Real-Time High Quality Rendering

GAMES202, Lingqi Yan, UC Santa Barbara

Lecture 8:
Real-Time Global Illumination
(screen space)




Announcements

- Homework 2 has been released!
- Homework 3 will be about Screen Space Reflection (SSR)
- GAMES101 homework submission
  - Still recruiting graders!
  - Now do not need to have taken GAMES101 before

Last Lecture

- Precompuated Radiance Transfer (cont.)
  - SH for glossy transport
  - Wavelet
- Real-Time Global Illumination (in 3D)
  - Reflective Shadow Maps (RSM)
  - Light Propagation Volumes (LPV)
  - Voxel Global Illumination (VXGI)

Today

- Finishing up
  - Light Propagation Volumes (LPV)
  - Voxel Global Illumination (VXGI)
- Real-Time Global Illumination (screen space)
  - Screen Space Ambient Occlusion (SSAO)
  - Screen Space Directional Occlusion (SSDO)
  - Screen Space Reflection (SSR)

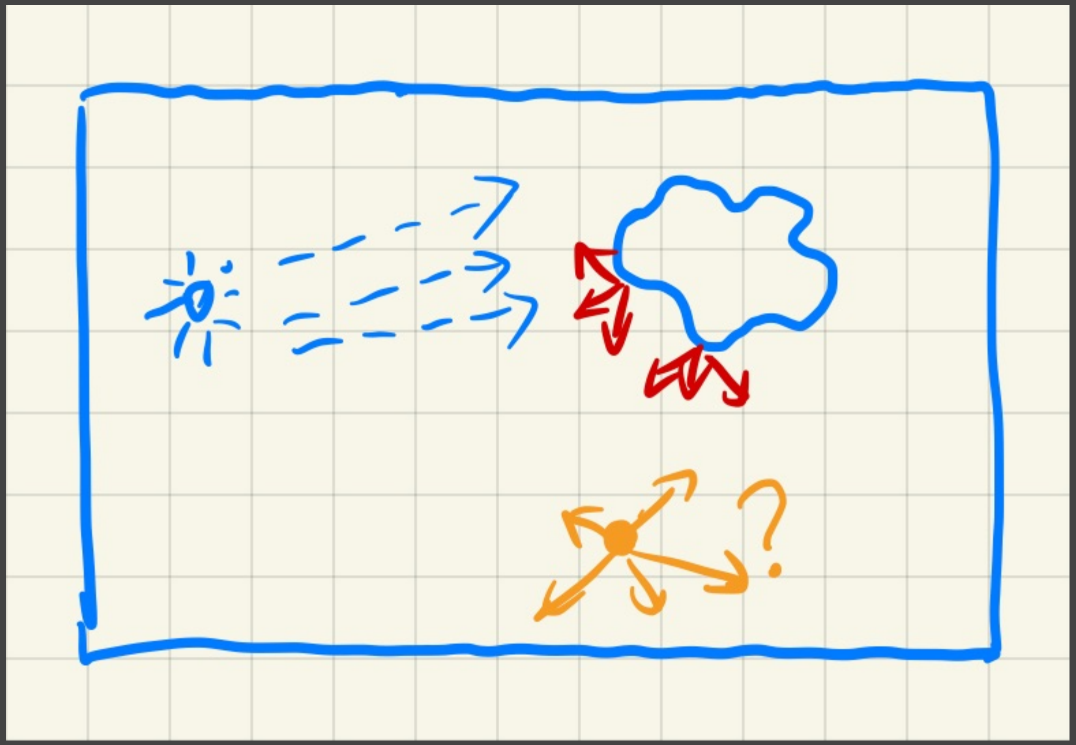
Light Propagation Volumes (LPV)

Light Propagation Volumes (LPV)

- First introduced in CryEngine 3
  - Fast performance and good quality

Light Propagation Volumes (LPV)

- Key problem
  - Query the radiance from any direction at any shading point
- Key idea
  - Radiance travels in a straight line and does not change
- Key solution
  - Use a 3D grid to propagate radiance from
    directly illuminated surfaces to anywhere else



Light Propagation Volumes (LPV)

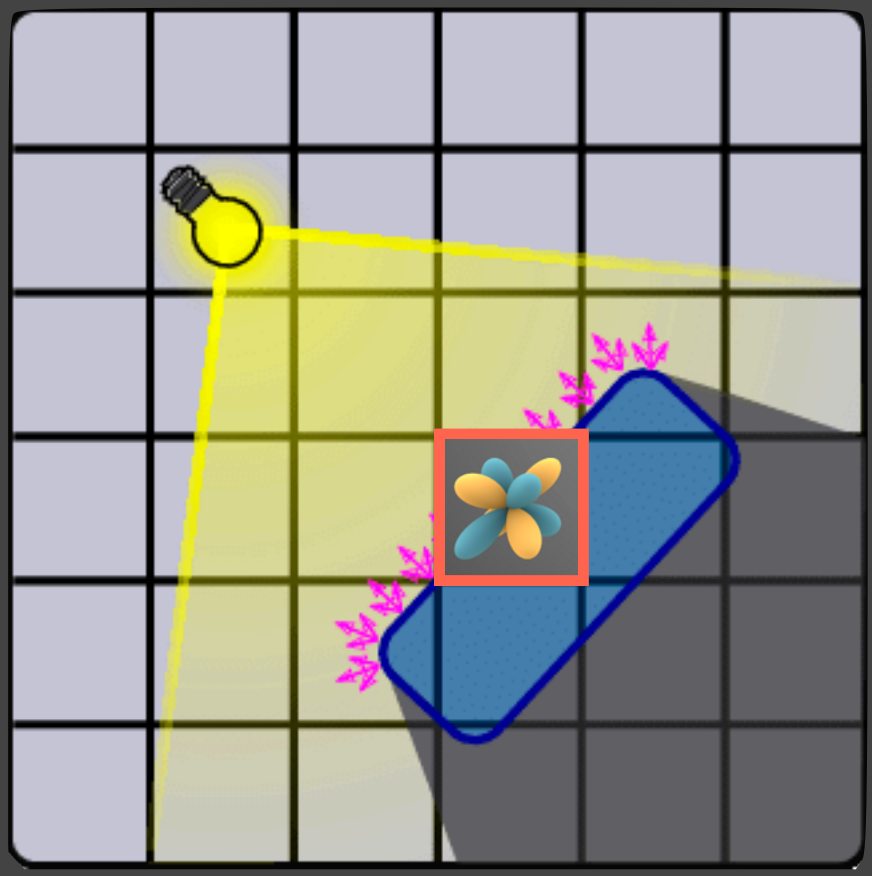
- Steps
  1. Generation of radiance point set scene representation
  2. Injection of point cloud of virtual light sources into radiance volume
  3. Volumetric radiance propagation
  4. Scene lighting with final light propagation volume

Light Propagation Volumes (LPV)

- Step 1: Generation
  - This is to find directly lit surfaces
  - Simply applying RSM would suffice!
  - May use a reduced set of diffuse surface patches (virtual light sources)

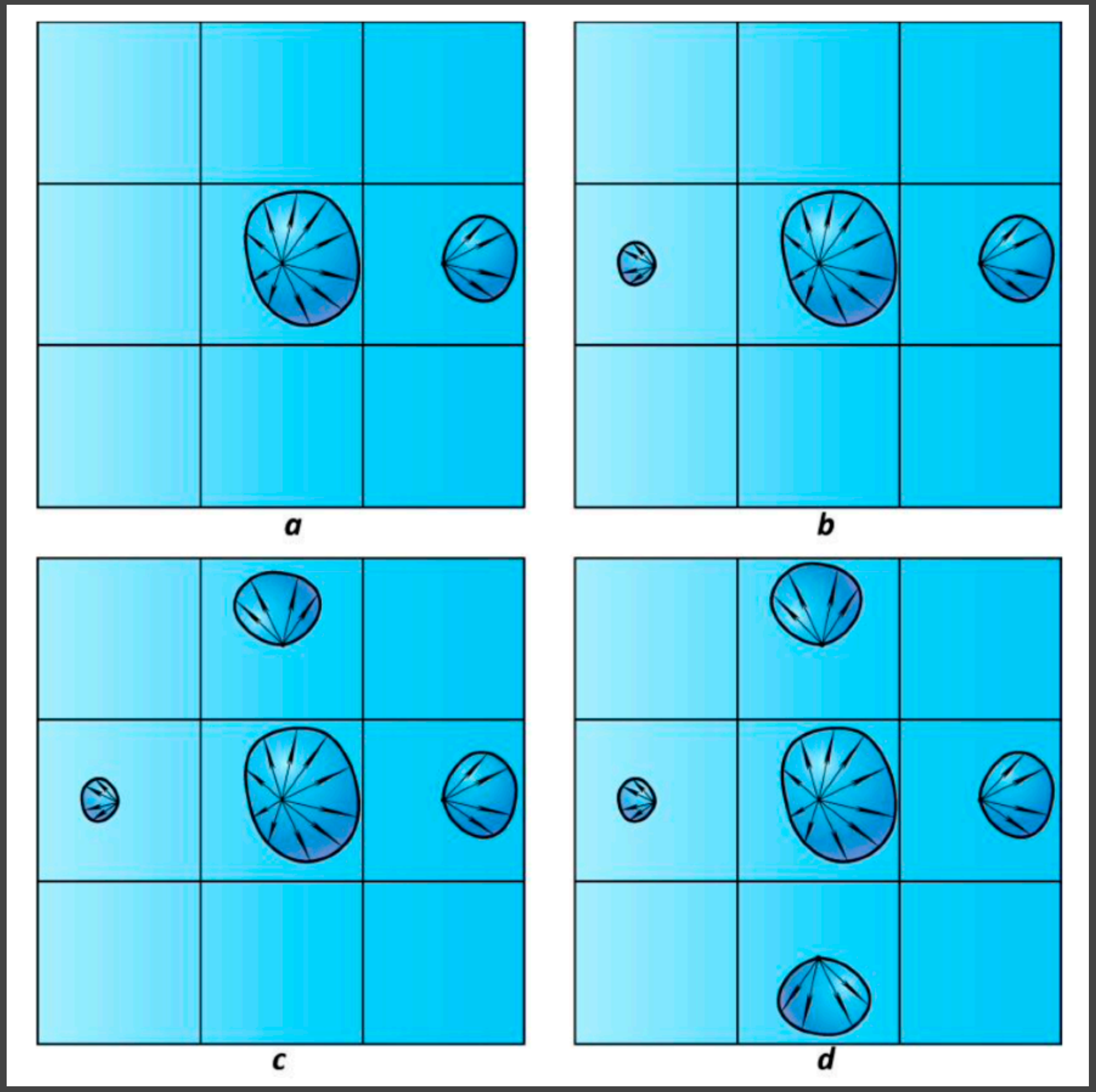
Light Propagation Volumes (LPV)

- Step 2: Injection
  - Pre-subdivide the scene into a 3D grid
  - For each grid cell, find enclosed virtual light sources
  - Sum up their directional radiance distribution
  - Project to first 2 orders of SHs (4 in total)



Light Propagation Volumes (LPV)

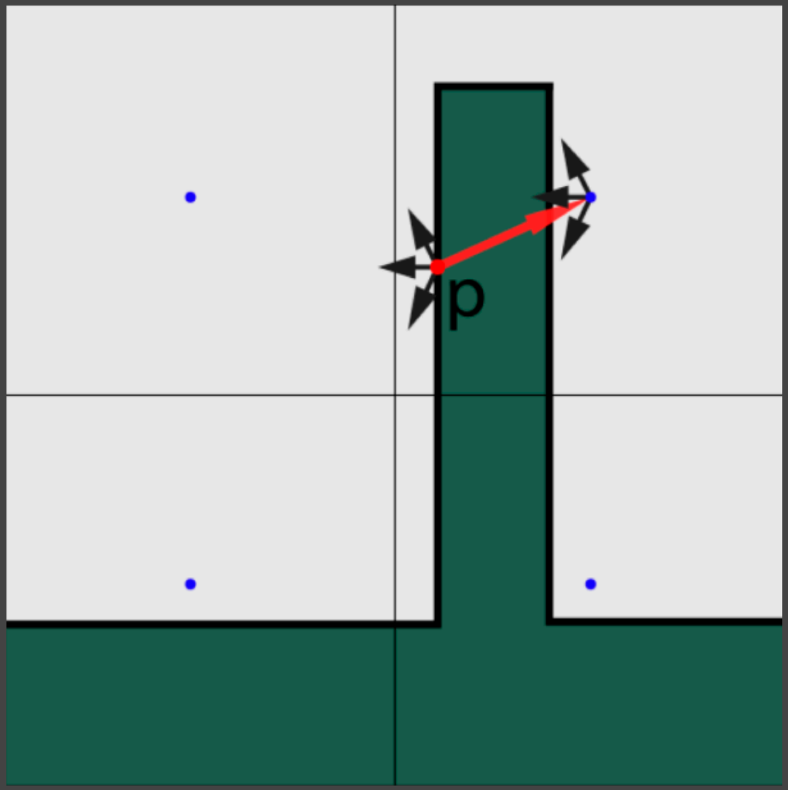
- Step 3: Propagation
  - For each grid cell, collect the radiance received from each of its 6 faces
  - Sum up, and again use SH to represent
  - Repeat this propagation several times till the volume becomes stable



Light Propagation Volumes (LPV)

- Step 4: Rendering
  - For any shading point, find the grid cell it is located in
  - Grab the incident radiance in the grid cell (from all directions)
  - Shade

- Any problems?
  - Hint: look at point p

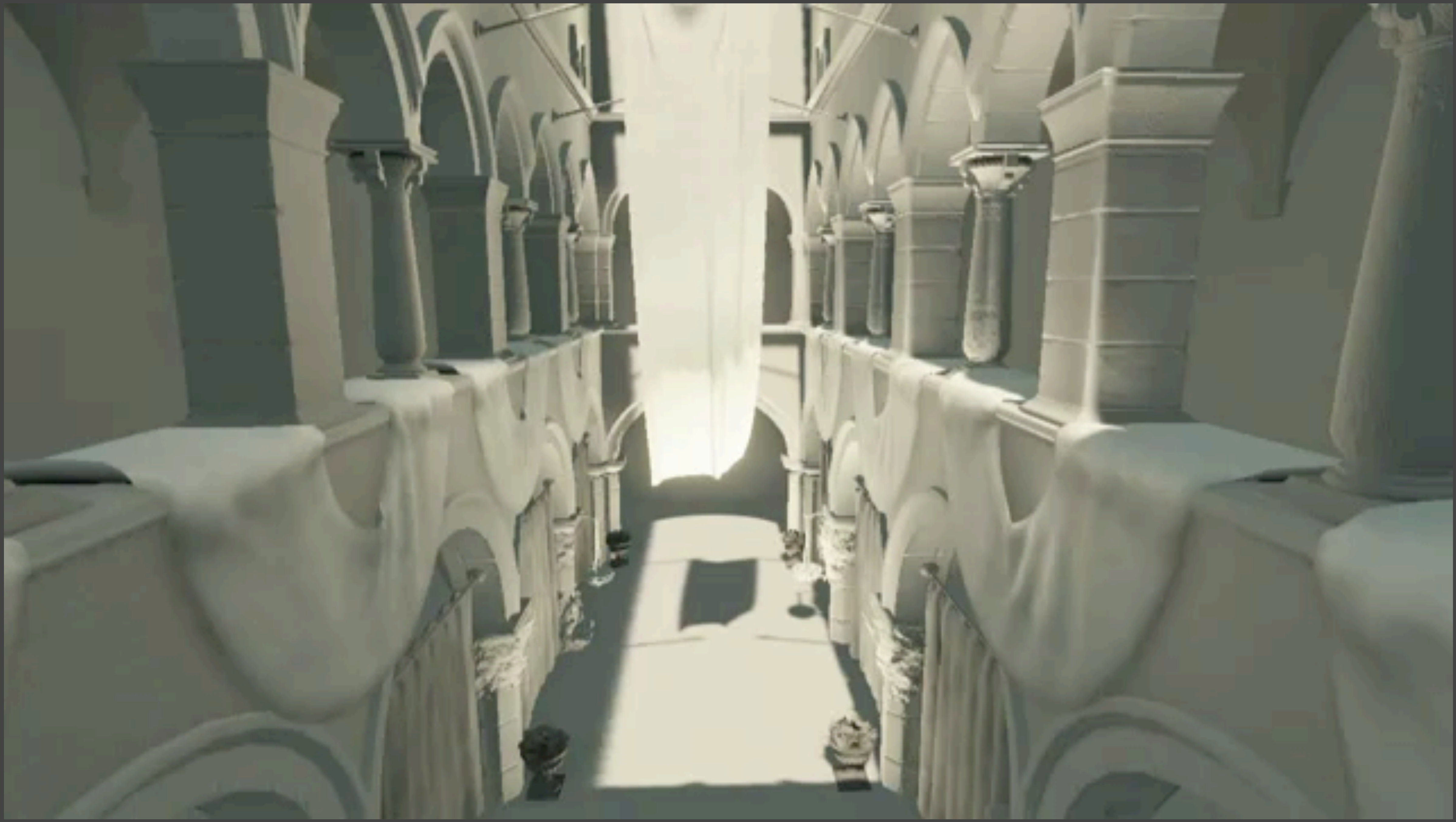


Light Propagation Volumes (LPV)

- Light leaking

LPV Reference

Light Propagation Volumes (LPV)

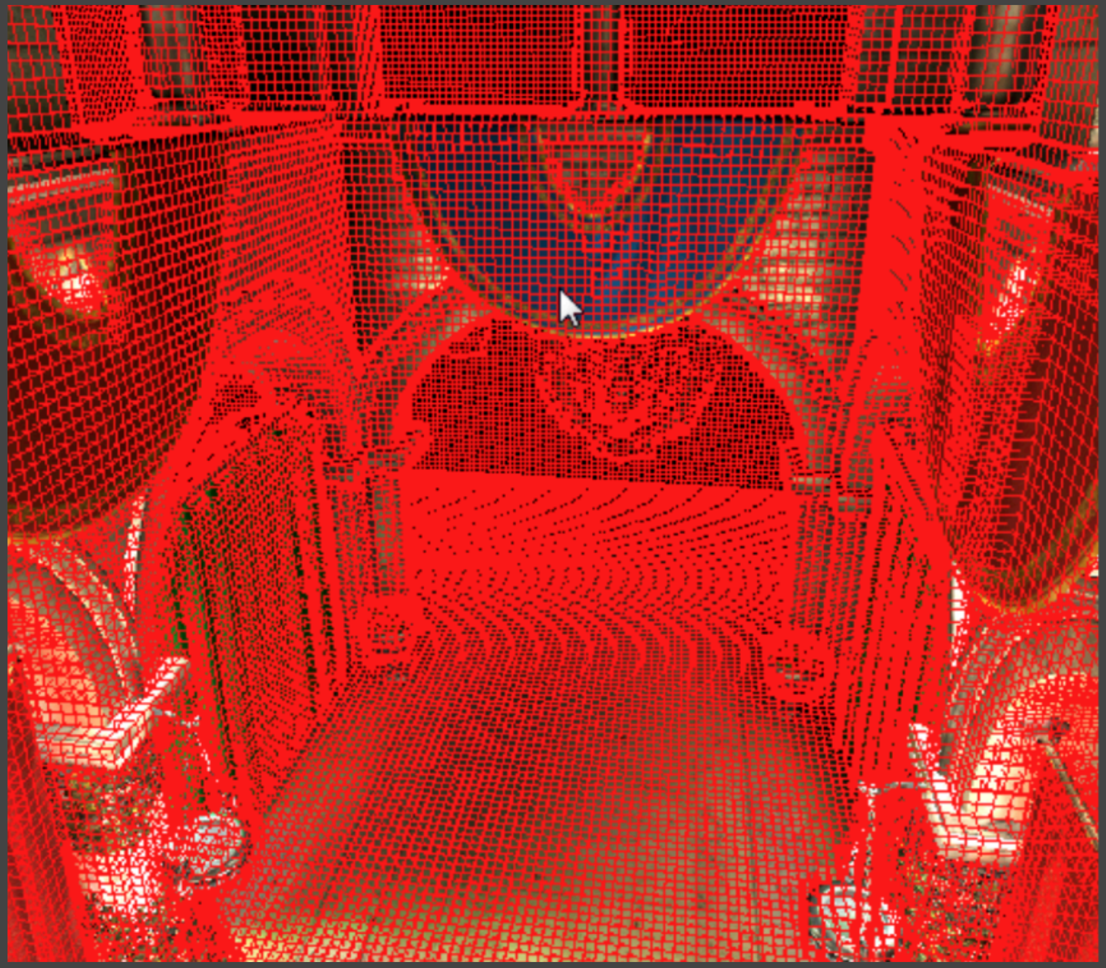


Questions?

Voxel Global Illumination (VXGI)

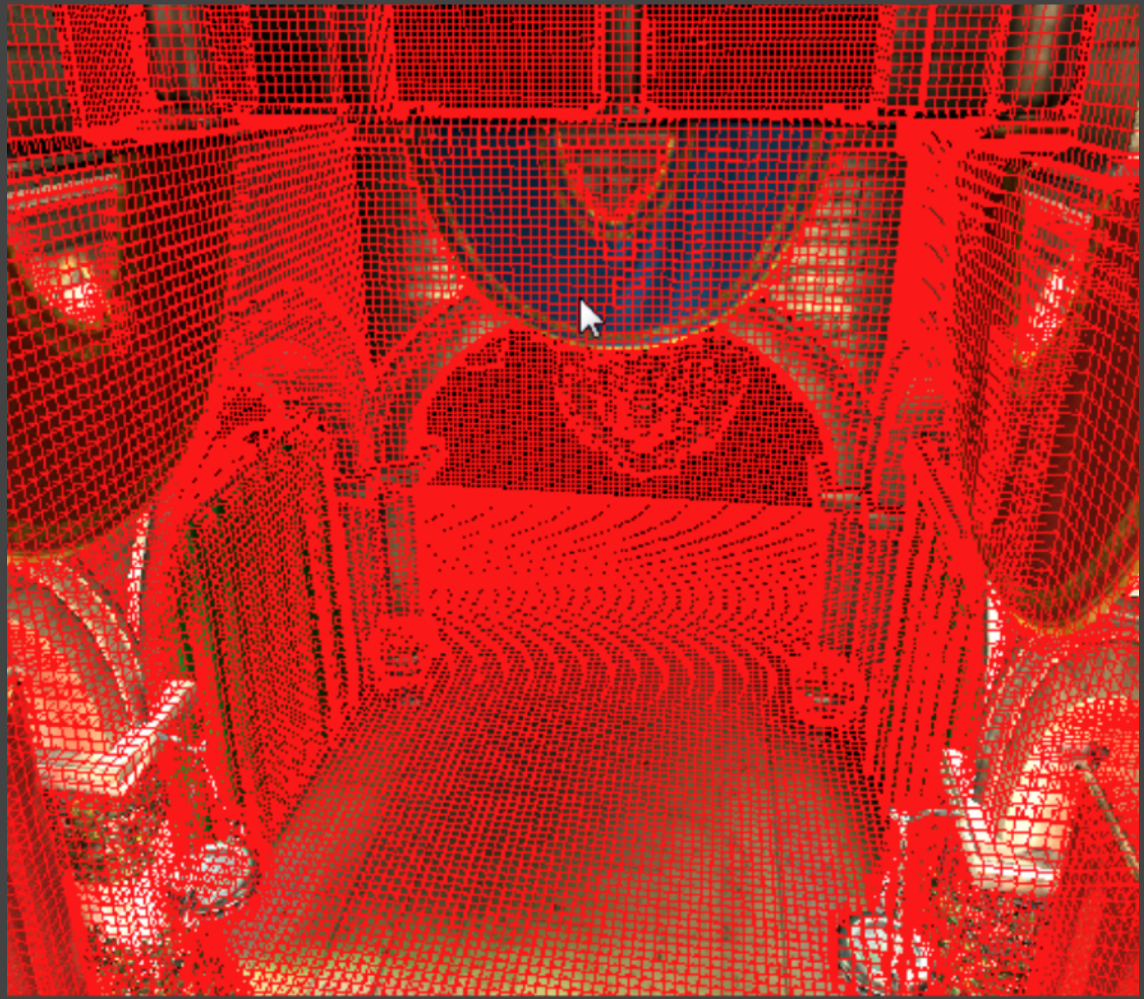
VXGI

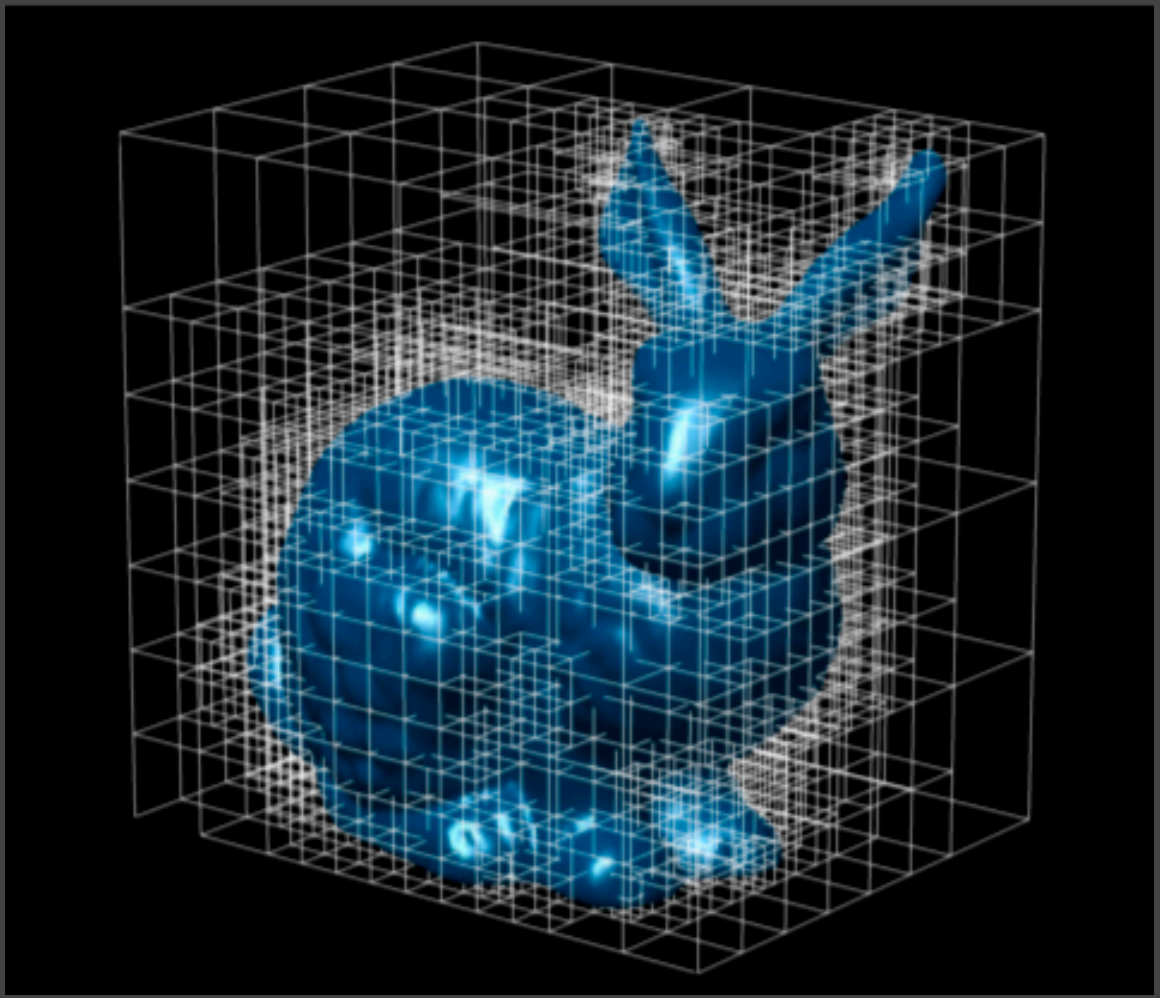
- Still a two-pass algorithm
- Two main differences with RSM
  - Directly illuminated pixels -> (hierarchical) voxels
  - Sampling on RSM -> tracing reflected cones in 3D (Note the inaccuracy in sampling RSM)



VXGI

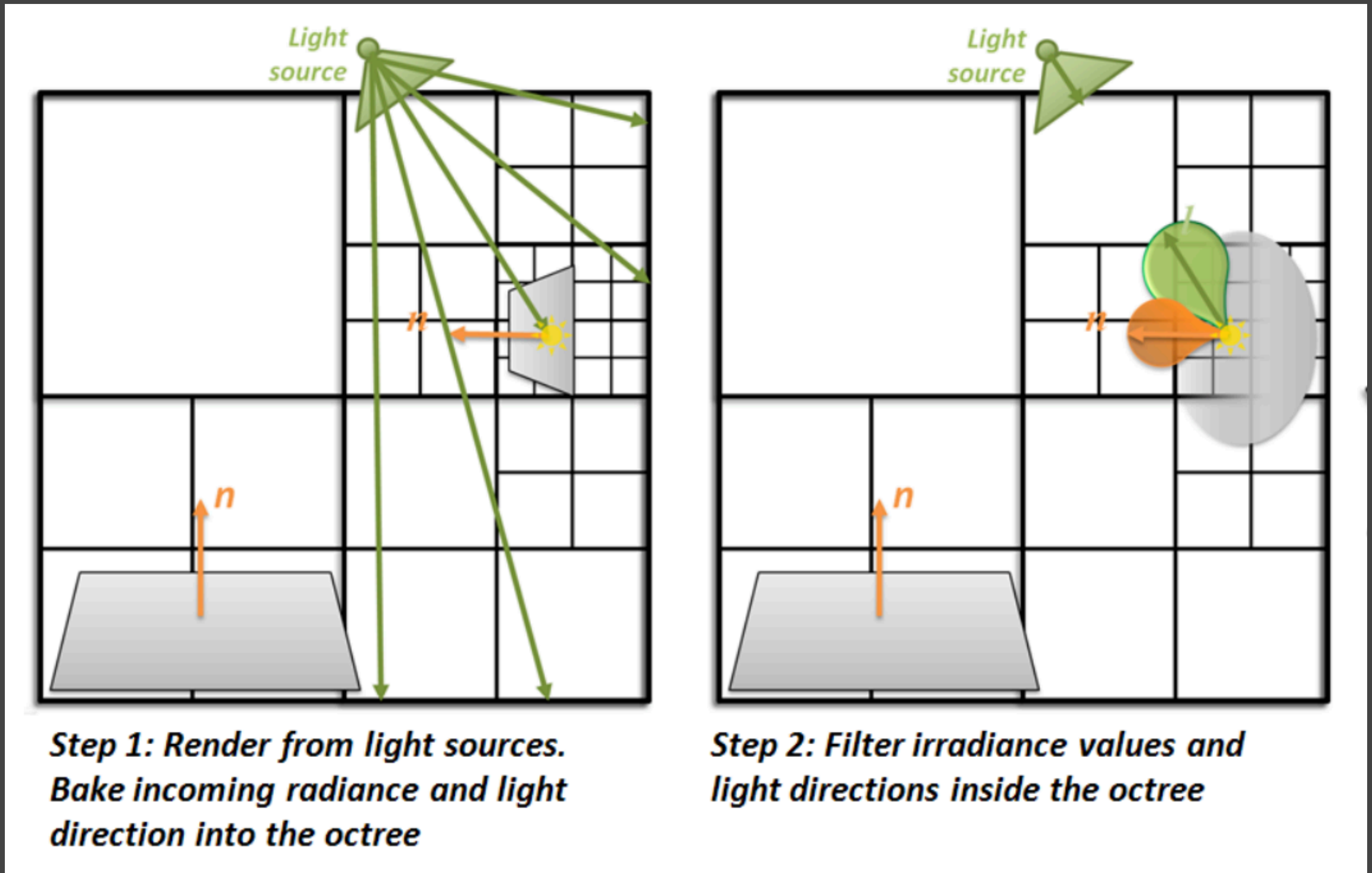
- Voxelize the entire scene
- Build a hierarchy





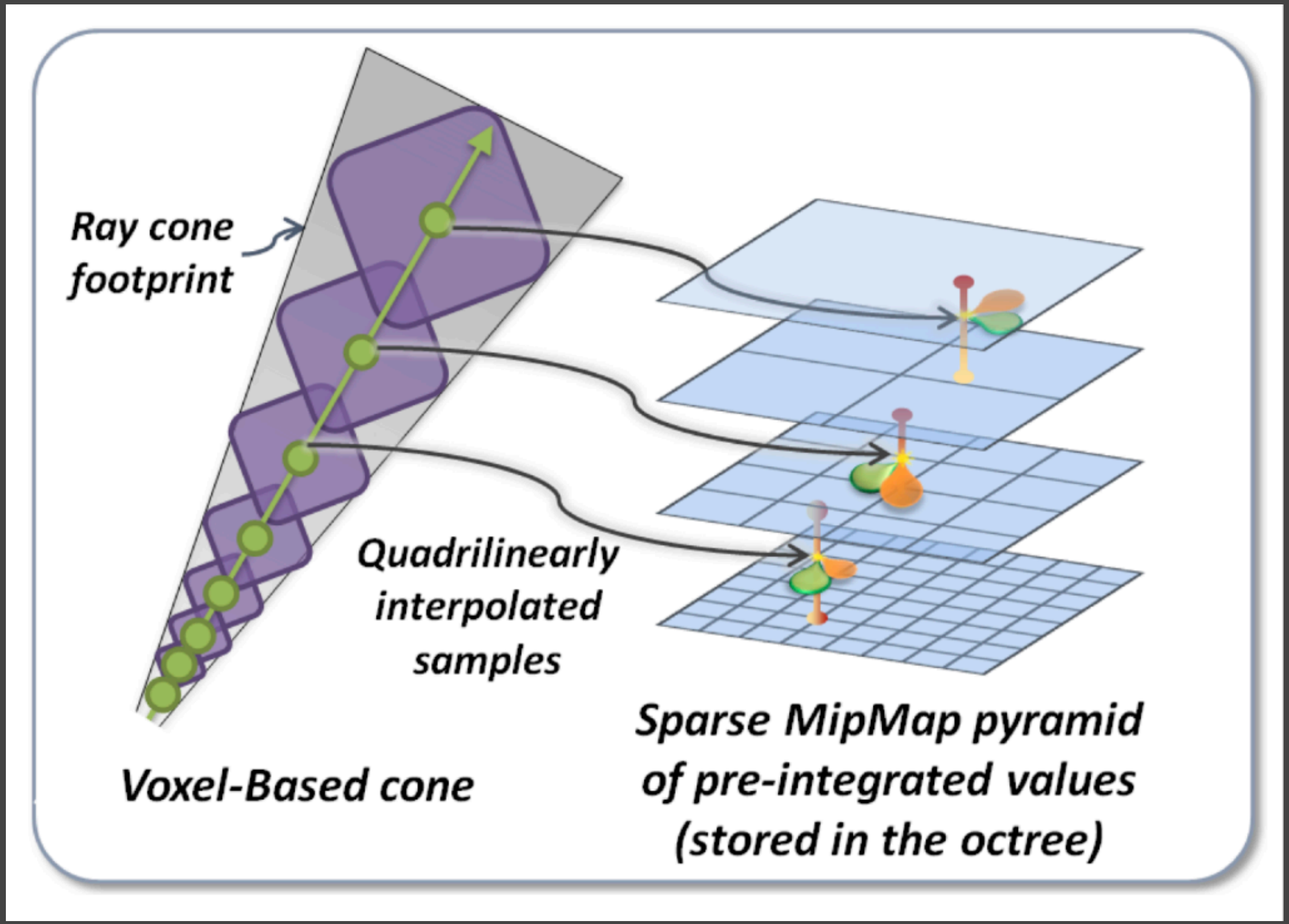
VXGI

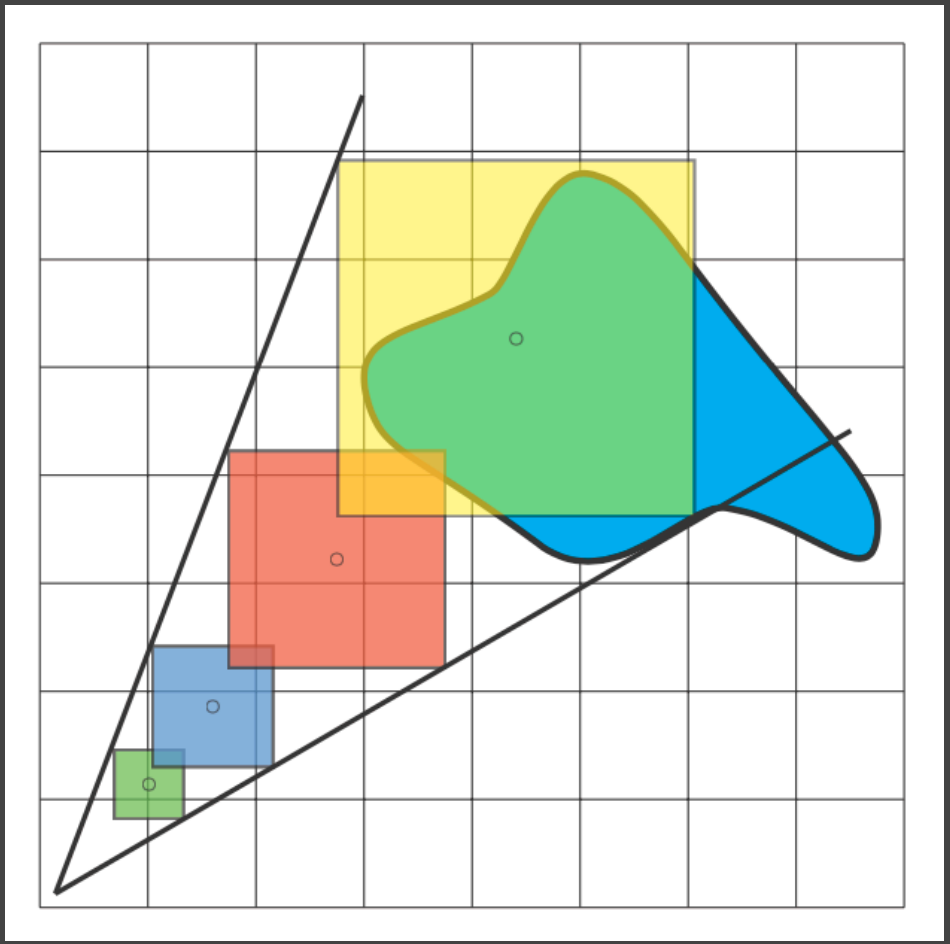
- Pass 1 from the light
  - Store the incident and normal distributions in each voxel
  - Update on the hierarchy



VXGI

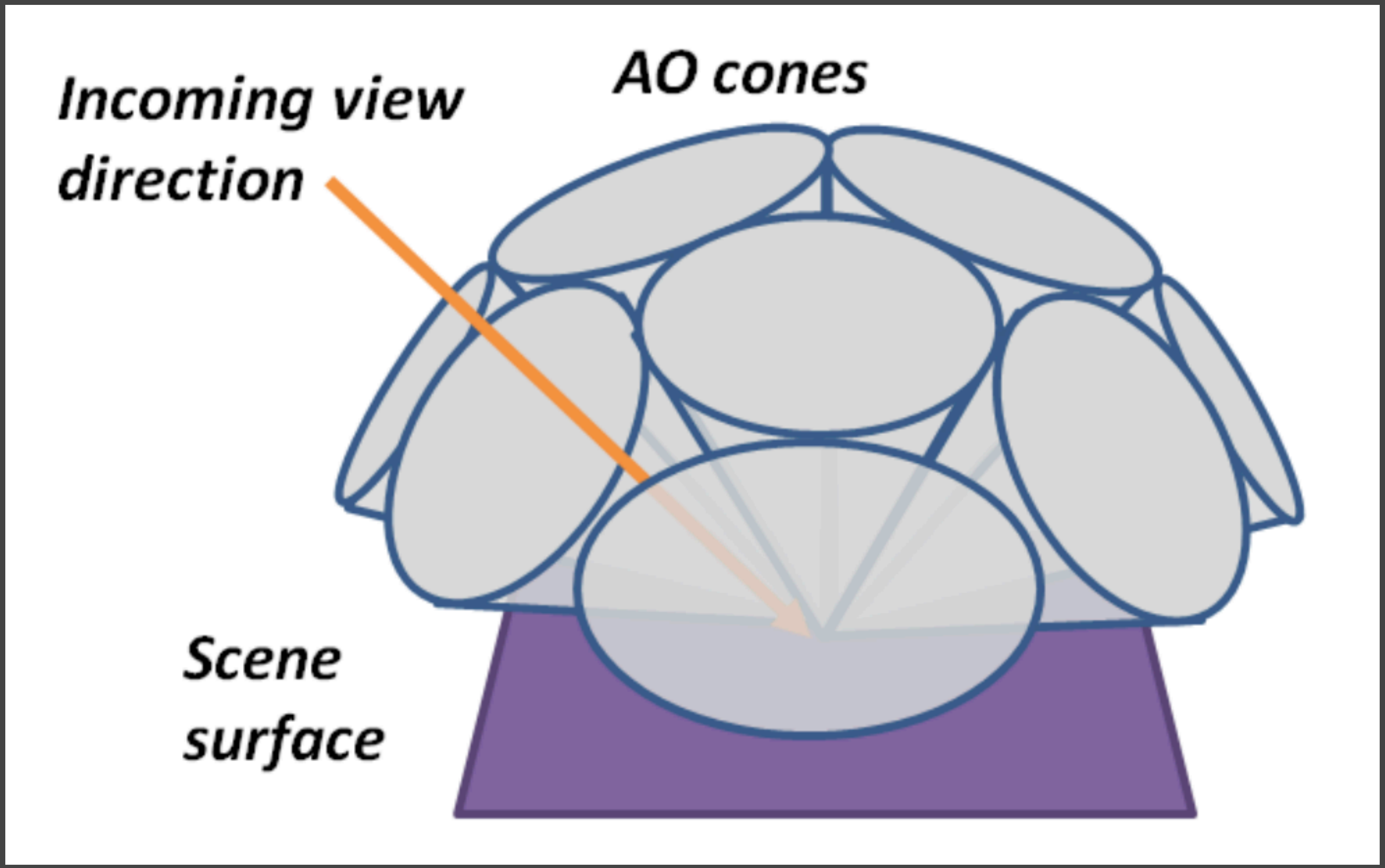
- Pass 2 from the camera
  - For glossy surfaces, trace 1 cone toward the reflected direction
  - Query the hierarchy based on the (growing) size of the cone





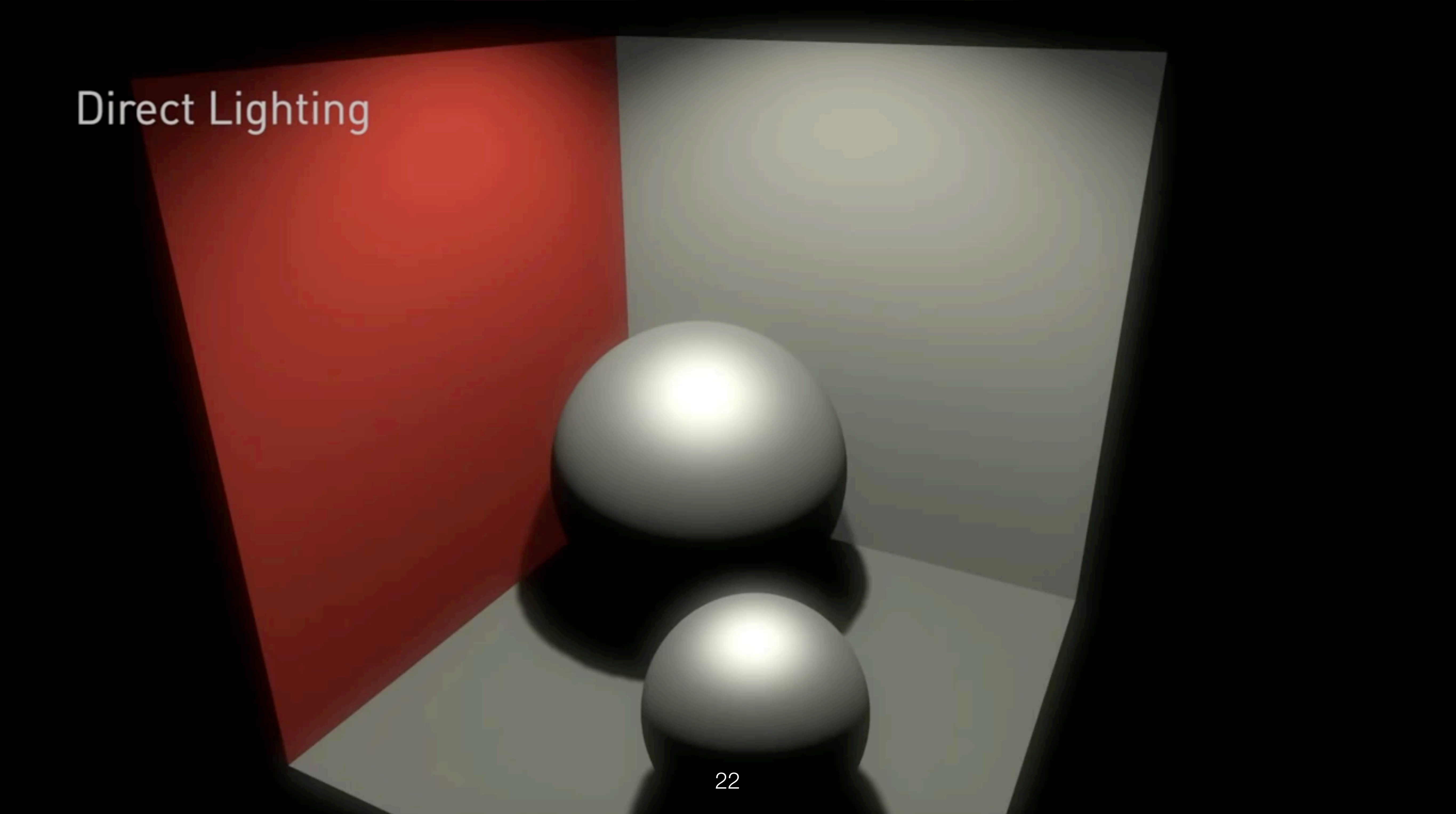
VXGI

- For diffuse, trace several cones (e.g. 8)



VXGI

- Pretty good results, close to ray tracing



Questions?

Today

- Finishing up
  - Light Propagation Volumes (LPV)
  - Voxel Global Illumination (VXGI)
- Real-Time Global Illumination (screen space)
  - Screen Space Ambient Occlusion (SSAO)
  - Screen Space Directional Occlusion (SSDO)
  - Screen Space Reflection (SSR)

GI in Screen Space

- What is "screen space"?
  - Using information only from "the screen"
  - In other words, post processing on existing renderings

[Xin et al. Lightweight Bilateral Convolutional Neural Networks for Interactive Single bounce Diffuse Indirect Illumination]

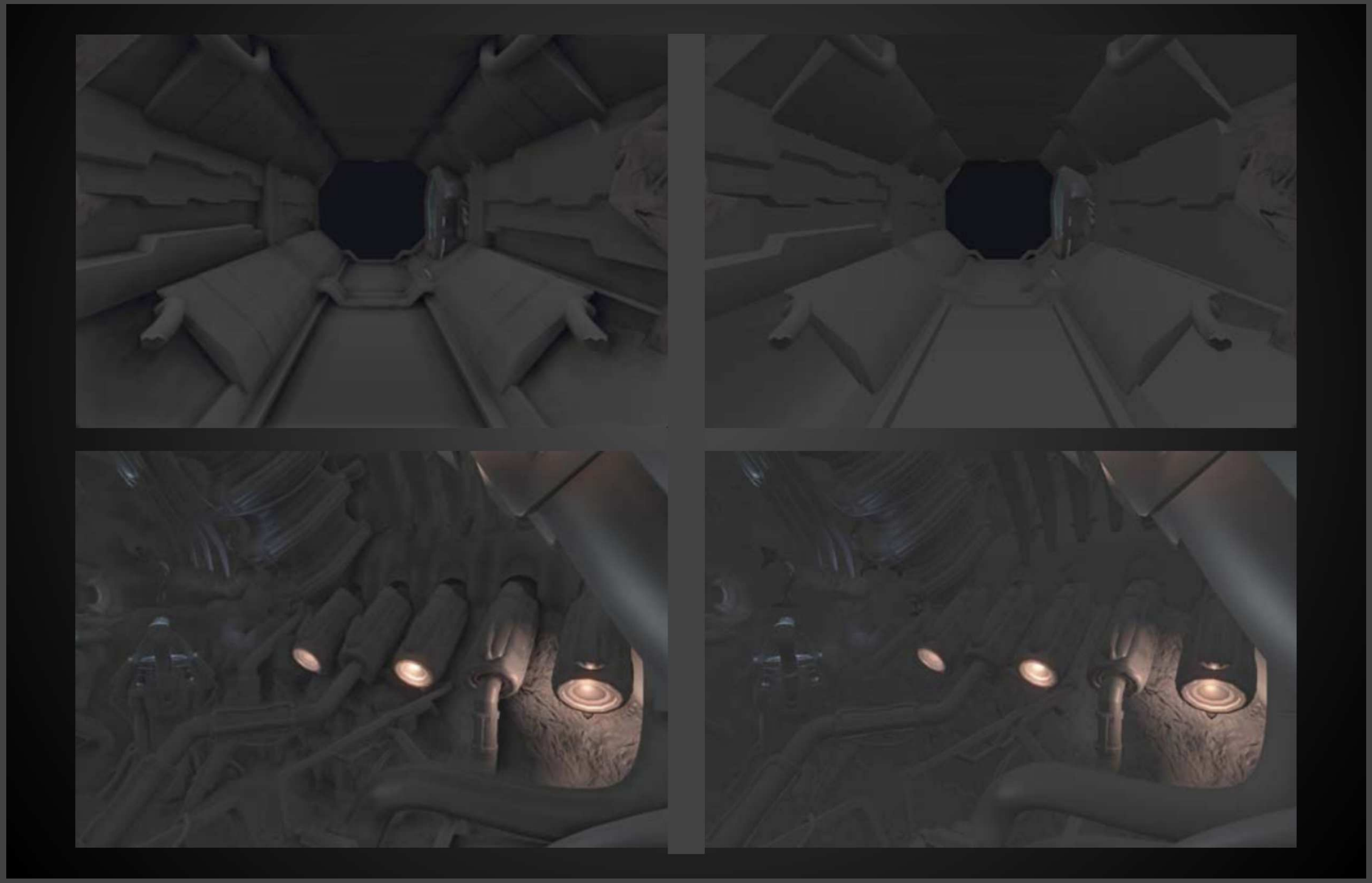
Screen Space Ambient Occlusion (SSAO)

Screen Space Ambient Occlusion

- First introduced by Crytek again

Screen Space Ambient Occlusion

- Why AO?
  - Cheap to implement
  - But enhances the sense of relative positions



[From CryEngine 2]

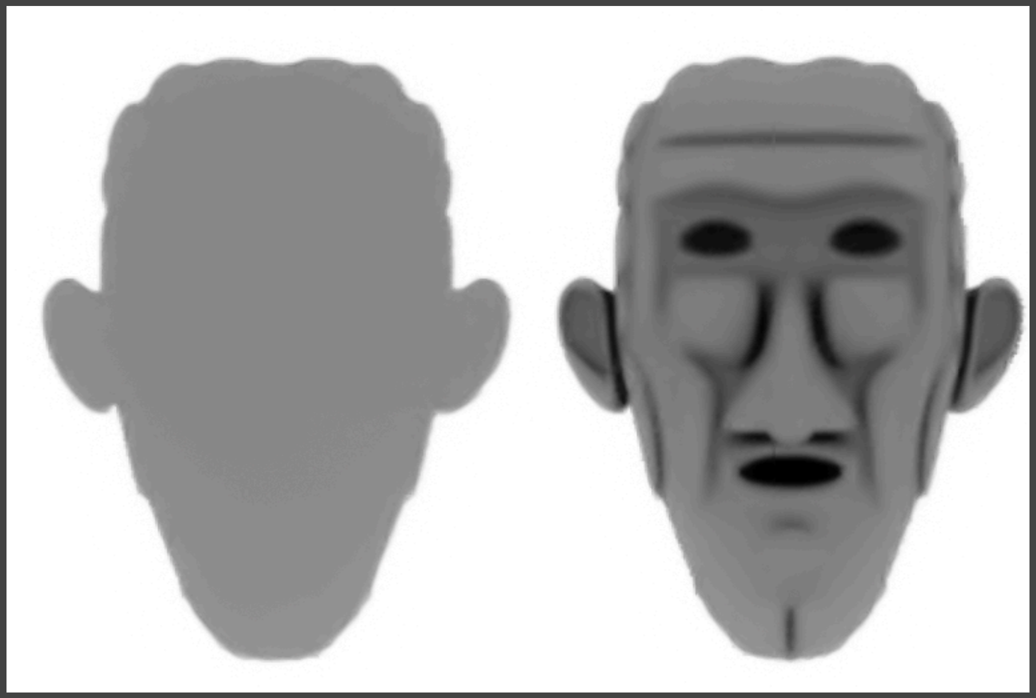
Screen Space Ambient Occlusion

- What is SSAO?
  - An approximation of global illumination
  - In screen space

- Key idea 1
  - We don't know the incident indirect lighting
  - Let's assume it is constant
    (for all shading points, from all directions)
  - Sounds familiar to you?

Screen Space Ambient Occlusion

- Key idea 2 & 3
  - Considering different visibility (towards all directions) at different shading points (why?)



Ambient term from Phong

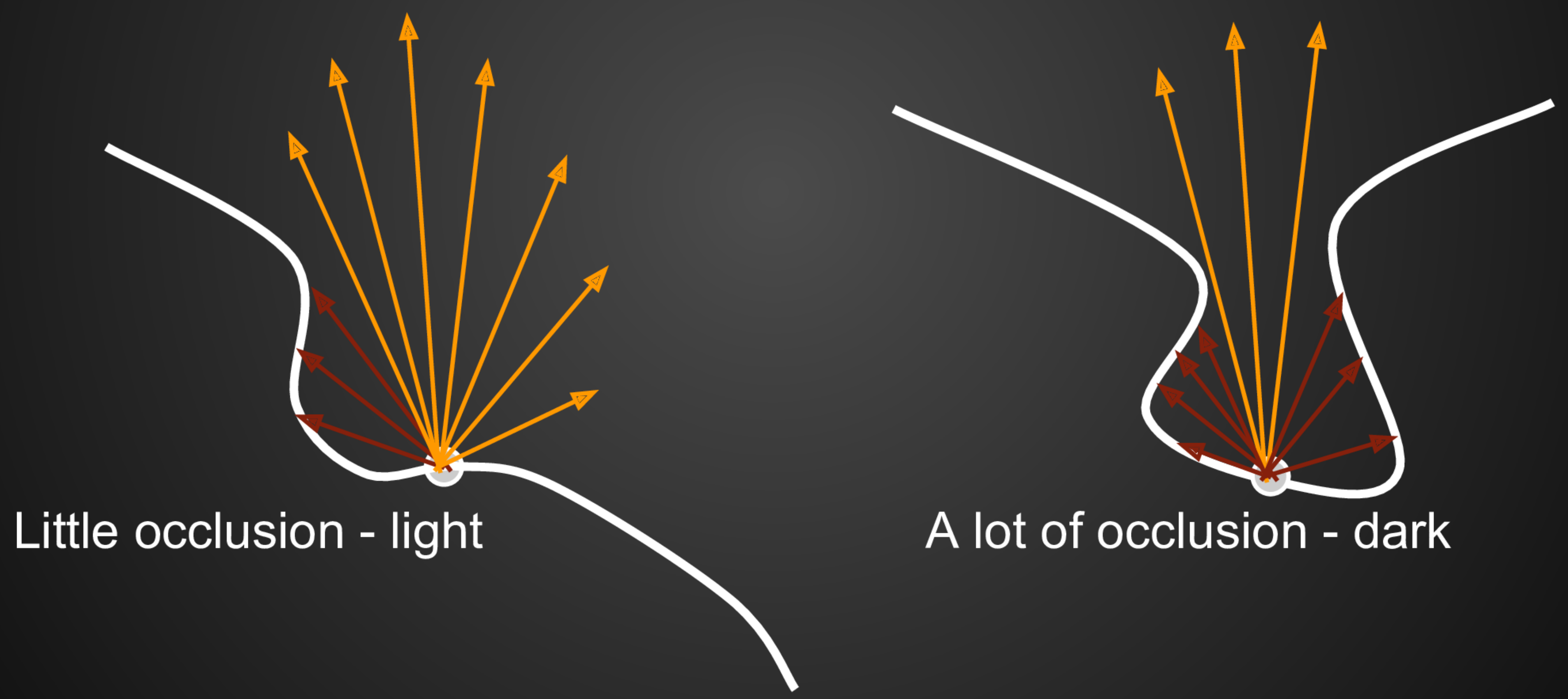
Ambient Occlusion

[Autodesk 3ds Max]

Ambient Occlusion

  - Also, assuming diffuse materials

Ambient occlusion



Screen Space Ambient Occlusion

- Theory
  - Still, everything starts from the rendering equation

$$L_o(\mathrm{p}, \omega_o) = \int_{\Omega^+} L_i(\mathrm{p}, \omega_i) f_r(\mathrm{p}, \omega_i, \omega_o) V(\mathrm{p}, \omega_i) \cos\theta_i \,\mathrm{d}\omega_i$$

  - And again, from “the RTR approximation / equation”!

$$\int_{\Omega} f(x) g(x) \,\mathrm{d}x \approx \frac{\int_{\Omega_G} f(x) \,\mathrm{d}x}{\int_{\Omega_G} \mathrm{d}x} \cdot \int_{\Omega} g(x) \,\mathrm{d}x$$

Screen Space Ambient Occlusion

- Separating the visibility term

$$L_o^{\text{indir}}(\mathrm{p}, \omega_o) \approx \frac{\int_{\Omega^+} V(\mathrm{p}, \omega_i) \cos\theta_i \, \mathrm{d}\omega_i}{\int_{\Omega^+} \cos\theta_i \, \mathrm{d}\omega_i} \cdot \int_{\Omega^+} L_i^{\text{indir}}(\mathrm{p}, \omega_i) f_r(\mathrm{p}, \omega_i, \omega_o) \cos\theta_i \, \mathrm{d}\omega_i$$

$$\square \triangleq k_A = \frac{\int_{\Omega^+} V(\mathrm{p}, \omega_i) \cos\theta_i \, \mathrm{d}\omega_i}{\pi}$$

(the weight-averaged visibility $\overline{V}$ from all directions)

$$\square = L_i^{\text{indir}}(p) \cdot \frac{\rho}{\pi} \cdot \pi = L_i^{\text{indir}}(p) \cdot \rho$$

(constant for AO)

Screen Space Ambient Occlusion

- A deeper understanding 1

$$\int_{\Omega} f(x)g(x)\,\mathrm{d}x \approx \frac{\int_{\Omega_G} f(x)\,\mathrm{d}x}{\int_{\Omega_G}\,\mathrm{d}x} \cdot \int_{\Omega} g(x)\,\mathrm{d}x$$

$$= \overline{f(x)} \cdot \int_{\Omega} g(x)\,\mathrm{d}x$$

(the average f(x) in
the support of G)

- Also, in AO, the approximation is accurate
(const $G = L \cdot f_r$)

Screen Space Ambient Occlusion

- A deeper understanding 2

$$\int_{\Omega} f(x)g(x)\,\mathrm{d}x \approx \frac{\int_{\Omega_G} f(x)\,\mathrm{d}x}{\int_{\Omega_G}\mathrm{d}x} \cdot \int_{\Omega} g(x)\,\mathrm{d}x$$

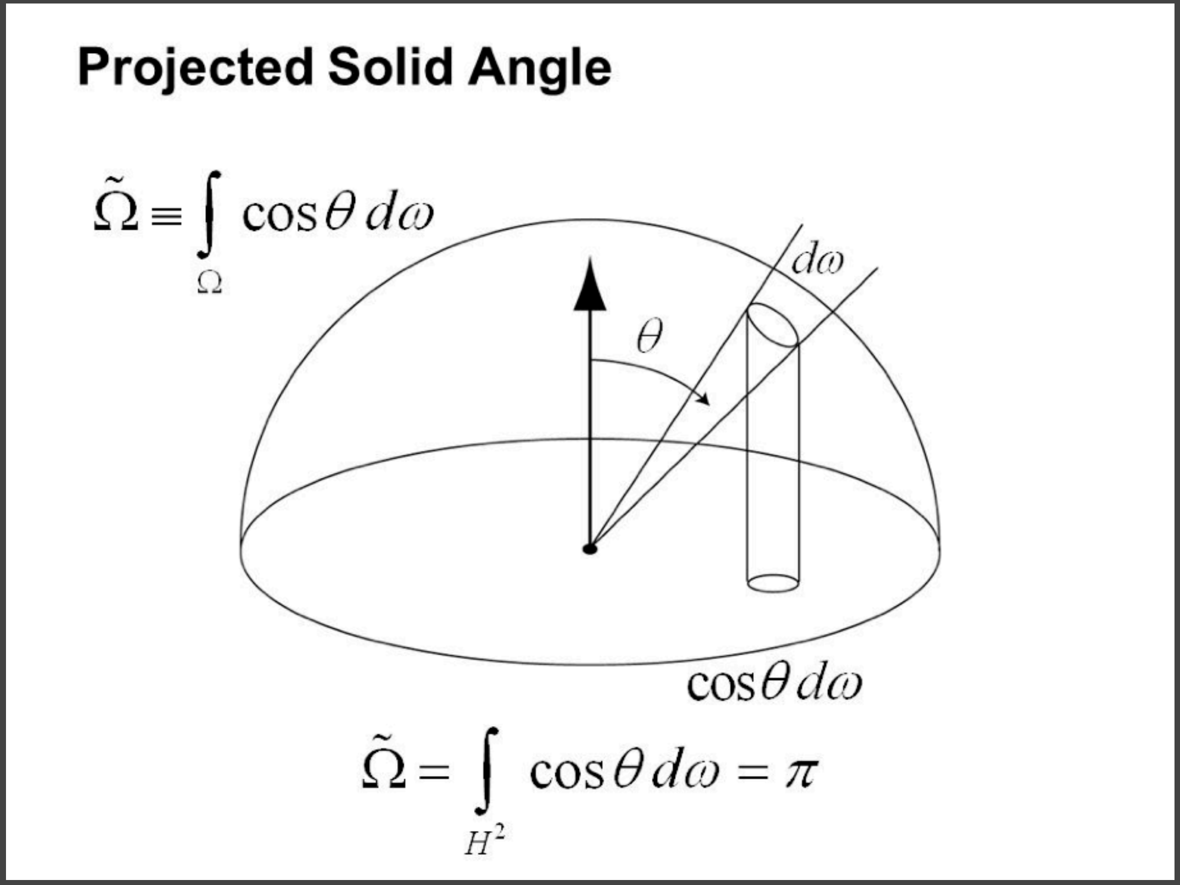
  - Why can we take the cosine term with $d\omega_i$?

$$L_o^{\text{indir}}(\mathrm{p},\omega_o) \approx \frac{\int_{\Omega+} V(\mathrm{p},\omega_i)\cos\theta_i\,\mathrm{d}\omega_i}{\int_{\Omega+}\cos\theta_i\,\mathrm{d}\omega_i} \cdot$$

$$\int_{\Omega+} L_i^{\text{indir}}(\mathrm{p},\omega_i) f_r(\mathrm{p},\omega_i,\omega_o)\cos\theta_i\,\mathrm{d}\omega_i$$

Screen Space Ambient Occlusion

- Why can we take the cosine term with $d\omega_i$?
- Projected solid angle $dx_\perp = \cos\theta_i \, \mathrm{d}\omega_i$
  - Unit hemisphere -> unit disk
  - Integration of projected solid angle == the area of the unit disk == π



Screen Space Ambient Occlusion

- Actually, a much simpler understanding
  - Uniform incident lighting — L_i is constant
  - Diffuse BSDF — $f_r = \frac{\rho}{\pi}$ is also constant
  - Therefore, taking both out of the integral:

$$L_o(\mathrm{p}, \omega_o) = \int_{\Omega^+} L_i(\mathrm{p}, \omega_i) f_r(\mathrm{p}, \omega_i, \omega_o) V(\mathrm{p}, \omega_i) \cos\theta_i \, \mathrm{d}\omega_i$$

$$= \frac{\rho}{\pi} \cdot L_i(p) \cdot \int_{\Omega^+} V(\mathrm{p}, \omega_i) \cos\theta_i \, \mathrm{d}\omega_i$$

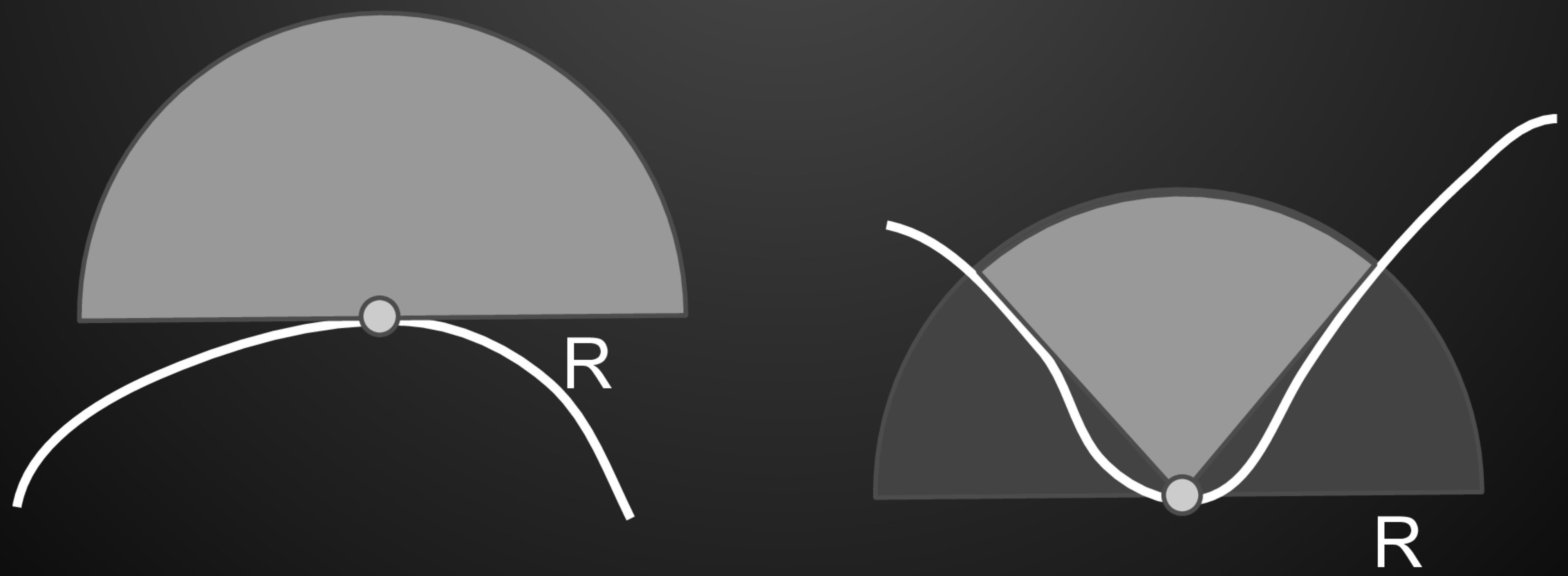
How to compute the occlusion values $k_A(p)$ in real time

- In object space
  - Raycasting against geometry
  - Slow, requires simplifications and/or spatial data structures
  - Depends of scene complexity
- In screen space
  - Done in a post-rendering pass
  - No pre-processing required
  - Doesn't depend on scene complexity
  - **Simple**
  - Not physically accurate

Ambient occlusion approximation: limited radius

Limit to local occlusion in a hemisphere of radius R.

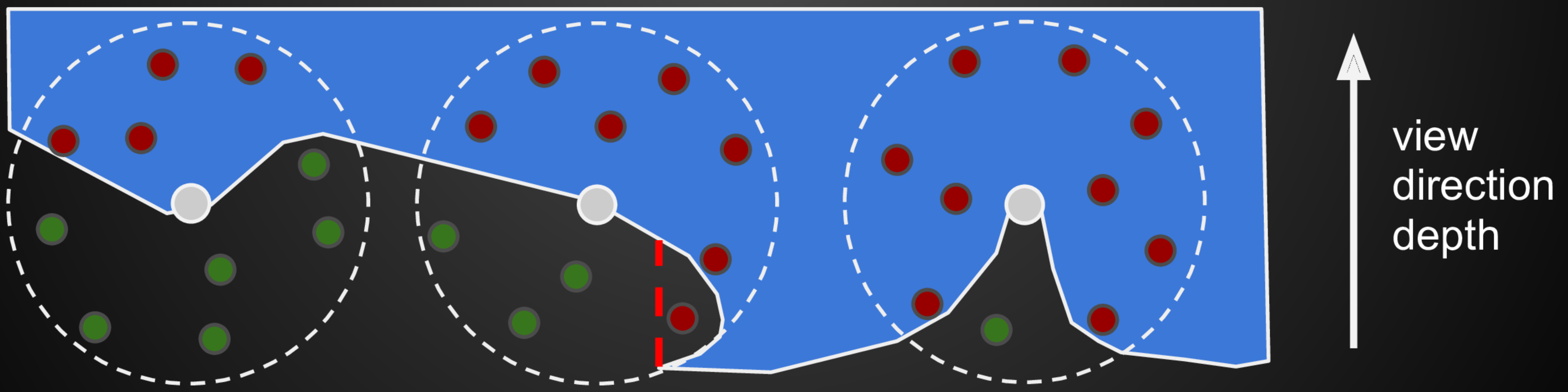
More efficient and works better in enclosed areas such as indoors, that would be fully occluded otherwise.



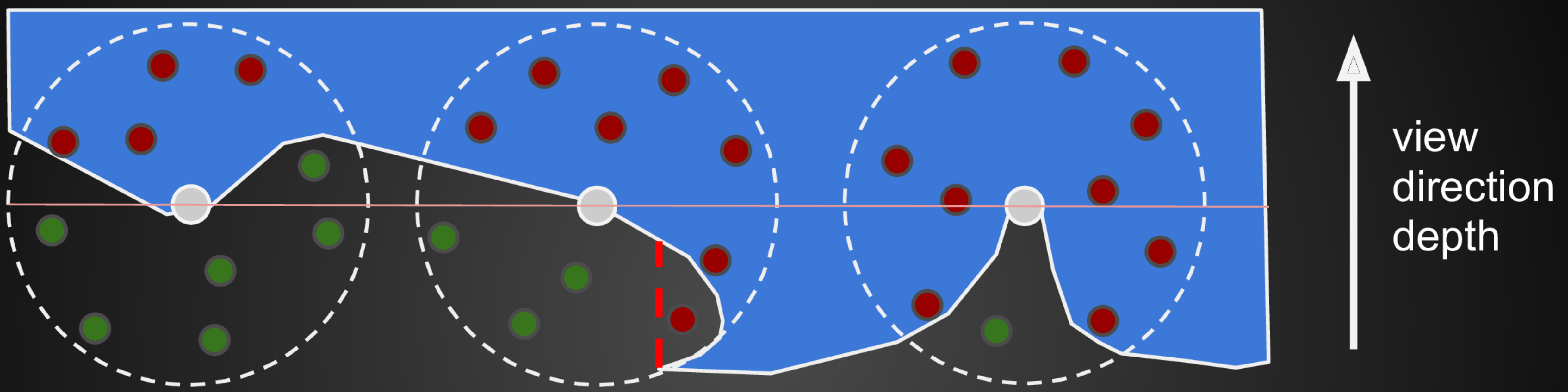
SSAO: Ambient occlusion using the z-buffer

Use the readily available depth buffer as an approximation of the scene geometry.

Take samples in a sphere around each pixel and test against buffer.



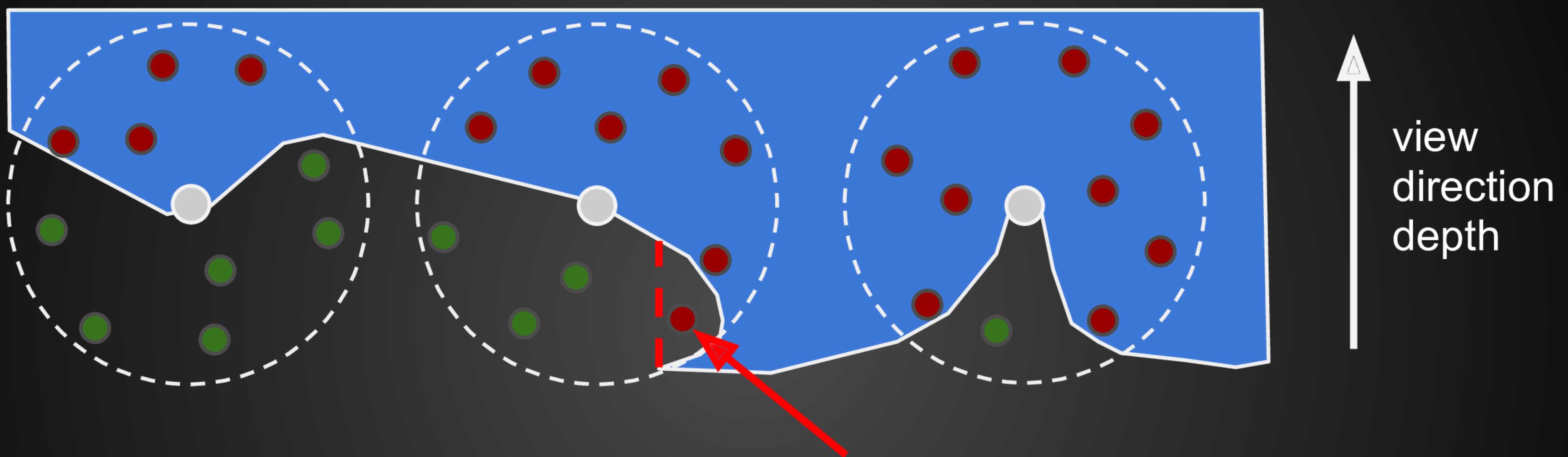
SSAO:
Ambient occlusion using the z-buffer



If more than half of the samples are inside, AO is applied, amount depending on ratio of samples that pass and fail depth test.

Uses sphere instead of hemisphere, since normal information isn't available.

SSAO:
Ambient occlusion using the z-buffer



Approximation of the scene geometry, some fails are incorrect. The one behind the red line for example. False occlusions.

Samples are not weighted by cos(theta), so not physically accurate, but looks convincing.

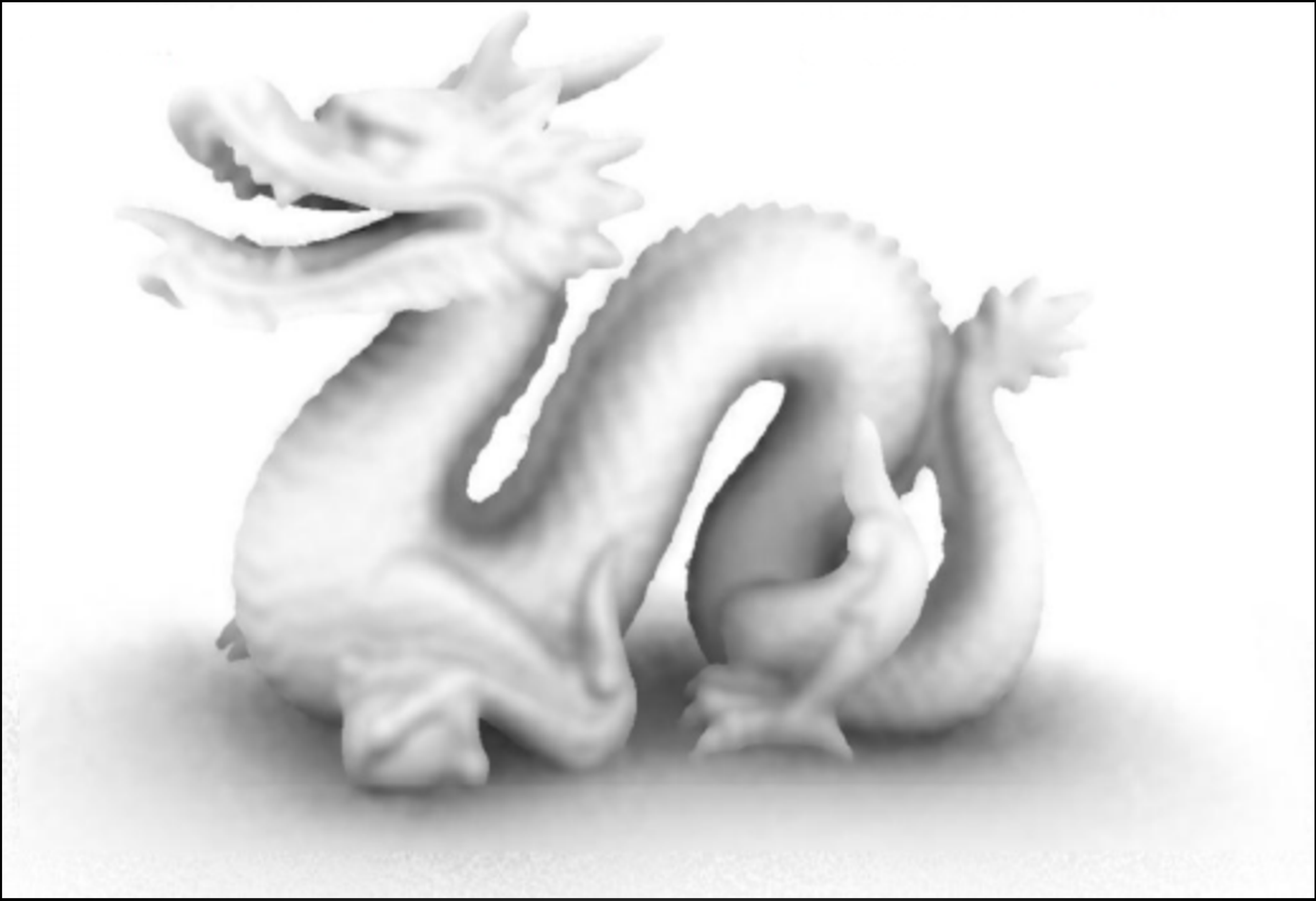
SSAO: False occlusions, halos

No SSAO

SSAO

Choosing samples

- More samples -> greater accuracy
- Many samples are needed for a good result, but for performance only about 16 samples are used.
- Positions from randomized texture to avoid banding.
- Noisy result, blurred with edge preserving blur.

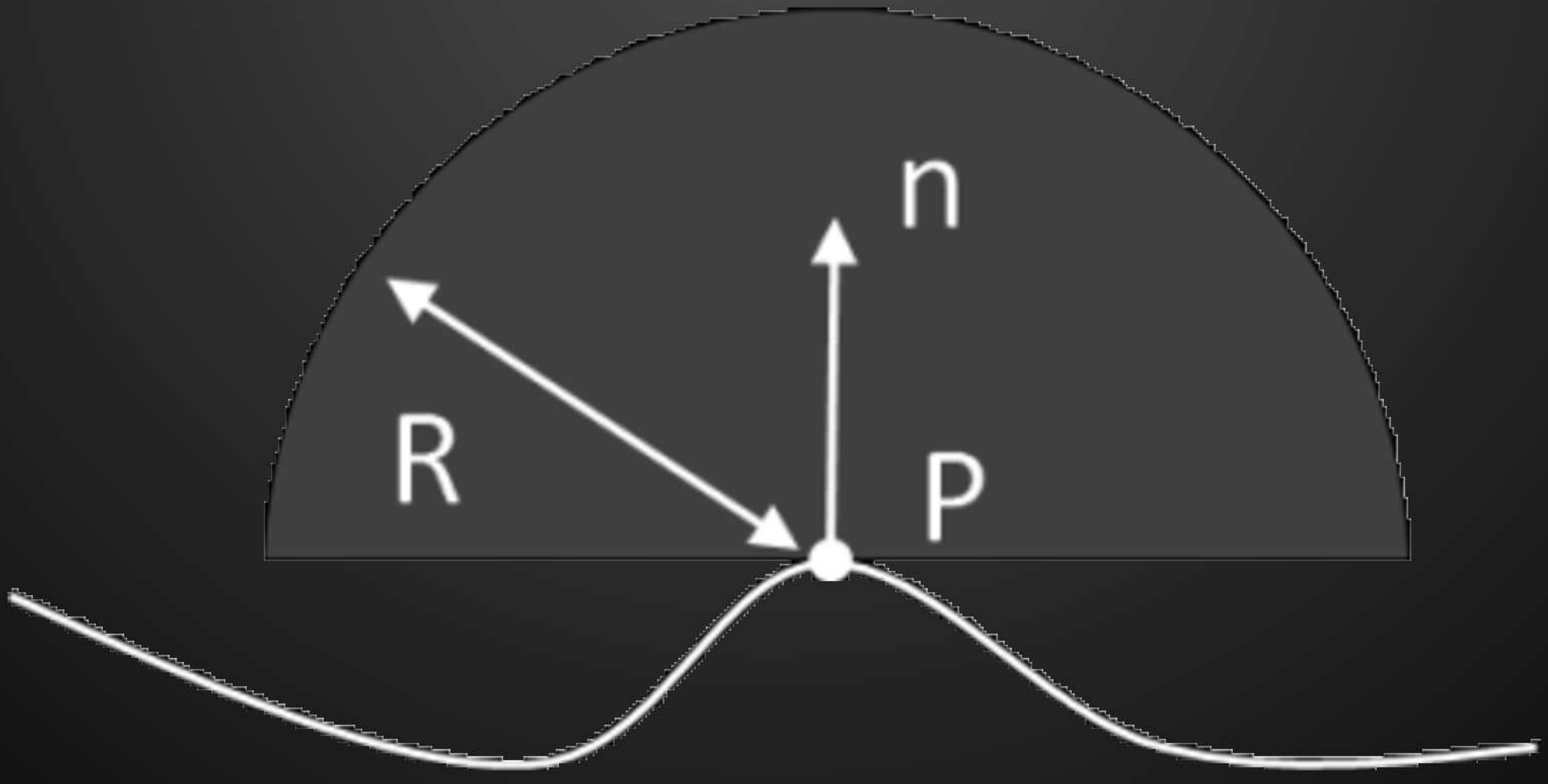


Horizon based ambient occlusion: HBAO

Also done in screen space.

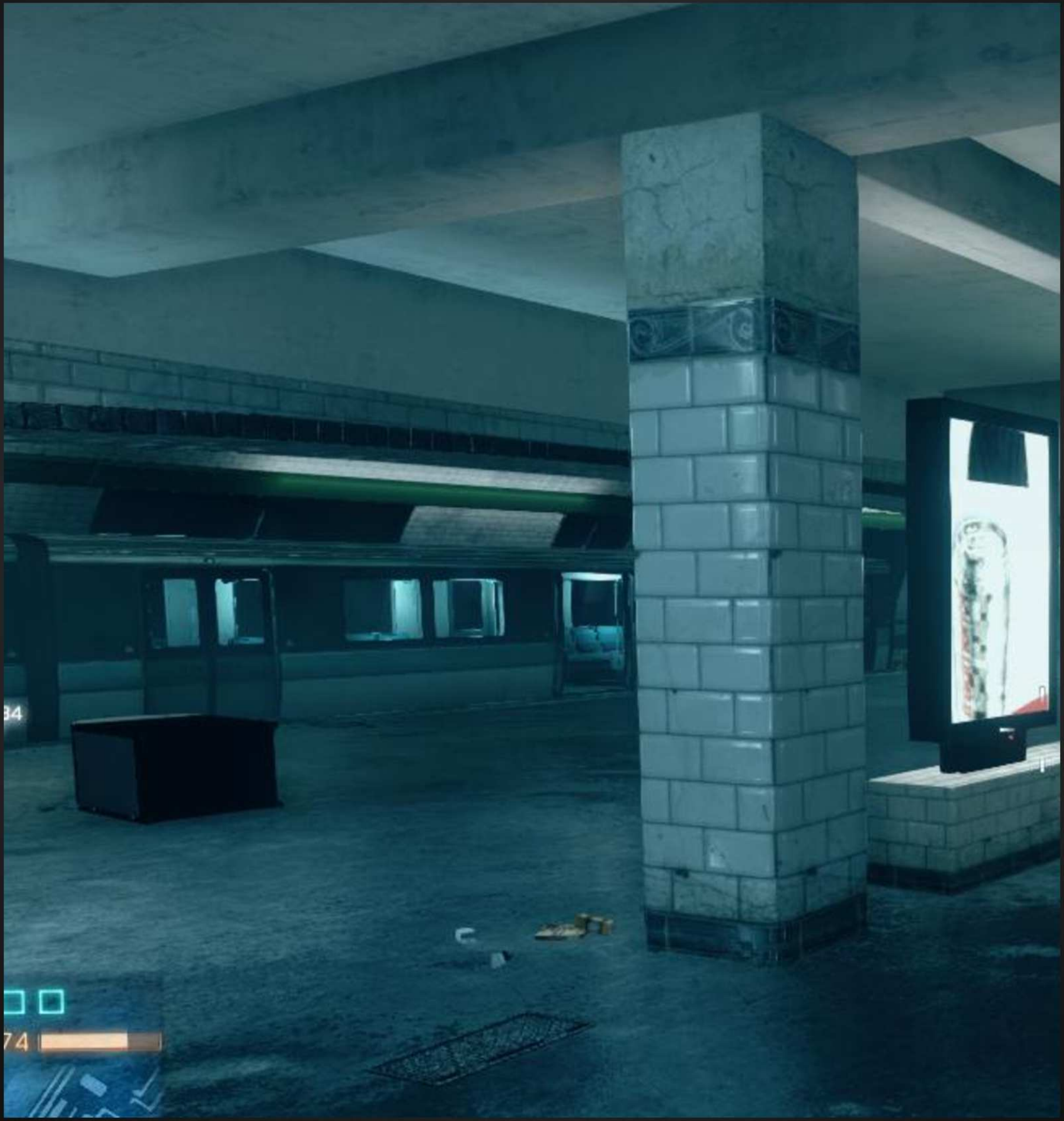
Approximates ray-tracing the depth buffer.

Requires that the normal is known, and only samples in a hemisphere.

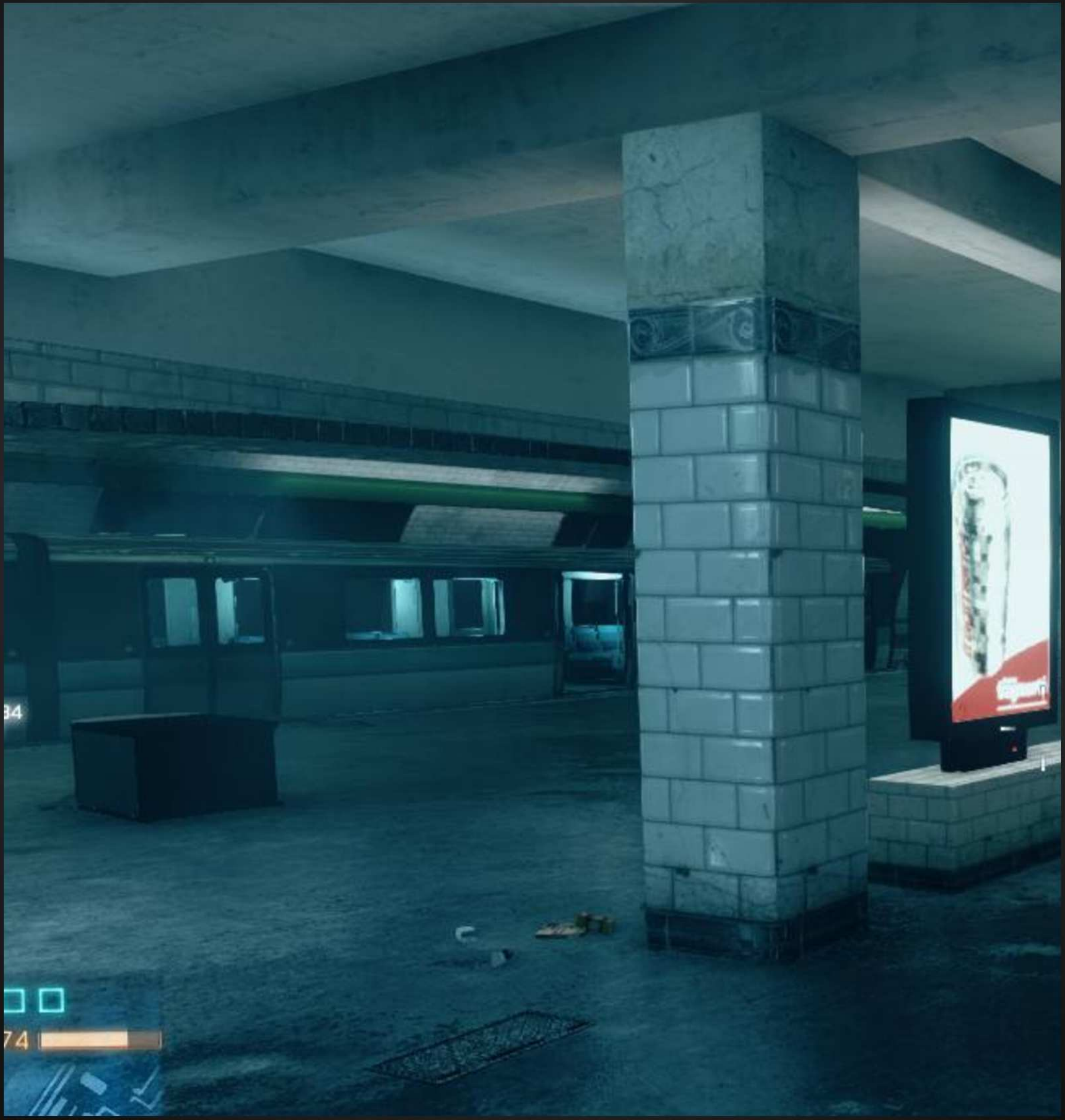


Battlefield 3 - No SSAO

Battlefield 3 - SSAO



Battlefield 3 - HBAO



Battlefield 3 - SSAO

Battlefield 3 - HBAO

Questions?

Next Lecture

- Real-time global illumination cont.
  - Screen Space Reflection (SSR)

[Onmyoji by NetEase]

Thank you!

(Many SSAO slides courtesy of the
Chalmers University of Technology)