpenGL的实现流程

- 投影到场景中
  - $T = P_L \cdot V_L \cdot V_C^{-1} = P_L \cdot V_{Lview} \cdot V_{Lproj} \cdot (V_{Cview} \cdot V_{Cproj})^{-1}$
  - $M_{eye} = V_{Lview} \cdot V_{Lproj} \cdot V_{bias}$
  - glTexGeni(GL_S, GL_TEXTURE_GEN_MODE, GL_EYE_LINEAR);
  - glTexGenfv(GL_S, GL_EYE_PLANE, M_{eye}.Row0);
  - glEnable(GL_TEXTURE_GEN_S);
  - …… // S/T/R/Q
- 偏移矩阵的作用
  - 解决Shadow Ance问题

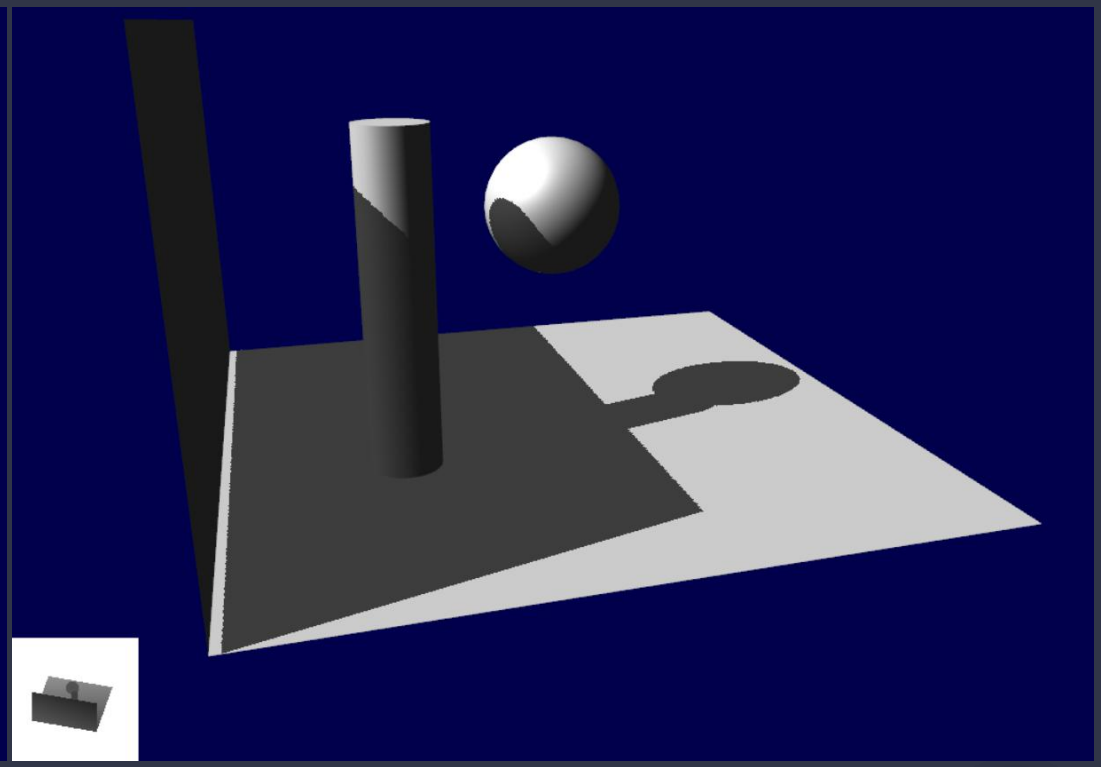




阴影图的主要问题



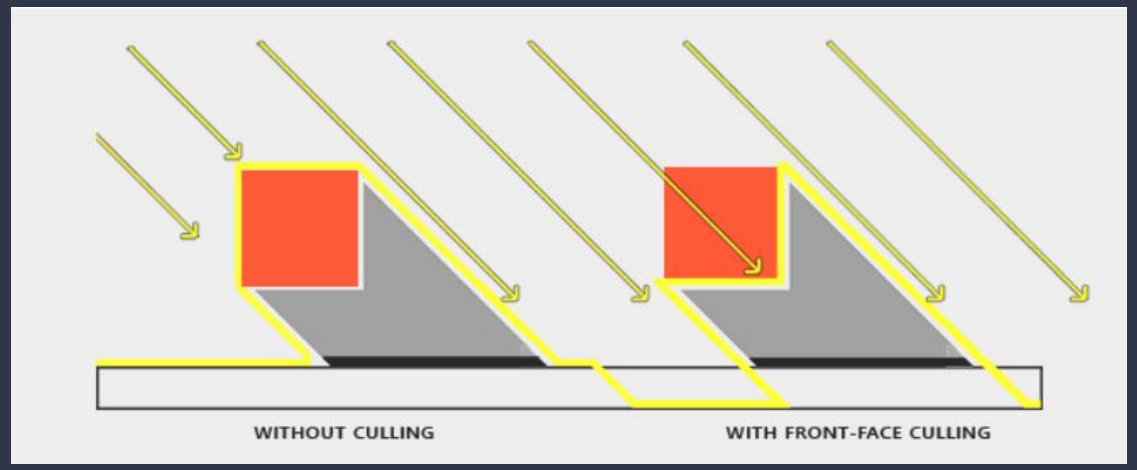
Shadow Acne

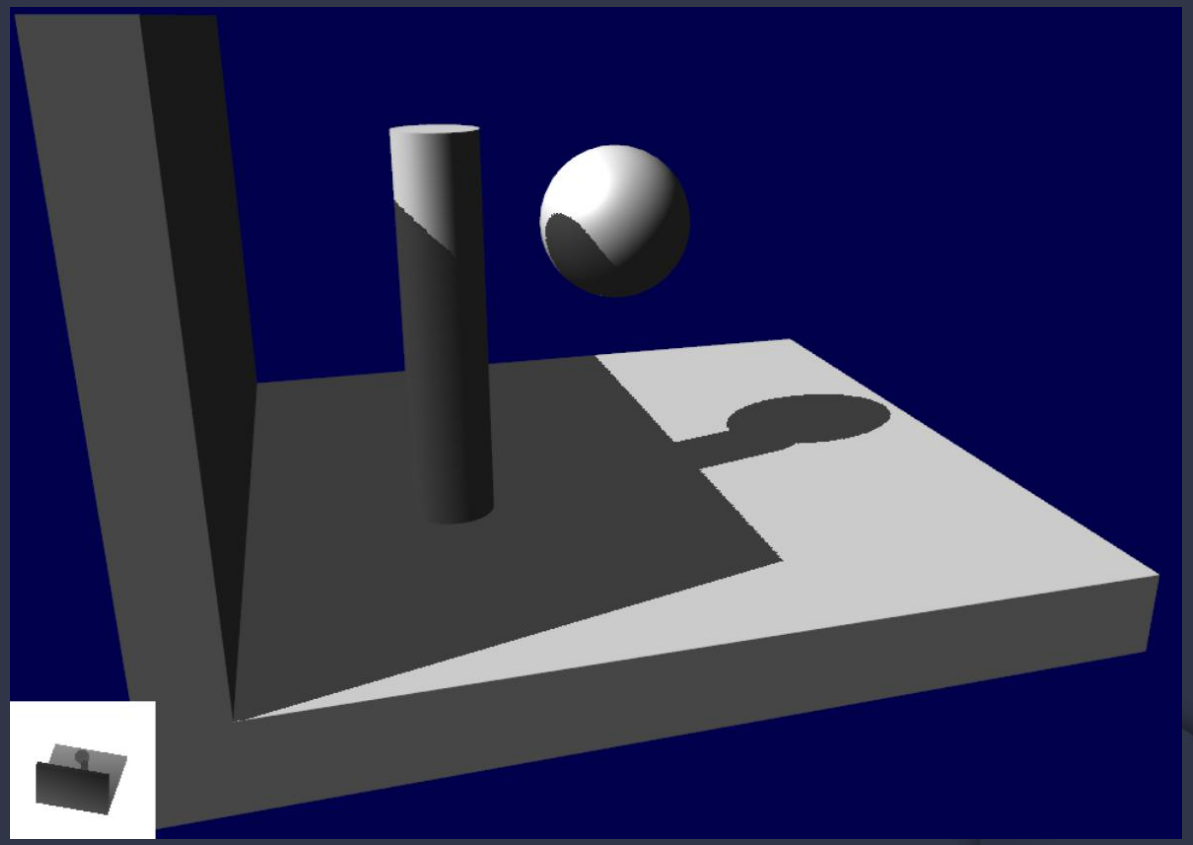


Peter Panning

阴影图的主要问题

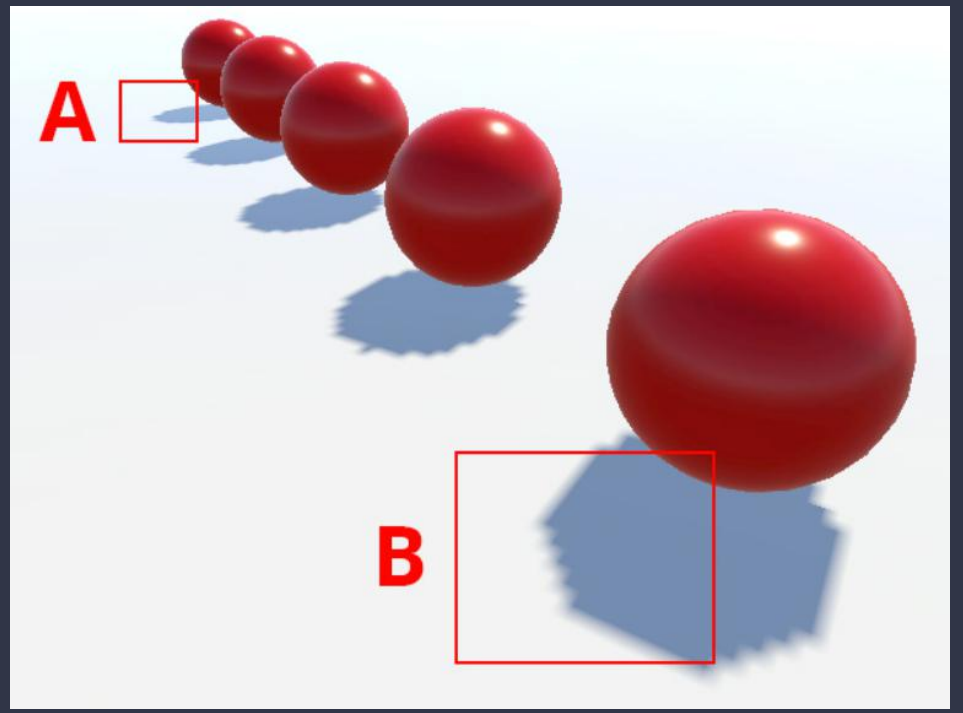
- 场景物体避免Thin Plane
- PASS1：只渲染背面
- PASS2：正常渲染

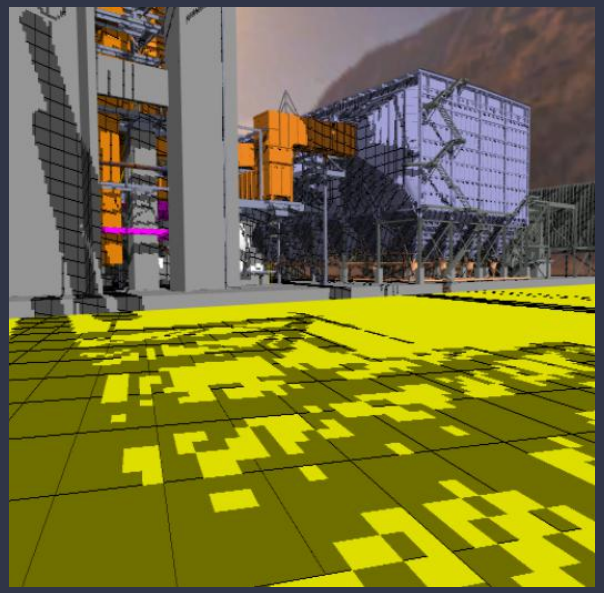


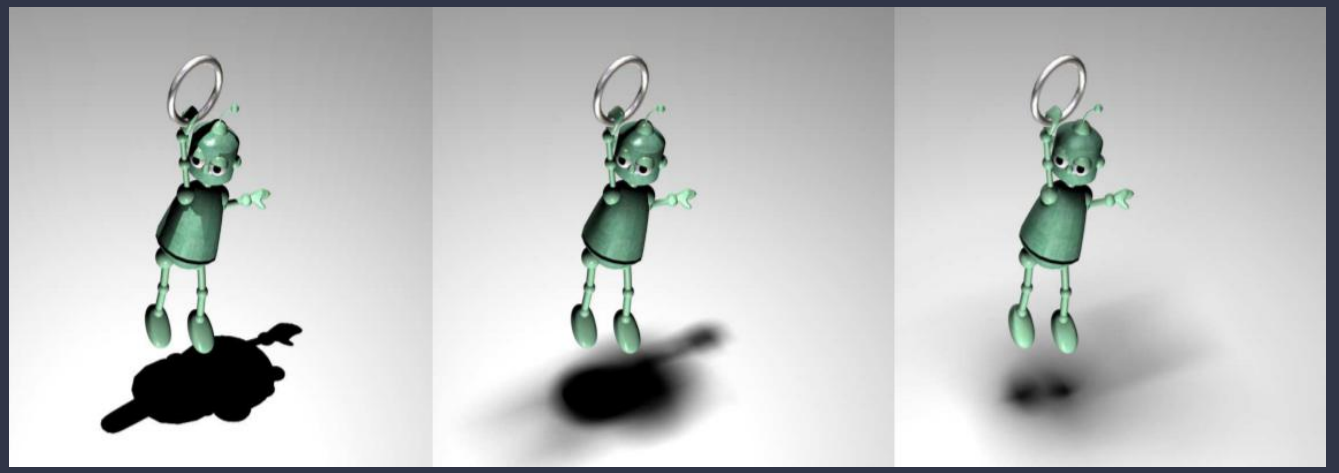


阴影图的主要问题

- 阴影分辨率和场景大小的矛盾
- 阴影边缘的柔和度问题
- 阴影利用的效率问题

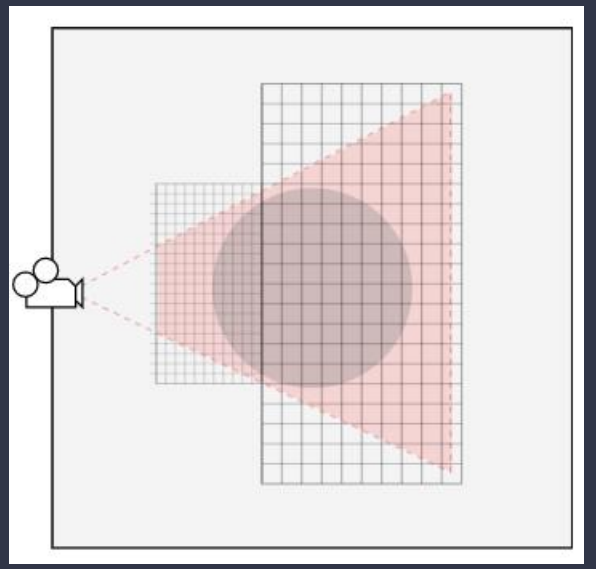


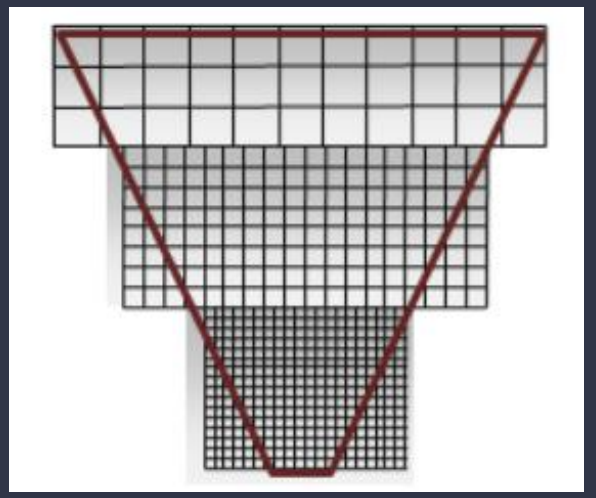


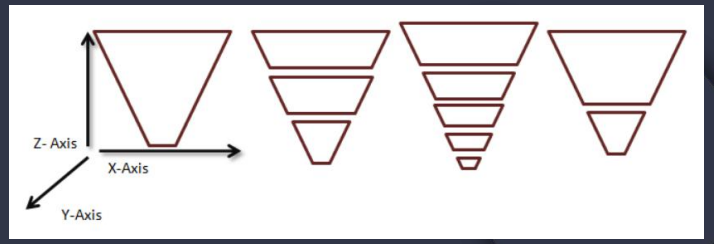


级联阴影图CSM

- 无法一直提升单张深度纹理的分辨率
- 将纹理划分为多个层级，依次渲染
  - Cascaded Shadow Maps
  - Parallel-Split Shadow Maps
- 延展问题：
  - 划分多少级
  - 怎么划分
  - SDSM

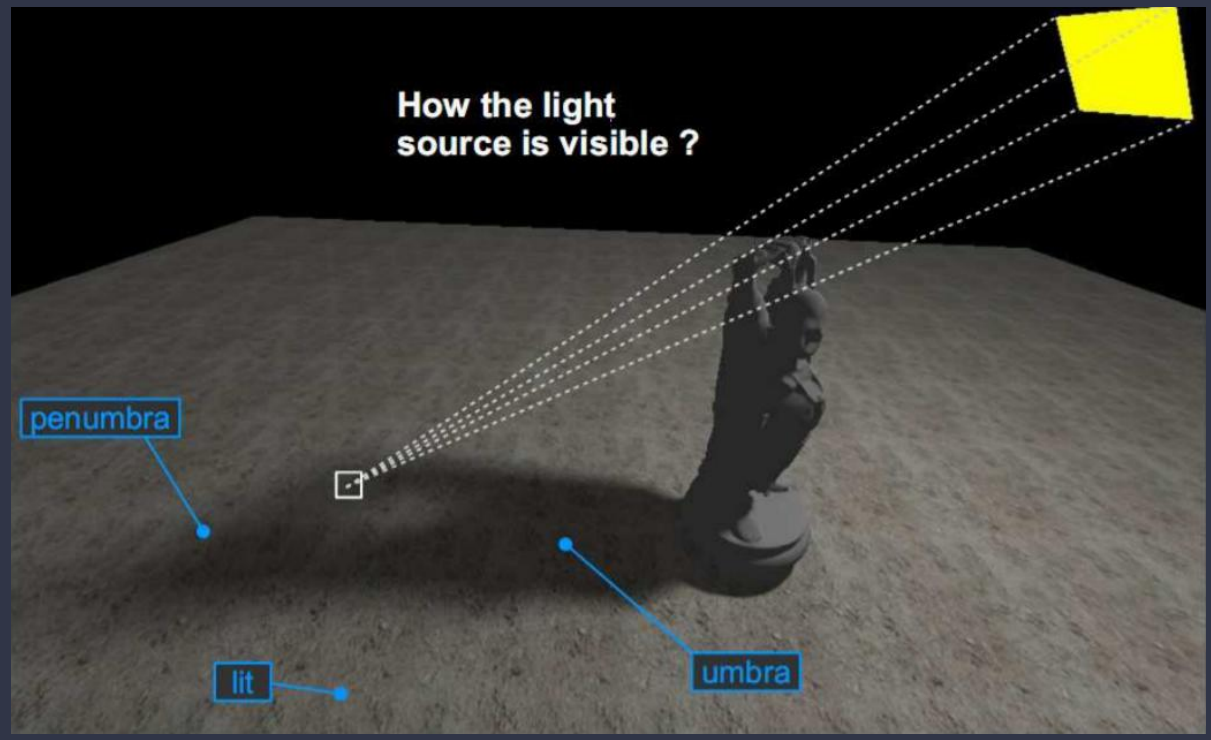






软阴影SSM

- 本影，半影的概念
- 对阴影硬边缘进行羽化
  - 后处理：深度纹理投影时进行羽化
  - 预处理：深度纹理生成时进行羽化
- 软阴影的两个核心需求：
  - 一致性需求
  - 按需计算半影区距离



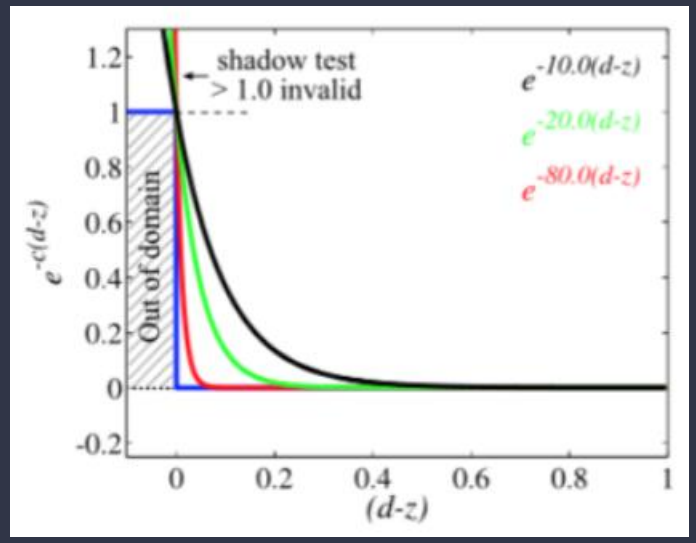
后处理软阴影

- Percentage-Closer Filtering（PCF）
  - for (y = -1.5; y <= 1.5; y += 1.0)
  - for (x = -1.5; x <= 1.5; x += 1.0)
  - sum += offset_lookup(shadowmap, shadowCoord, float2(x, y));
  - shadowCoeff = sum / 16.0;

- Percentage-Closer Soft Shadows（PCSS）
  - 计算某个点的平均遮挡距离
  - 找到可能的半影区域
  - 让PCF动态适应半影的大小

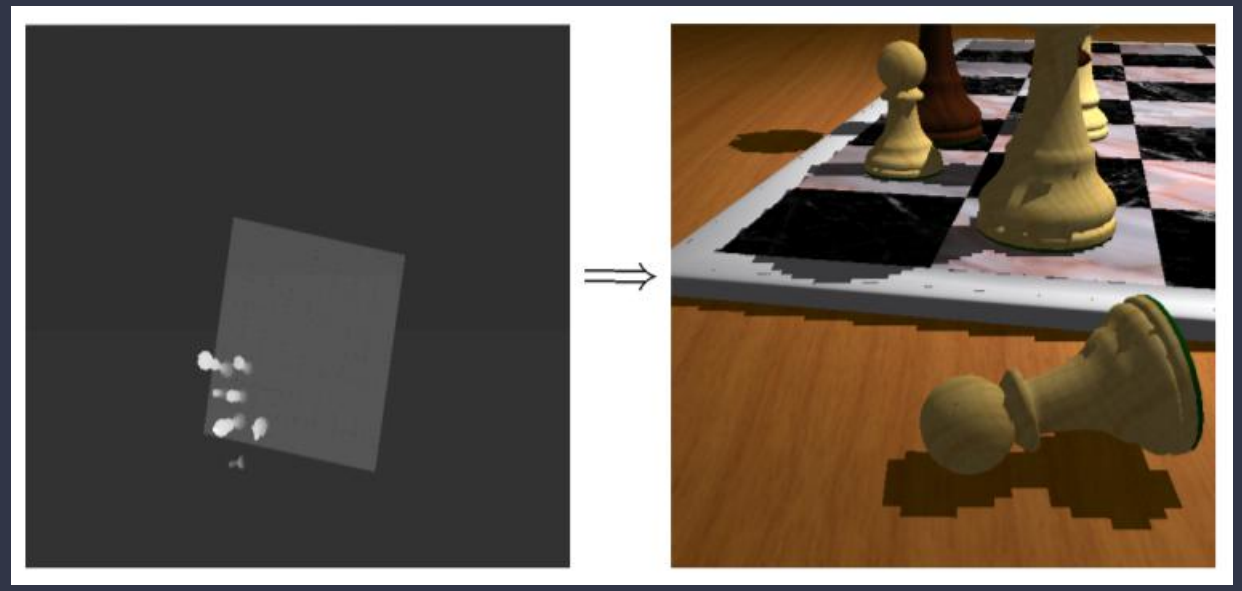
预处理软阴影

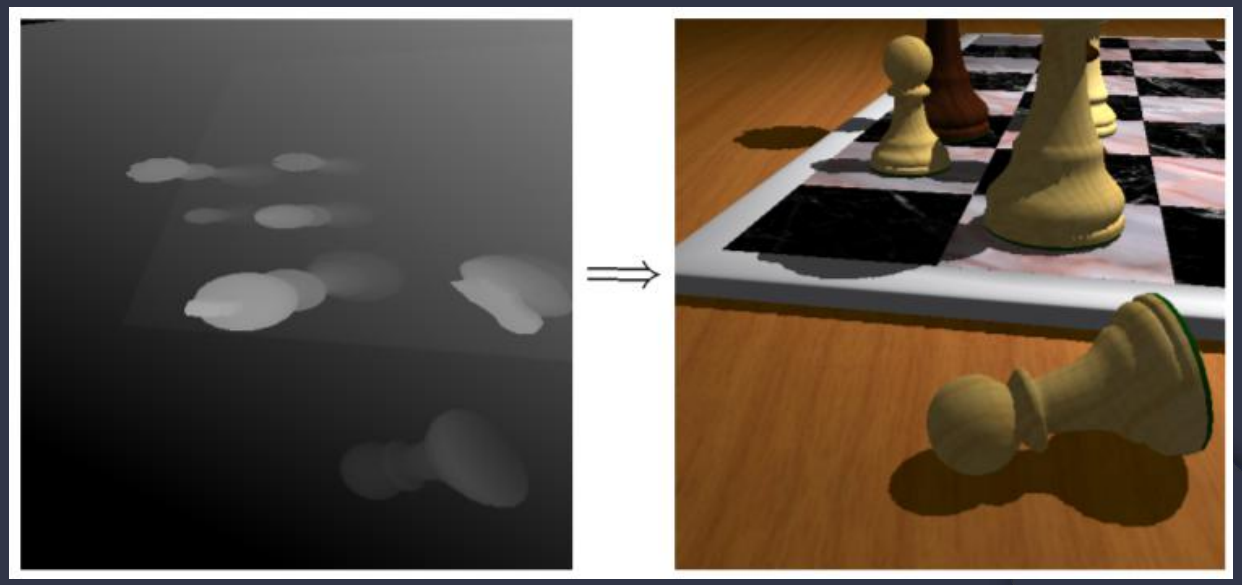
- Variance Shadow Mapping（VSM）
  - 运用统计学的切比雪夫不等式
  - 保存原始深度d和深度的平方d2，并进行滤波和mipmap
  - 结合d和d2计算方差与期望，对原始的x做概率估计
  - 纹理投影到场景后，深度值大于期望值为阴影区；接近期望值则根据概率决定
- Exponential Shadow Mapping（ESM）
  - $S(x)=e^{-cd}e^{cz}$，c为常数
  - 保存指数值$e^{-c}e^{cz}$，并进行滤波和mipmap
  - d>z时，处于阴影区；c越大，半影区域越收敛



另辟蹊径

- 解决分辨率问题的另一种途径：
  - 提升空间利用效率
  - 对光源相机的投影矩阵进行变形
- Perspective Shadow Maps (PSM)
- Trapezoidal Shadow Maps (TSM)





在OSG里实现阴影图

- shadowedScene = new osgShadow::ShadowedScene;
- shadowedScene->setReceivesShadowTraversalMask(ReceivesShadowTraversalMask);
- shadowedScene->setCastsShadowTraversalMask(CastsShadowTraversalMask);
- shadowedScene->addChild(sceneSoot);
- shadowedScene->addChild(lightSource);

- osg::ref_ptr<osgShadow::ViewDependentShadowMap> vdsm =
  new osgShadow::ViewDependentShadowMap;
- shadowedScene->setShadowTechnique(vdsm);

- osgShadow::ShadowSettings* settings = shadowedScene->getShadowSettings();
- .....

osgShadow的得与失

- 支持固定管线的ShadowMap
- 支持PSSM级联阴影
- 侵入式的场景节点设计
- 没有及时跟进最新算法
- 大地形上的阴影支持问题较多
- **计划奉献：**
  - **ESM+n级CSM+PCSS！**
  - **支持数字地球级别的阴影渲染**