如何打造一款秒级别的全局光烘培软件

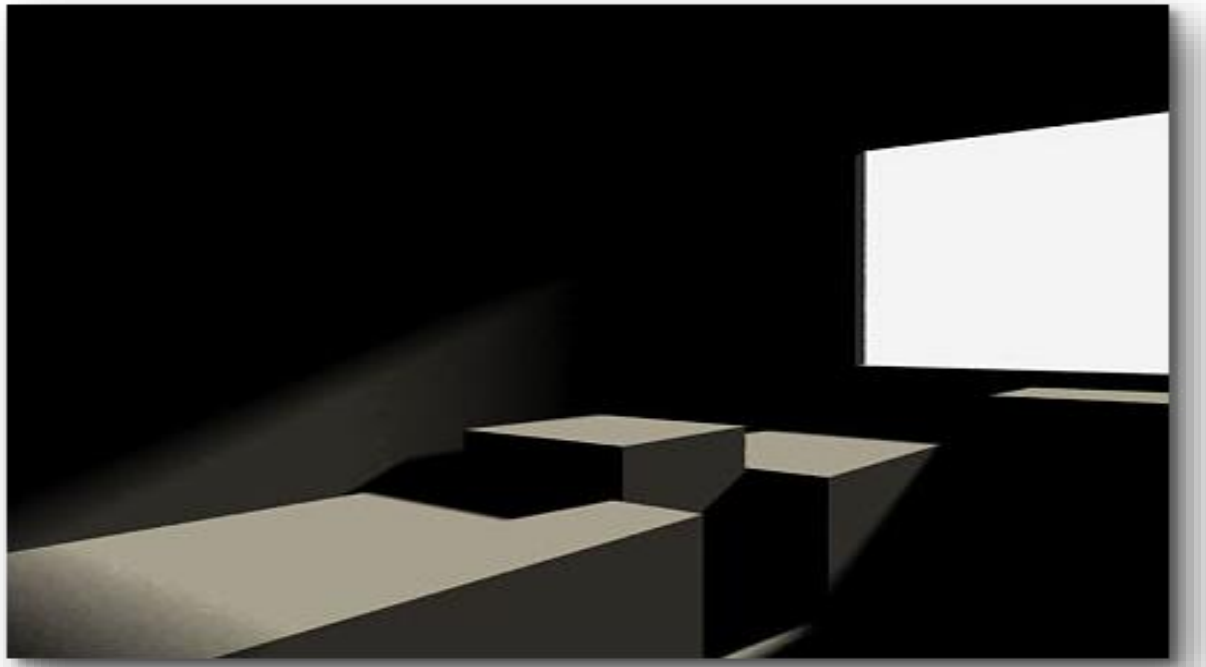
李文耀
网易游戏 游戏引擎研究员

游戏开发者大会•中国
GAME DEVELOPERS CONFERENCE CHINA
SHANGHAI INTERNATIONAL CONVENTION CENTER
SHANGHAI, CHINA · OCTOBER 25-27, 2015

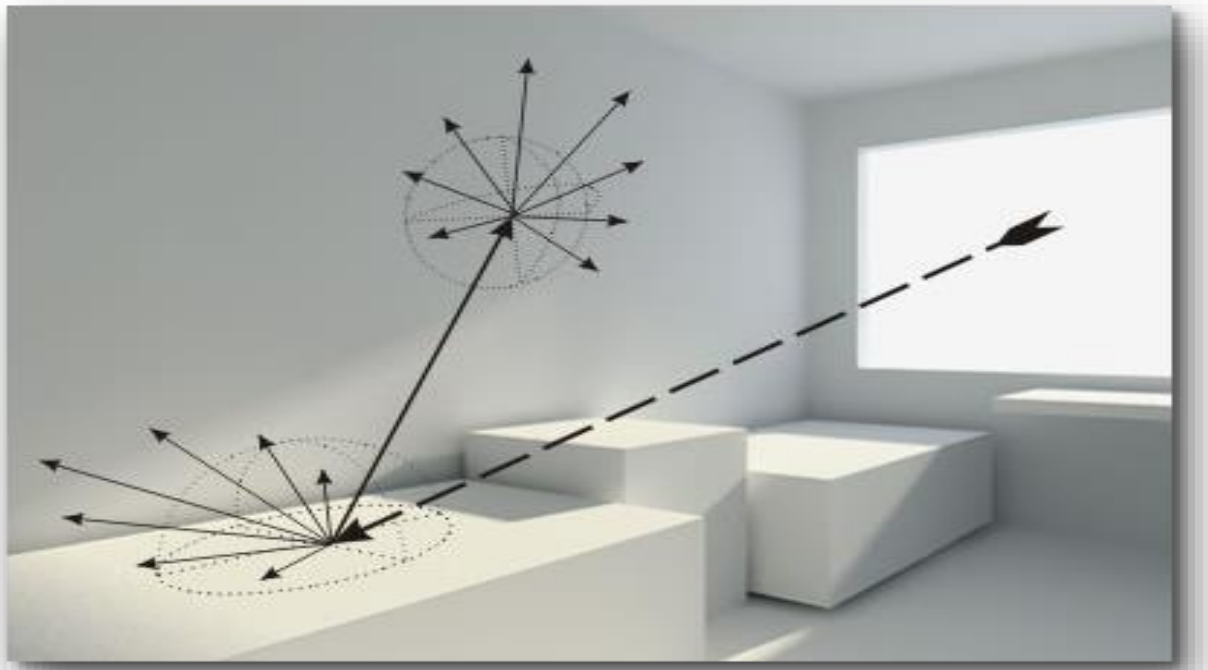
大纲

- 开发背景
- 烘培直接光照
- 烘培间接光照

出发点1——全局光照



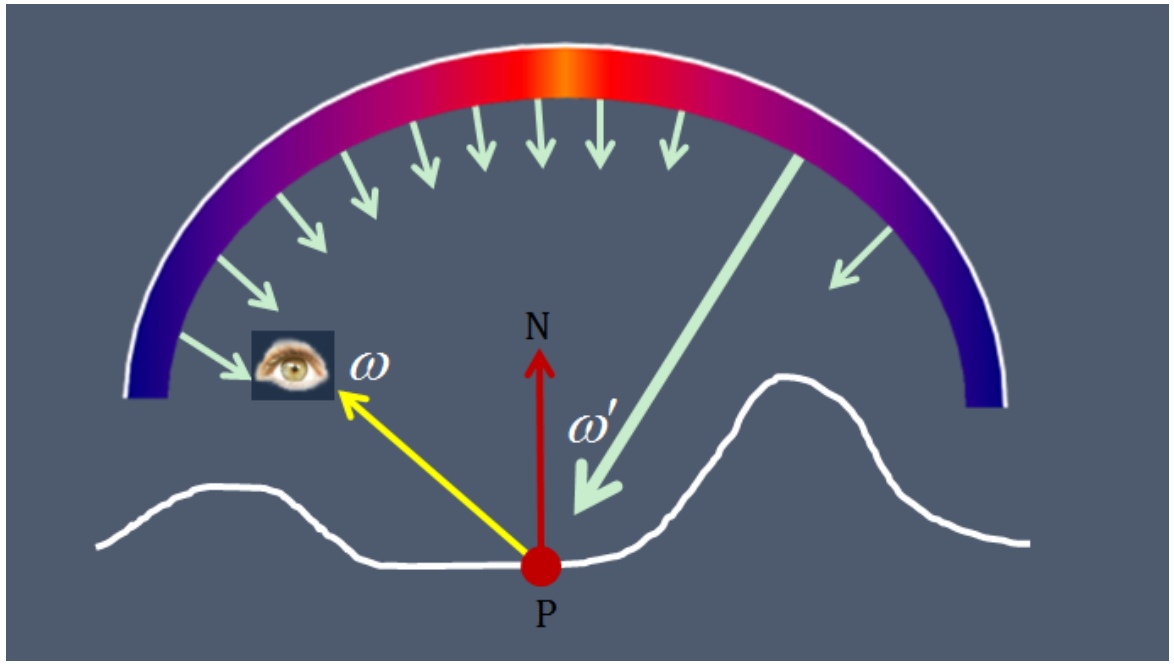
直接光照



直接光照+间接光照

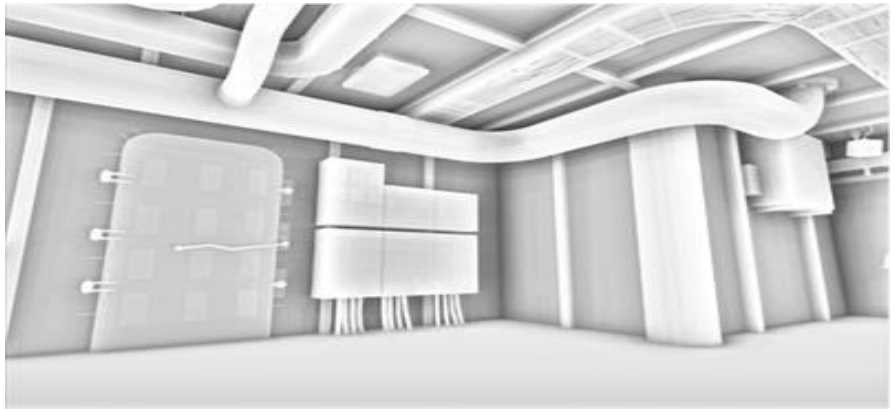
出发点1——全局光照

$$L_o(x,\omega_o) = L_e(x,\omega_o) + \int_\Omega f_r(x,\omega_o,\omega_i) L_i(x,\omega_i) \left|\cos\theta_i\right| d\omega_i$$

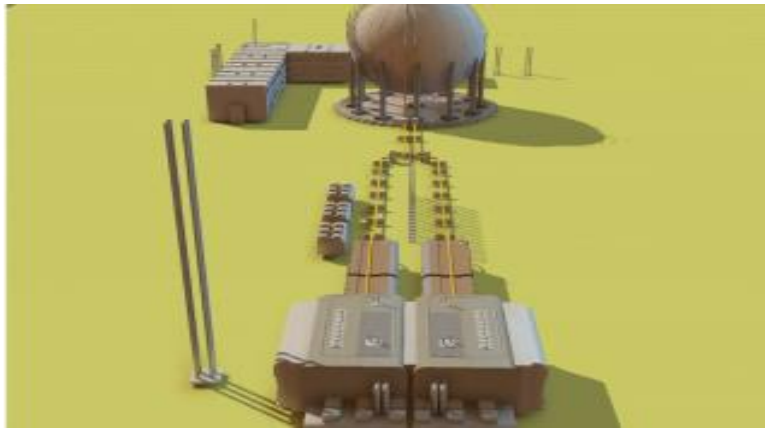


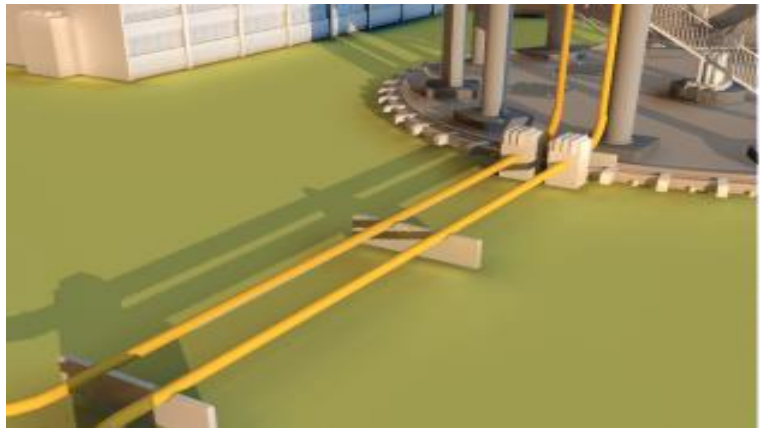
实时全局光照的探索

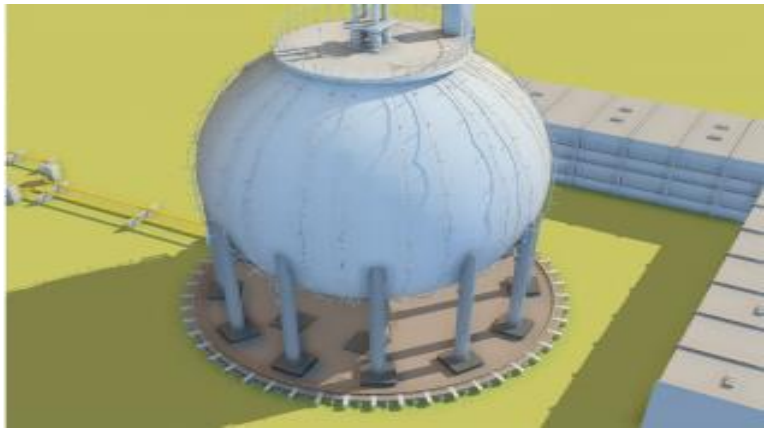
- SSAO [Kajalin09, Bavoil08; Filion08, Mittring07]



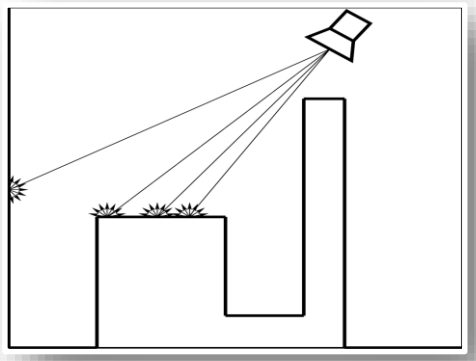
- SSIL [RGS09]



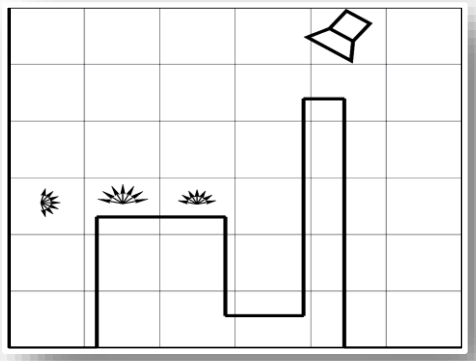


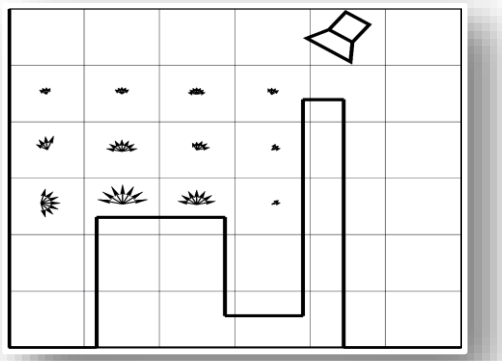


实时全局光照的探索

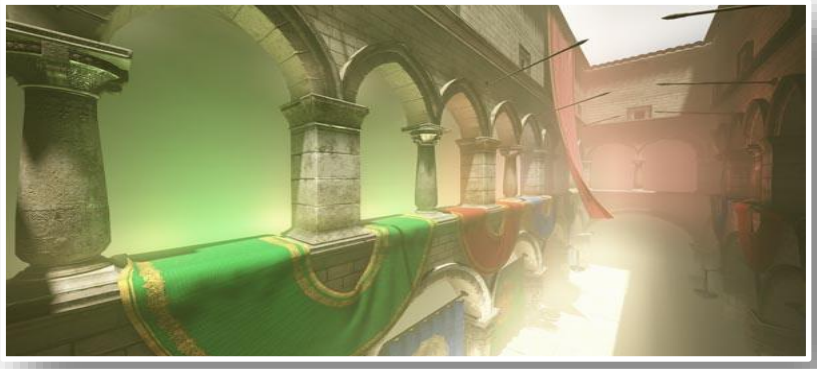


Reflective shadow maps

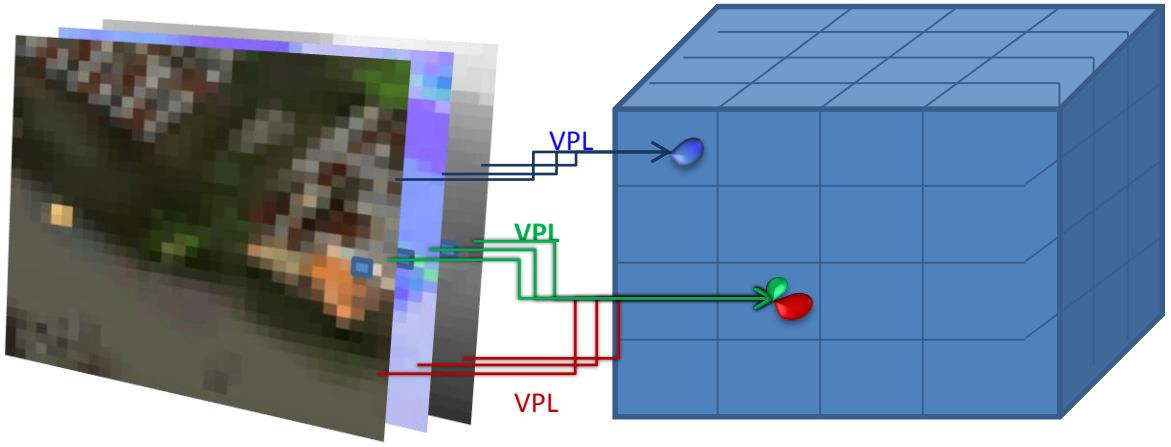




Radiance volume gathering

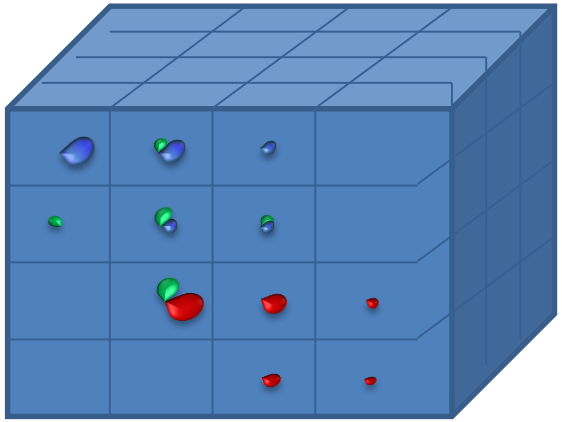


Iterative propagation



A set of regularly sampled VPLs of the scene from light position

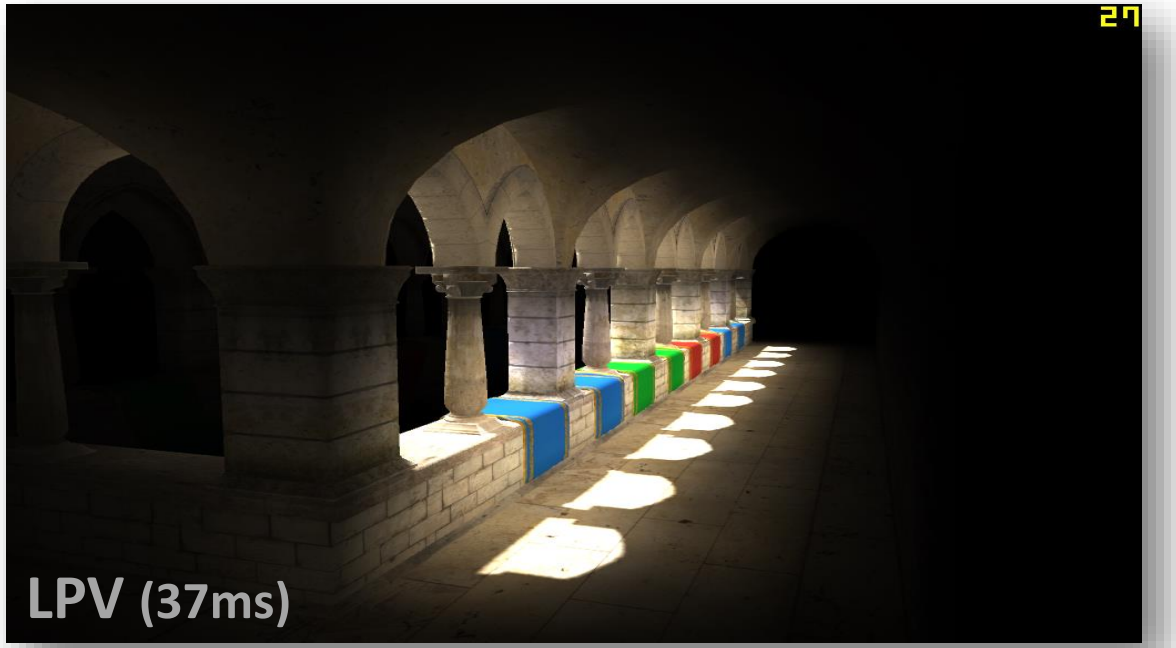
Discretize initial VPL distribution by the regular grid and SH

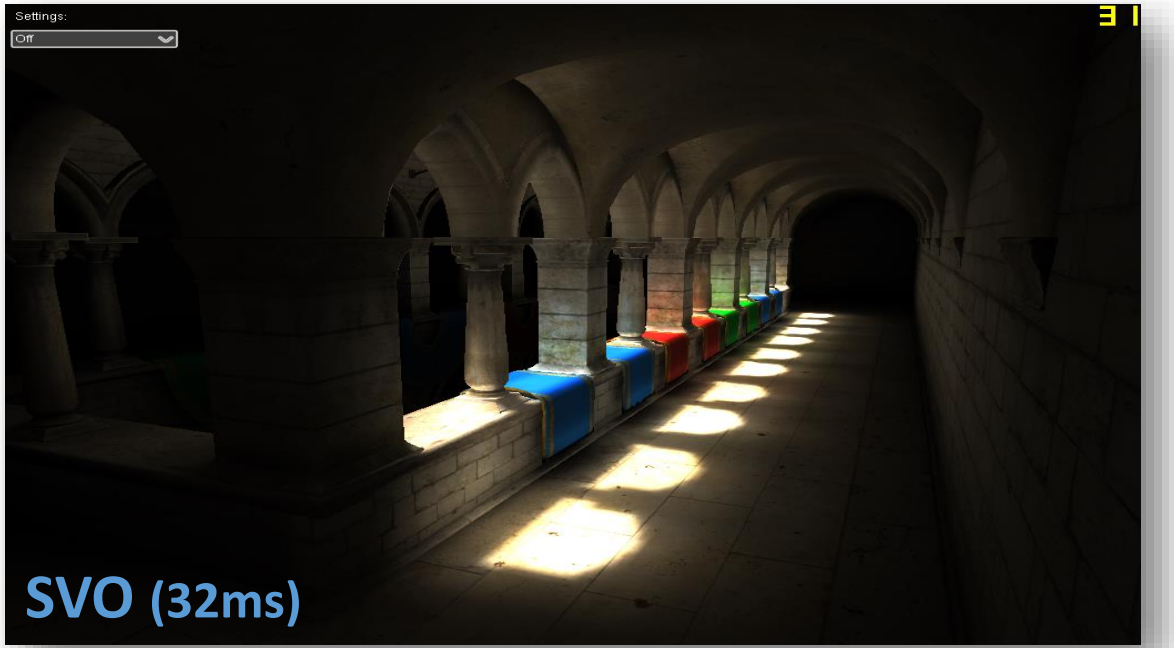


Propagate light iteratively going from one cell to another

实时全局光照的探索

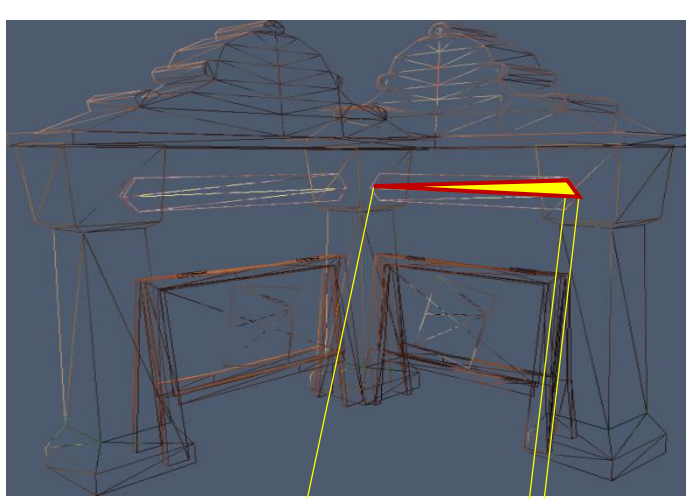
- Sparse Voxel Octree + Cone Trace

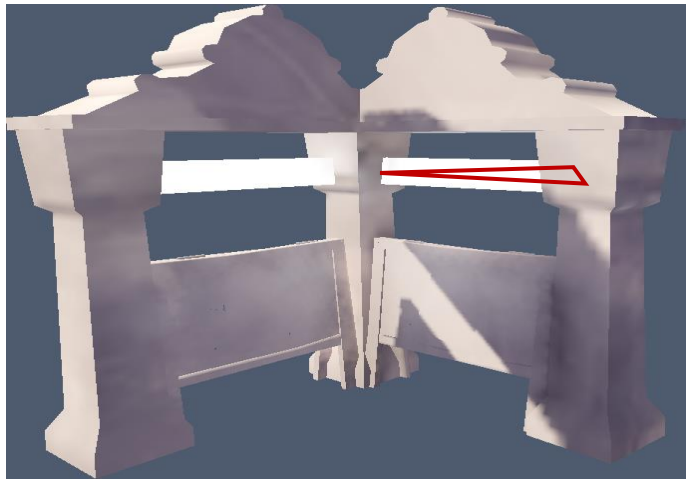




Light Map

- 实时全局光是次次世代的事情
- Light Map——预计算光照





美术制作

烘培软件

实时渲染

出发点2——低端显卡

- 希望游戏在集成显卡上可以跑到30帧
  - ~~Deferred Lighting~~
  - ~~Forward Lighting~~
  - Light Map + Forward

- 3D手游，light map是唯一的选择
  - Power VR G6450, 249.6 GFLOPS

为什么要自己研发

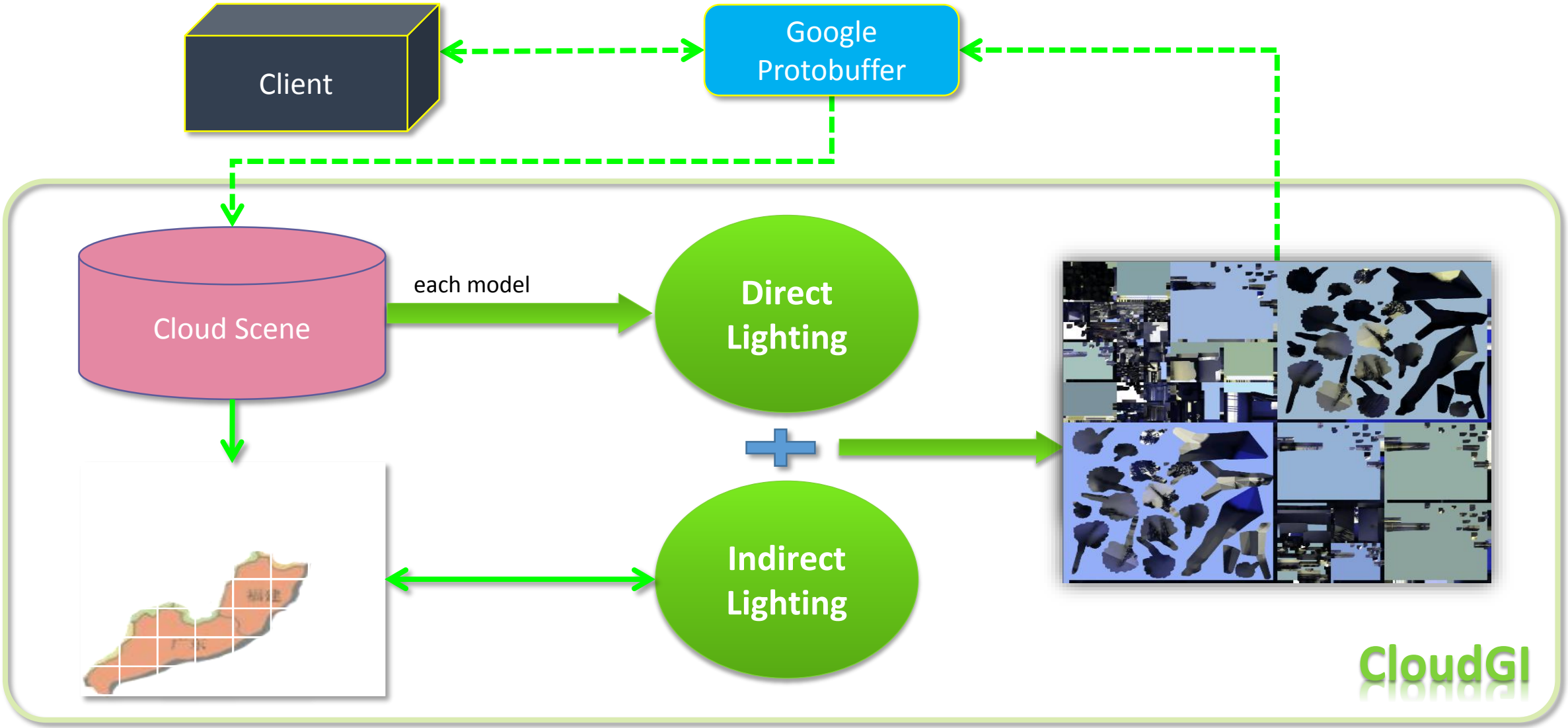
- 目前市场上的商业烘培软件

烘培巨慢

Patch上G

材质固定

CloudGI全景图


Client
Google Protobuffer
Cloud Scene
each model
Direct Lighting
Indirect Lighting
CloudGI

视频演示1

烘培直接光照

Deferred Shading

太阳光阴影

- 针对camera frustum的策略不再有效
- 针对perspective的策略也不再有效

太阳光阴影

- 每个模型一张ShadowMap
- 一张巨大的ShadowMap

5000 X 4000

- 10000X10000 = 200M 显存
- 内存做Cache + Shadow Mask

LightMap裂缝

LightMap裂缝

- 在世界空间连续，在uv空间不连续

LightMap裂缝

- 在世界空间连续，在uv空间不连续

  - Gutter，用最近的烘培点颜色填充背景

LightMap裂缝

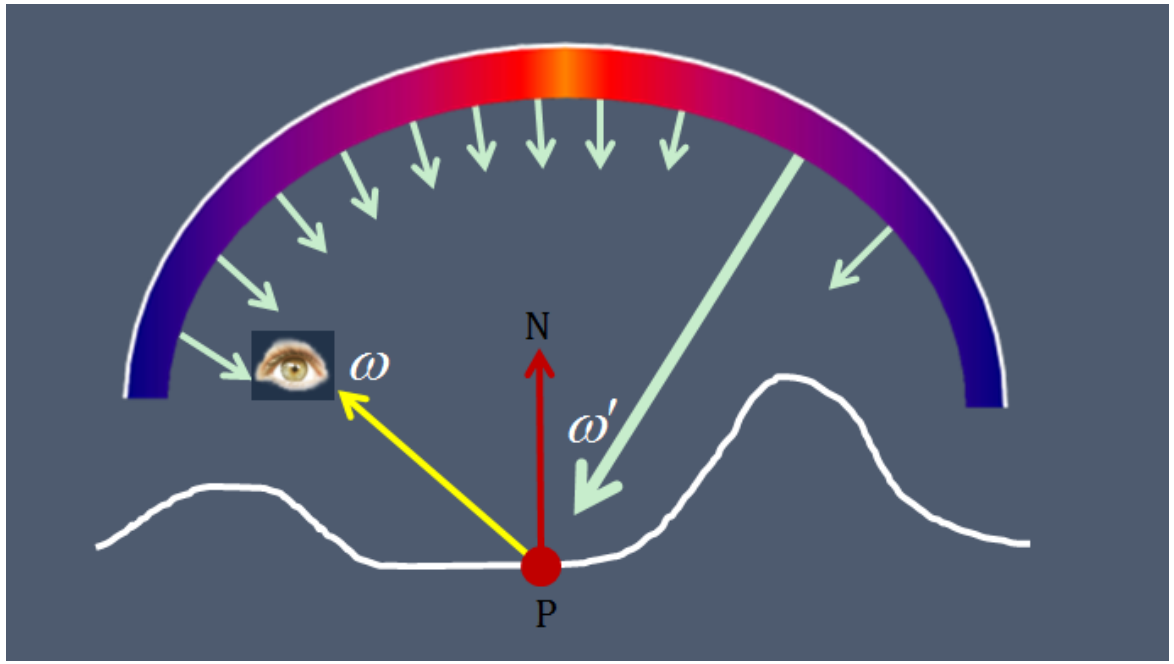
LightMap裂缝

LightMap裂缝

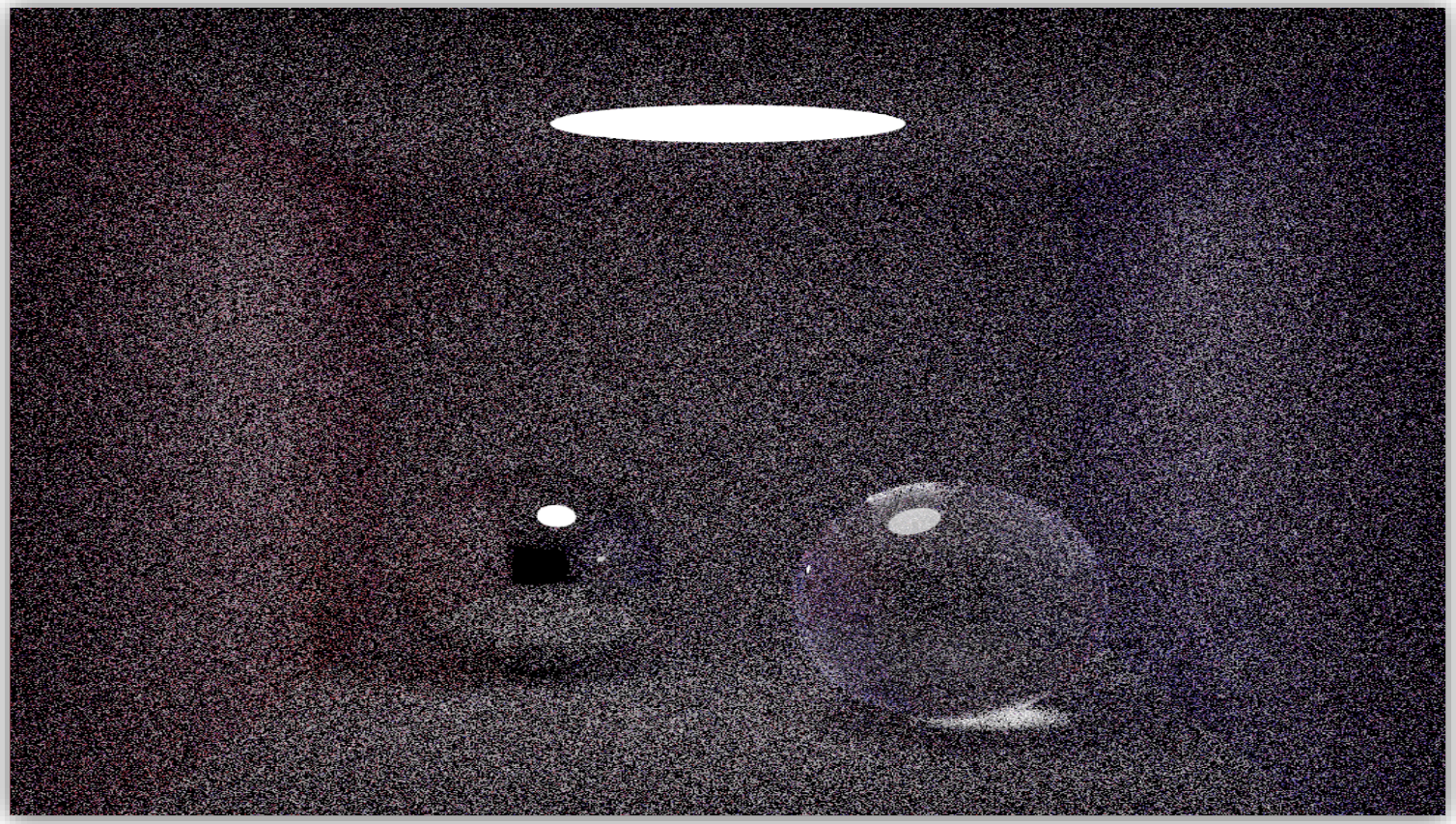
- 细小的边在lightmap上小于一个像素
- Super Sample->Gutter->Down Sample

Point Cloud based GI

$$L_o(x, \omega_o) = L_e(x, \omega_o) + \int_{\Omega} f_r(x, \omega_o, \omega_i) L_i(x, \omega_i) \, |\cos \theta_i| \, d\omega_i$$



Point Cloud based GI



Raytrace

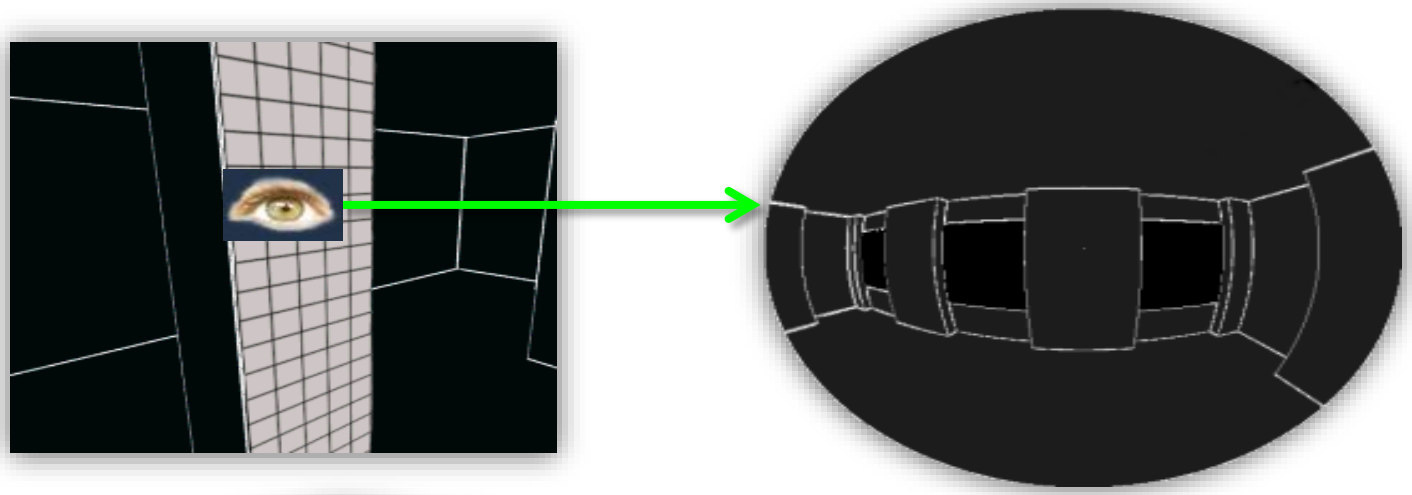
$O(n*m^k) * O(s)$

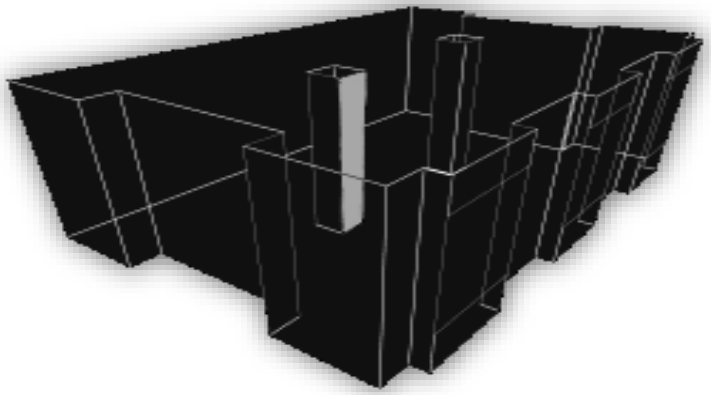
- 什么效果都能做
- 指数增长
- **Noise**

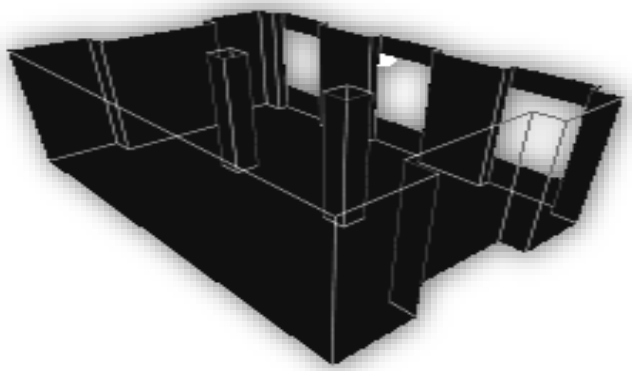
Point Cloud based GI

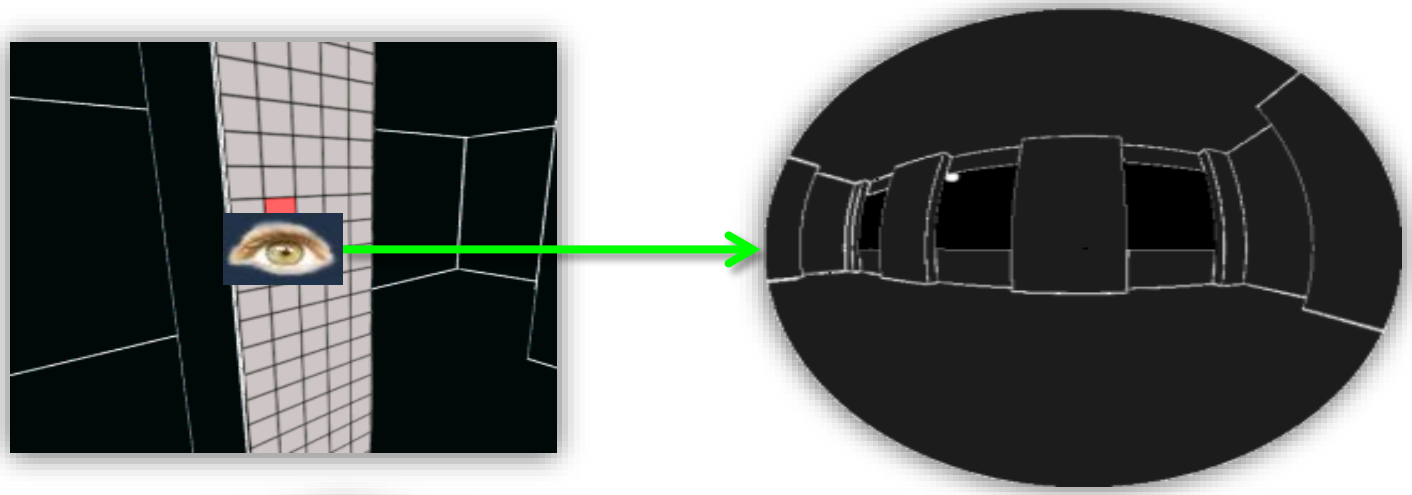
- Radiosity（辐射度）

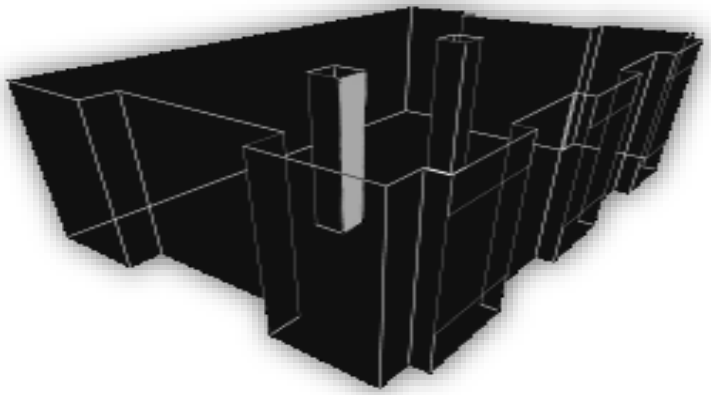
- 不区分灯光物体
- 每个surfel被所有其他surfel照亮

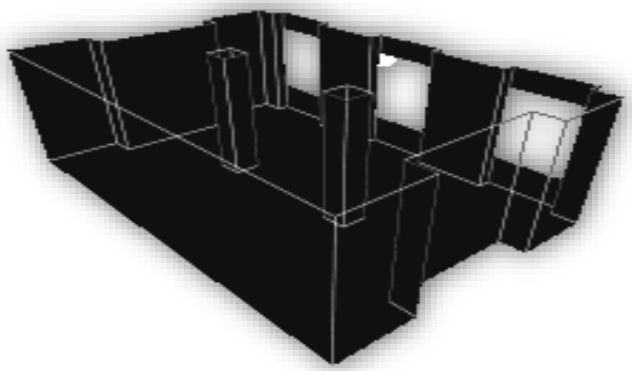


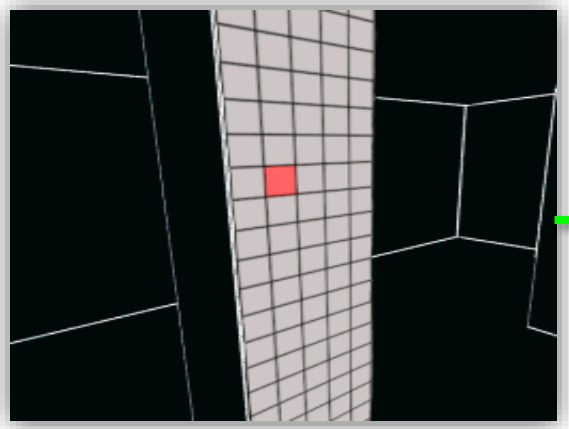


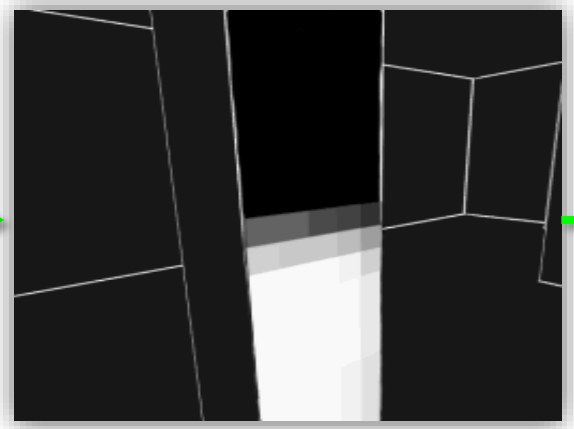


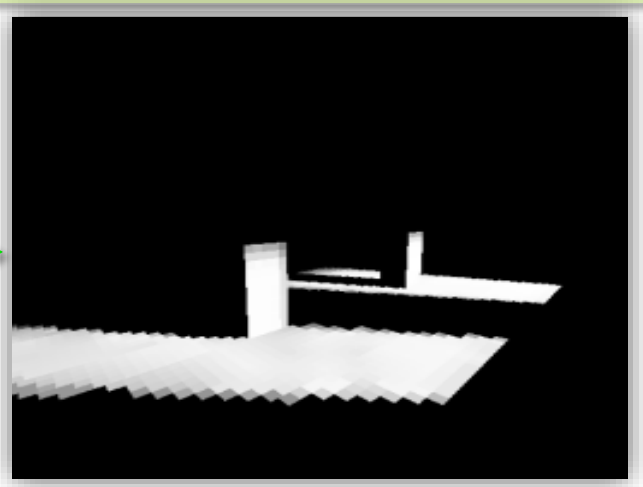


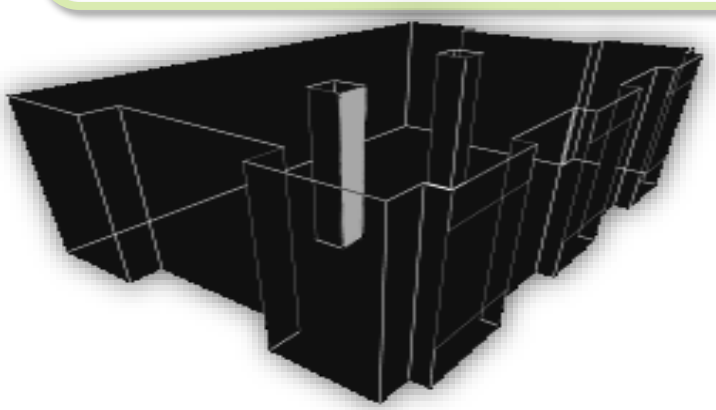


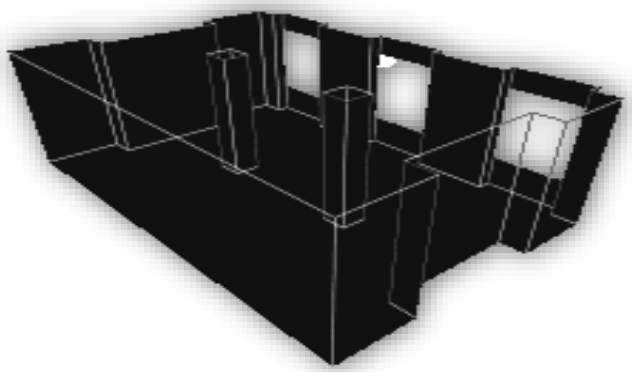


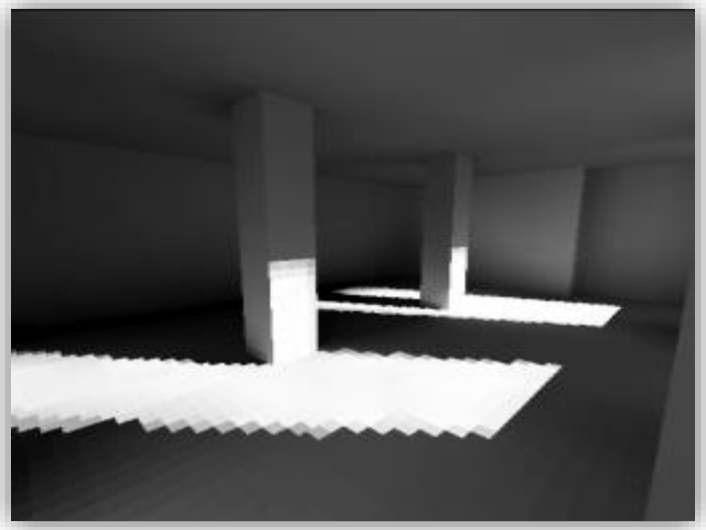




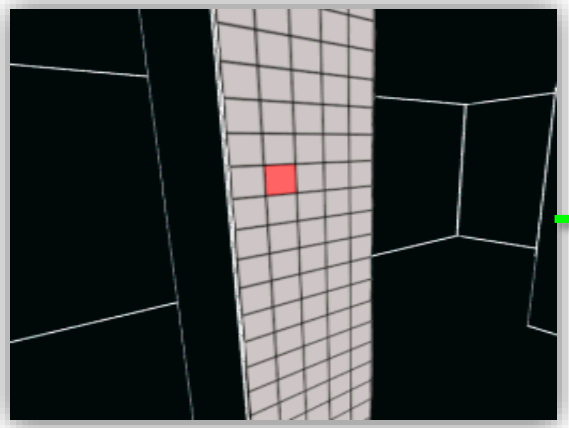


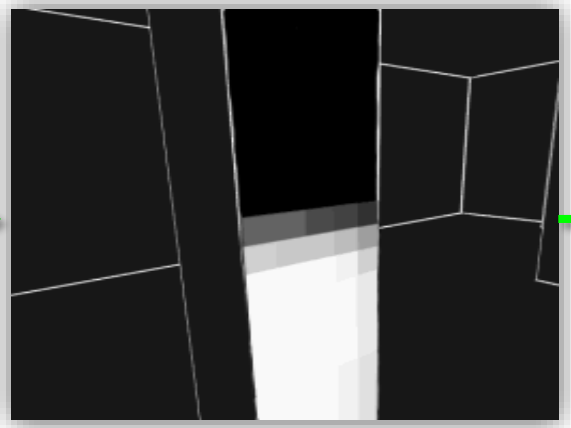


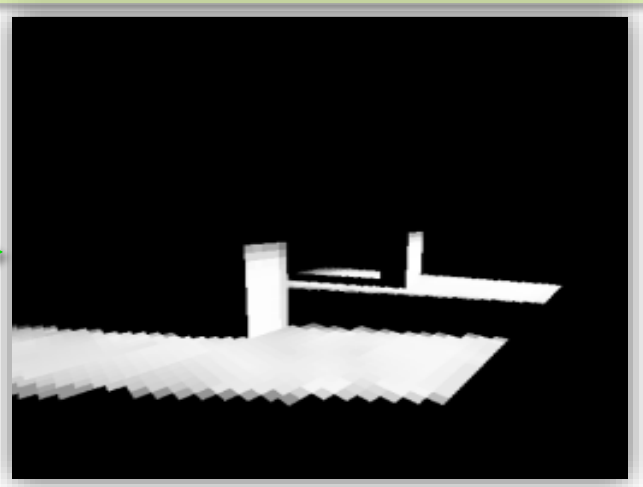


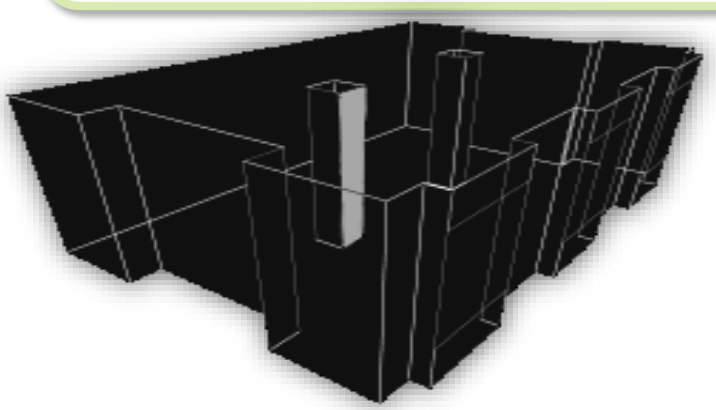


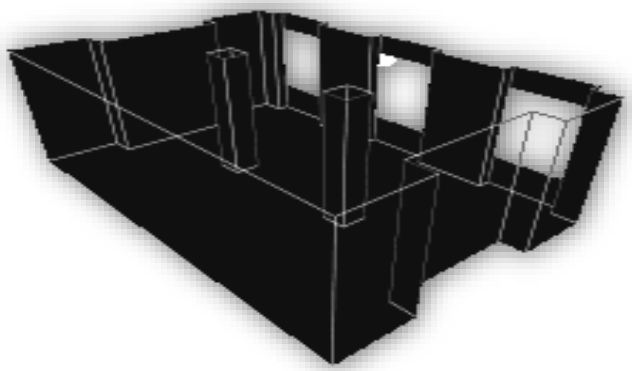
2 bounce

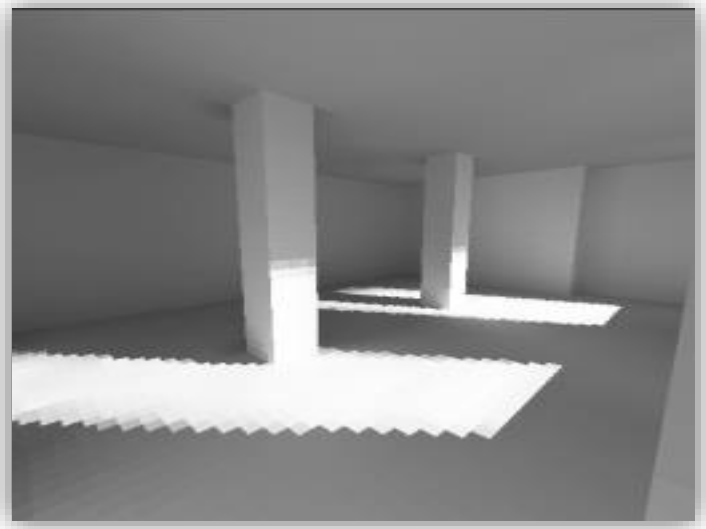








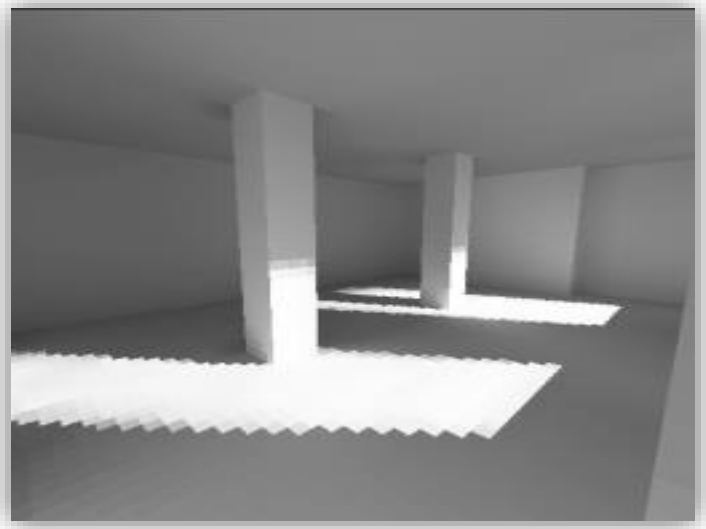




16 bounce

Point Cloud based GI

- Radiosity（辐射度）



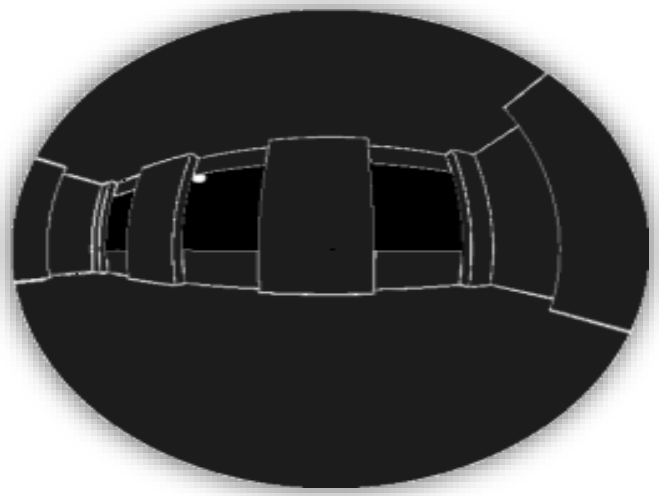
O(n*n*k)

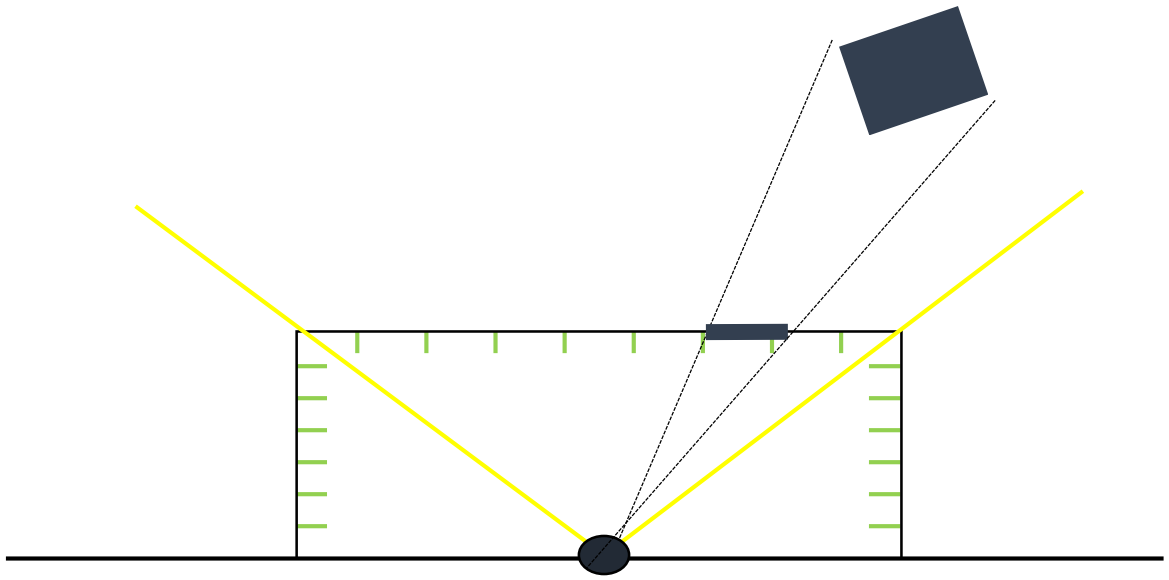
- 线性增长
- 没有noise

- 只能做漫反射的GI效果

Point Cloud based GI

- Radiosity（辐射度）
  - 如何计算surfel可见图

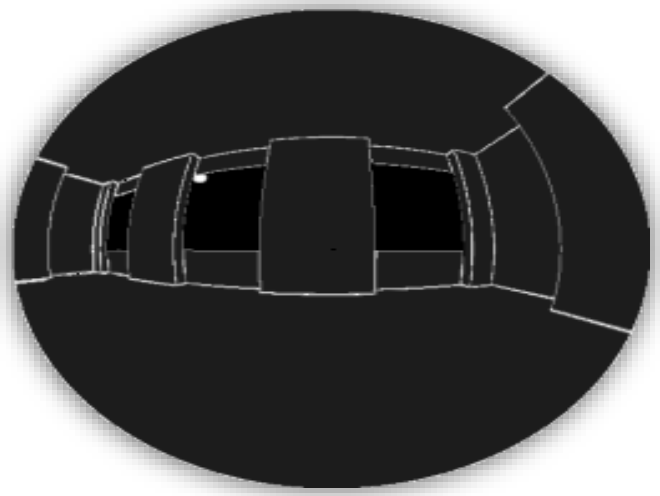


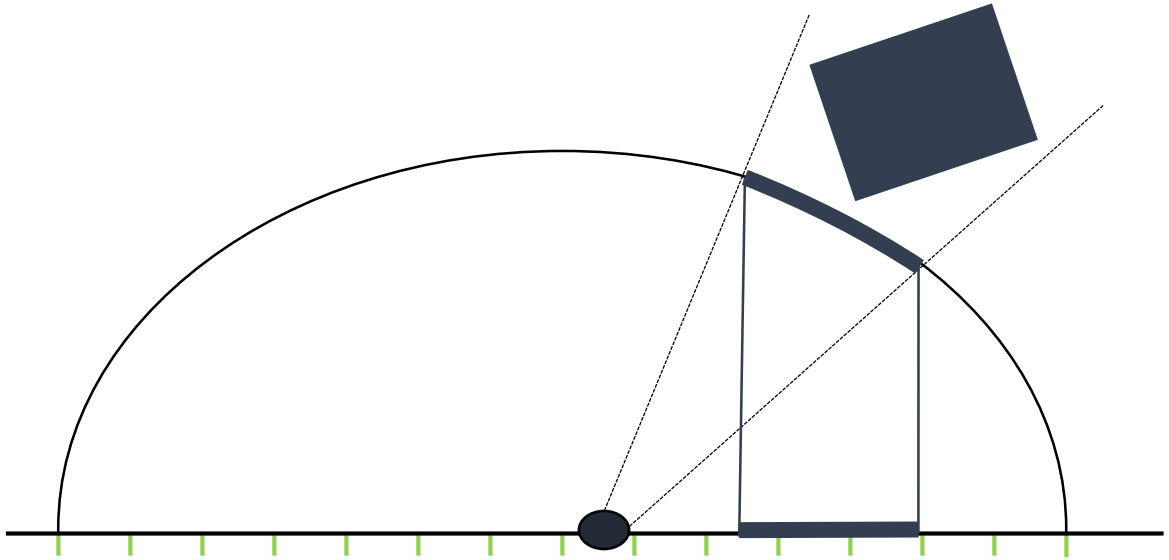


Hemicube, raster 5次

Point Cloud based GI

- Radiosity（辐射度）
  - 如何计算surfel可见图





Hemispherical, raster 1次

Point Cloud based GI

- 点云生成
- 构建Octree
- 计算间接光照

Point Cloud Generation

- 如何生成点云？
  - 跟直接光照一样

```
struct Surfel {
    float3 position;
    float3 normal;
    float3 radiance;
    float area;
}
```

DX11 Unordered Access View + Atomic Add
CUDA Memory

Deferred Shading

Point Cloud Generation

- 如何计算Area?

```
struct Surfel {
    float3 position;
    float3 normal;
    float3 radiance;
    float area;
}
```

Geometry Shader

$$\text{Area} = \frac{\text{Triangle Area}}{\left(\text{UV Area} / \text{UV per pixel}\right)}$$

视频演示2

Octree Building

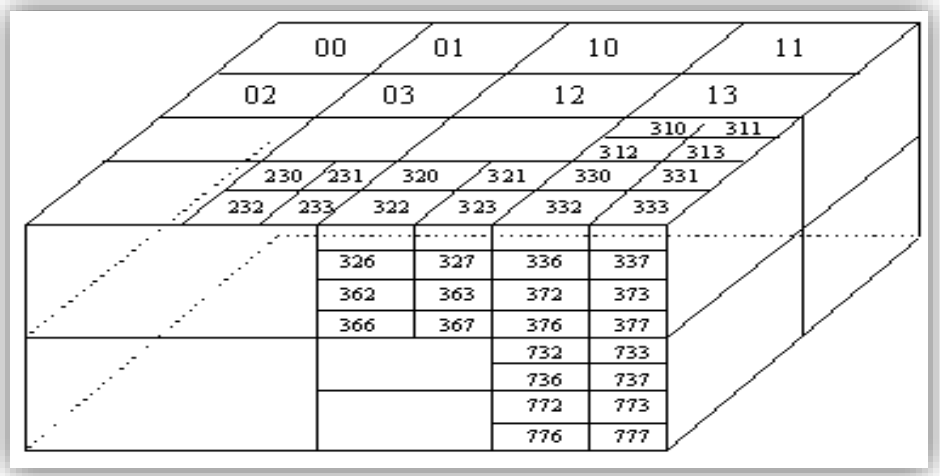
- CPU上，自上而下
  - 并发不友好
  - 动态内存分配
  - 需要同步，或者原子操作

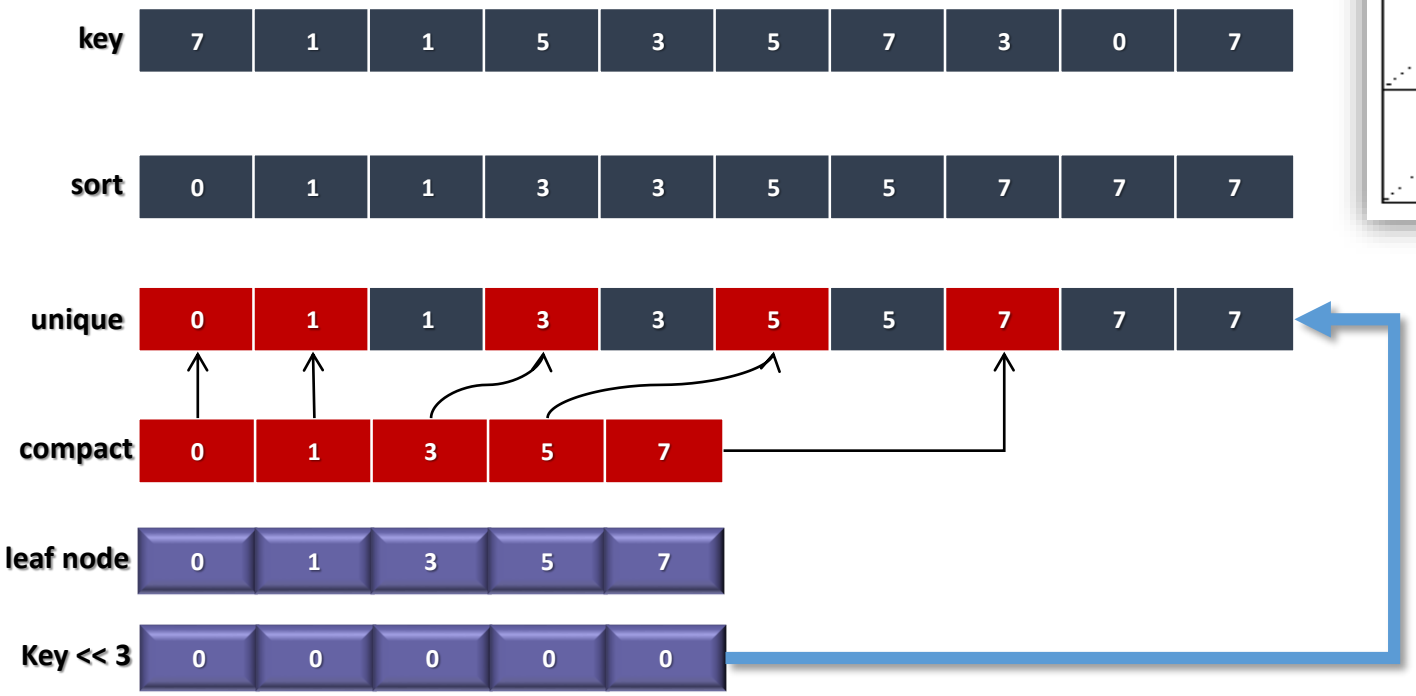
Octree Building

- GPU上，自下而上
  - 天然并发
  - 无需同步
  - 显存不浪费

Octree Building

- 创建叶节点
  - 3位一层编码，32位表示10层



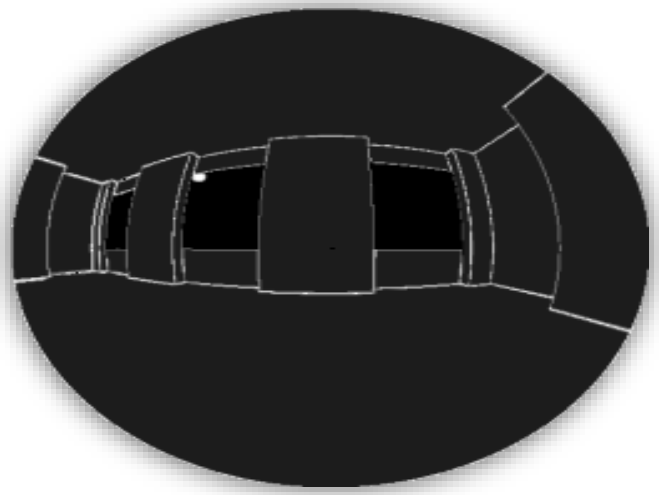


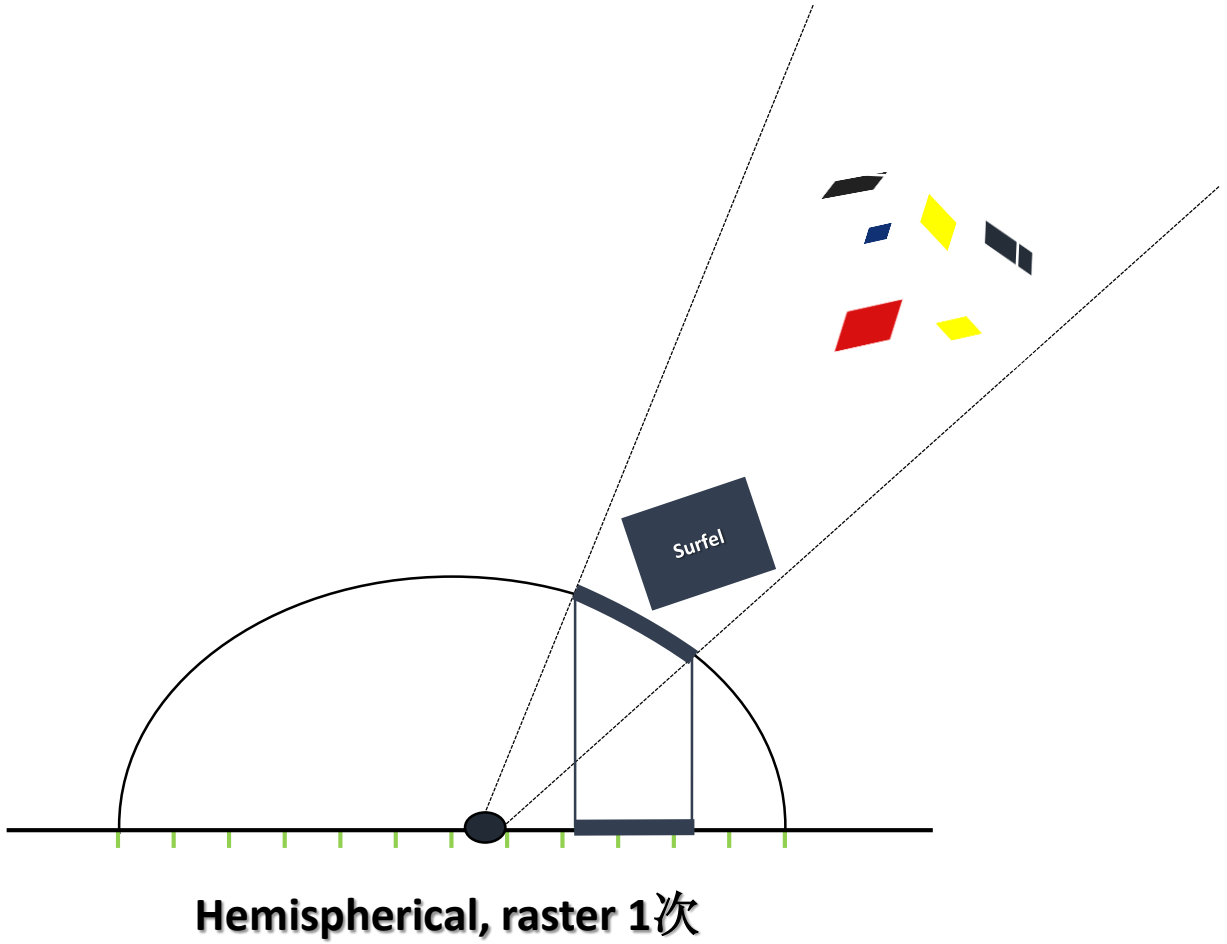
Thrust——CUDA based STL

视频演示3,Octree

Octree Building

- 为什么要建立Octree
  - 0(n*n*k)->0(n*logn*k)
  - 这个Node的面积跟颜色是什么呢？





Hemispherical, raster 1次

Octree Building

- Spherical Harmonics
  - 一组定义在球面上的正交基函数
  - $Y_l^m(\theta,\varphi) = \cdots; l \in N, -l \le m \le l$

$$f(\theta,\varphi) \approx \sum_{l=0}^{n} \sum_{m=-l}^{l} f_l^m Y_l^m(\theta,\varphi)$$

$$f_l^m = \int f(\theta,\varphi)\, Y_l^m(\theta,\varphi) d\omega$$

Octree Building

- Spherical Harmonics
  - 把一个函数$f(\theta,\varphi)$变成几个参数
  - 两个函数相加变成几个参数相加

$$f(\theta,\varphi) \approx \sum_{l=0}^{n}\sum_{m=-l}^{l} f_l^m Y_l^m(\theta,\varphi)$$

$$f_l^m = \int f(\theta,\varphi)\, Y_l^m(\theta,\varphi) d\omega$$

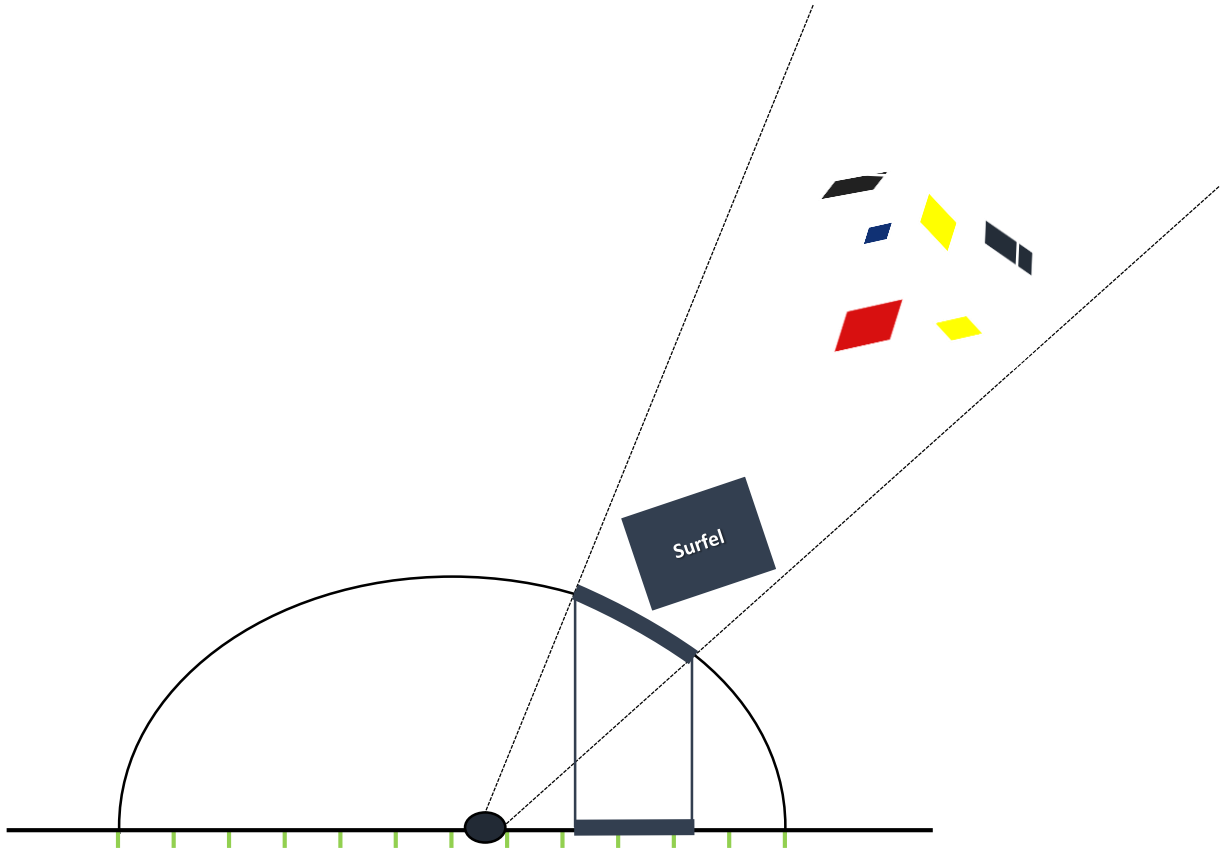
$$a_l^m = \int A_i |dot(\vec{d},\vec{n})|\, Y_l^m(\theta,\varphi) d\omega$$

$$p_l^m = \int B_i A_i dot(\vec{d},\vec{n})_+\, Y_l^m(\theta,\varphi) d\omega$$

Indirect Lighting

- 计算间接光照
  - 遍历Octree，得到id map
  - calcRadianceFromID
  - 平均 * PI

$$f(\theta,\varphi) \approx \sum_{l=0}^{n} \sum_{m=-l}^{l} f_l^m Y_l^m(\theta,\varphi)$$



Hemispherical, raster 1次

算法优化

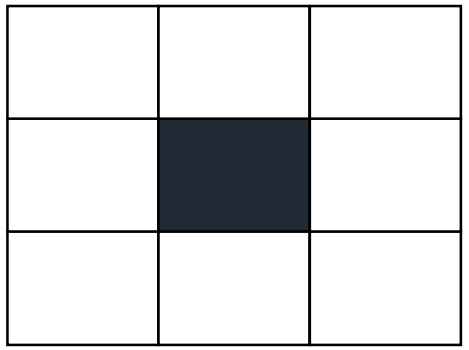
- 烘培一个小场景需要半个小时!!!
  - 烘培精度 == 间接光照精度
  - 1张超采样的lightmap就有1600W的点云
- 分离烘培精度、间接光照精度
  - K-near 插值，考虑normal，position
  - 预设阀值

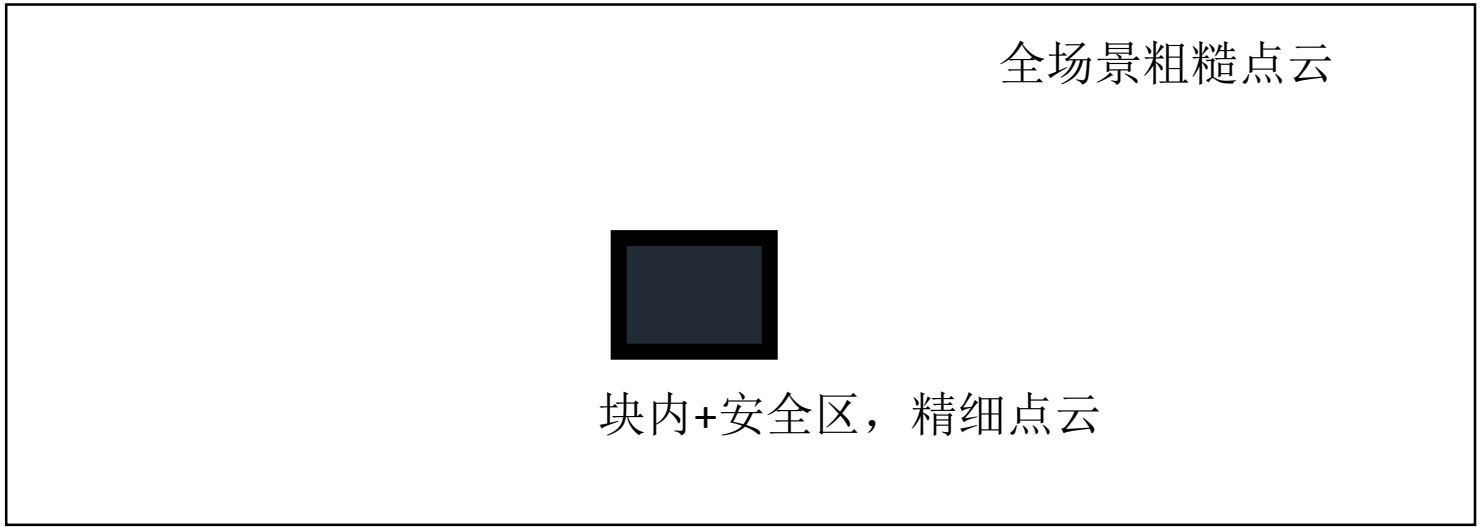
视频演示4

分块烘培

- 1000W点云=400M
- 1000m×1000m的正方形
  - 4G显存
- ◉ 分块烘培是必须的

```
struct Surfel {
    float3 position;
    float3 normal;
    float3 radiance;
    float area;
}
```

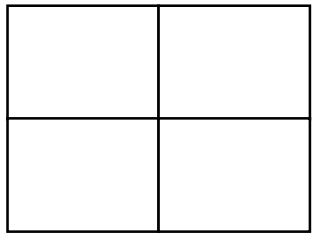




半透明材质

- 1行代码搞定
  - surfel.area *= alpha;

- ◉ 半透明阴影
  - 传统做法是两张shadow map
  - 以alpha的概率写入深度图



半透明材质

自定义材质

- Radiosity的计算时间跟材质复杂度无关

```
PS_OUTPUT_GBUFFERGEN PsGBuffergen( VS_OUTPUT Input )
{
    float3 worldPos,worldNormal,vBinormal;
    GetWorldProperties(worldPos, worldNormal,vBinormal, Input );

    //diy codes

    return OutputPsGBufferResult( worldPos, worldNormal, diffuseColor.rgb,
        emissiveColor, vBinormal );
}
```

总结

- 实时GI是次次世代的技术

- CloudGI比cpu raytrace快1000倍
  - 贴合网游的实用功能（实时预览、minpatch、自定义材质）

- 用Octree来加速的Radiosity算法
  - K-near插值加速

- 分块烘培解决显存问题

References

- Per H. Christensen, Point-Based Approximate Color Bleeding, Pixar Animation Studios.
- KUN ZHOU, MINMIN, XIN , and BAININ 2010. Data-Parallel Octrees for Surface Reconstruction.

Thank you