



Ambient occlusion

Proximity shadows

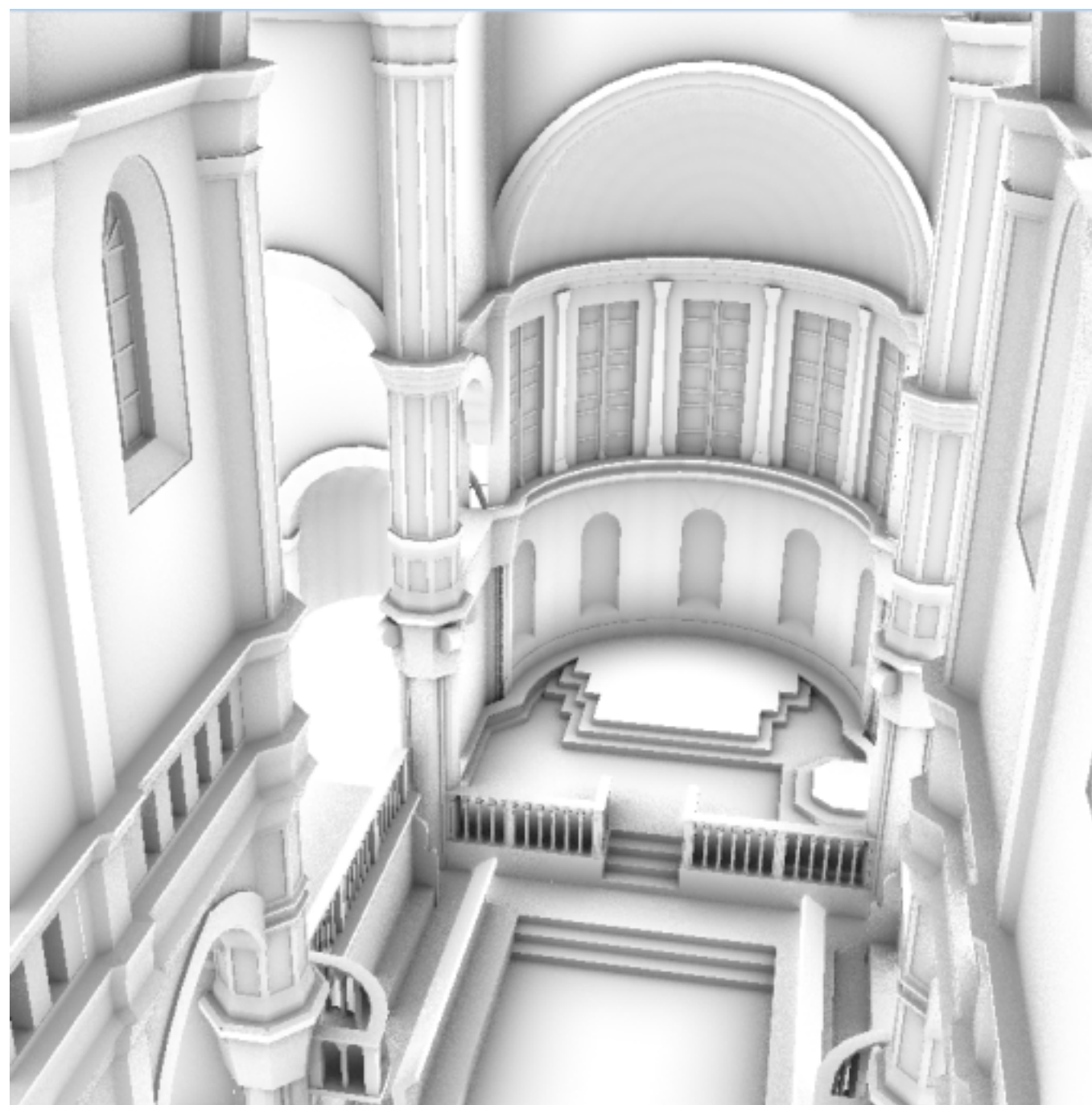
Andra ytor nära en yta - då gissar vi att den skall skuggas

Ger en bra approximation till global ljussättning.



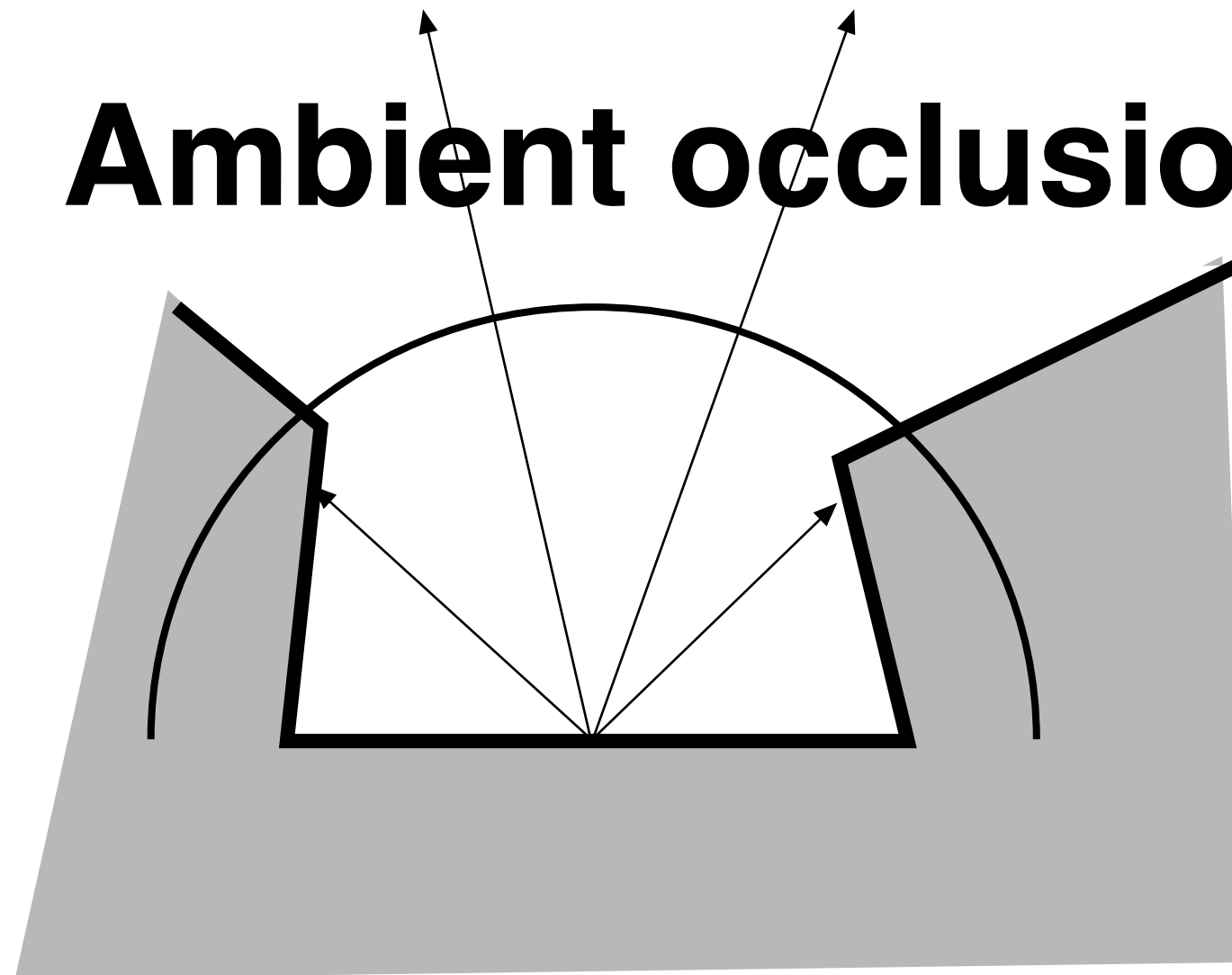
Ambient occlusion (Proximity shadows)

Kolla om det finns
många ytor nära/
om det är "trångt"





Ambient occlusion

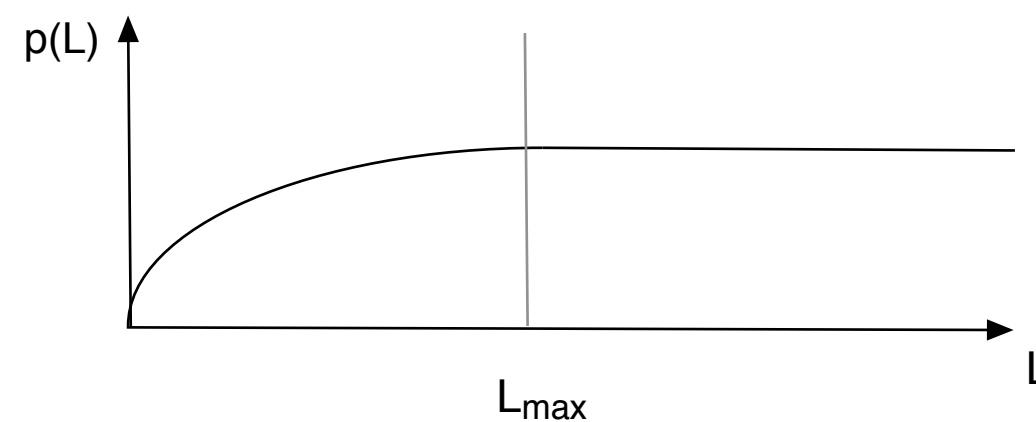


Integrera funktion av avståndet i alla riktningar över halvklot



Avståndsberoende

Avstånd genom avklingande funktion $p(L)$ (där L är uppmätt avstånd):



Detta betyder: Avstånd bortom $L_{\max}$ ökar inte "obscurance"

Figur i boken från borttappad referens. Skall vara Zhukov, samma som ovanför!



Ambient occlusion i object space

Ray-casting från varje punkt

Extremt beräkningstungt, kräver en bra strukturering av världen

Förenkling av objekt effektivt och ger ytterst liten försämring



Screen Space Ambient Occlusion

Image space

Beräkna AO genom beräkningar med Z-buffer

+ 2D-data i stället för 3D, enkel struktur

- Skymda objekt kan inte påverka



Screen Space Ambient Occlusion

Kräver exakt match mellan Z-buffer och världen (precis som shadow buffers gör)

Scale and bias

Beräkna djup från Z-bufferinnehåll



Vertex shader, scale and bias

```
#version 150

in vec3 in_Position;
out vec2 texCoord;

void main(void)
{
    gl_Position = vec4(in_Position, 1.0);
    texCoord = vec2(in_Position / 2.0 + vec3(0.5));
}
```




Fragment shader, hämta djup

```
float readDepth( in vec2 coord )
{
    // These numbers must match the main program's
    perspective projection
    float zNear = 1.0;
    float zFar = 100.0;

    float z_from_depth_texture = texture(texture0, coord).x;

    // scale and bias from texture to normalized coords
    float z_sb = 2.0 * z_from_depth_texture - 1.0;

    // Get back to real Z
    float z_world = 2.0 * zNear * zFar /
                    (zFar + zNear - z_sb * (zFar - zNear));
    return z_world;
}
```



readDepth() följer formlerna för normaliserade koordinater (röda boken samt PFNP)

$$\begin{bmatrix} x_h \\ y_h \\ z_h \\ h \end{bmatrix} = \begin{bmatrix} \frac{2near}{right-left} & 0 & A & 0 \\ 0 & \frac{2near}{top-bottom} & B & 0 \\ 0 & 0 & C & D \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$A = \frac{right+left}{right-left}$$

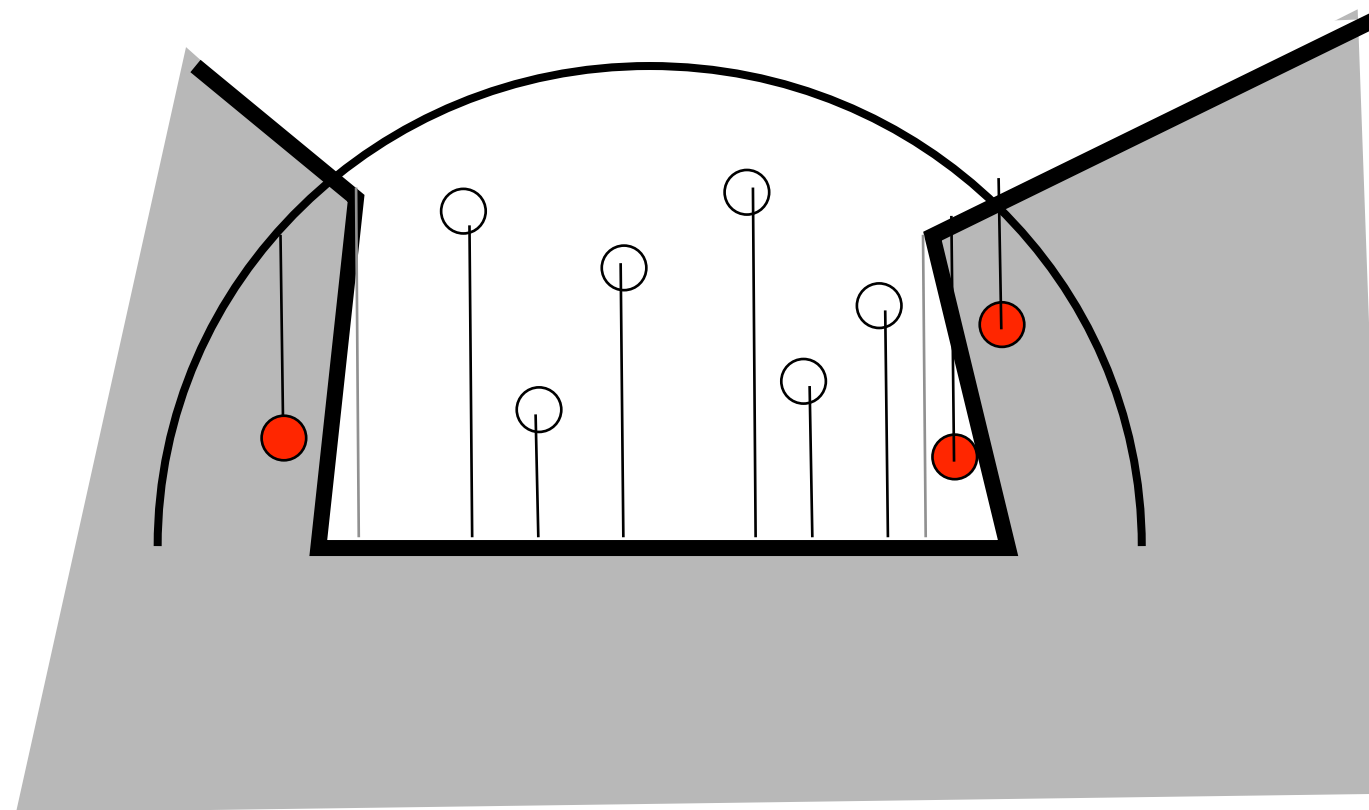
$$B = \frac{top+bottom}{top-bottom}$$

$$C = -\frac{far+near}{far-near}$$

$$D = -\frac{2 \cdot far \cdot near}{far-near}$$

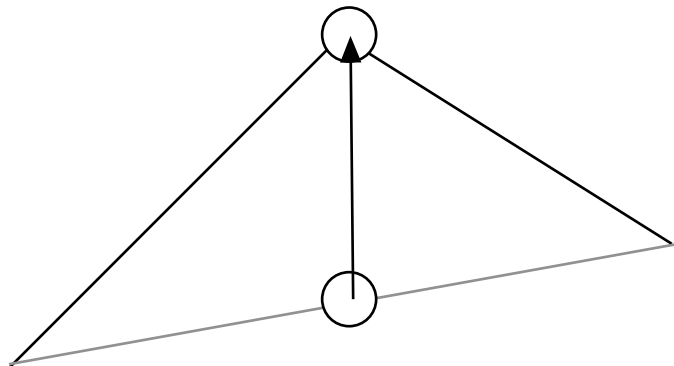


**Skapa "moln" av punkter kring centrum,
mät höjd**

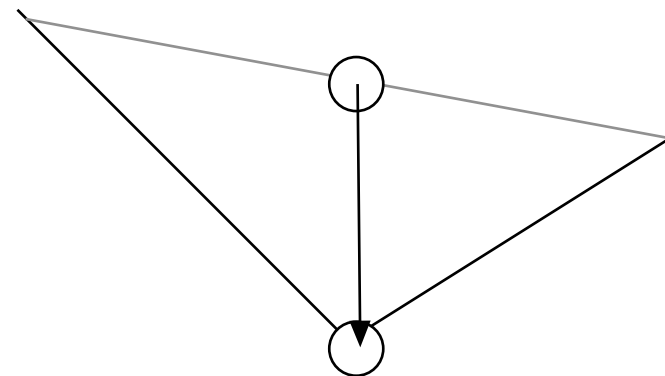




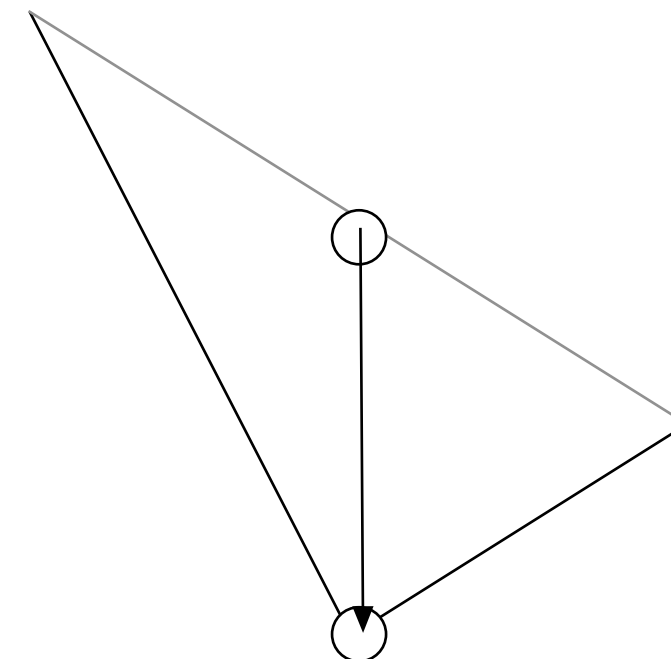
Min egen metod: Mät höjd för punkter på båda sidor, jämför medel med centrum



Mitten lägre - ingen obscurance



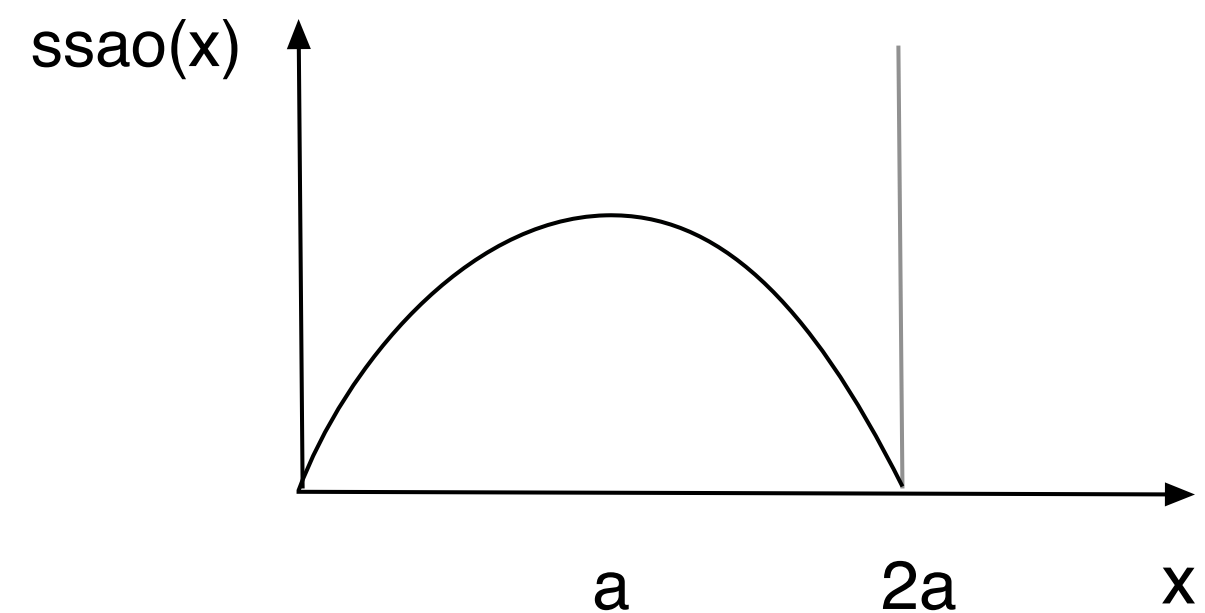
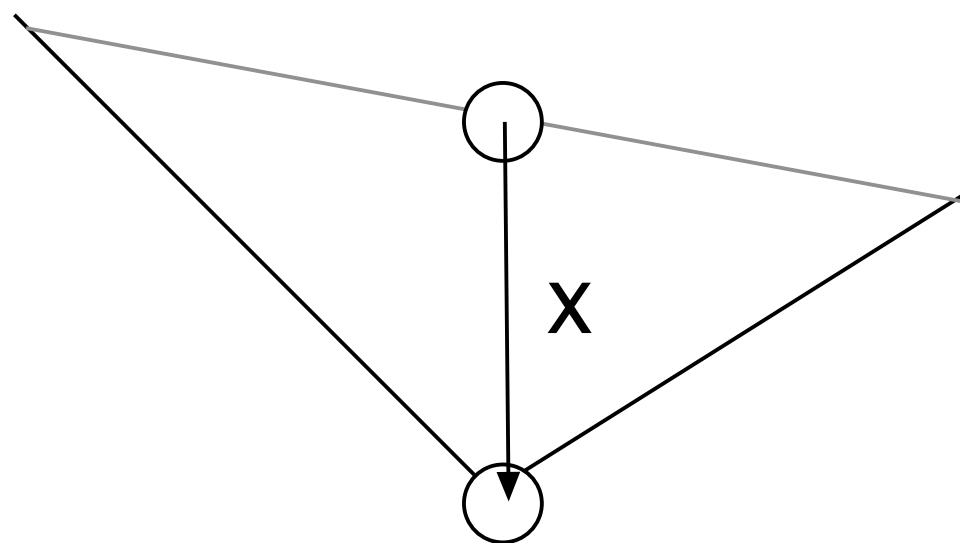
Mitten högre - obscurance



Stor skillnad - ingen obscurance (troligen skymmande objekt)

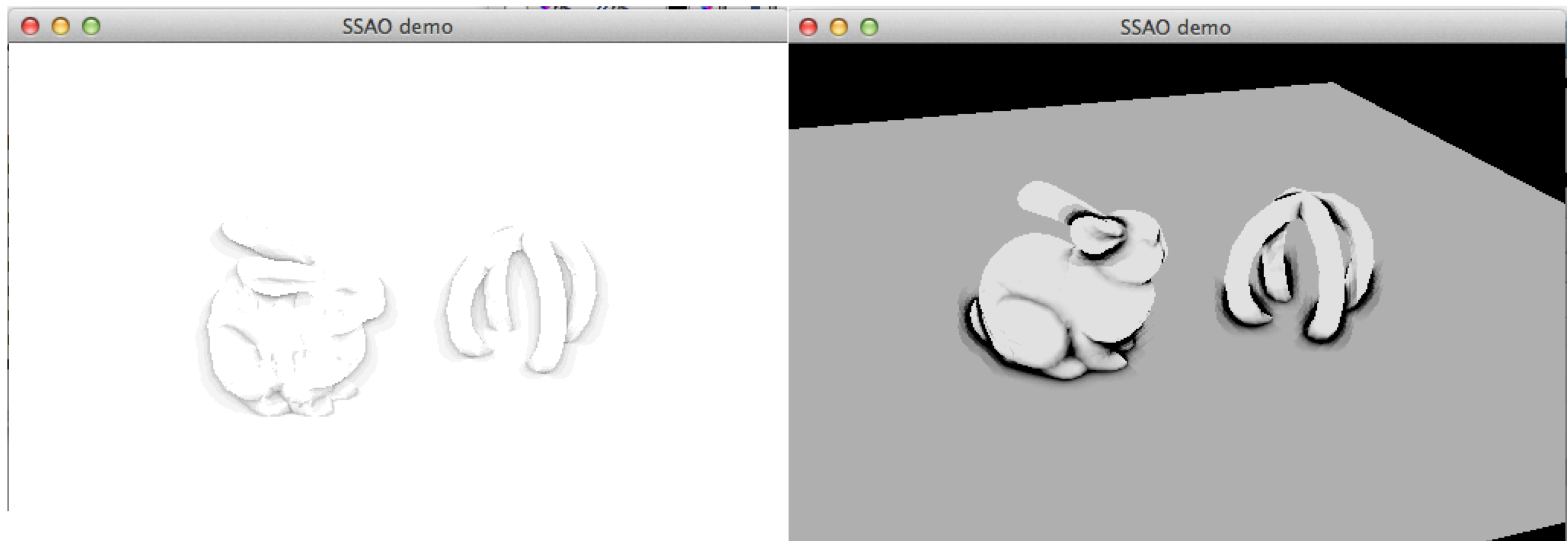


Kvadratisk funktion, ger max vid "medeldjup" skillnad





SSAO för enkel scen (överdriven)





Ambient self-occlusion

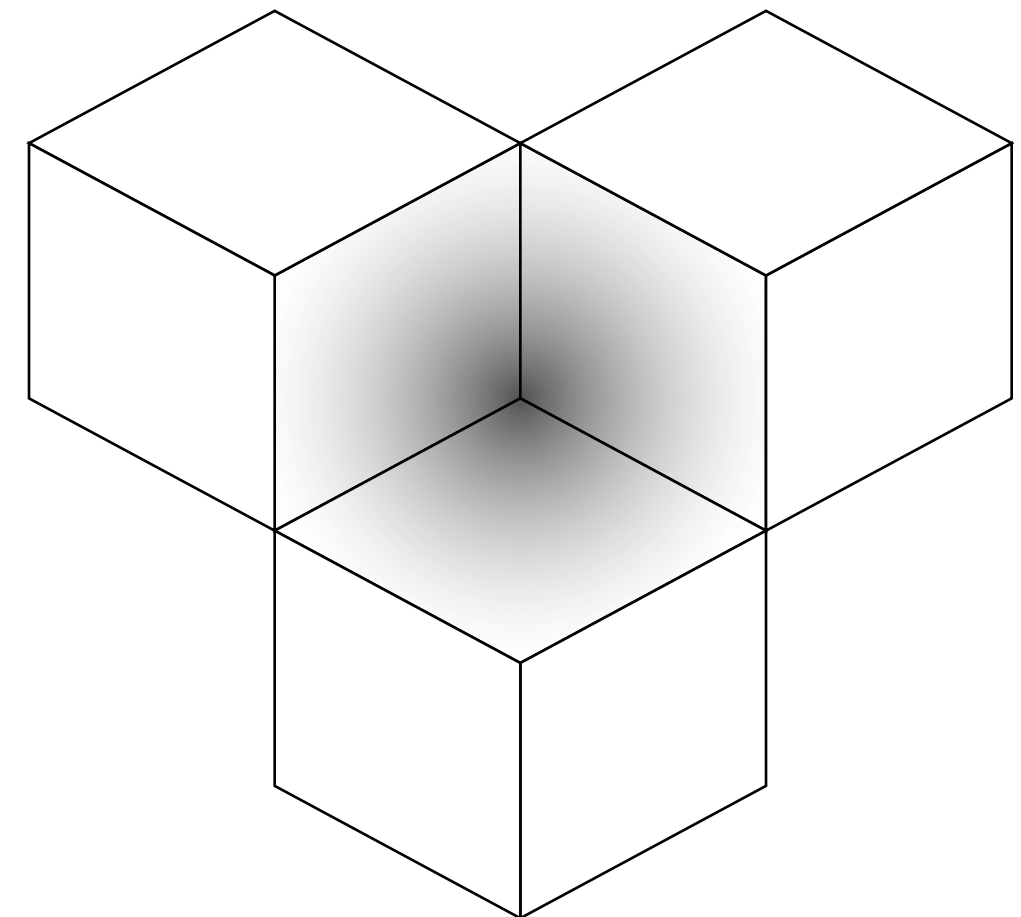
Förenklad modell

Beräkna enbart mot objektet självt

+ Kan förberäknas (men det kan radiosity också)

+ Kan beräknas mycket snabbt för enkel geometri

- Ger inga skuggor mellan objekt



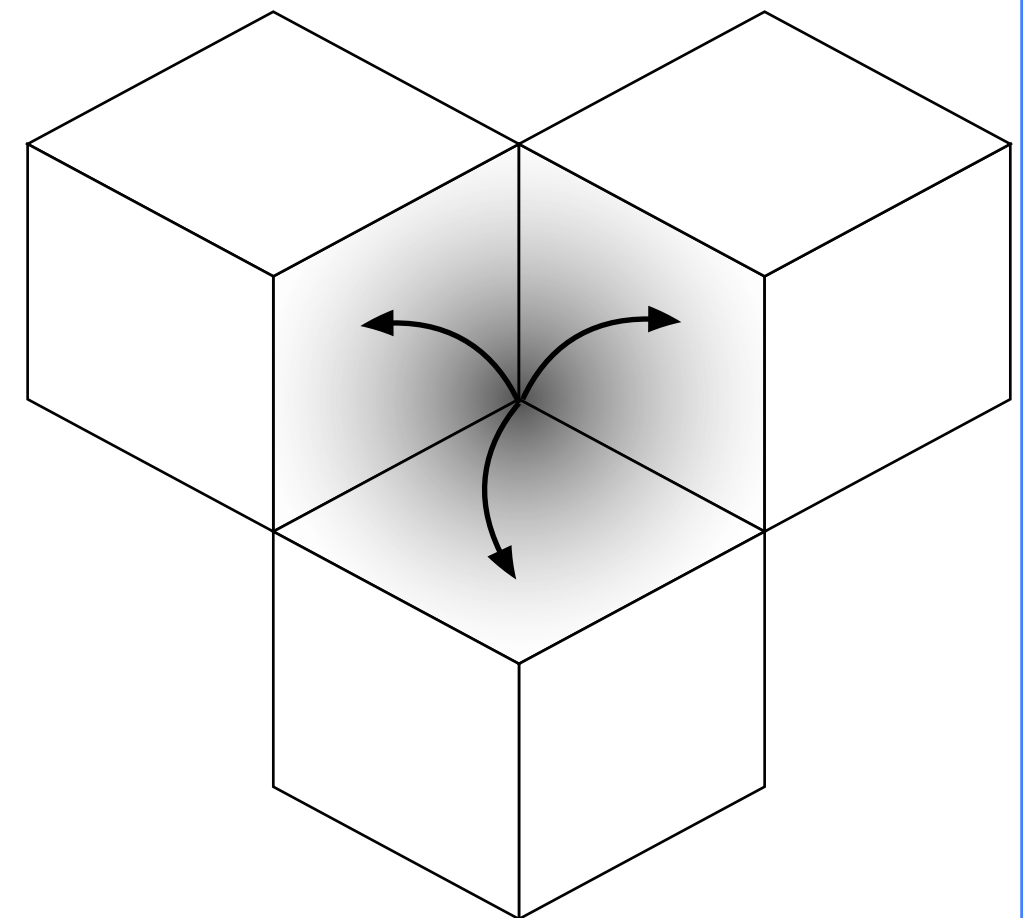
Speciellt trevlig för låg-polygon-världar!



Ambient self-occlusion ett fall för half-edge

Meshnavigering gör det snabbt och lätt att hitta grannar.

Hitta alla polygoner som gränsar till samma vertex. Undersök om de är en konkavitet.
(T.ex. med höjd längs vertexens normalvektor.)





Ambient occlusion, sammanfattning

Approximation av global belysning enbart baserad på närhet till skymmande objekt

Object space, krävande men exakt

Screen Space, snabb approximation

Observera att AO alltid är en approximation!



Raytracing på GPU

Raytracing ger skuggor som del av metoden.

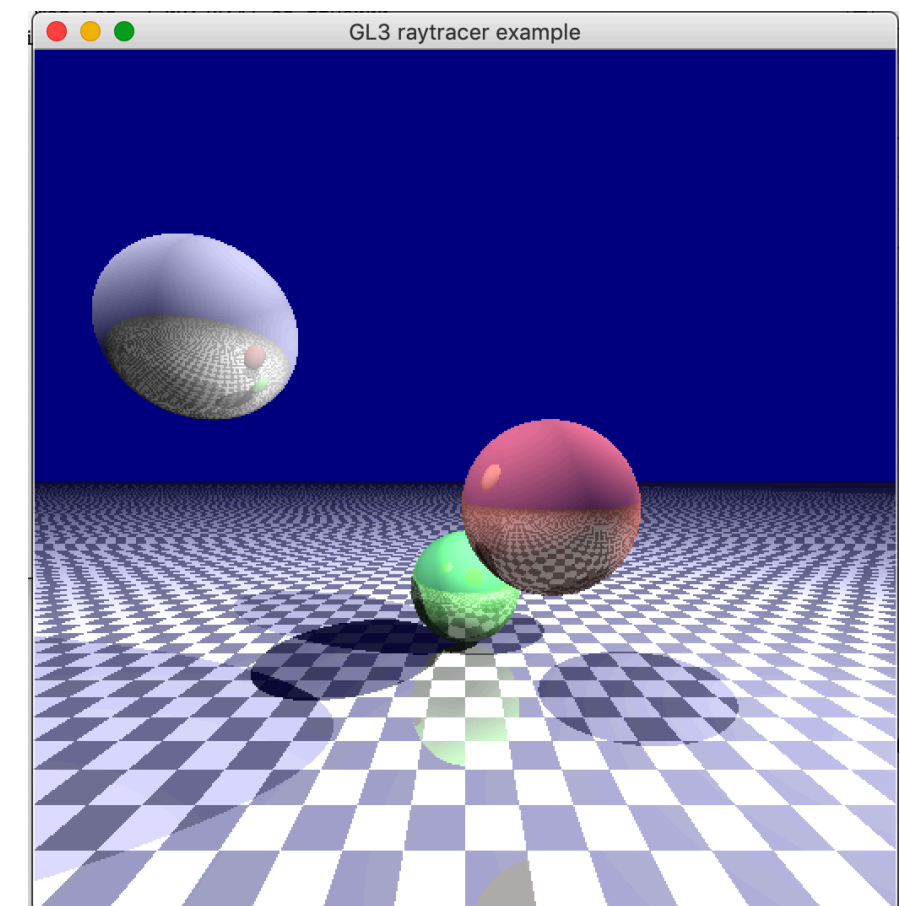
Möjligt redan på äldre kort

Aktualiserad av RTX-korten, RT cores

Accelererar BVH

Går längre än vanlig strålföljning: Photon mapping eller path tracing

Mer om detta i en senare föreläsning





Skugggenerering

Ett problem med många lösningar.

Metod kan styras av behov, prestandatillgång, typ av scen.

Mycket händer nu pga RT cores och ökade prestanda.