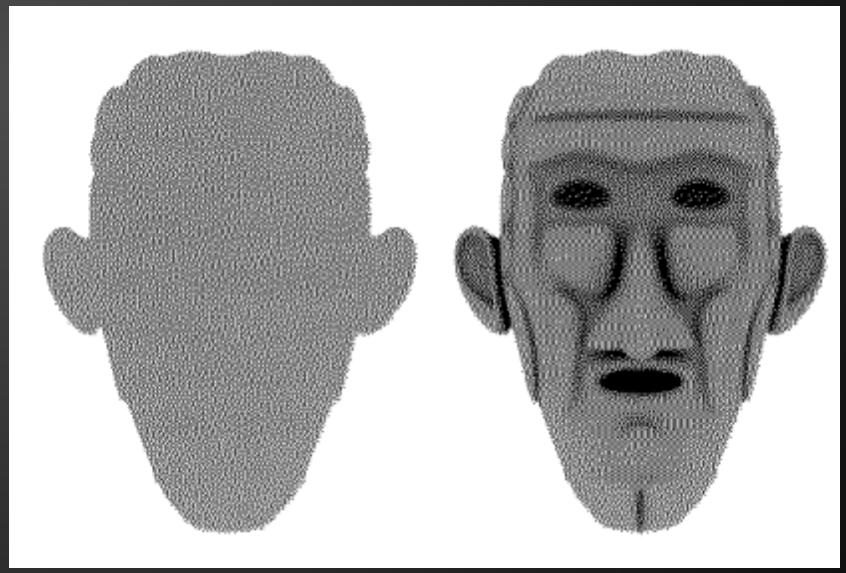
Screen Space Ambient Occlusion

Daniel Kvarfordt & Benjamin Lillandt

Ambient light

- Same from all directions.
- Lambertian shading doesn't show form well.

- Need shadows to see form.
- Global illumination can be used, but is expensive.

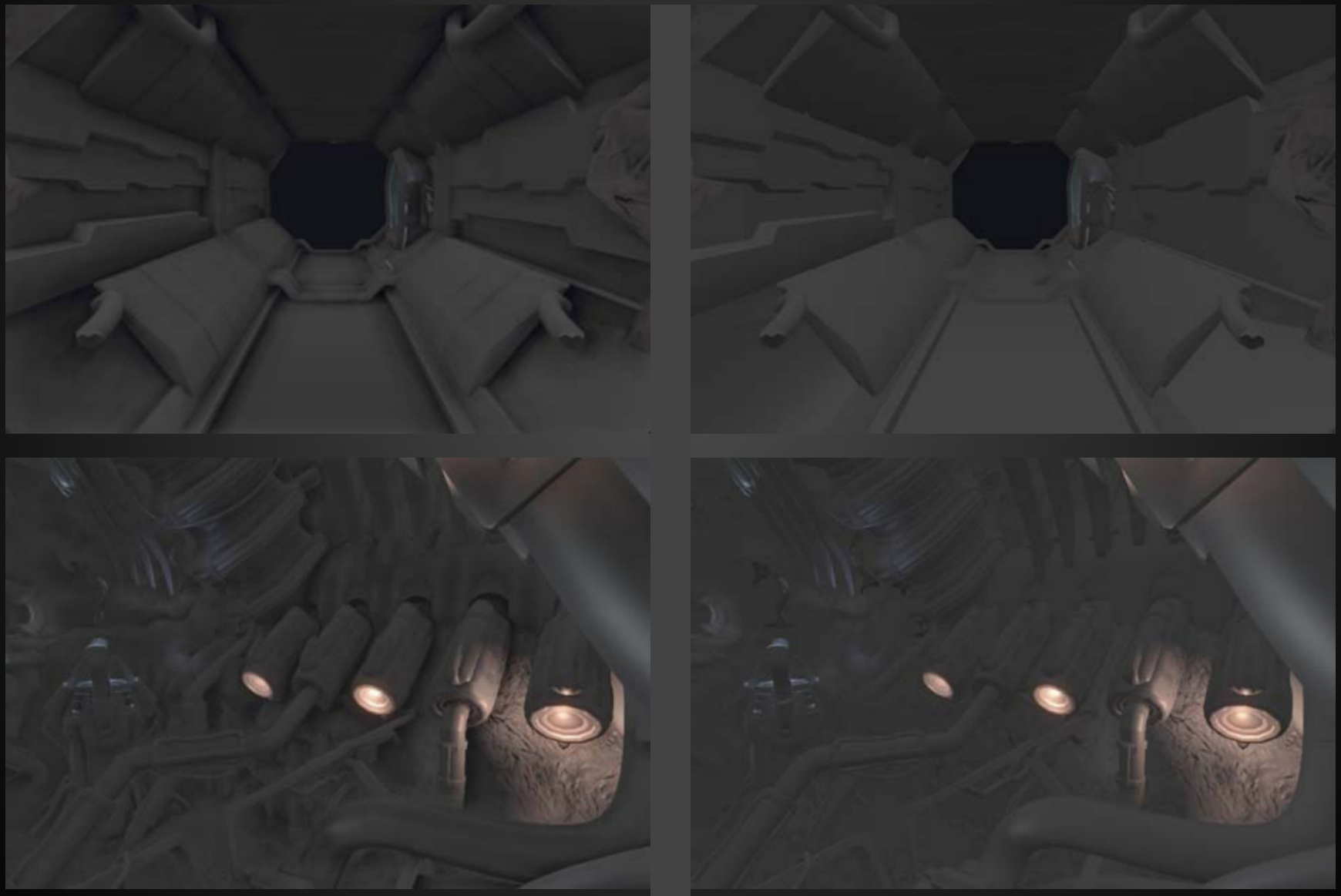


Ambient occlusion

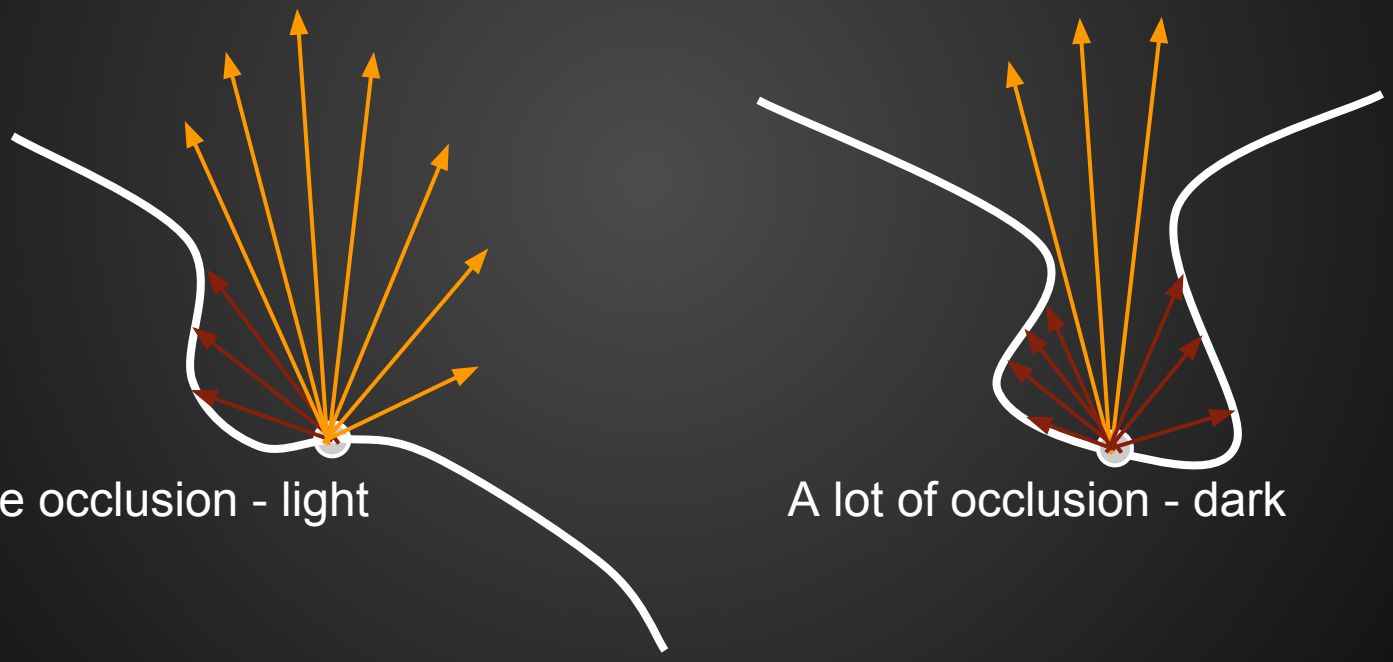
AO is the amount of ambient light that is stopped from reaching a point.

Calculate diffuse shadows from ambient light.

Examples: Cryengine 2



Ambient occlusion



Little occlusion - light

A lot of occlusion - dark

Irradiance from ambient light without ambient occlusion

$$E(\boldsymbol{p}, \boldsymbol{n}) = \int_{\Omega} L_A cos\theta_i d\omega_i = \pi L_A$$

E(p,n) surface irradiance

Omega hemisphere at p

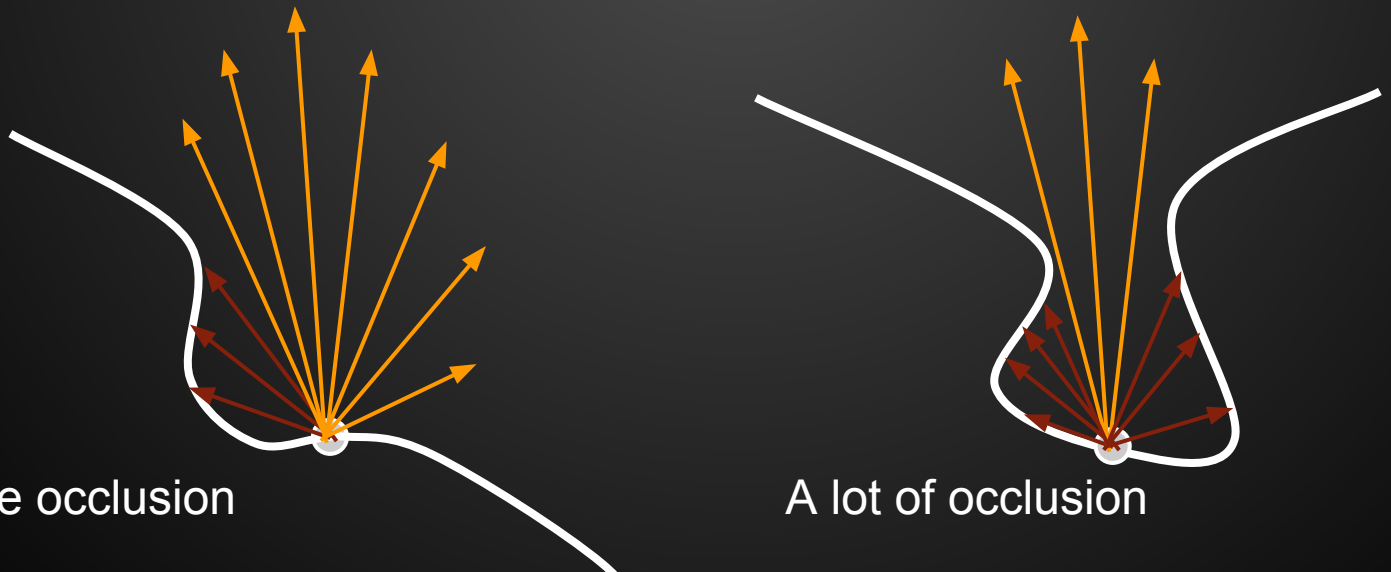
L_A ambient incoming irradiance

Constant value for all points and normals

Irradiance from ambient light with ambient occlusion

$$E(\mathbf{p}, \mathbf{n}) = L_A \int_\Omega v(\mathbf{p}, \mathbf{l}) cos\theta_i d\omega_i$$

Visibility v(p,l) is 0 or 1 depending on if the direction l is occluded or not.



Little occlusion

A lot of occlusion

Ambient occlusion value (factor)

$$k_A(\mathbf{p}) = \frac{1}{\pi} \int_{\Omega} v(\mathbf{p}, \mathbf{l}) cos\theta_i d\omega_i$$

Darkening factor k_A between 0 (fully occluded) and 1 (fully visible).

Using the ambient occlusion value k_A, the ambient irradiance equation becomes simple:

$$E(\mathbf{p}, \mathbf{n}) = k_A(\mathbf{p})\pi L_A$$

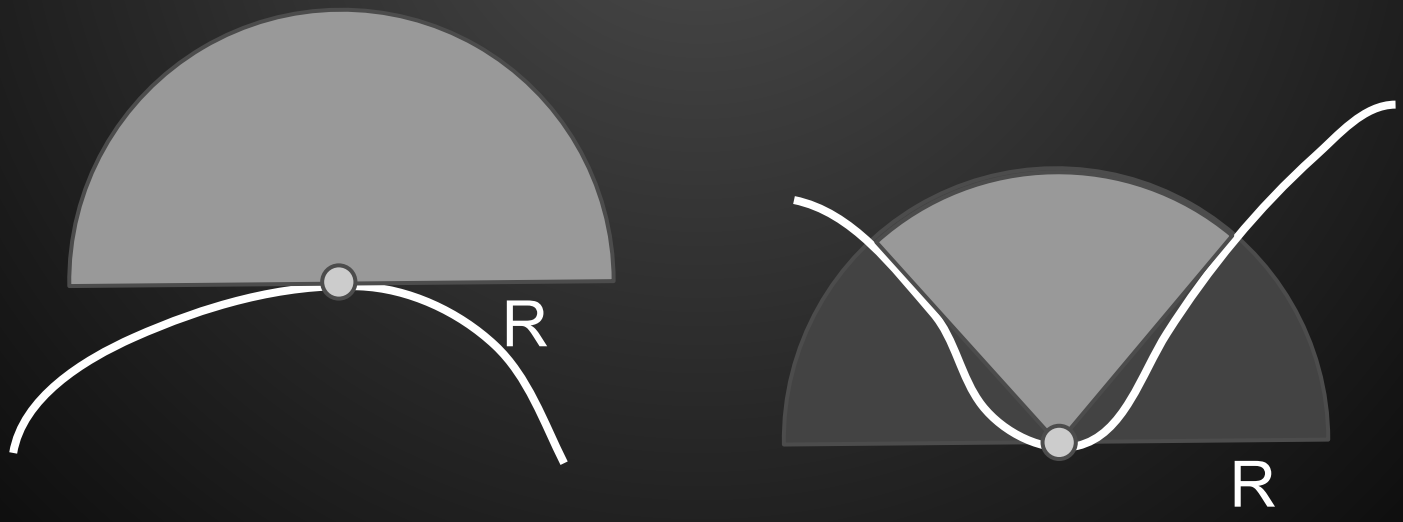
How to compute the occlusion values $k_A(p)$ in real time

- 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.

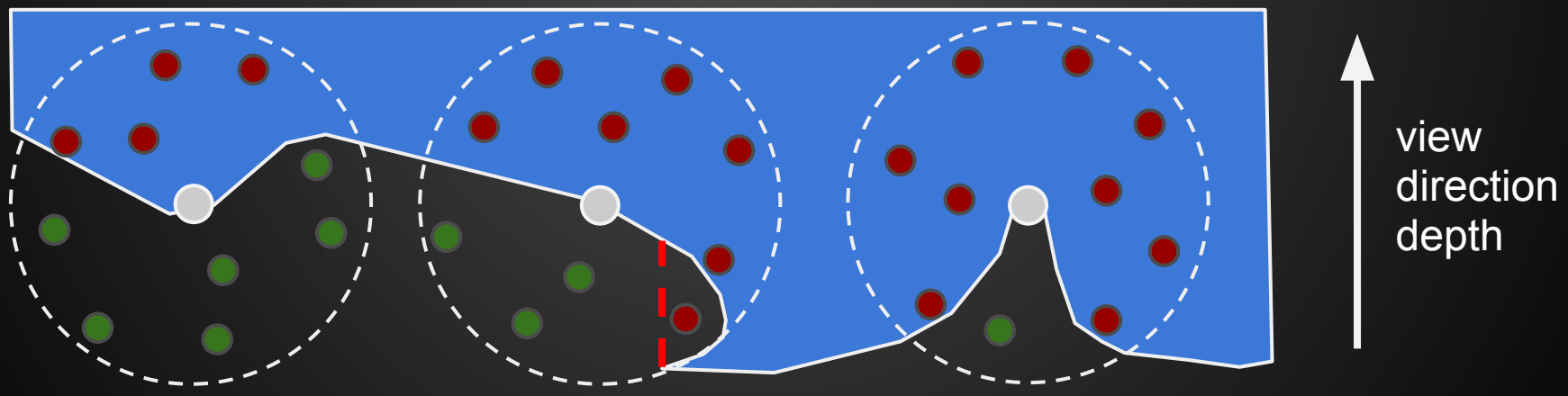
More efficient and works better in enclosed areas such as indoors, that would be fully occluded otherwise.



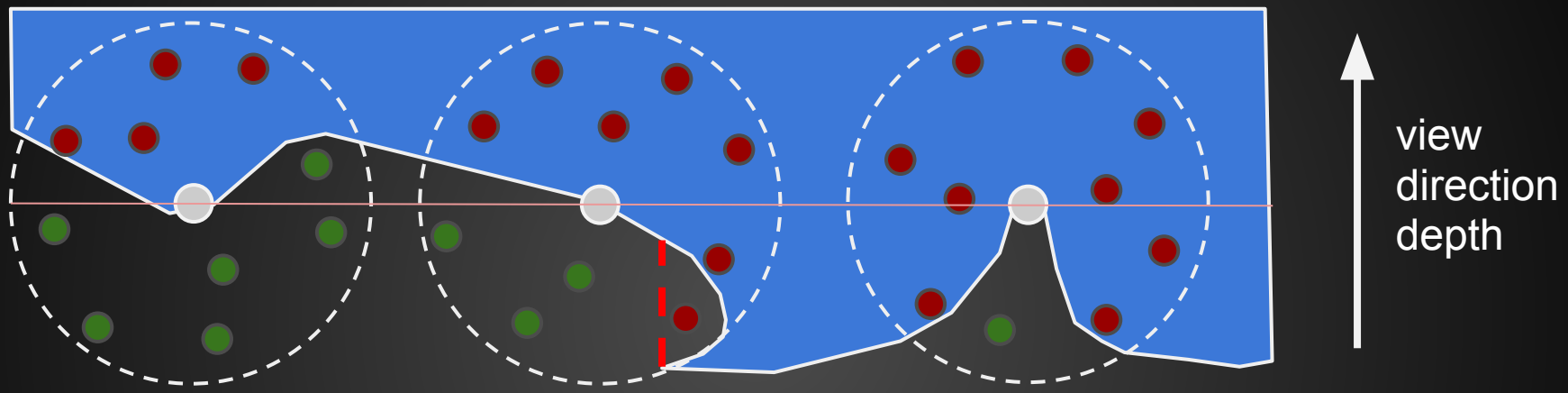
SSAO:
Ambient occlusion using the z-buffer

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.



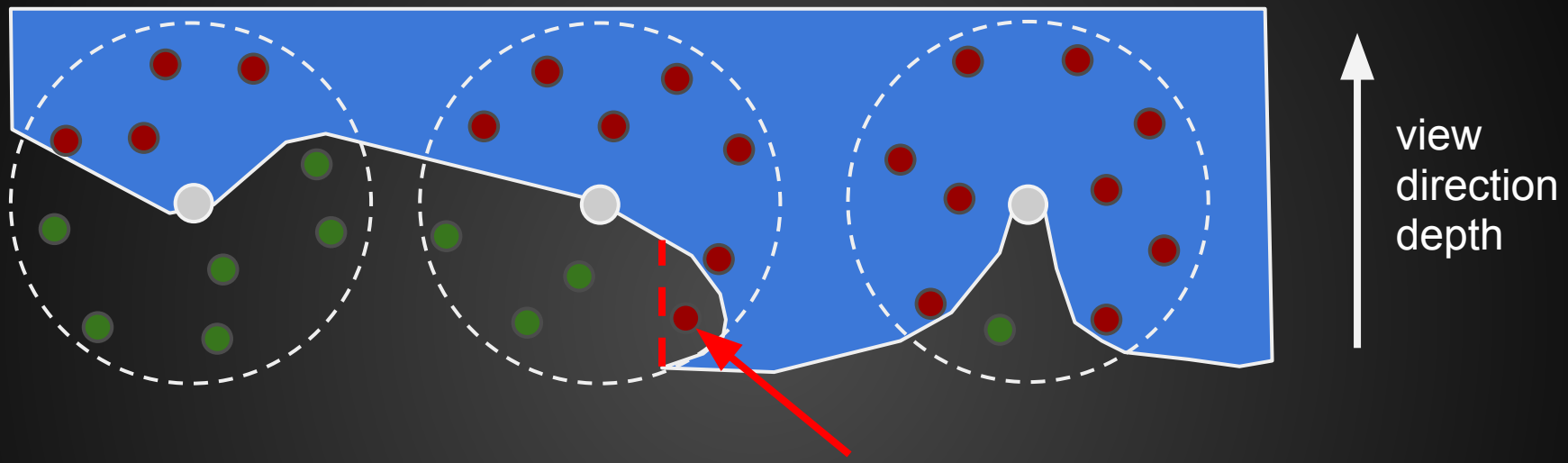
SSAO:
Ambient occlusion using the z-buffer



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



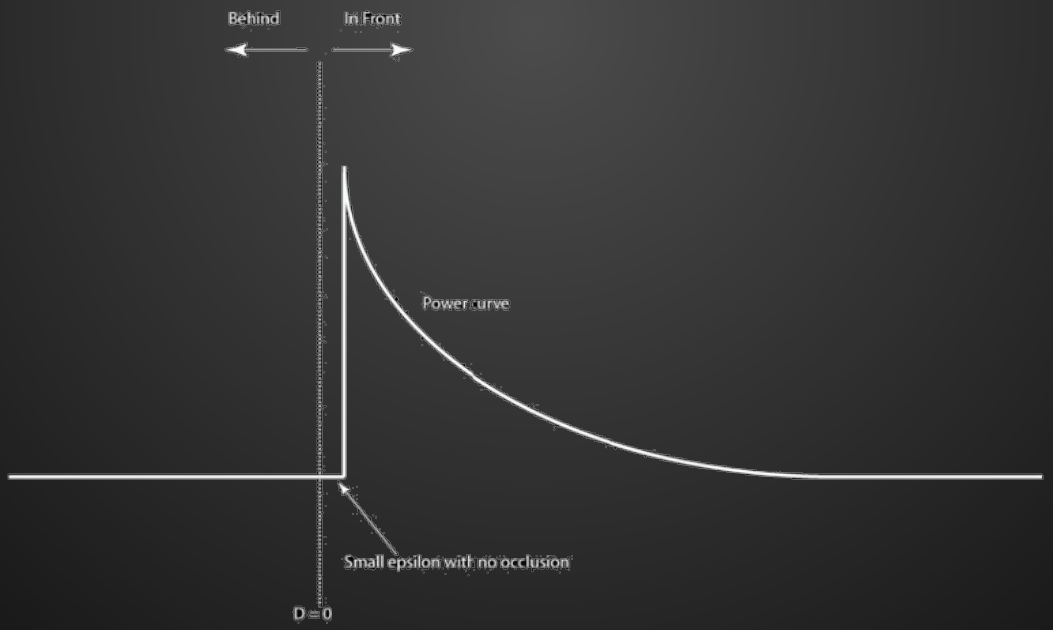
Approximation of the scene geometry, some fails are incorrect. The one behind the red line for example. False occlusions.

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

Occlusion function

Occlusion function used to relate to sample depth delta and distance from the central point

- Negative depth deltas give zero occlusion
- Small positive depth delta produces high occlusion term
- Large positive depth delta tend to zero (calculation happens in screen space)
- Simple exponentials or lookups used



SSAO: False occlusions, halos

No SSAO

SSAO

SSAO: Full geometry not known

SSAO

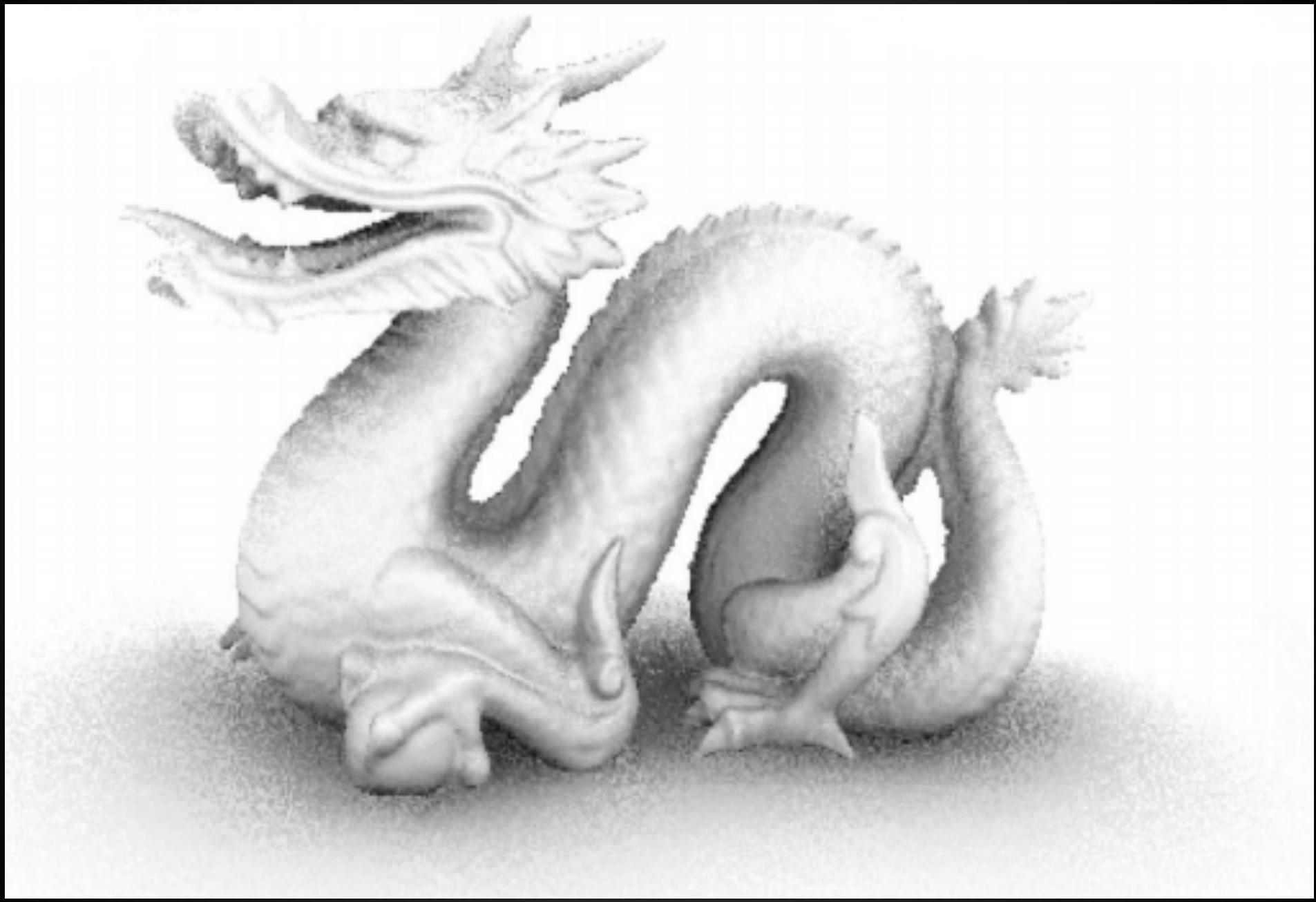
"Real" AO

Can be solved by rendering multiple layers of depth, "Depth peeling".

Images from "Multi-Layer Dual-Resolution Screen-Space Ambient Occlusion" Louis Bavoil, Miguel Sainz 2009

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.

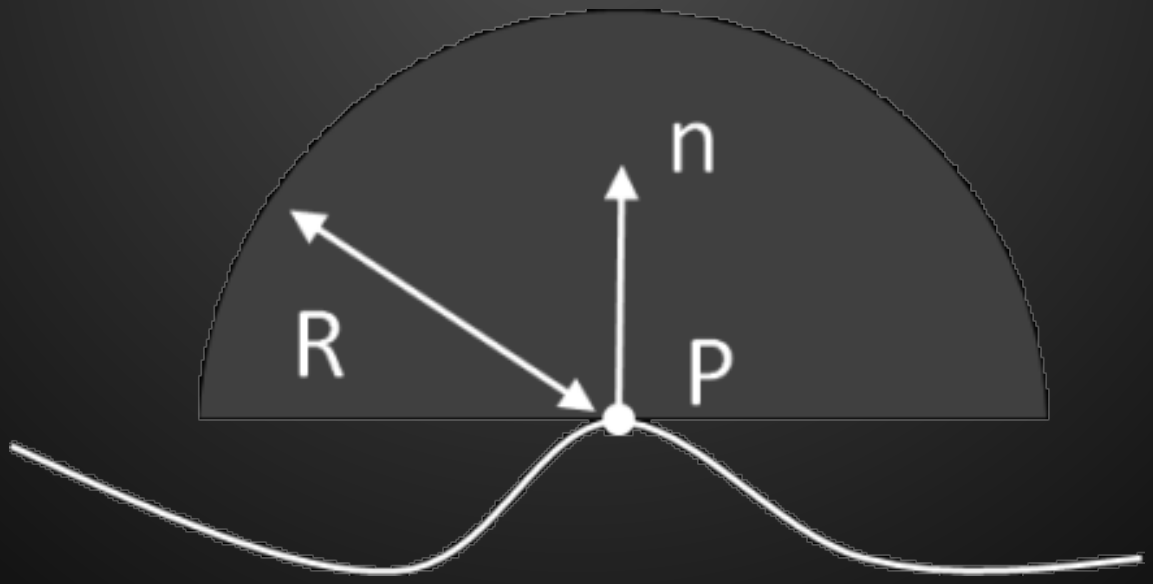


Horizon based ambient occlusion: HBAO

Also done in screen space.

Approximates ray-tracing the depth buffer.

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



Horizon based ambient occlusion: HBAO

Attenuates the contribution for each ray for a more accurate result. (normally based on distance from P to occlusion and angle from normal, *I think*)

$$A = 1 - \frac{1}{2\pi} \int_{\Omega} V(\vec{\omega}) W(\vec{\omega}) d\omega$$

V is the visibility function, 0 or 1.
W is an attenuation function.

Battlefield 3 - No SSAO

Battlefield 3 - SSAO

Battlefield 3 - HBAO

Battlefield 3 - SSAO

Battlefield 3 - HBAO

Half resolution AO

AO is usually low frequency, so SSAO/HBAO is performed at half resolution for significant speedup.

Smart blur is done using full resolution z-buffer to avoid edge bleeding.

Half-res SSAO flickering

Small high frequency geometry like grass can cause flickering / shimmering due to the half resolution.

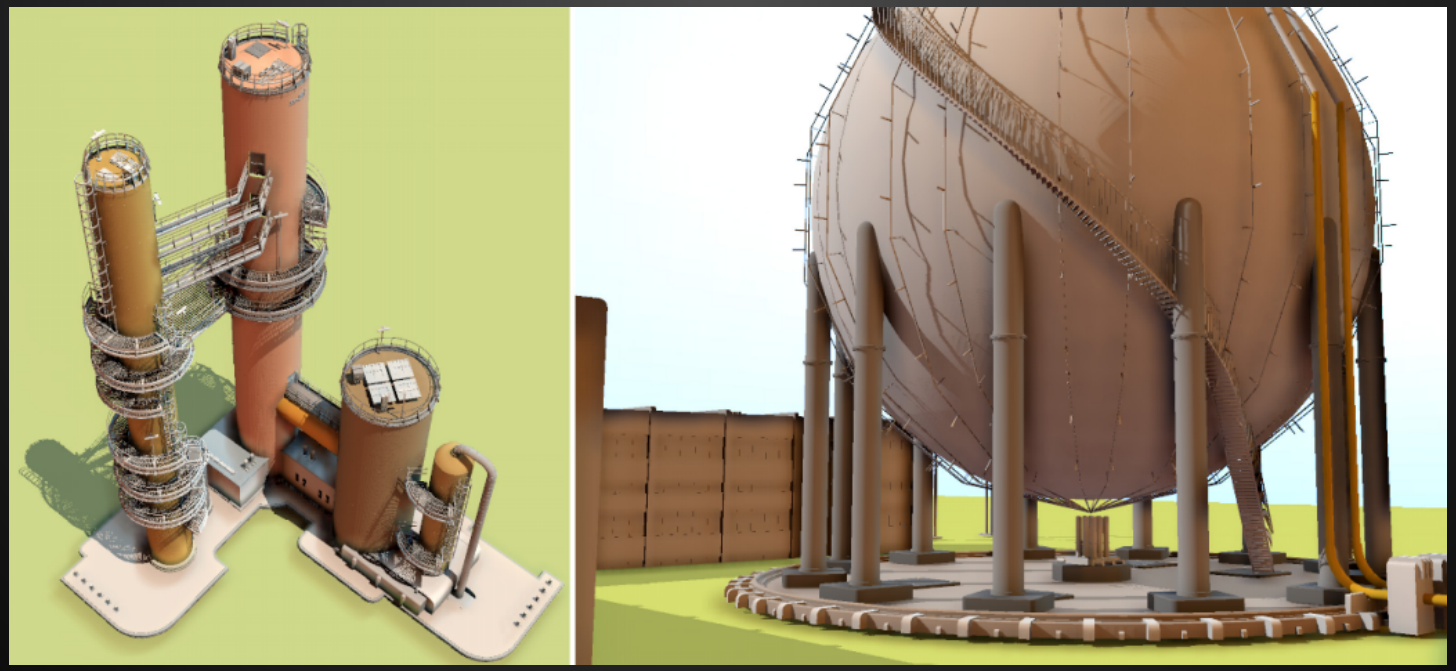
Half-res SSAO flickering

Solution in Battlefield 3: Temporal filtering

- AO is dependent only on scene geometry, not camera position.
- Use AO from last frame. Reproject it to the current view and interpolate between new and old AO.

Other GI approximations in screen space

- SSDO
  - Screen space directional occlusion
  - Directional shadows
  - Indirect color bleeding (bounces)
  - See "Approximating Dynamic Global Illumination is Screen Space" by Ritschel et al 2009.



Demo

Questions?