Ambient Occlusion

Global Illumination

- We usually divide illumination techniques in two groups:
  - Direct illumination
    - Objects lit by rays coming directly from a light source (one “bounce”)
  - Indirect illumination
    - Rays that have been reflected/refracted by other sources in the scene (more than one “bounce”)
- Global illumination includes both illumination types, while Local only focuses on direct illumination.

Global Illumination

- Any global illumination technique you have implemented?
  - Reflections
  - Refractions
  - Shadows?
    - Light changes depending on other objects in scene other than light sources, so YES. But…
- In practice we mainly call GI only to diffuse inter-reflection

Global Illumination

- What is our approach to global illumination in Phong shading model?
  - Ambient light
    - Not realistic
    - Flat and position independent
    - Cheap

Global Illumination

- In reality, the amount of light that reaches a point on the surface of an object depends on the environment
- Objects that can self-occlude or occlude nearby objects
- Think about following scenario:
  - Is the amount of light reaching a corner of the room the same as the center of the wall?
- Why is this happening?

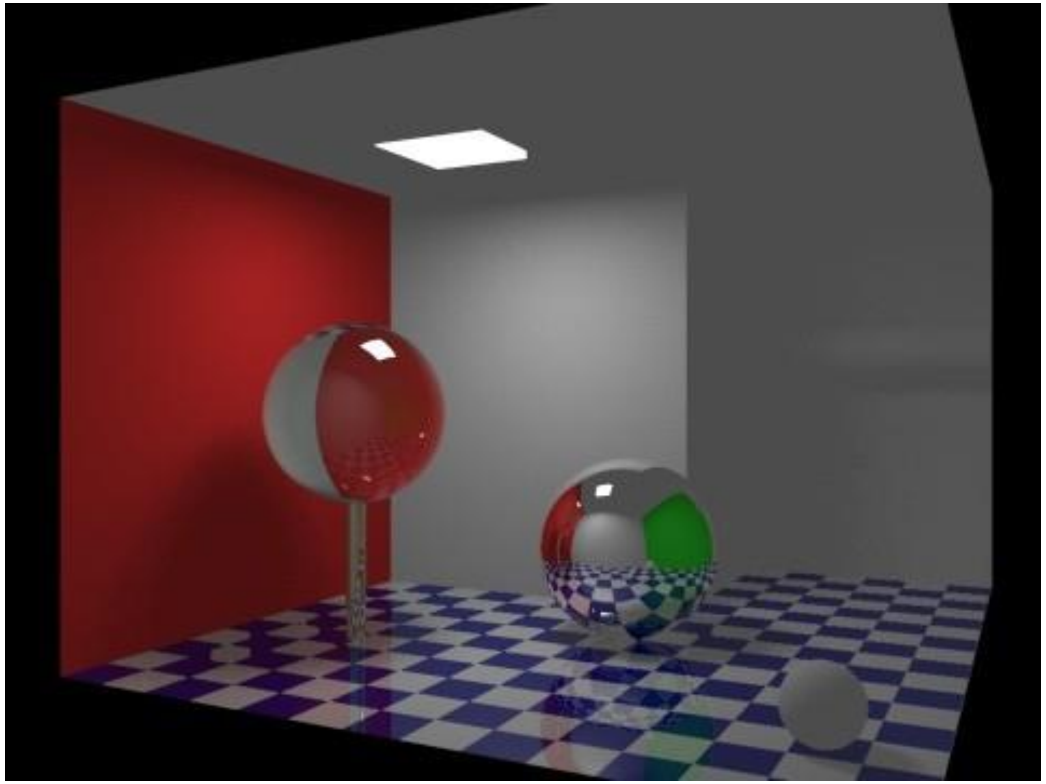
Wikipedia

Global Illumination

Global Illumination

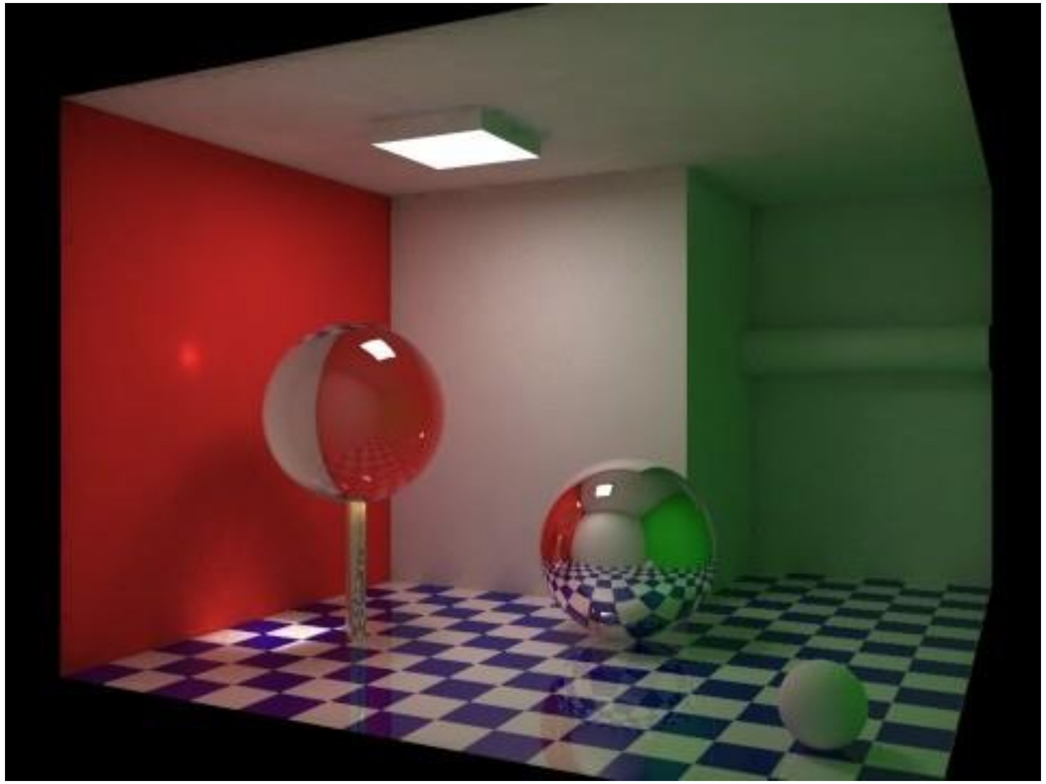
- How can we implement global illumination?
  - A bunch of different techniques:
    - Image-based lighting
    - Radiosity
    - Ray tracing
    - Light Probing
    - Cone tracing
    - Photon mapping
    - Signed Distance Fields
    - **Ambient occlusion**
    - …

Local Illumination



Wikipedia

Global Illumination



Wikipedia

Ambient Occlusion

- Shading rendering technique that calculates how exposed a surface is to incoming light
- What light component does it affect?
  - AMBIENT!
- Our ambient light lacks directional and spatial variation
- We want to add “shadows” to the ambient

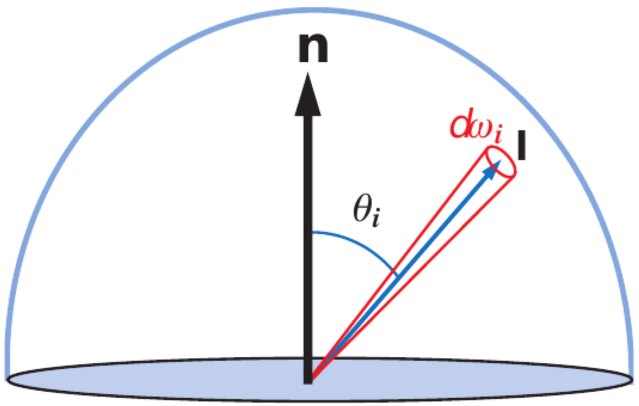
Ambient Occlusion

Ambient Occlusion

- Ambient Occlusion is a technique for approximating the effect of self-occlusion
- Gives perceptual clues of depth, curvature and spatial proximity
- We will compute the accessibility of every point on the surface of an object
- How can we compute accessibility?
  - Ray tracing offline (might be real-time nowadays) and baking results to a texture (light maps)
  - Approximated using Screen Space techniques

Ambient Occlusion

- What is irradiance?
  - Randiant flux received per unit area
  - Energy received on a surface from all incoming directions of the hemisphere



Wikipedia

Ambient Occlusion

- Ambient light defined as a constant incoming radiance for all incoming directions (without ambient occlusion)

$$E(p,n) = \int_{\Omega} L_A(n \cdot l)\, d\omega_i = \pi L_A$$

Where:

$E(p,n)$ is the surface irradiance

L_A is incoming ambient radiance

Ambient Occlusion

$$E(p,n) = L_A \int_{\Omega} (n \cdot l)\, d\omega_i$$

- We will add a visibility factor to this equation

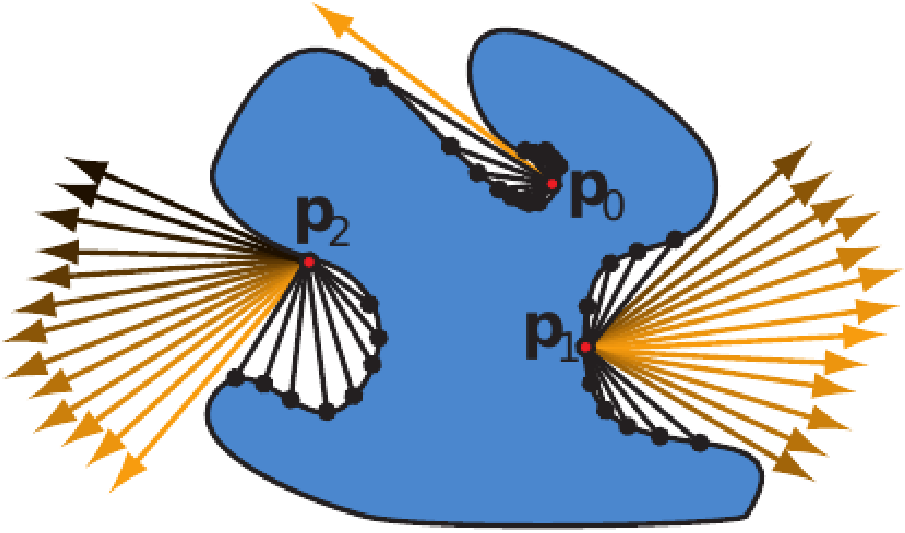
$$k_A(p) = \frac{1}{\pi} \int_{\Omega} \boldsymbol{v}(\boldsymbol{p},\boldsymbol{l})(n \cdot l)\, d\omega_i$$

- So that the irradiance varies with surface location
- $k_A(p)$ between 0 (occluded) and 1 (visible)

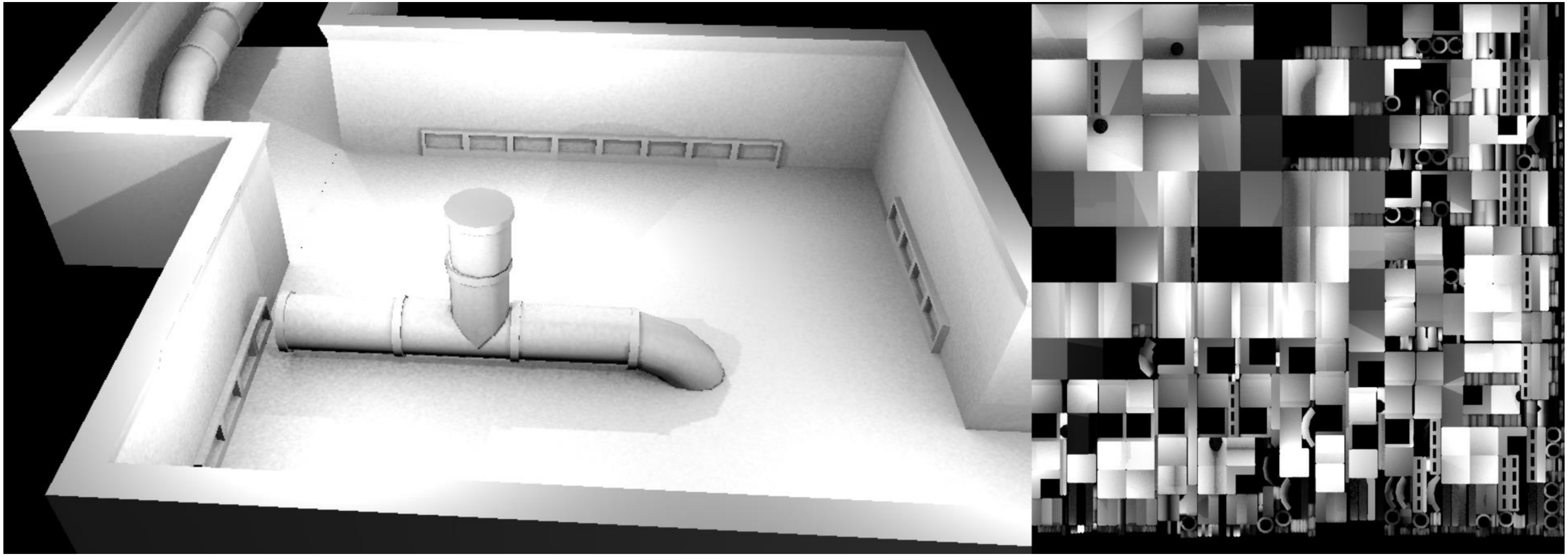
Visibility Function

- Various ways to approach this integral
  - Precomputation
    - Ray casting
    - Bent normal
    - Lightmaps
  - Real time
    - Unsharp Masking
    - SSAO
    - HBAO

Bent Normal

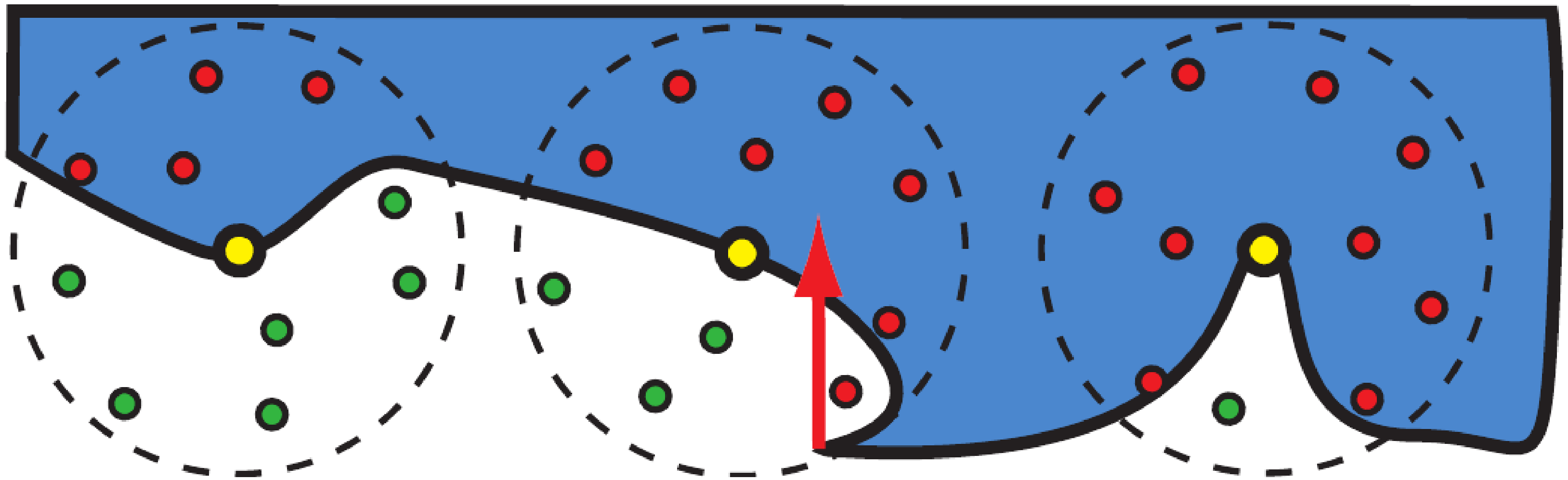


Lightmaps



Screen-Space Ambient Occlusion (SSAO)

- Consists on a post-process algorithm where visibility will be computed based on depth
- Developed by Crytek

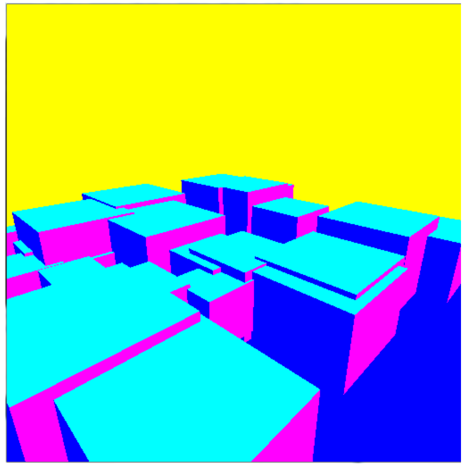


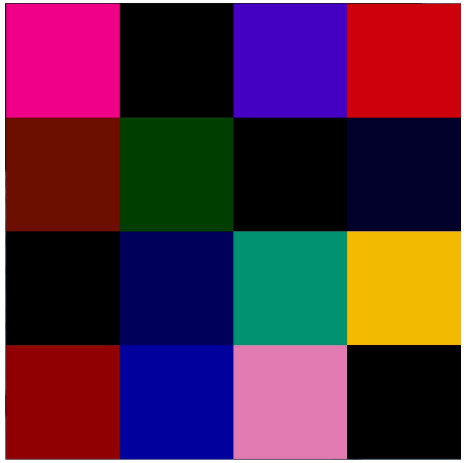
SSAO

- Check different random position values in a region
  - If more than 50% of the values are LESS than the current fragment depth value
    - More than 50% of the neighboring fragments are closer to the camera than the current fragment
    - Current fragment is probably blocked so compute occlusion factor

SSAO

- Needed inputs
  - Cam position
  - View space position
  - Random values
    - Noise texture
    - Random buffer





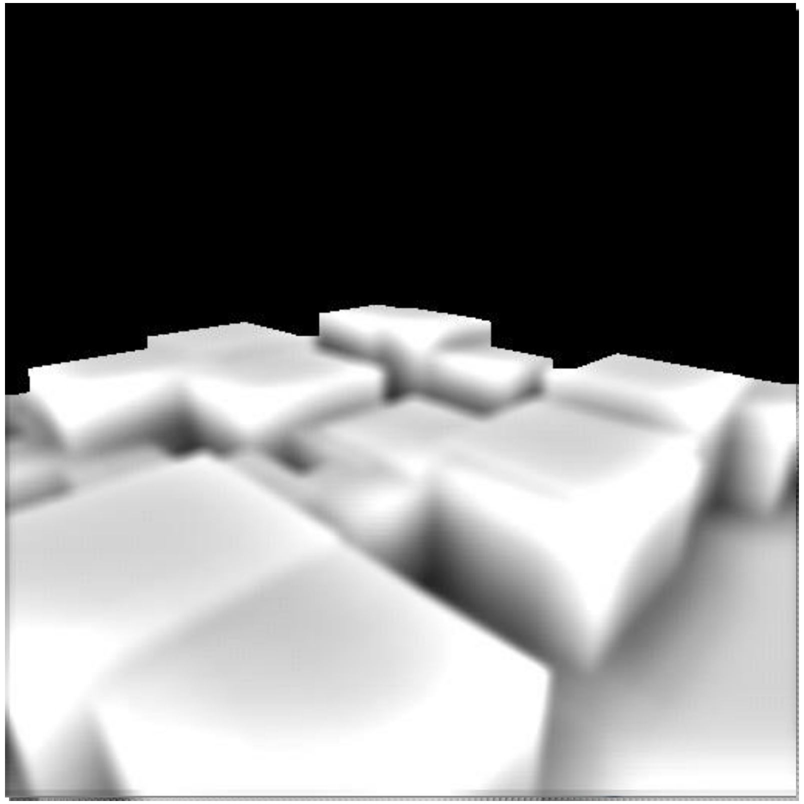
SSAO

```
// Calculate occlusion value
float occlusion = 0.0f;
// remove banding
const float bias = 0.25f;
for(int i = 0; i < SSAO_KERNEL_SIZE; i++)
{
    vec3 samplePos = TBN * uboSSAOKernel.samples[i].xyz;
    samplePos = fragPos + samplePos * SSAO_RADIUS;

    // project
    vec4 offset = vec4(samplePos, 1.0f);
    offset = ubo.projection * offset;
    offset.xyz /= offset.w;
    offset.xyz = offset.xyz * 0.5f + 0.5f;

    float sampleDepth = -texture(samplerPositionDepth, offset.xy).w;

    occlusion += (sampleDepth >= samplePos.z + bias ? 1.0f : 0.0f);
}
occlusion = 1.0 - (occlusion / float(SSAO_KERNEL_SIZE));
```

https://github.com/SaschaWillems/Vulkan/tree/master/data/shaders/glsl/ssao

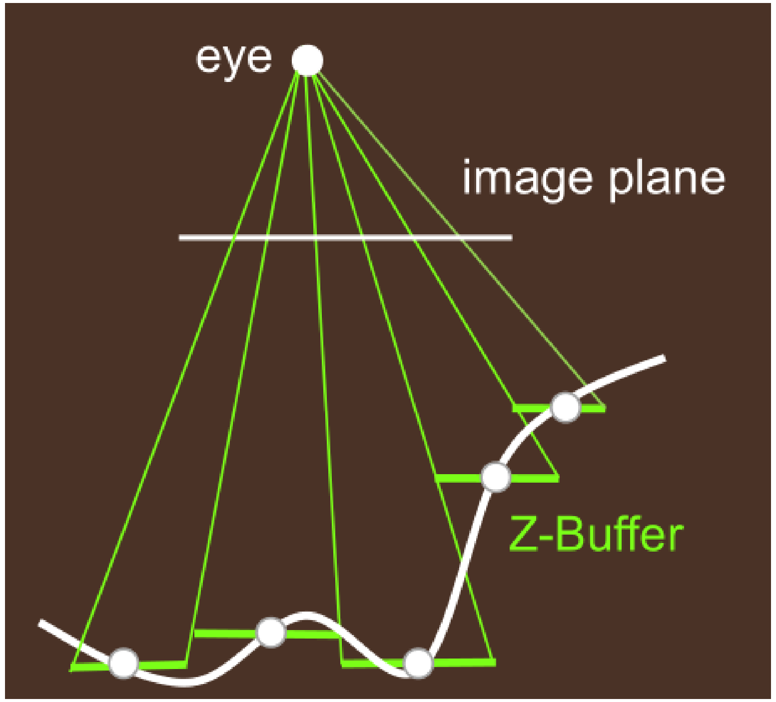
SSAO

- Aliasing and banding artifacts
- Solution?
  - BLUR!
  - Gaussian blur… for now
- Use this texture to multiply to ambient



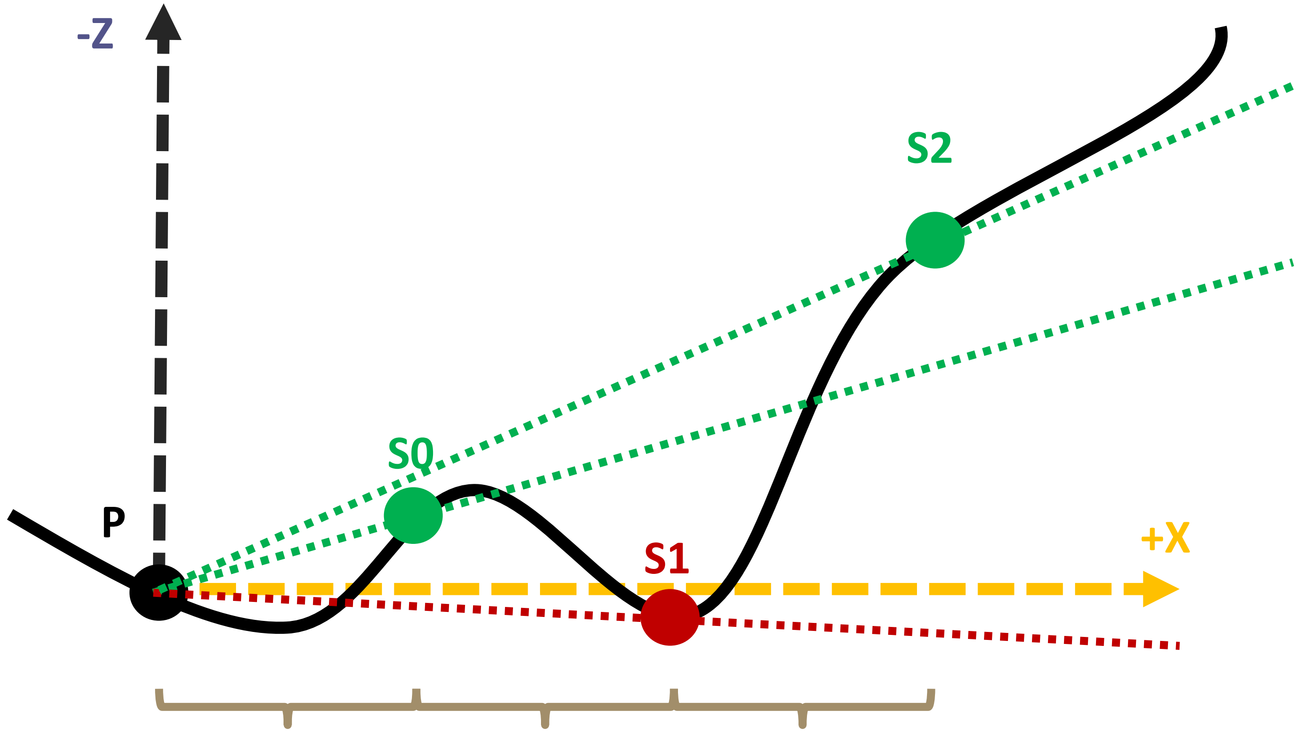
Horizon-Based Ambient Occlusion (HBAO)

- Developed by NVIDIA
- Uses depth buffer as a heightfield to evaluate visibility
  - Can use positions Z also



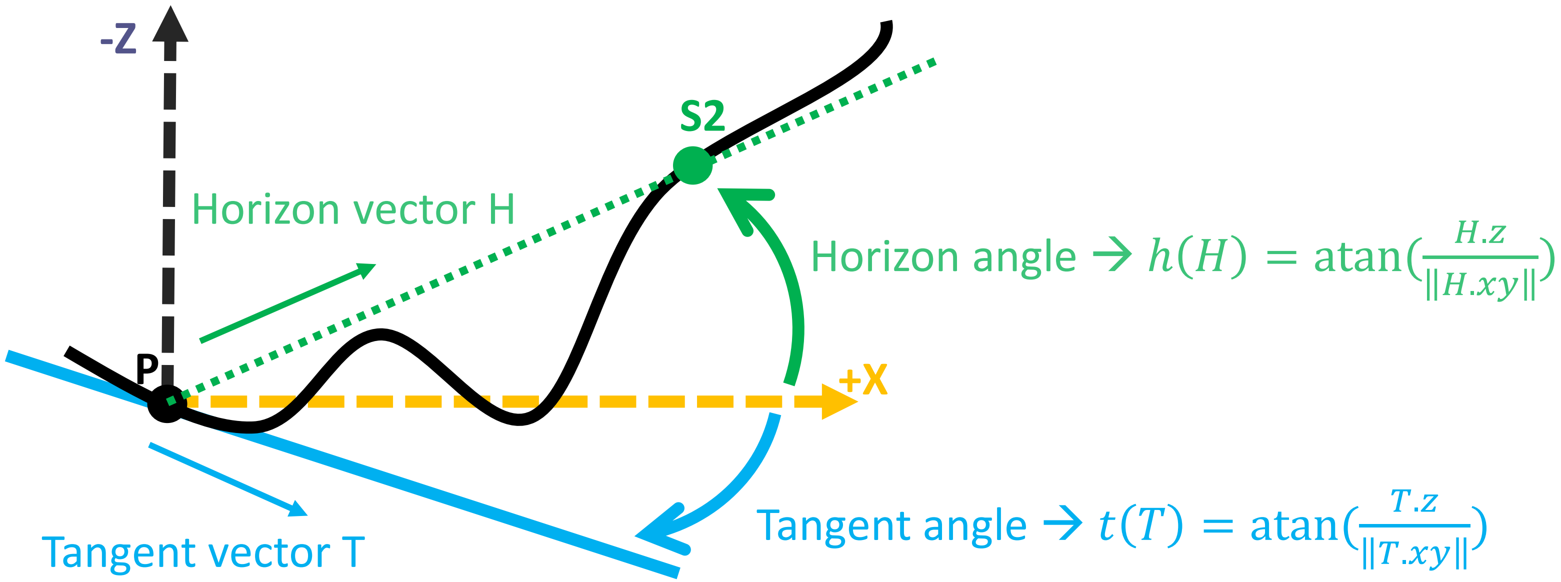
HBAO

- March along multiple directions on the heightfield



HBAO

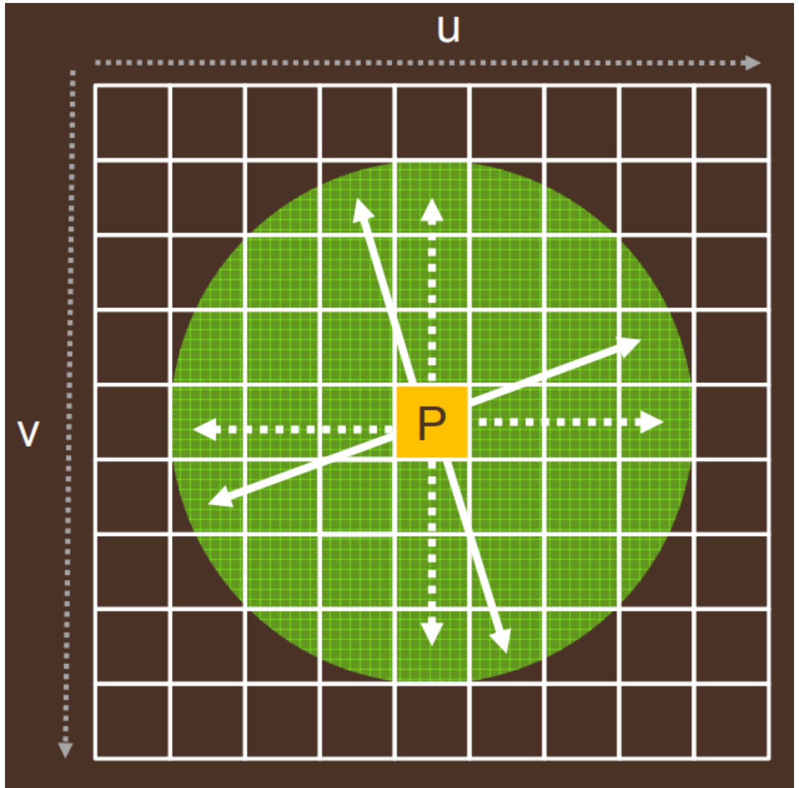
- Tangent vector will be relevant for the final occlusion



$$AO = \sin h - \sin t$$

HBAO

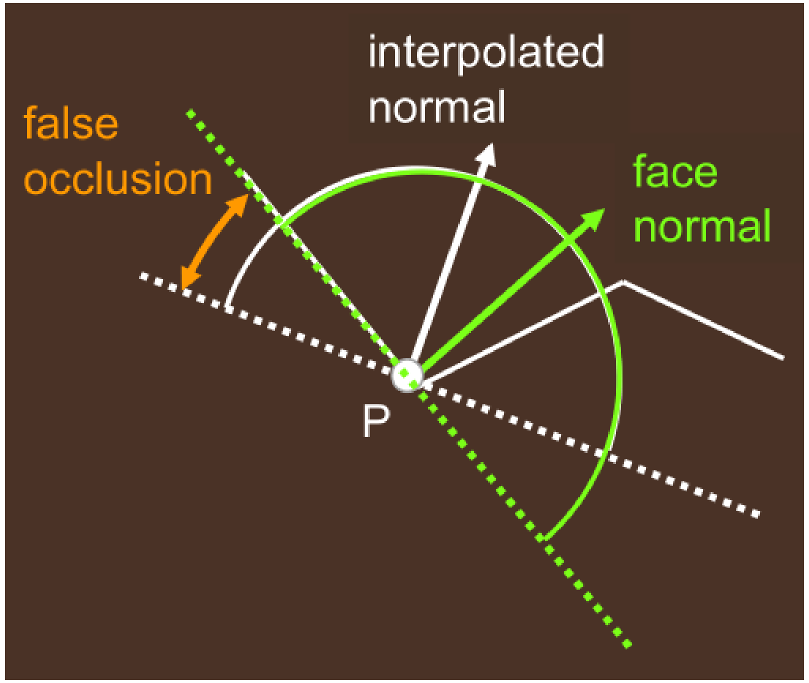
- A maximum radius of the disk needs to be defined
  - Usually in eye space
- Marching (sampling) will happen along multiple directions
- Each direction will have a uniform sampling
- Use per pixel randomization for sampling
  - Rotate the directions of sampling by a random factor
  - Jitter sample by a random offset



Example with 4 directions / pixel

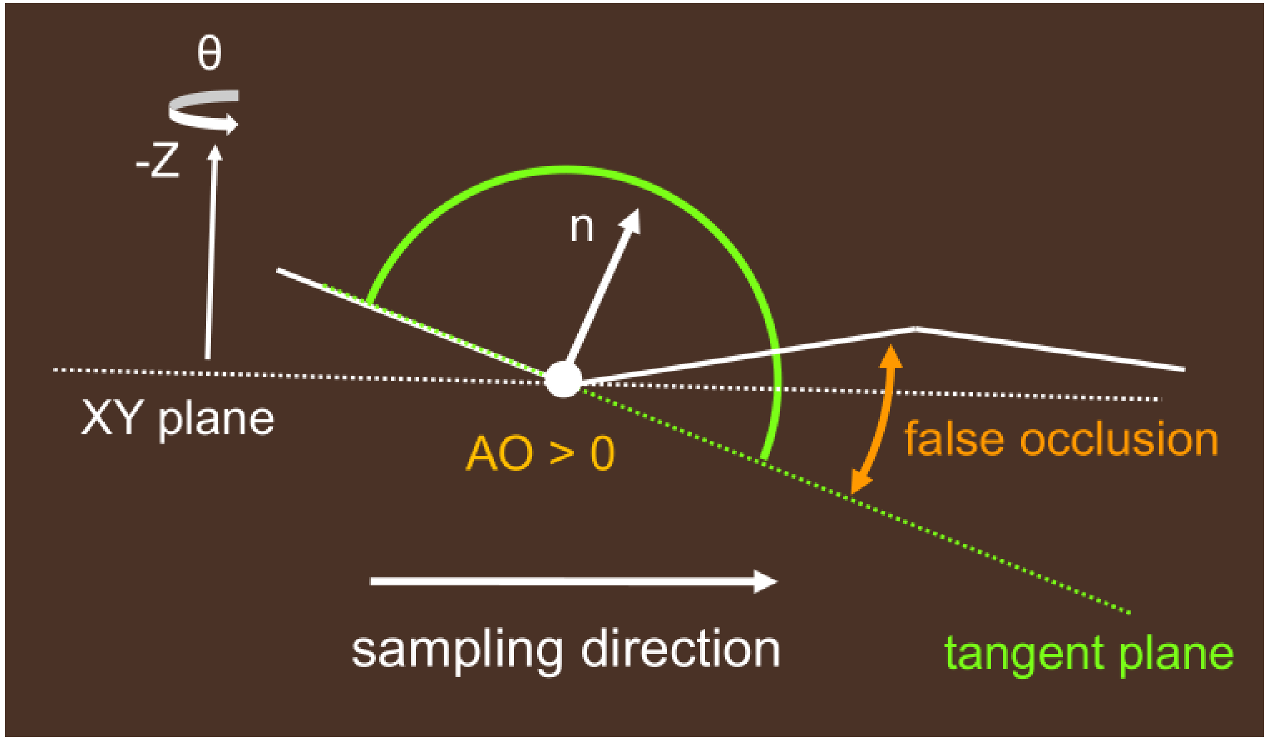
HBAO

- Using face normal is recommended
  - Can use dFdx/dFdy to get those
  - Interpolated normal may generate false occlusion
- Normal maps may also create a good effect (might be noisy)

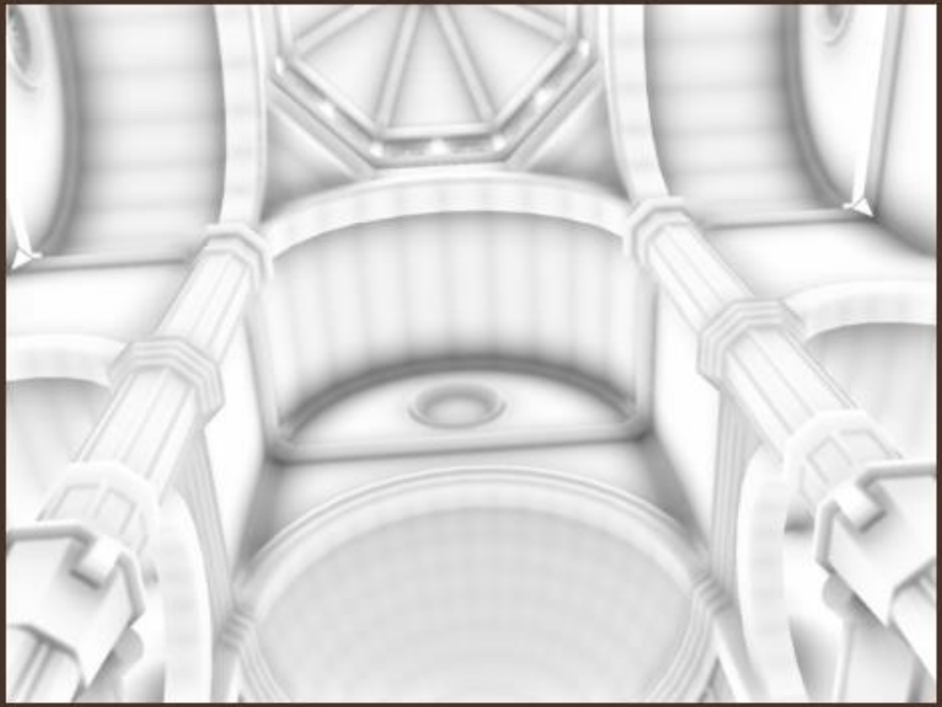


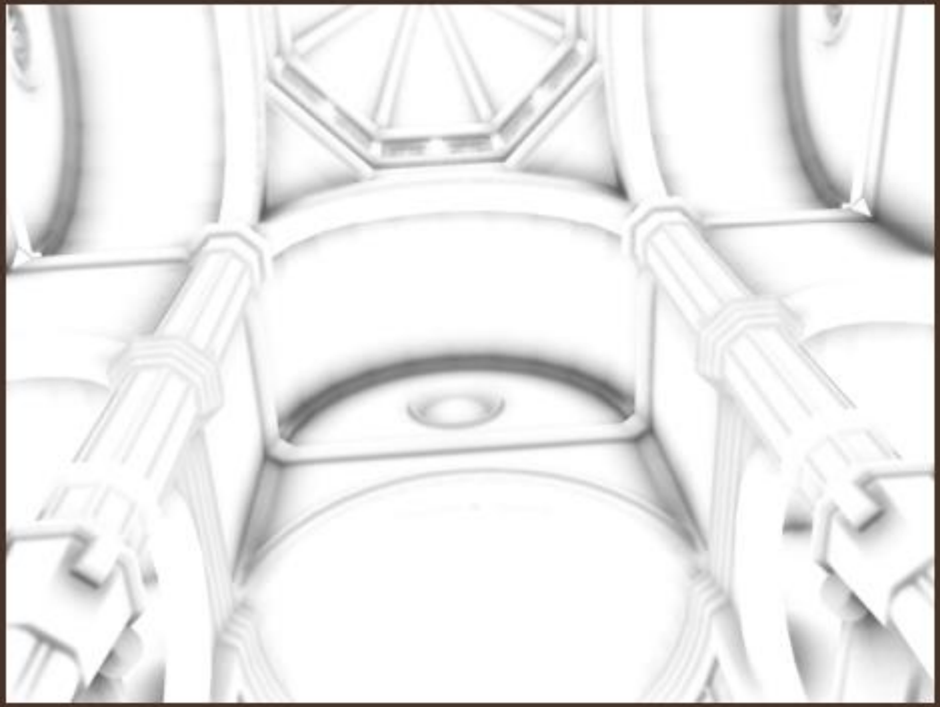
HBAO

- Add a adjustable bias to ignore “false” or small occlusions



HBAO Bias

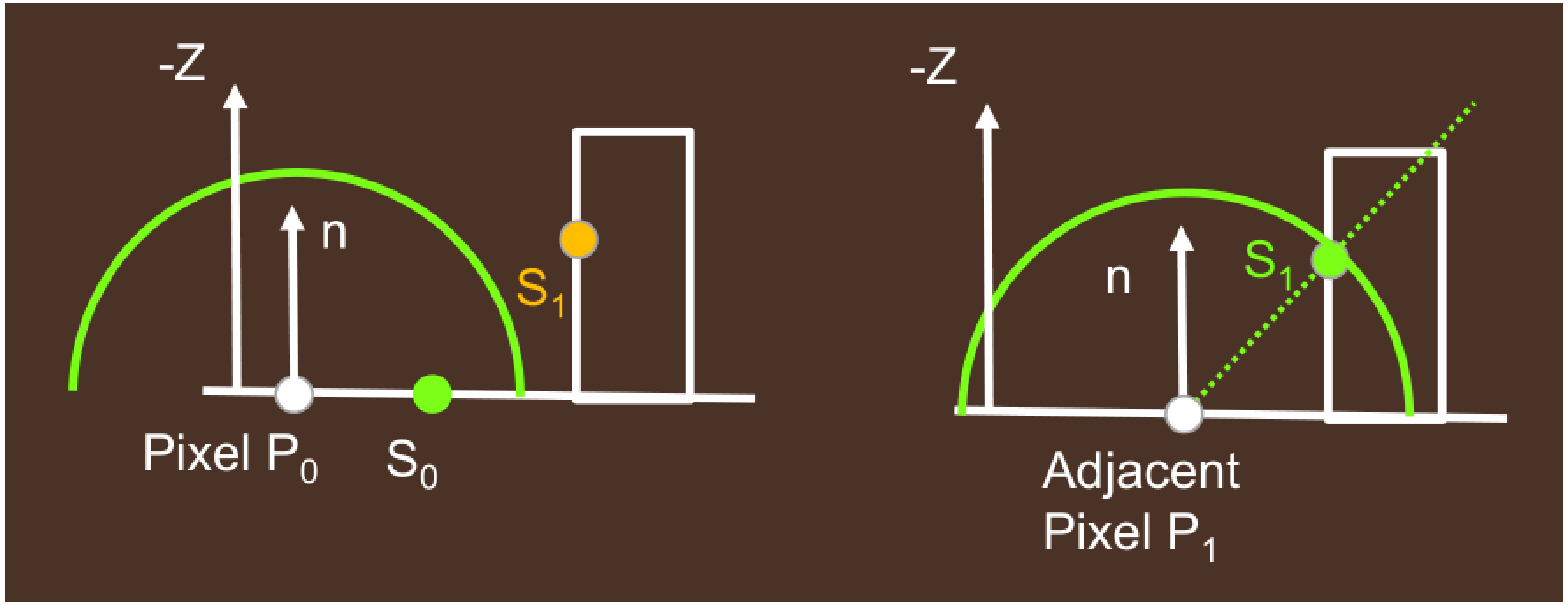

Without angle bias


With angle bias = 30 deg

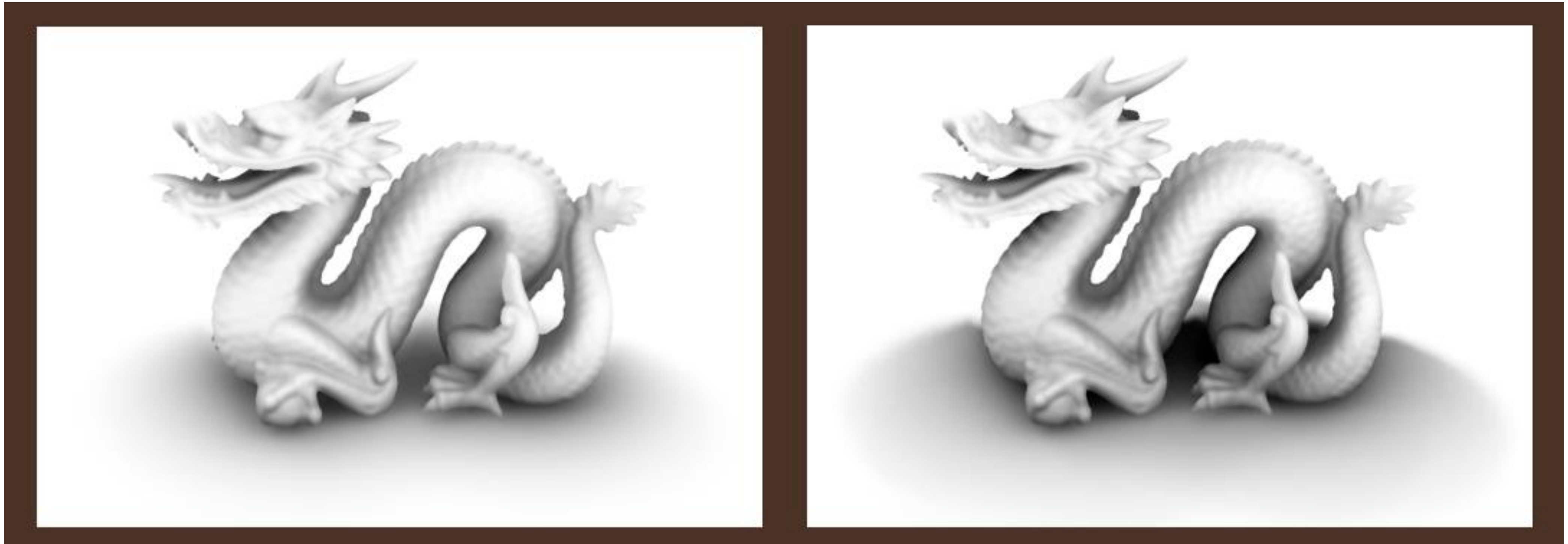
HBAO

- Discontinuity problems may also arise
- Attenuate AO contribution with the function

$$W(r) = 1 - r^2 \text{ where } r = \frac{|S_1 - P_0|}{Radius}$$



HBAO Attenuation



With Attenuation
$W(r) = 1 - r^2$

Without Attenuation
$W(r) = 1$

HBAO

- Input parameters:
  - Radius scale:
    - This is a scale factor for the radius R. The radius is the distance outside which occluders are ignored.
  - Angle bias:
    - For low-tessellated geometry, occlusion variations tend to appear at creases and ridges, which betray the underlying tessellation. To remove these artifacts, we use an angle bias parameter which restricts the hemisphere.
  - Number of directions:
    - This is the number of randomly-rotated 2D directions in image space distributed around the current pixel. The higher this parameter, the lower is the noise in the ambient occlusion.
  - Number of steps:
    - This is the maximum number samples per direction. The actual number depends on the projected size of the sphere of radius R around the current surface point.
  - Attenuation:
    - This scale factor W0 is applied to the per-sample attenuation function. The occlusion contribution of a given sample is attenuated by W0 * W(r/ R) where W(x) = 1 – x2.
  - Contrast:
    - This value allows to scales up the ambient occlusion values.

HBAO

- Like with SSAO randomly sampling will produce noise
  - BLUR!
- Regular Gaussian blur will produce halos because edges information is completely ignored
- Bilateral Filter!

References

- Screen Space Ambient Occlusion by Martin Mittring
- Image-Space Horizon-Based Ambient Occlusion by Louis Bavoil & Miguel Sainz
- Image Enhancement by Unsharp Masking the Depth Buffer by Thomas Luft, Carsten Colditz and Oliver Deussen