How to create a second level global light baking software

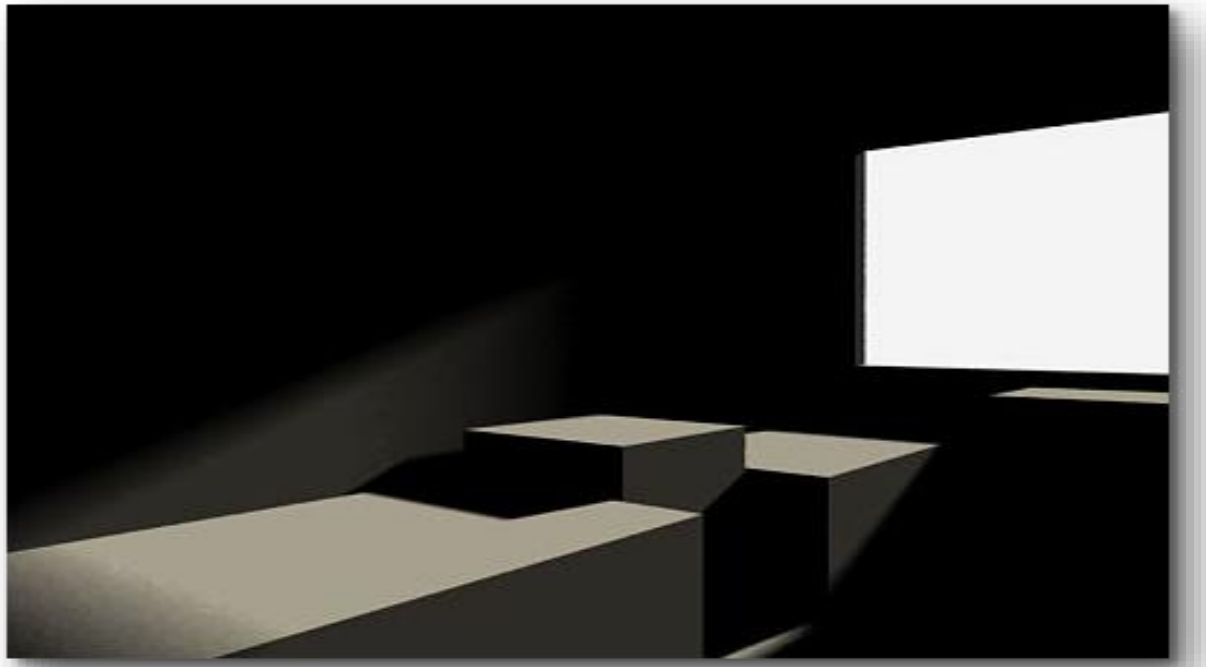
Li Wenyao
Game engine researcher, Netease Game

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

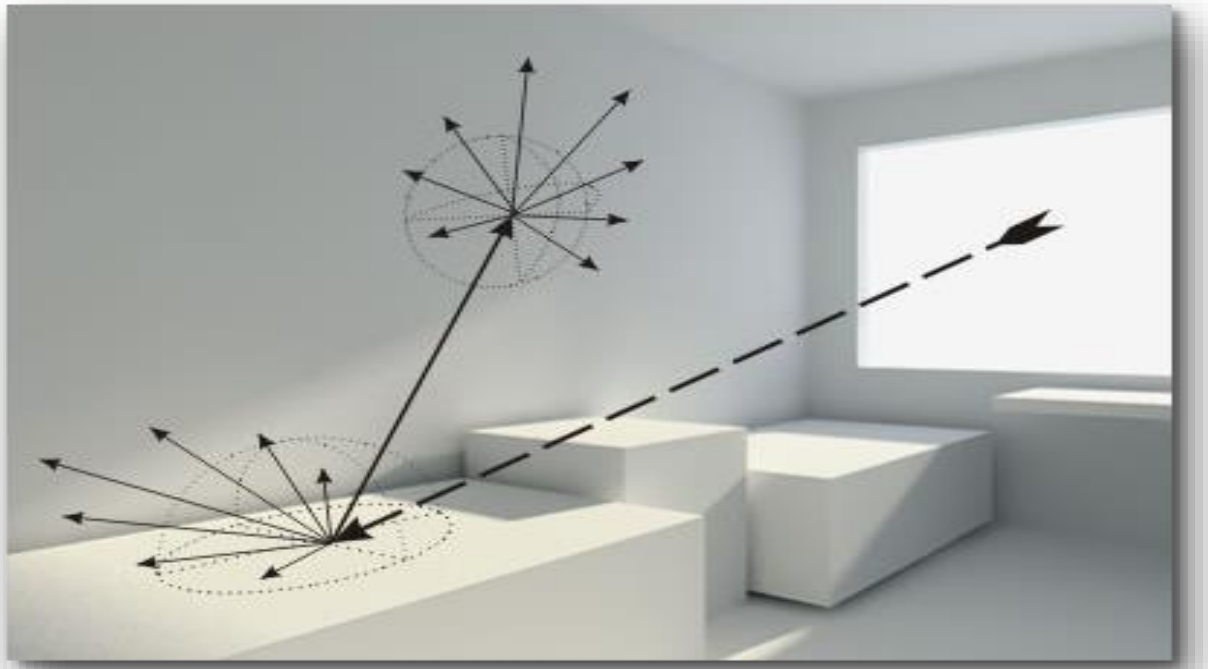
Outline

- Development background
- Baking direct lighting
- Baking indirect lighting

Starting point 1——Global lighting



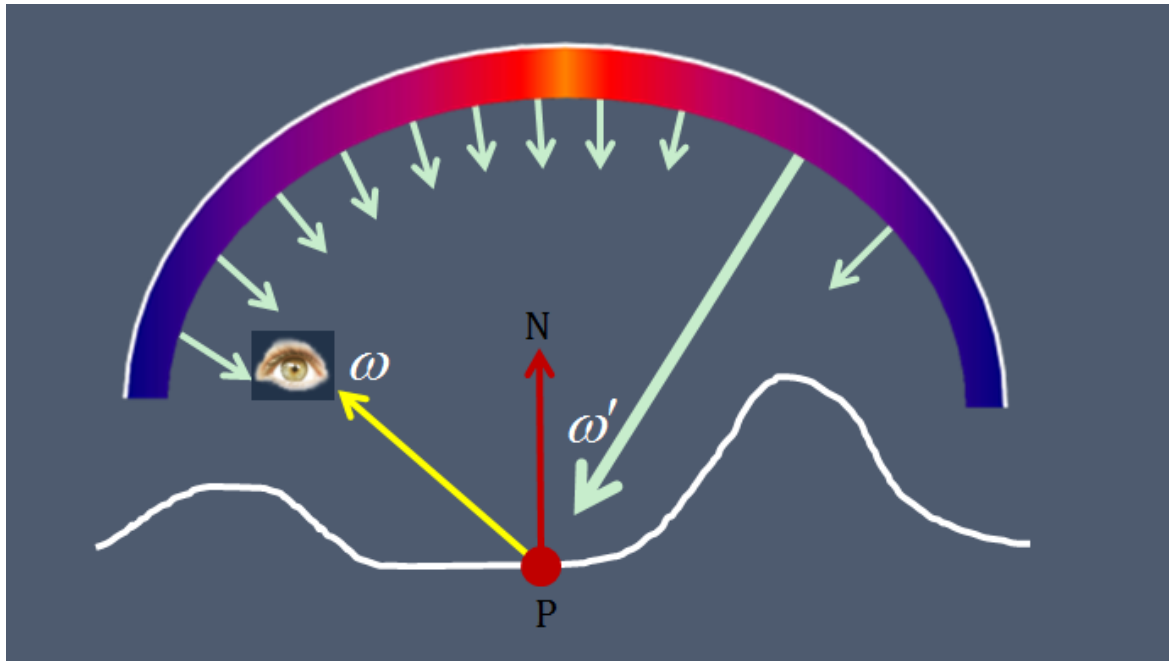
Direct lighting



Direct lighting
+ Indirect lighting

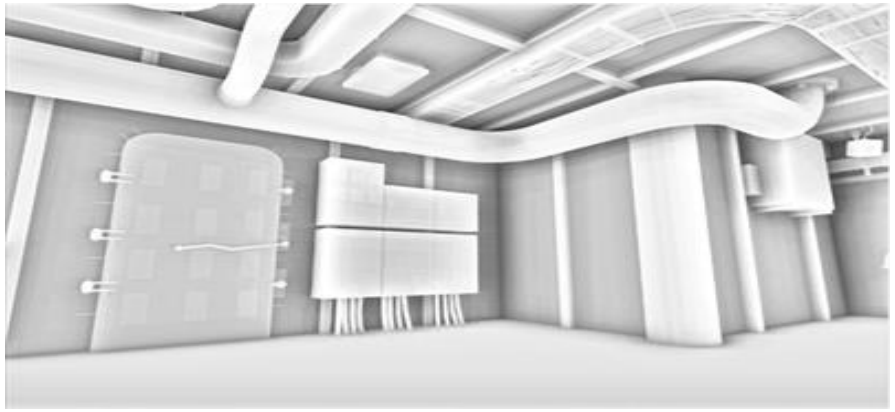
Starting point 1——Global lighting

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

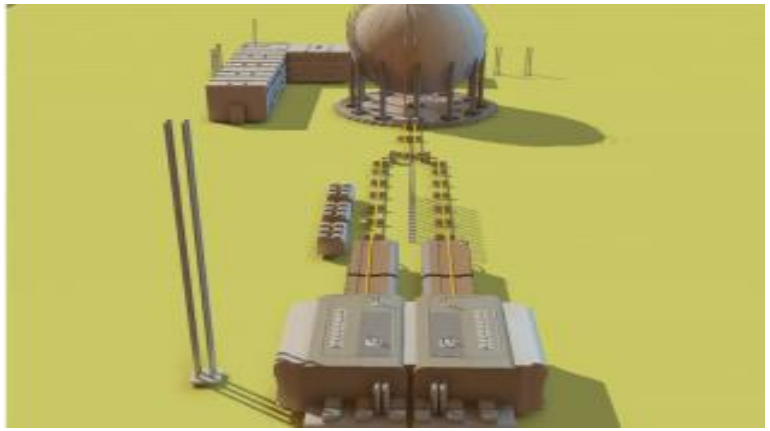


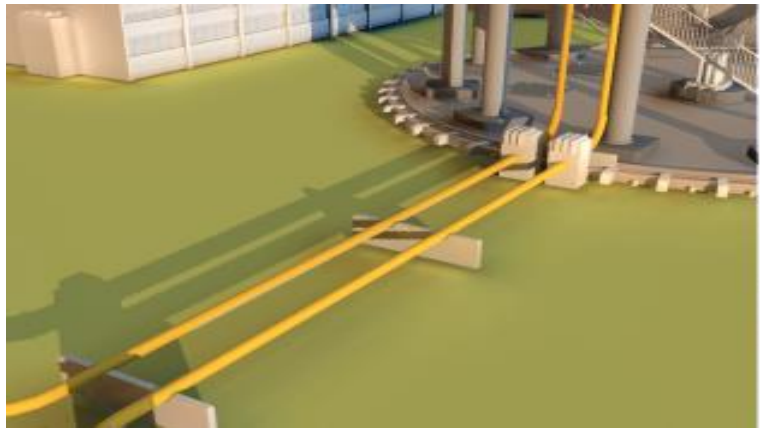
Exploration of real-time global lighting

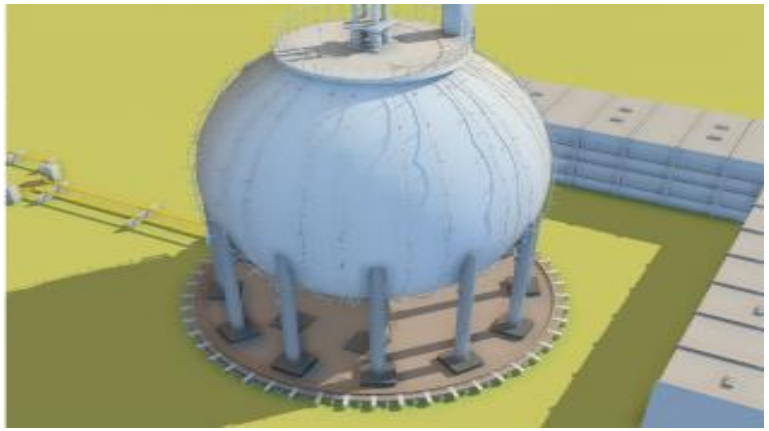
- SSAO [Kajalin09, Bavoil08; Filion08, Mittring07]



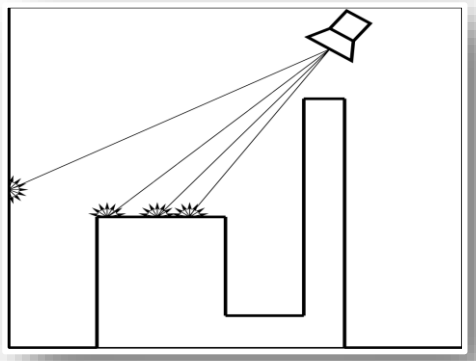
- SSIL [RGS09]



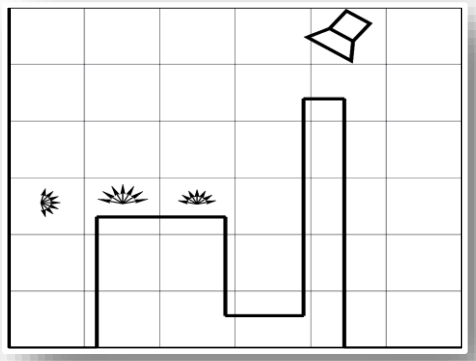


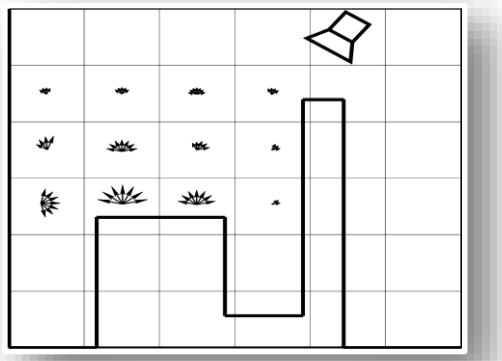


Exploration of real-time global lighting

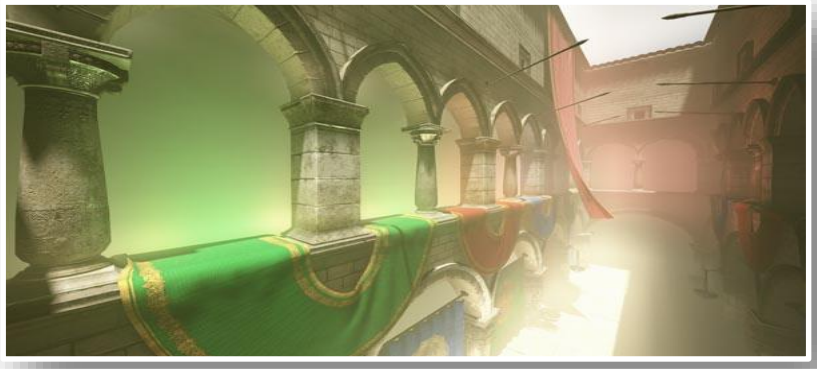


Reflective shadow maps

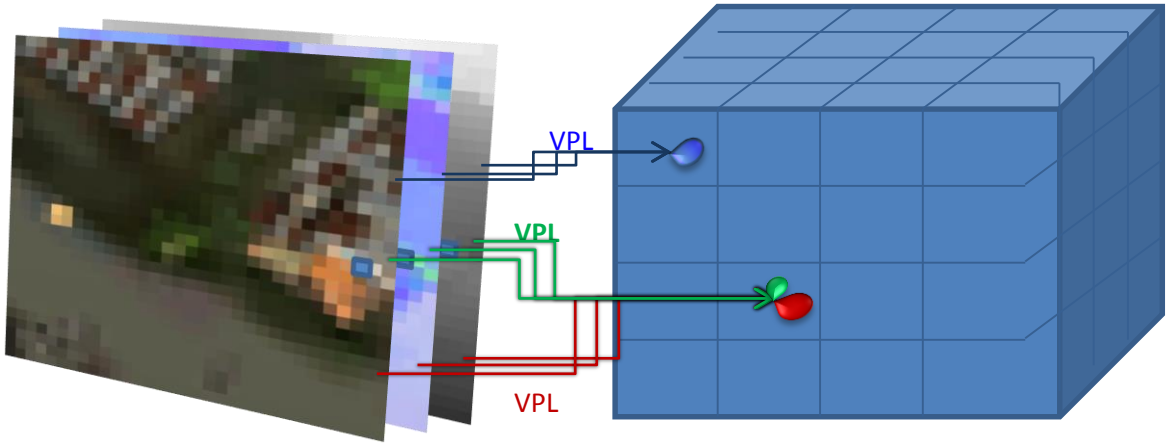




Radiance volume gathering

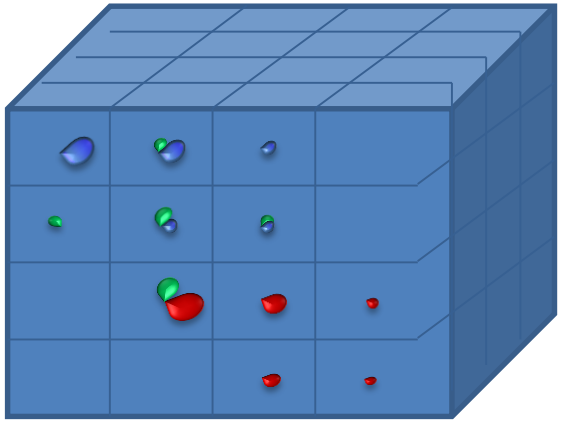


Iterative propagation



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

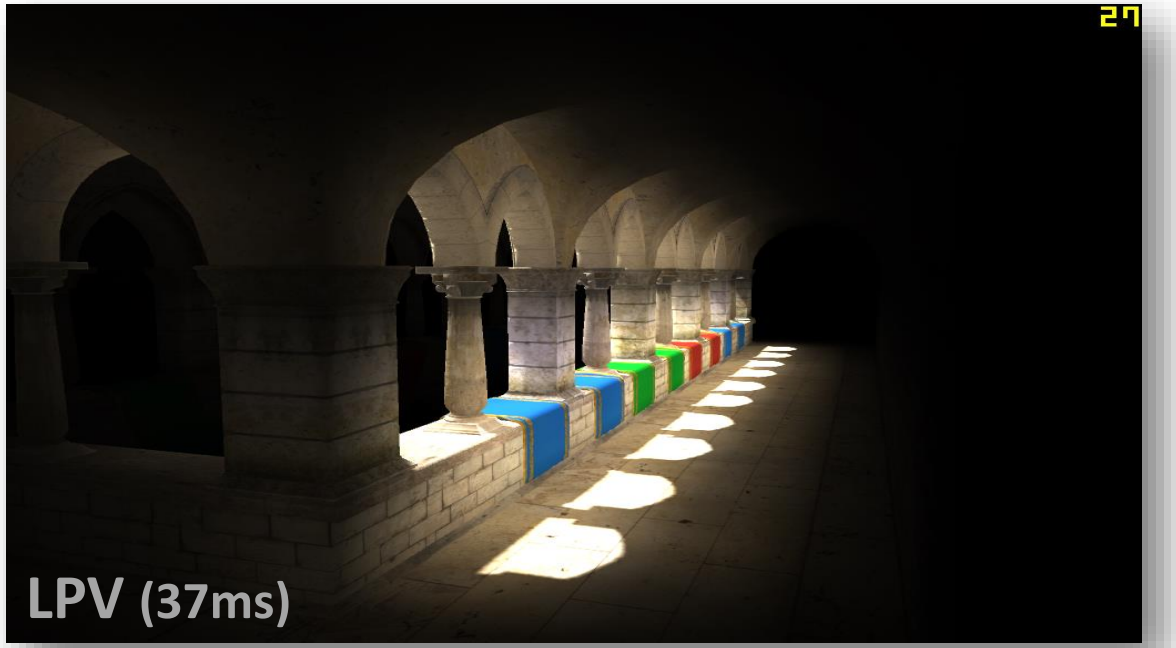
Discretize initial VPL distribution by the regular grid and SH

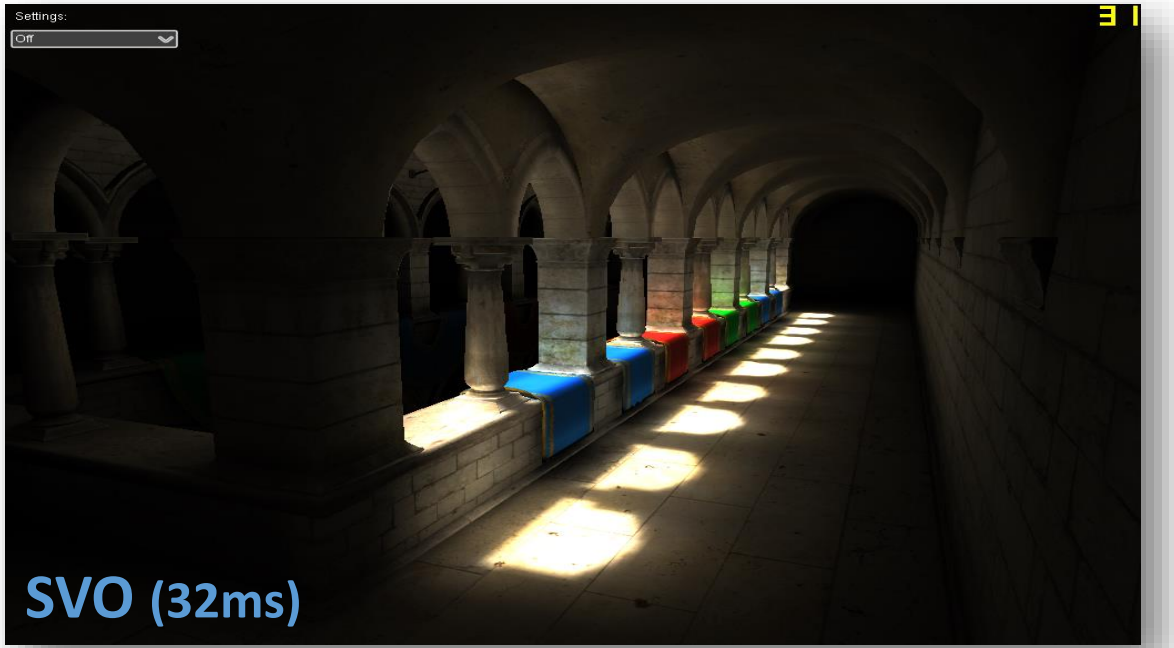


Propagate light iteratively going from one cell to another

Exploration of real-time global lighting

- Sparse Voxel Octree + Cone Trace

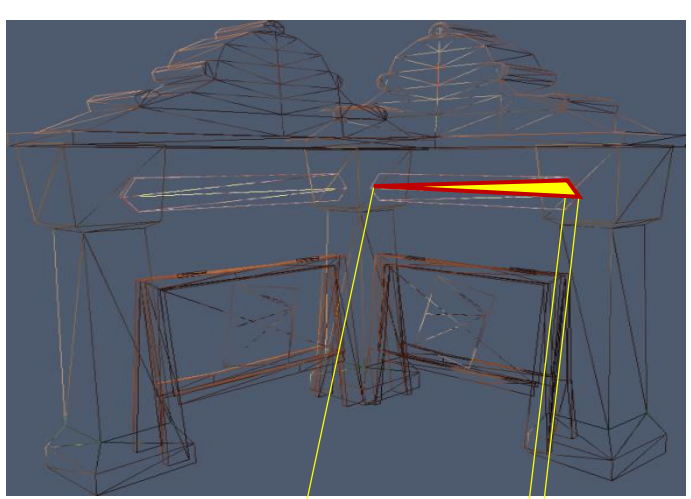


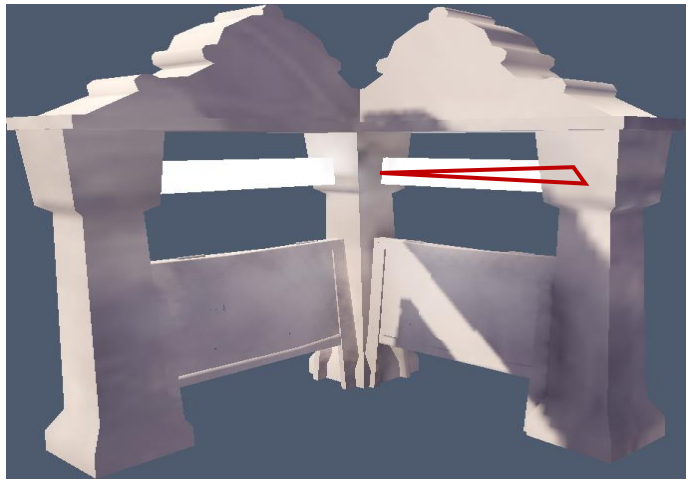


Light Map

- Real-time global lighting is a next next generation technology

- Light Map——Precomputed lighting





Art production

Baking software

Real-time rendering

Starting point 2——low-end graphics

- We hope the frame rate of games running with integrated graphics can reach 30 frames
  - ~~Deferred Lighting~~
  - ~~Forward Lighting~~
  - **Light Map + Forward**

- For 3D mobile games, light map is the only option
  - Power VR G6450, 249.6 GFLOPS

Why developed by ourselves?

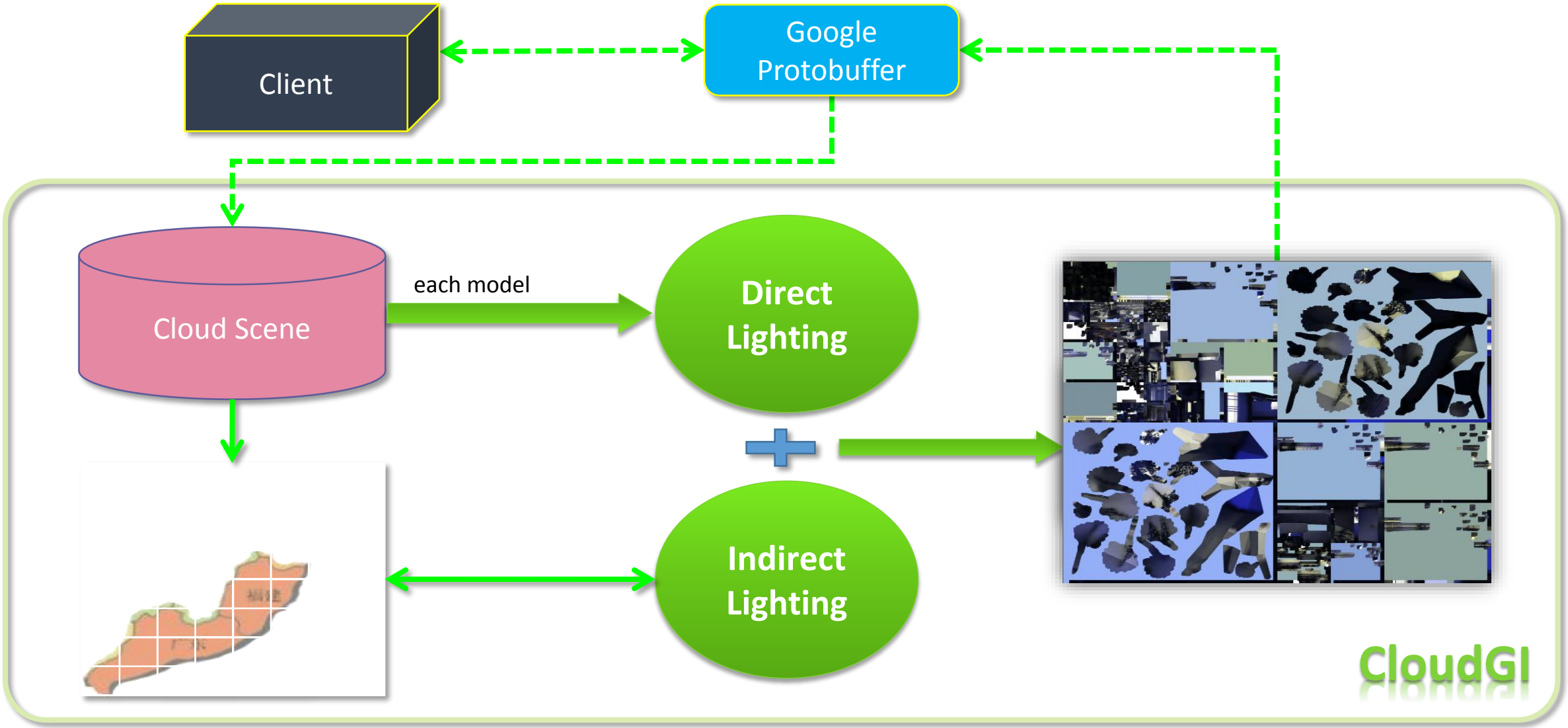
- Commercial baking software currently available on the market

Very slow baking

Patch more than one G

Fixed material

CloudGI panorama



Video Demo 1

Baking direct lighting

Deferred Shading

Sunlight shadow

- Strategies for camera frustum are not longer effective

- Strategies for perspective are not longer effective

Sunlight shadow

- Every model has a ShadowMap
- A huge ShadowMap

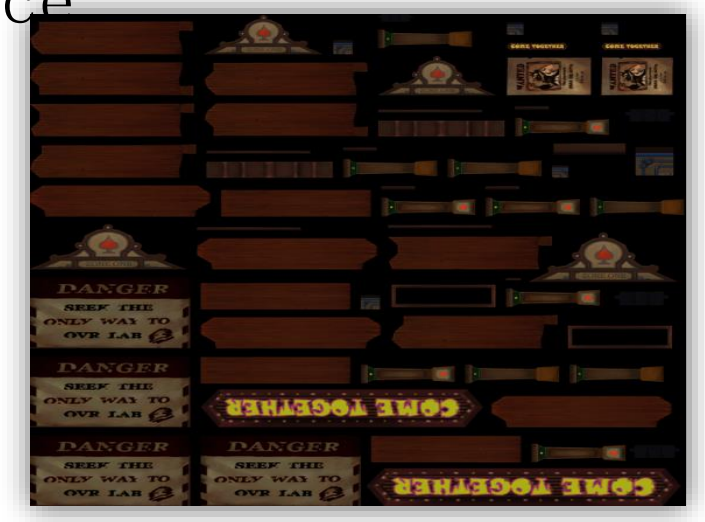
5000 X 4000

- 10000X10000 = 200M graphics memory
- Memory for Cache + Shadow Mask

LightMap crack

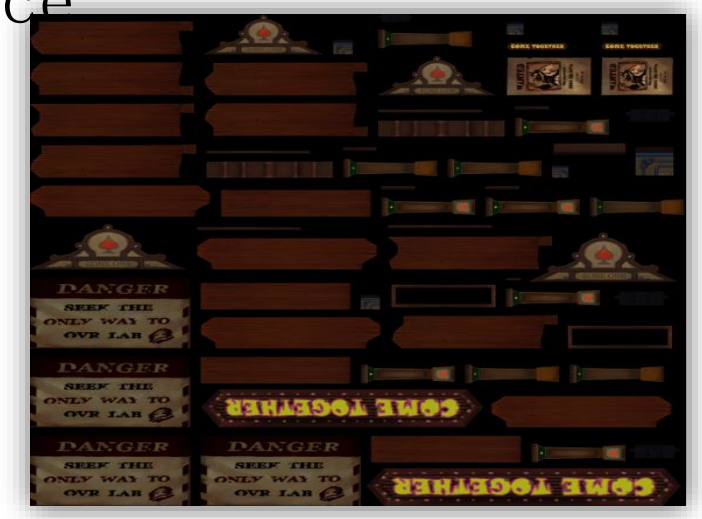
LightMap crack

- Continuous in the world space, discontinuous in the uv space



LightMap crack

- Continuous in the world space, discontinuous in the uv space



  - Gutter, using the color of nearest baking point to fill the background

LightMap crack

LightMap crack

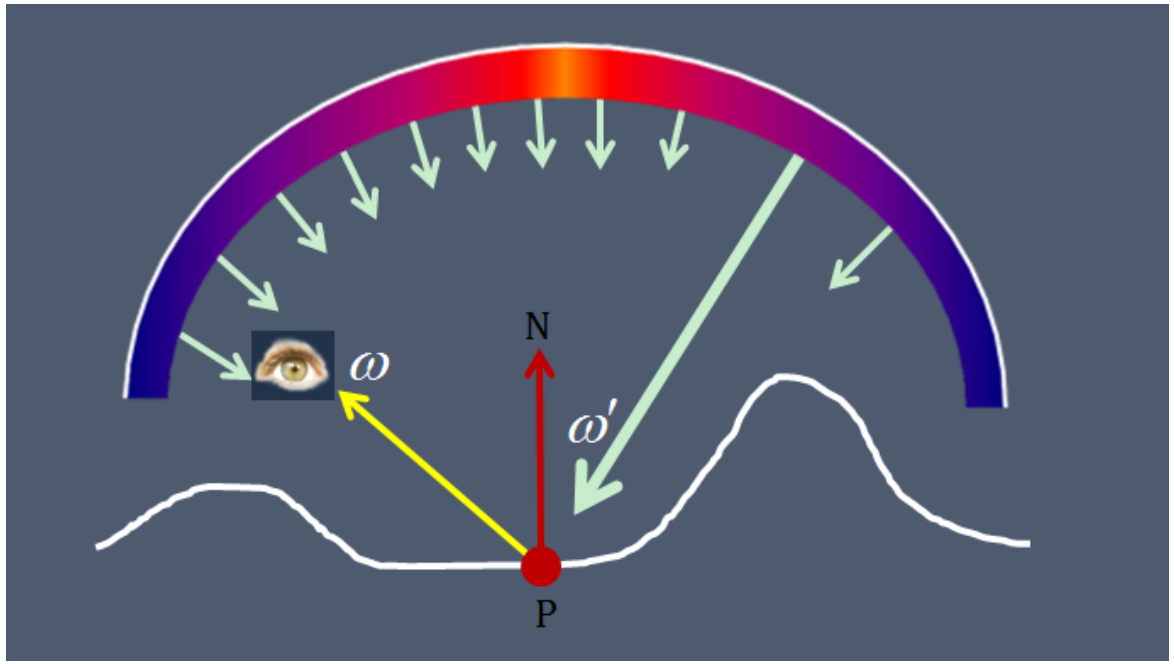
LightMap crack

- Fine edge in lightmap is smaller than one pixel

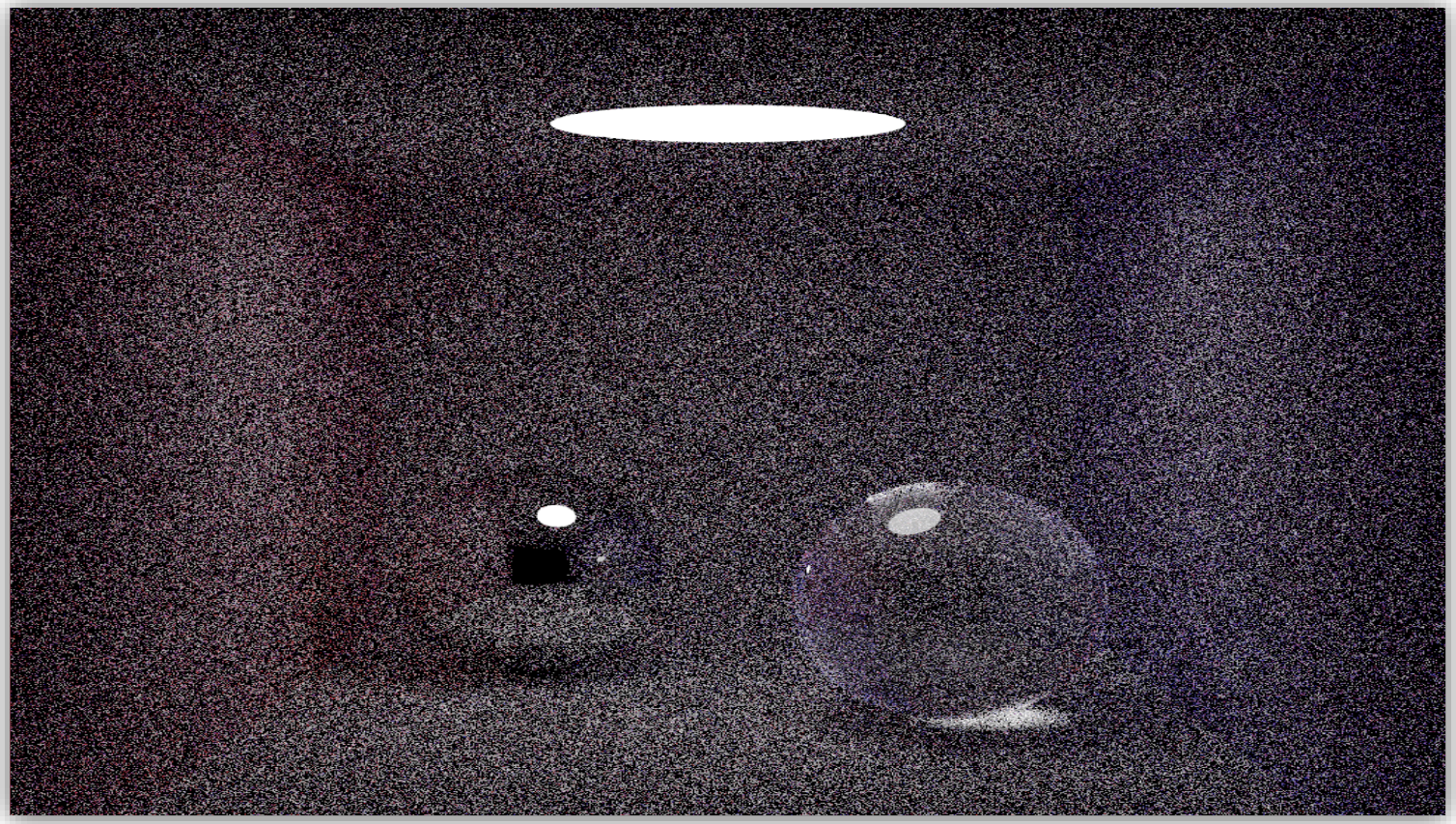
- Super Sample->Gutter->Down Sample

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

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

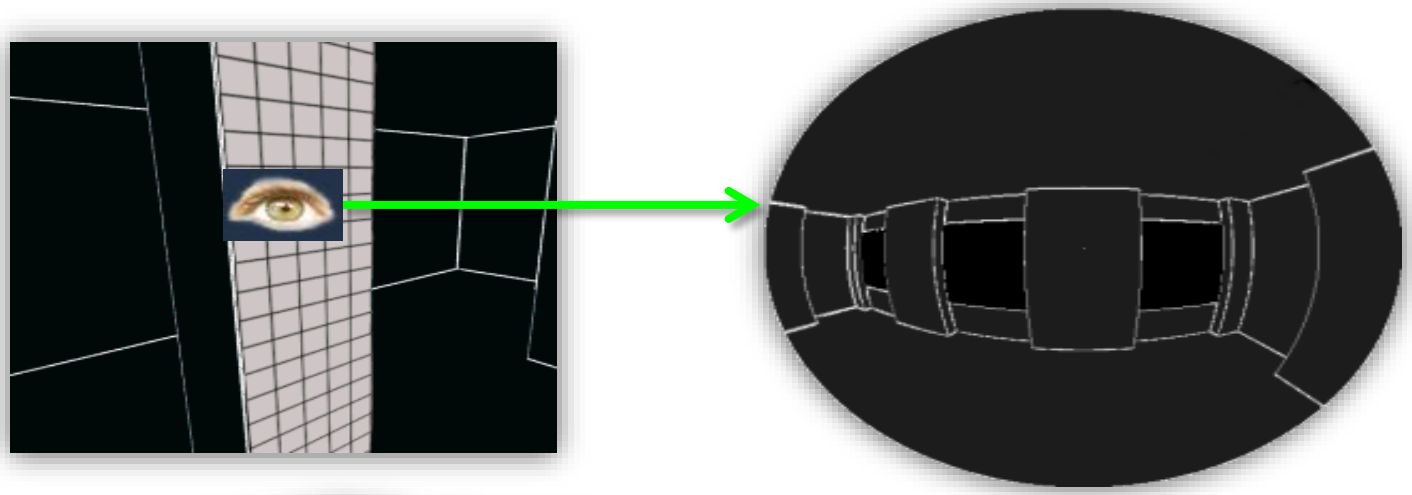
- Can do all the effects

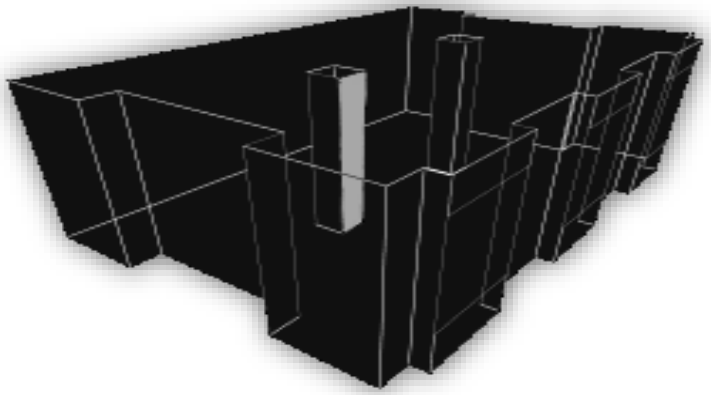
- **Exponential growth**
- **Noise**

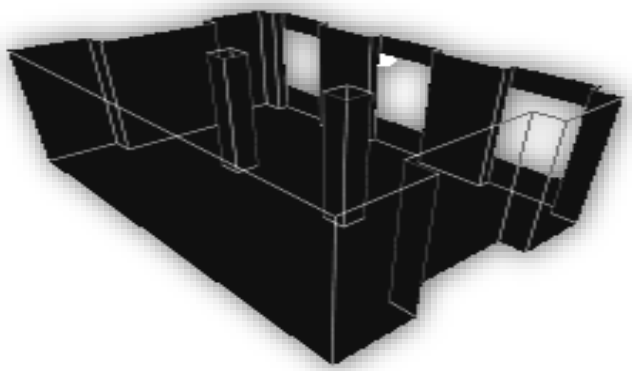
Point Cloud based GI

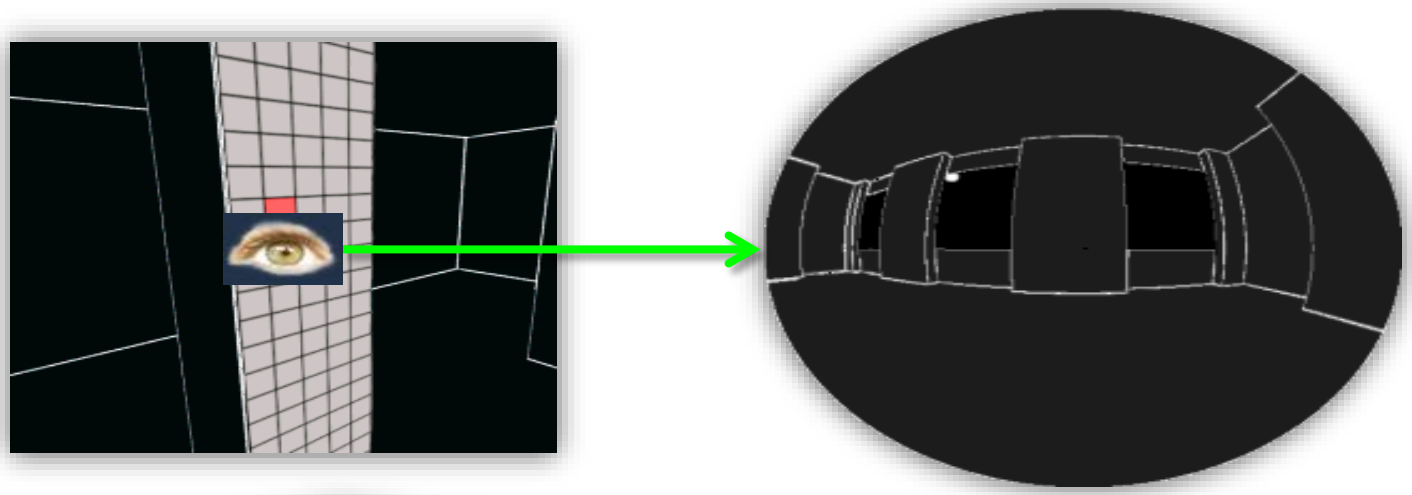
- Radiosity

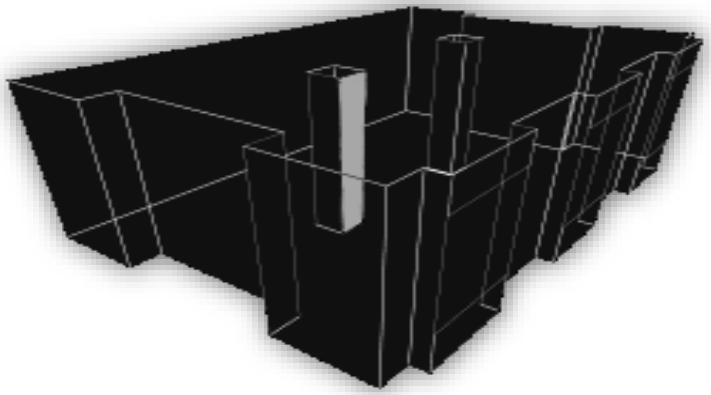
- Does not distinguish light and object
- Every surfel is lighted by all the other surfels

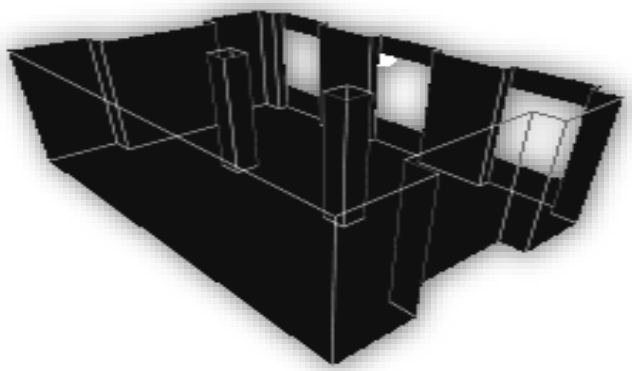


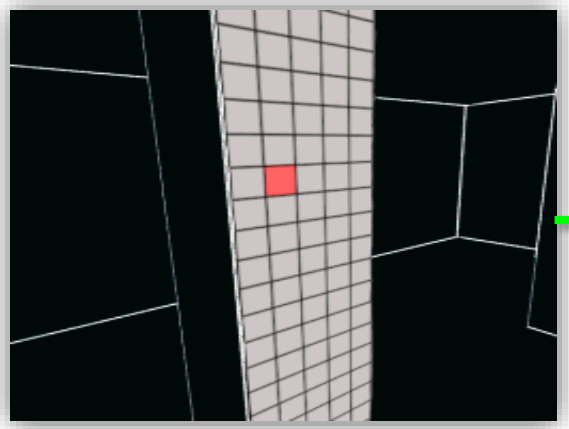


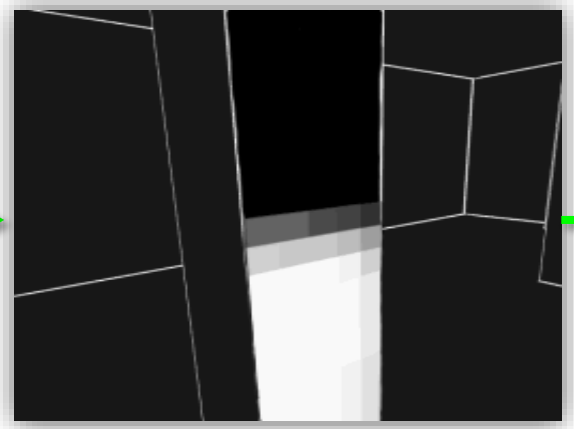


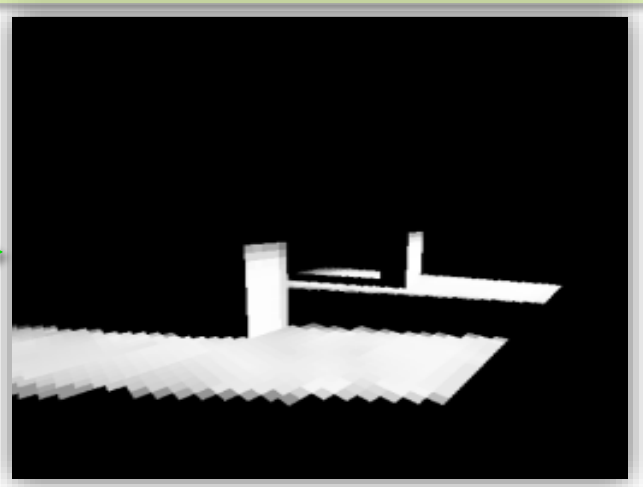


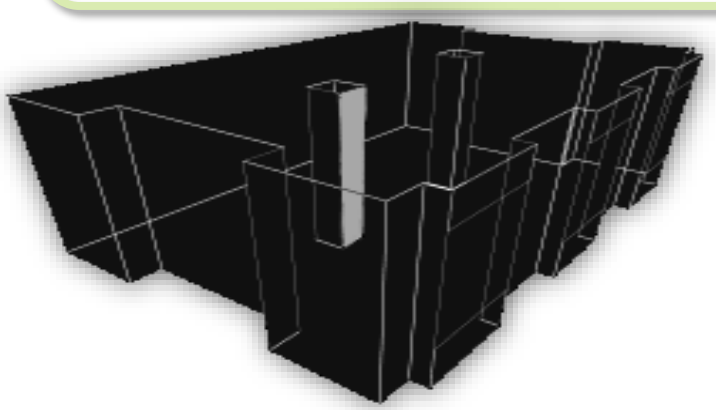


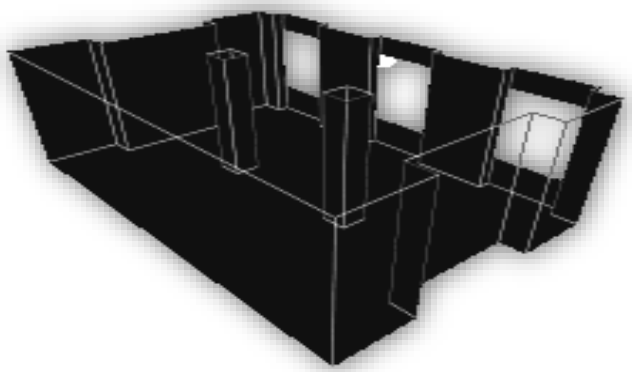


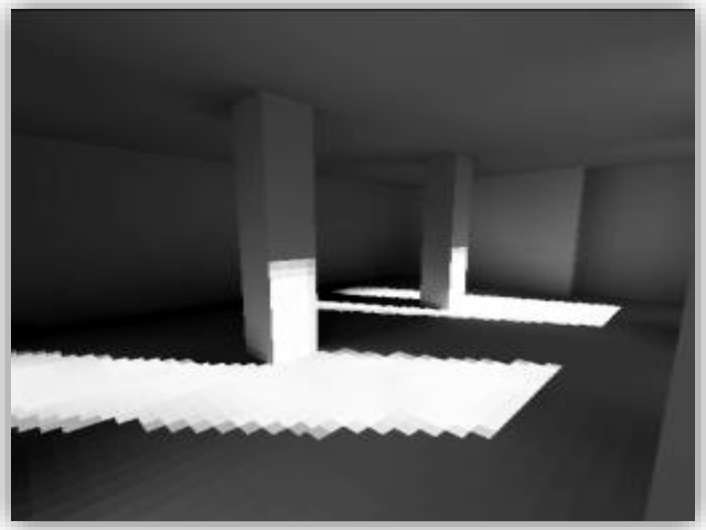




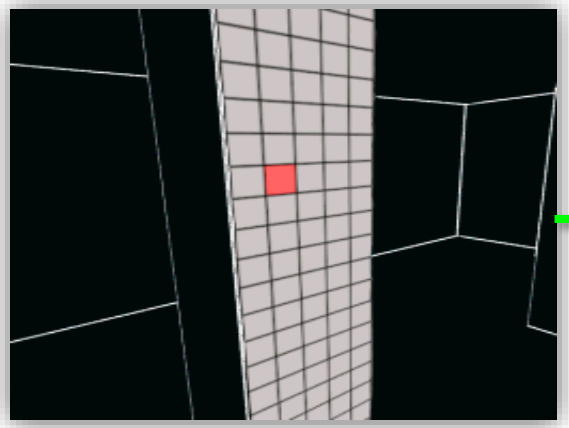


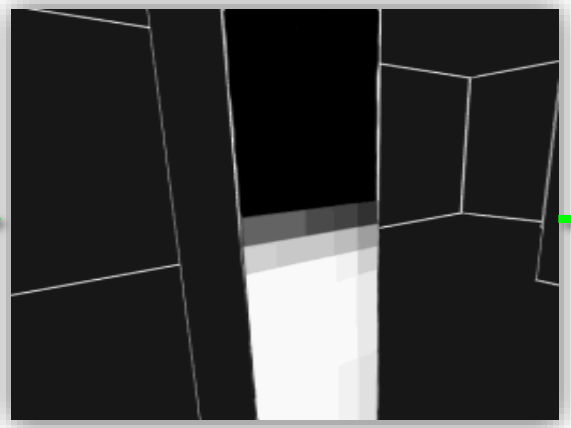


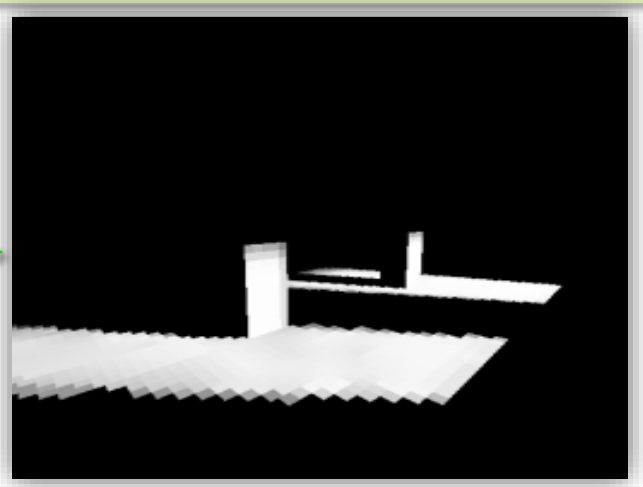


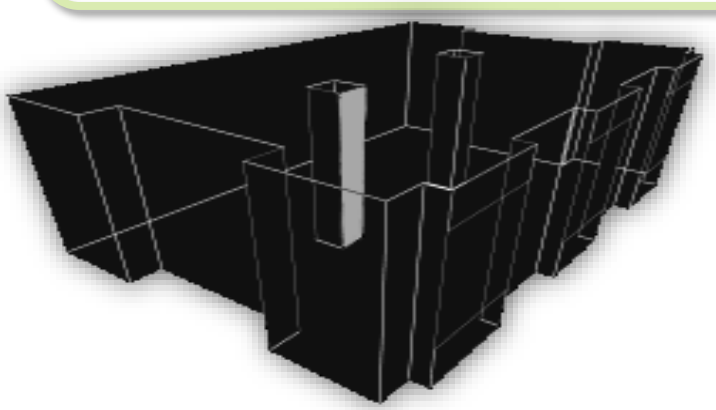


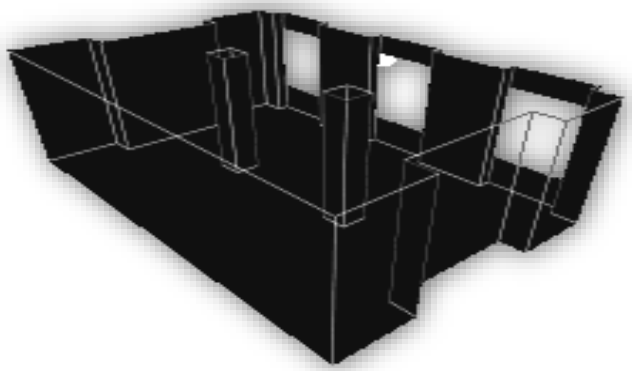
2 bounce

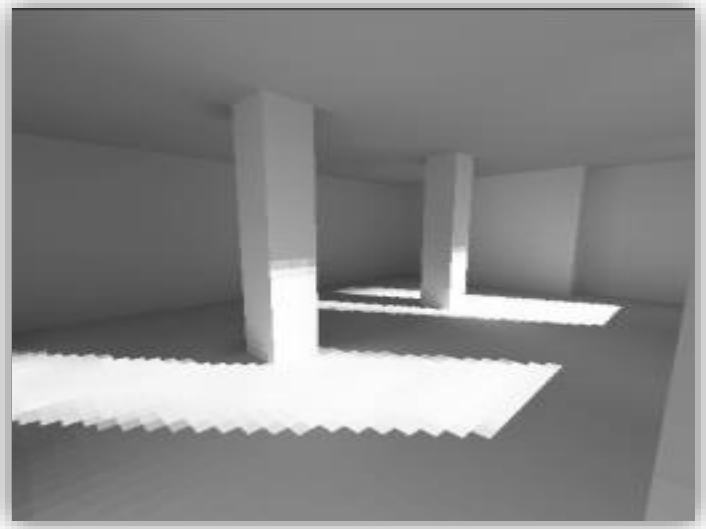








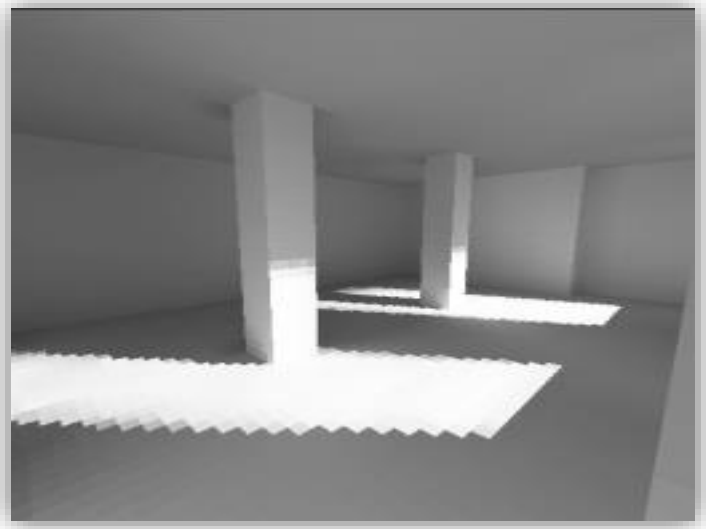




16 bounce

Point Cloud based GI

- Radiosity



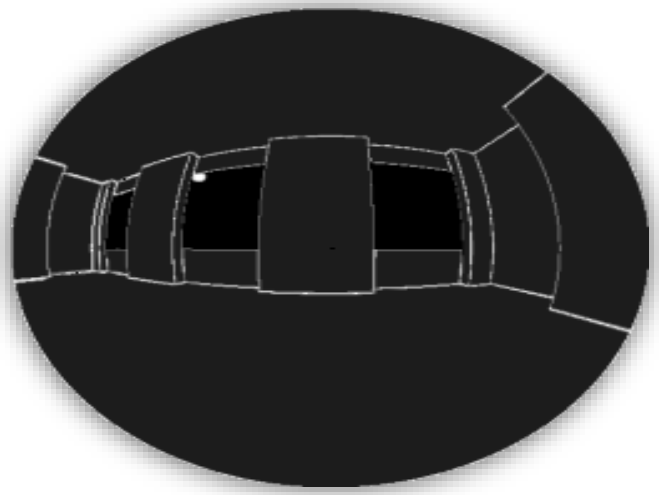
O(n*n*k)

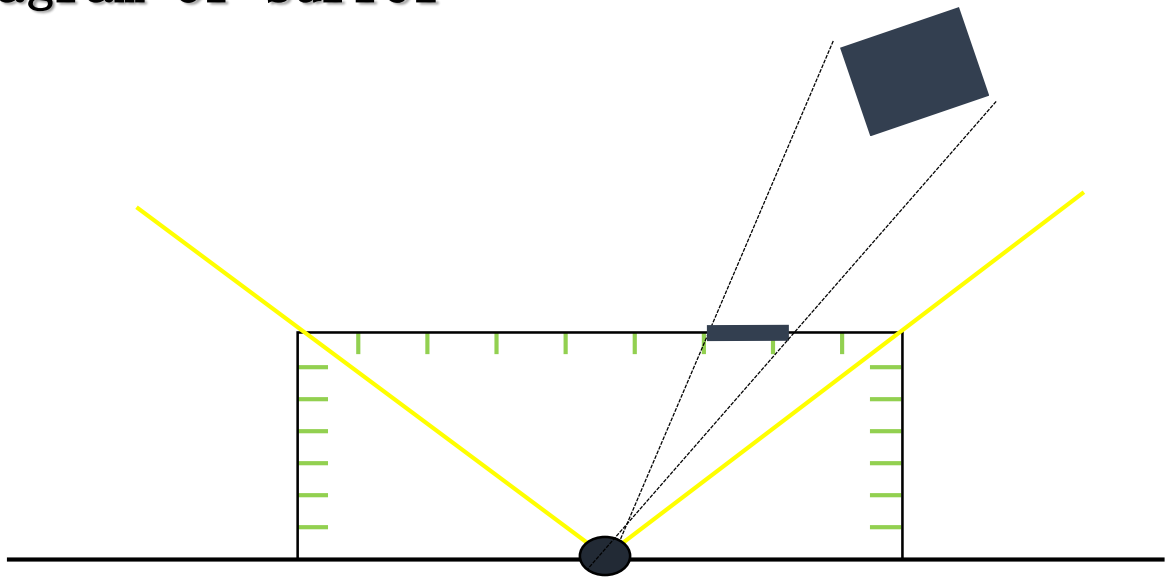
- Linear growth
- Without noise

- **Only can do diffuse reflectance GI effects**

Point Cloud based GI

- Radiosity
  - How to calculate visible diagram of surfel

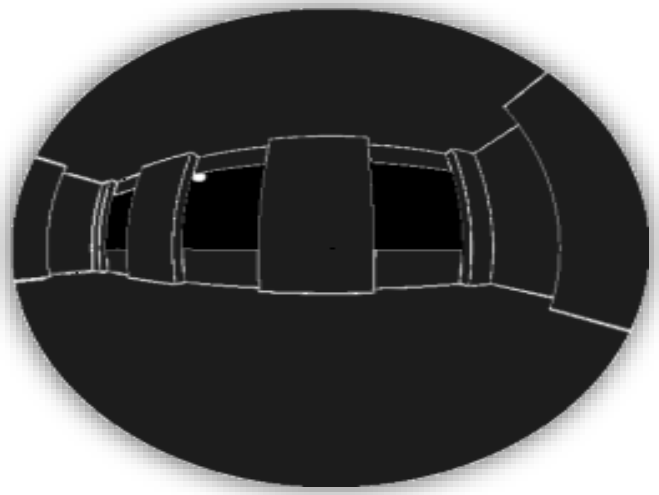


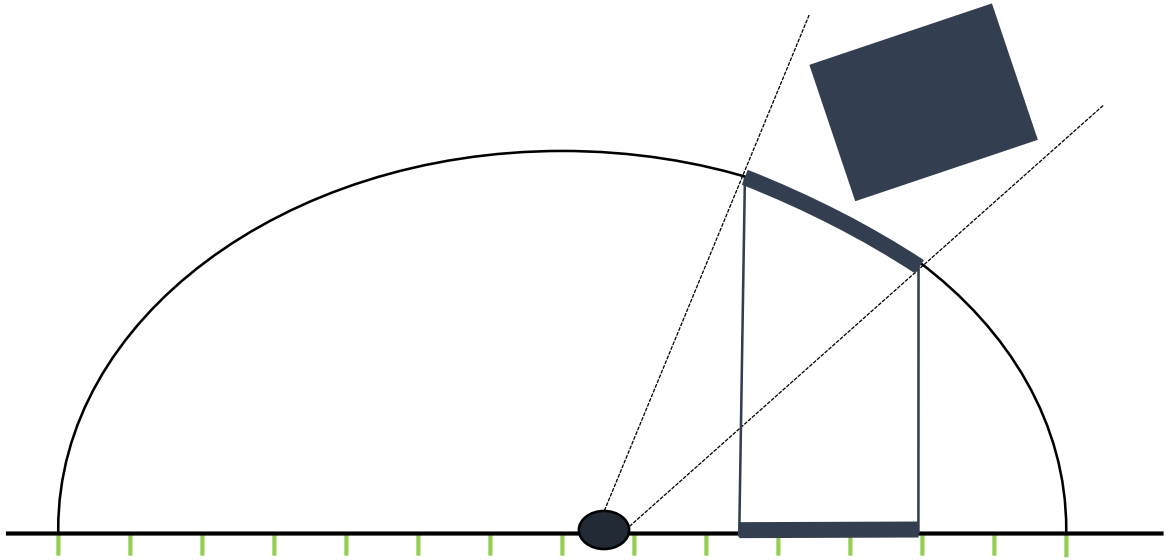


Hemicube, raster 5 times

Point Cloud based GI

- Radiosity
  - How to calculate visible diagram of surfel





Hemispherical, raster one time

Point Cloud based GI

- Point cloud generation
- Creation of Octree
- Calculation of indirect lighting

Point Cloud Generation

- How to generate a point cloud
  - Same as direct lighting

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

DX11 Unordered Access View + Atomic Add
CUDA Memory

Deferred Shading

Point Cloud Generation

- How to calculate Area?

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

Geometry Shader

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

Video Demo 2

Octree Building

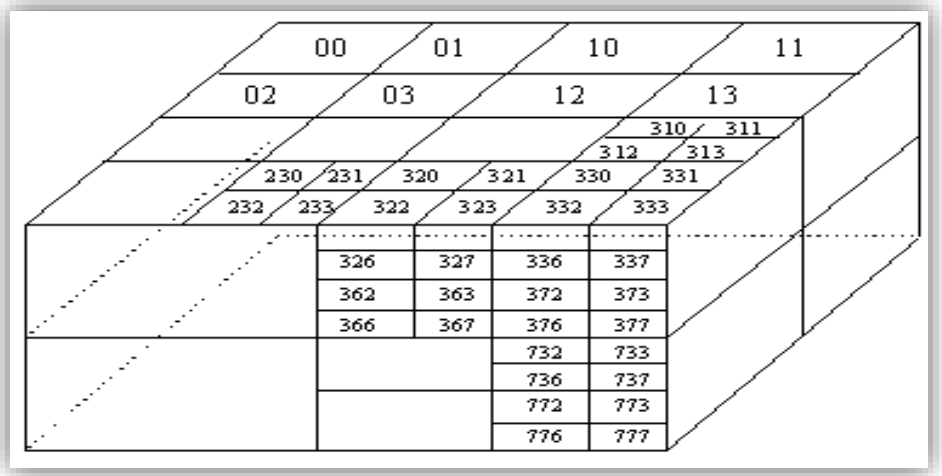
- CPU top, from top to bottom
  - Unfriendly concurrency
  - Dynamic memory allocation
  - Need synchronization or atomic operati

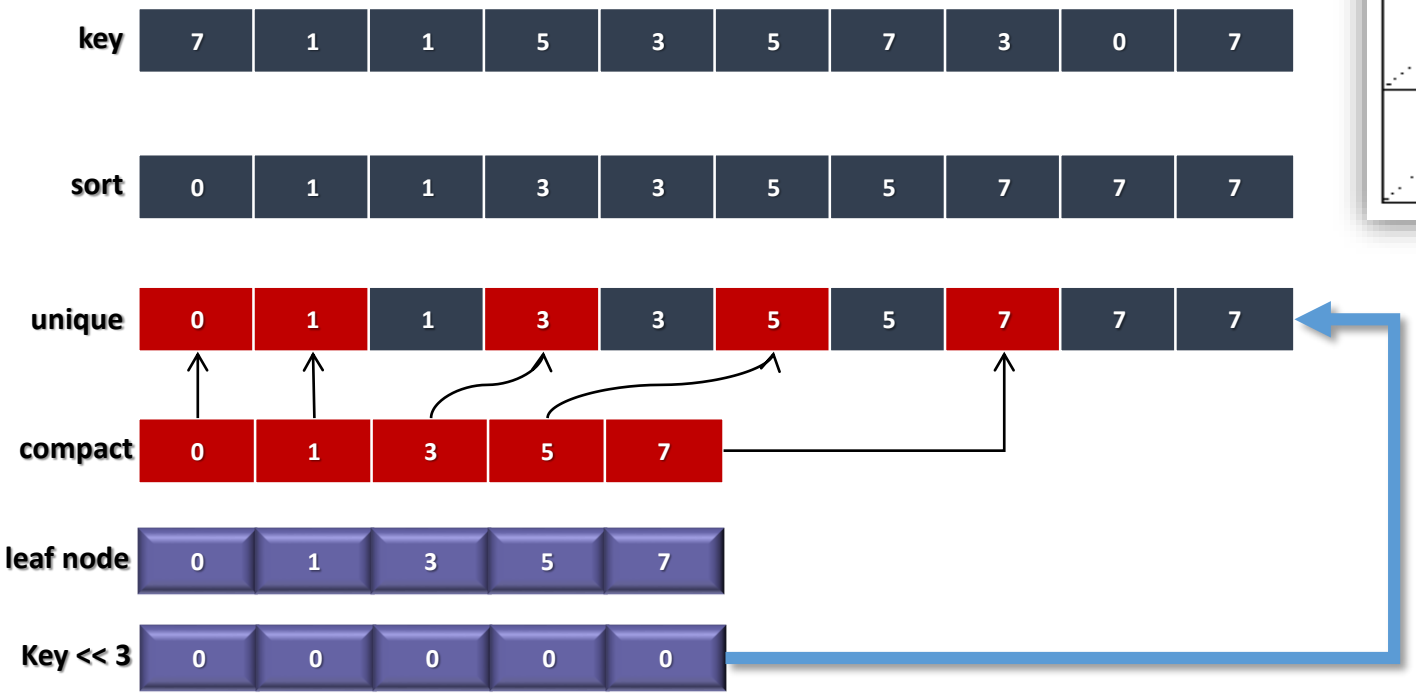
Octree Building

- CPU top, from bottom to top
  - Natural concurrency
  - Does not need synchronization
  - Graphics memory is not wasted

Octree Building

- Creation of leaf node
  - 3 bits for one layer coding, so 32-bit means 10 layers



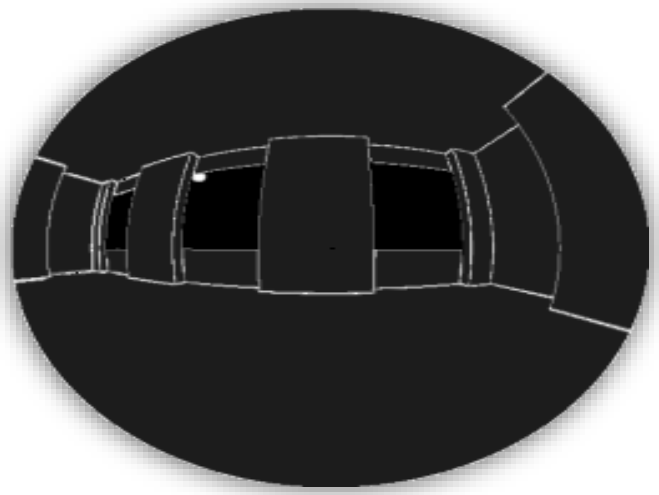


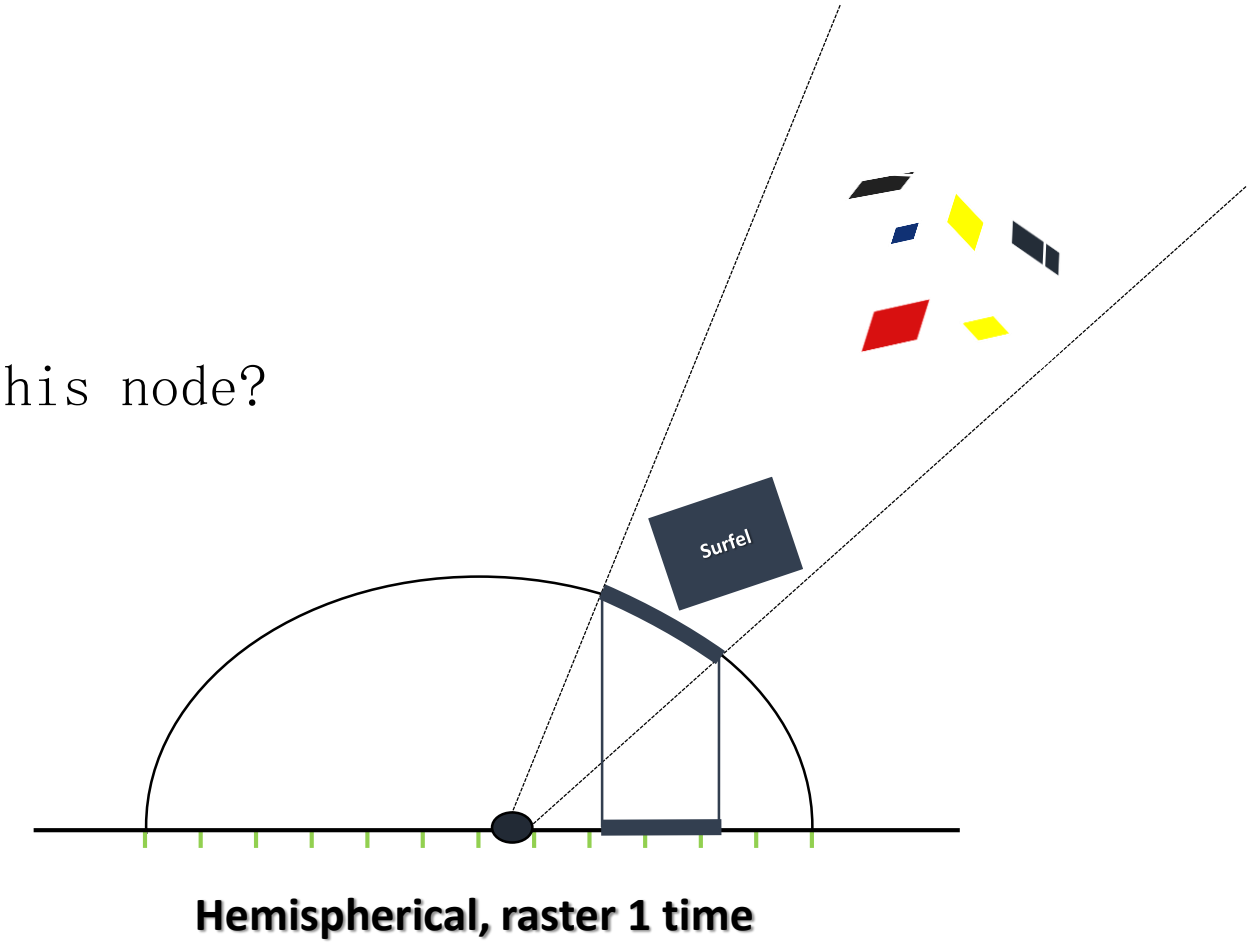
Thrust——CUDA based STL

Video Demo 3, Octree

Octree Building

- Why to build Octree
  - O(n*n*k)->O(n*logn*k)
  - What is the area and color of this node?

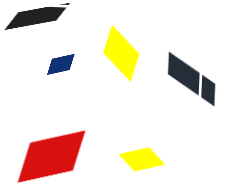




Hemispherical, raster 1 time

Octree Building

- Spherical Harmonics
  - A set of orthogonal basis functions defined in the sphere
  - $Y_l^m(\theta,\varphi) = \cdots; l \in N, -l \le m \le l$



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

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

Octree Building

- Spherical Harmonics
  - Turn one function $f(\theta,\varphi)$ into several parameters
  - Turn sum of two functions into sum of several parameters

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

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

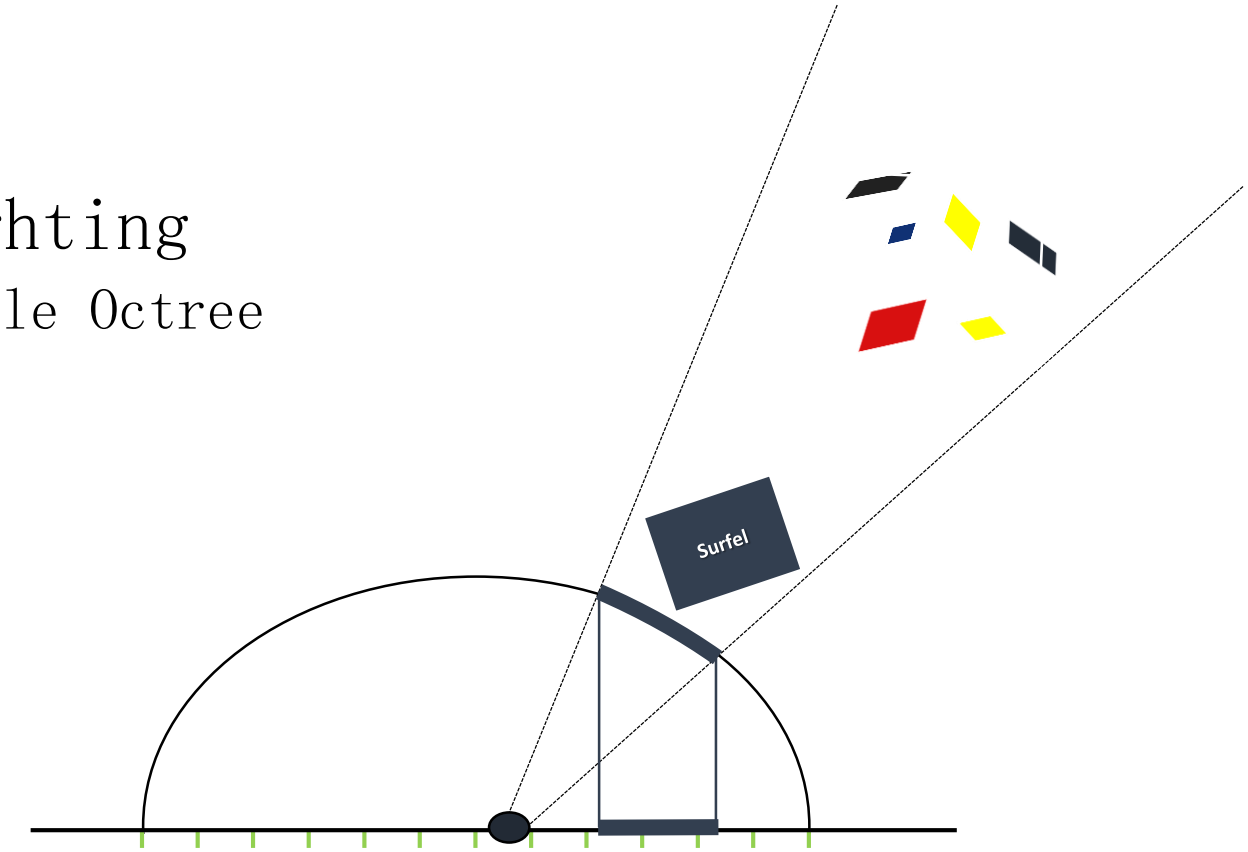
$$a_l^m = \int A_i |dot(\vec{d},\vec{n})|\, Y_l^m(\theta,\varphi) d\omega$$

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

Indirect Lighting

- Calculation of indirect lighting
  - Get id map by examining the whole Octree
  - calcRadianceFromID
  - Average * PI

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



Hemispherical, raster 1 time

Algorithm Optimization

- Baking a small scene needs half an hour!!!
  - Baking accuracy == Indirect lighting accuracy
  - 1 oversampled lightmap has 16 million point clouds
- Separate baking accuracy, indirect lighting accuracy
  - K-near interpolation, considering normal, position
  - Preset threshold

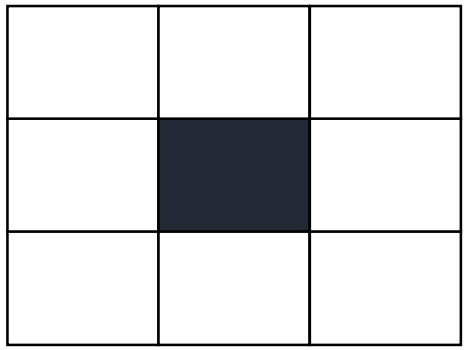
Video Demo 4

Block Baking

- 1000W point clouds=400M
- A square of 1000m×1000m
  - 4G graphics memory

◉ Block baking is necessary

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



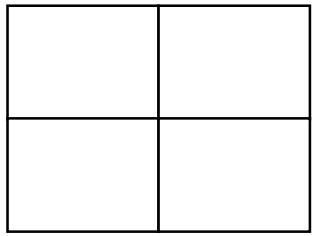
The whole scene, rough point cloud

Within the block + safe area, fine point cloud

Translucent material

- 1 line of code is enough
  - surfel.area *= alpha;

- ◉ Translucent shadow
  - Traditional method is using two shadow maps
  - Write depth map with alpha probability



Translucent material

Custom material

- The calculation time of Radiosity has nothing to do with the complexity of material

```
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 );
}
```

Summary

- Real-time GI is a next next generation technology

- CloudGI is faster than cpu raytrace by 1000 times
  - Practical function suitable for online games (real-time preview, minpatch, custom material)

- Use Octree to accelerate Radiosityal gorithm
  - K-near interpolation acceleration

- Block baking solves the problem of graphics memory

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