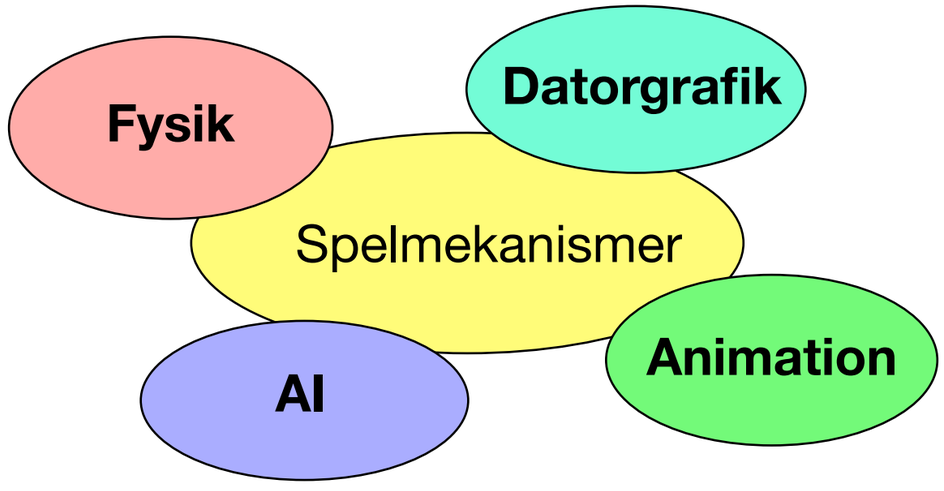
TSBK 03
Teknik för avancerade datorspel

Ingemar Ragnemalm, ISY



Föreläsning 3

"Konventionell" skuggenerering

- Skuggor, definitioner
  - Plana skuggor
  - Skuggmappning
  - Skuggvolymer
  - Mjuka skuggor
- Ambient occlusion

Föreläsningsfrågor

1. Vad är skillnaden mellan inre och yttre penumbran?

2. Vilka skuggmetoder gör perfekta skarpa skuggor?

3. Vad är fördelen med skuggmappning (shadow mapping)?

4. Varför kan man få artefakter i skuggmappning?

5. Hur hanteras ljuskällan i ambient occlusion?

Viktiga verktyg från föreläsning 2

Texturmatris / projektionsmatris

Stencilbuffer

Rendera till textur

Z-buffern som textur

Skuggor är viktiga eftersom:

- bidrar med realism
- viktiga “depth cues” som gör det lättare att förstå scenen (speciellt för flygande föremål)

men 3D-API’erna löser inte problemet åt oss automatiskt!

Skuggor

Vi vet redan att vi kan göra skuggor på flera sätt:

- Strålföljning
- Radiosity
- Light mapping baserad på dessa

men ingen av dem ger dynamiska skuggor!

(Oräknat strålföljning i realtid - helst med RTX.)

Tar strålföljningsbaserade metoder över helt?

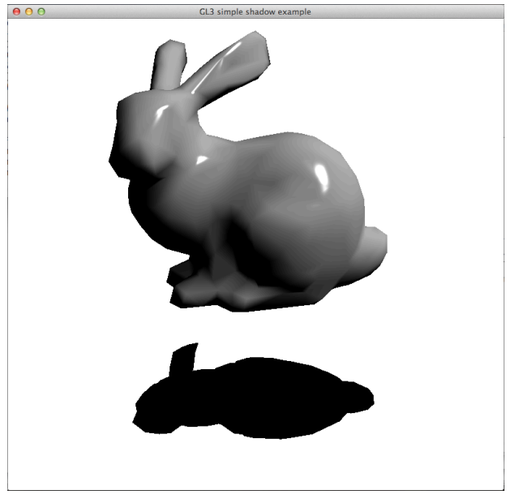
100% skifte? Inte än!

- RTX eller billigare GTX? RTX verkar ha tagit över helt!
  - Low-end (telefoner mm)?
  - Andra prioriteringar kräver beräkningskraften?
  - Hybridsystem, utnyttjar RT för delar av scenen.

En billig metod:

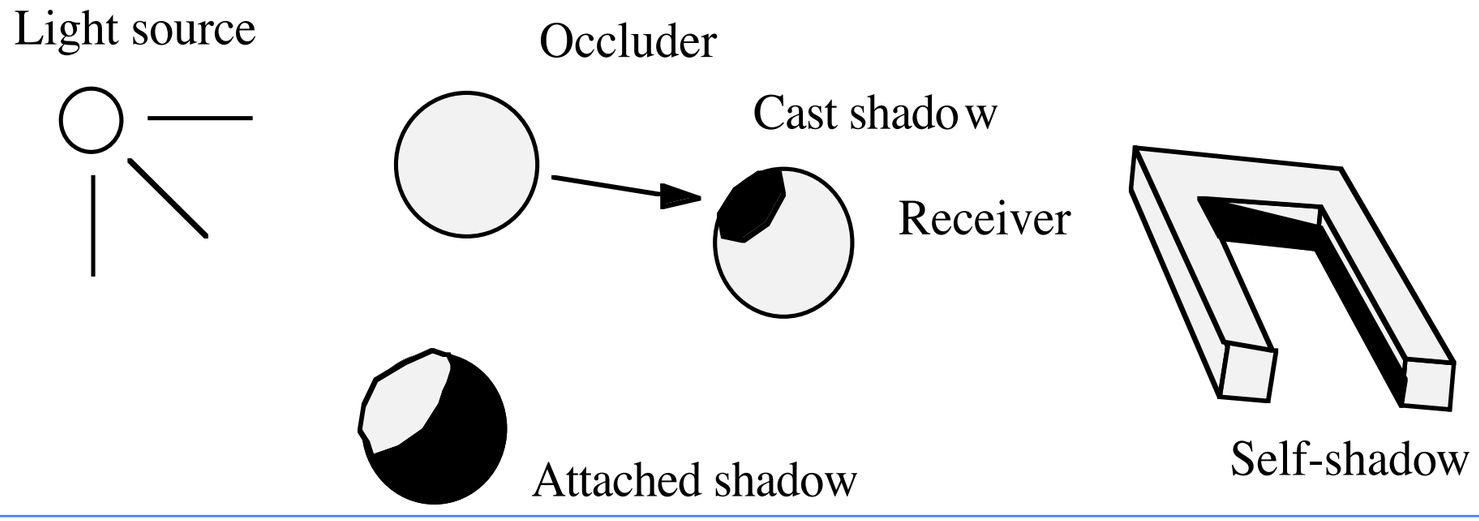
Rita tillplattad modell på ytan.

Hyfsat på ”oändlig” yta eller med stencilbuffer.



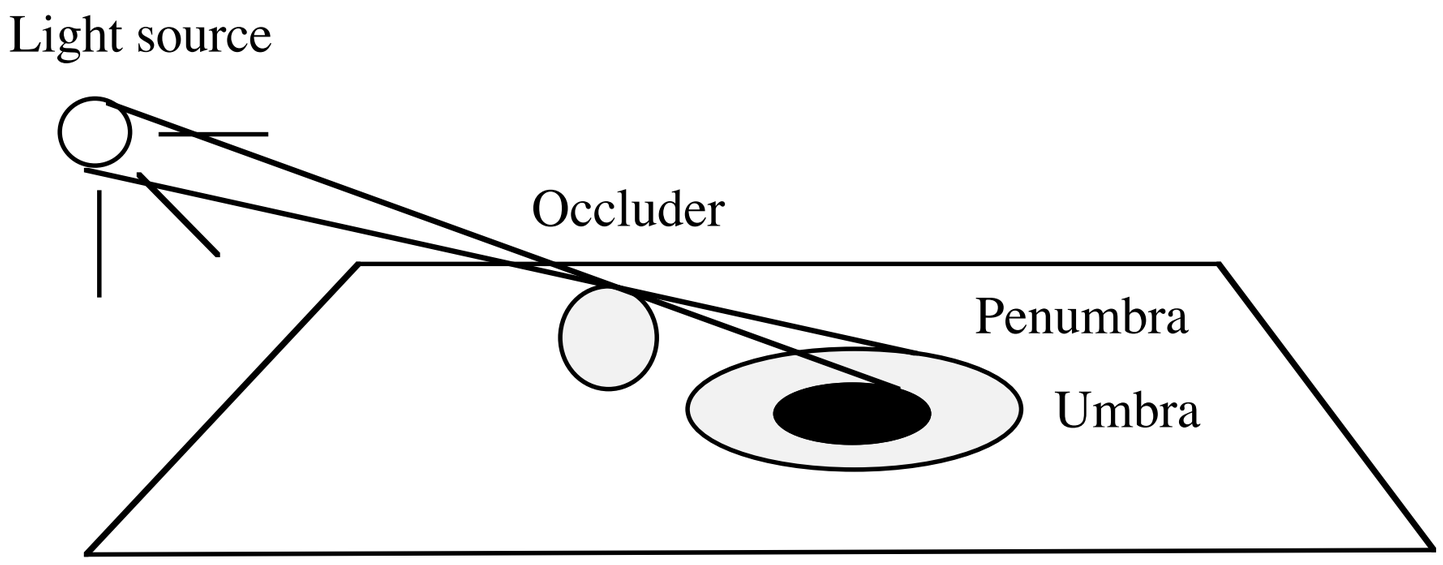
Definitioner

Occluder
Receiver
Attached shadow
Self-shadow



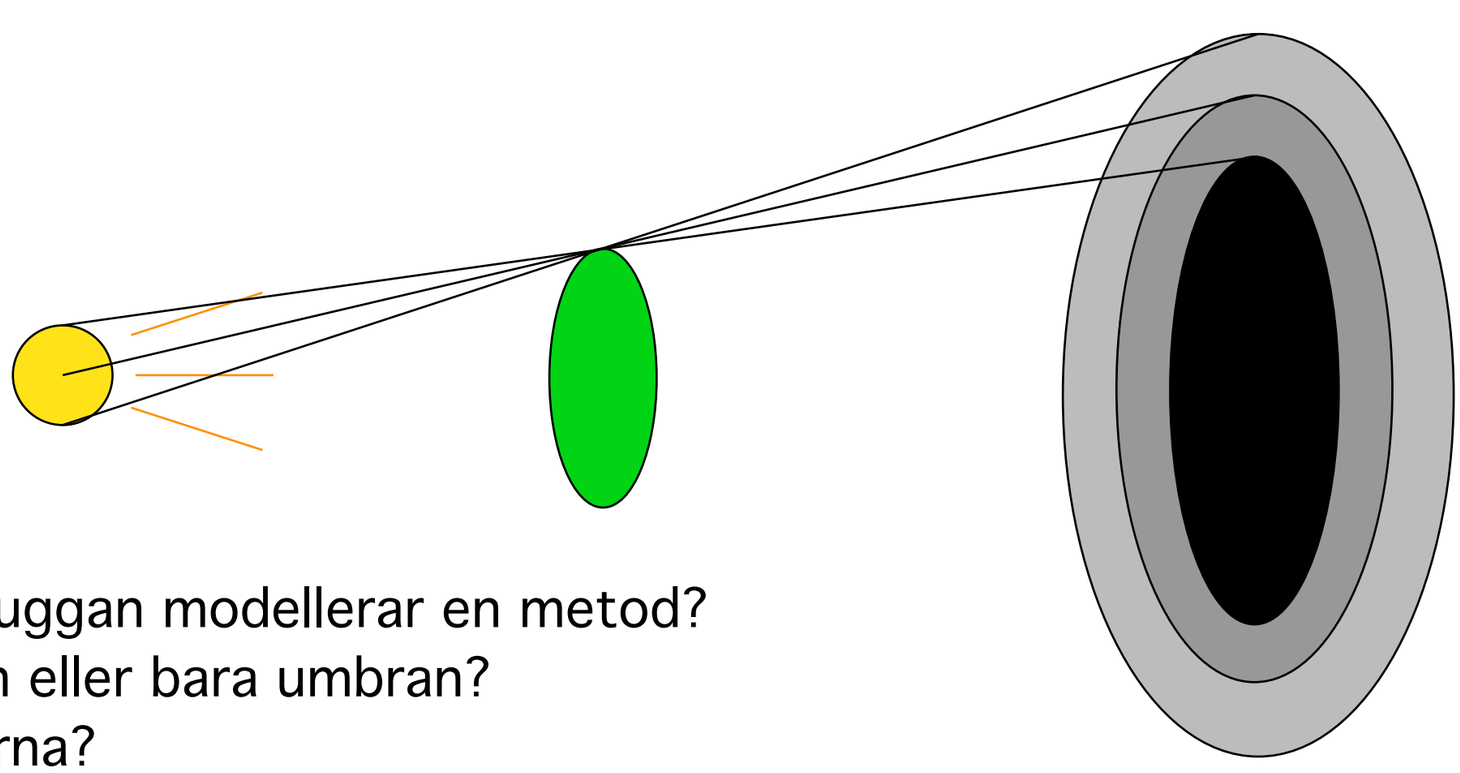
Definitioner

Umbra
Penumbra (yttre & inre)



Penumbran

Inre penumbra - skugga om punktkälla
Yttre penumbra - belyst om punktkälla



Vilka delar av skuggan modellerar en metod?
Finns penumbran eller bara umbran?
Finns båda zonerna?

Tre huvudsakliga sorters skuggor:

• Attached shadow. Detta klarar vi redan med den vanligaste belysningsmodellen.

Men det gäller inte för:

• Cast shadow, skugga från objekt till objekt.

• Self-shadowing.

Vi behöver effektiva metoder (dvs realtid) för detta som ger bra resultat.

Dynamiska skuggor

Skuggmetoder med låga prestandakrav.

Metoder för dynamiska skuggor:

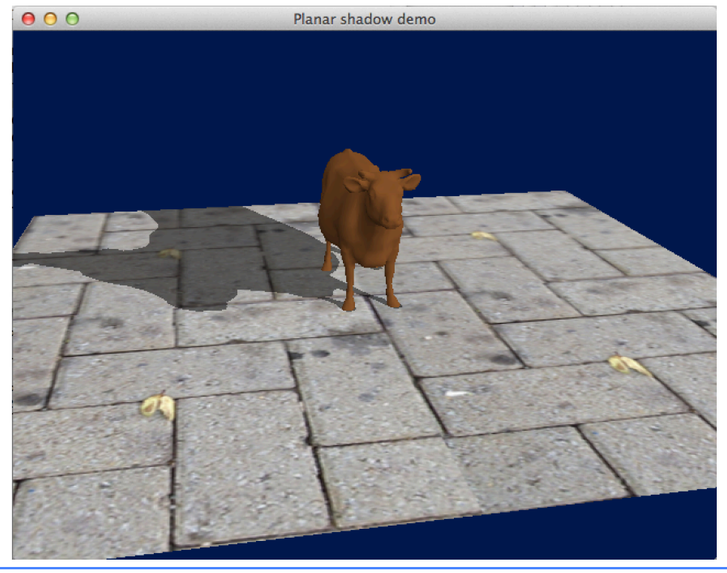
- Projektion på plana ytor
  - Shadow maps
  - Shadow volumes

Projektion på plan yta

Projicera objektet på en yta!

Projektionsmatris multipliceras in i stil med rotation runt godtycklig axel.

Objektet renderas utan textur, i önskad skuggas färg, blendas med bakgrunden.



Projektion på plan yta

Delproblem att lösa:

- Utför projektionen av objektet till vald yta
- Maska ritandet till denna yta med stencilbuffern
- Rita objektet på ytan med transparens

Projektionsmatrisen

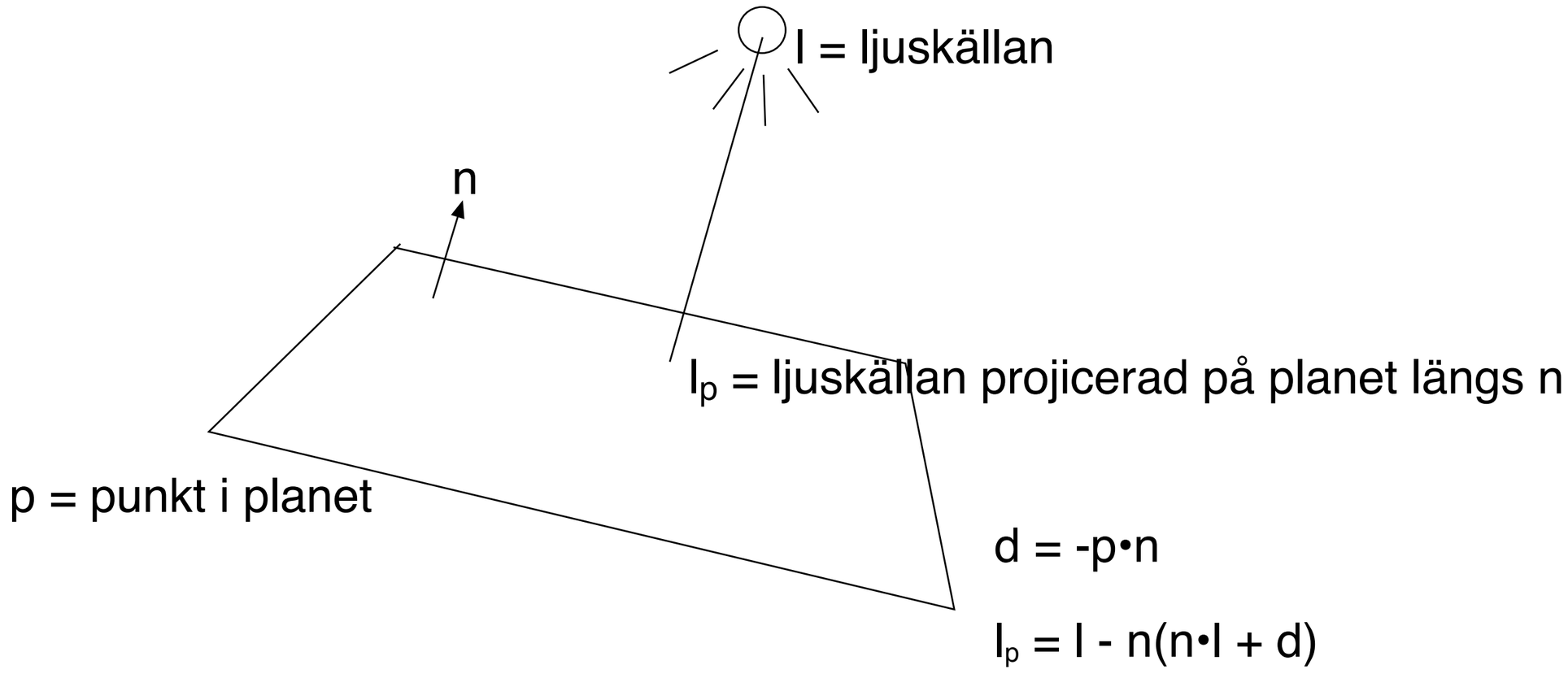
Enkla projektionmatriser längs Z:

$$\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & f & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix} \qquad \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & f \end{bmatrix}$$

Rotera och translatera så det stämmer med önskat plan.

Höger matris projicerar på Z=0.
Vänster har ljuskällan i origo.

Variabler i projektionen



$d = -p \cdot n$

$l_p = l - n(n \cdot l + d)$

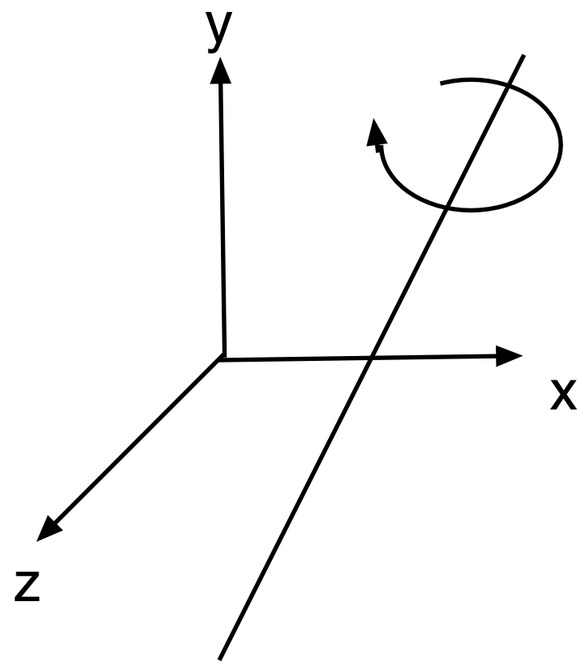
Projektionstransform

Samma princip som många liknande fall i grundkursen.

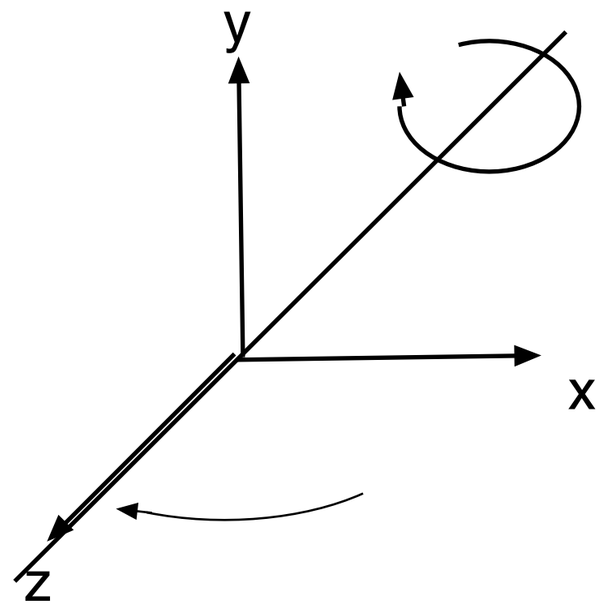
Låt l_p vara ljuskällan projicerad på planet.

- Translatera l_p till origo
- Rotera normalvektorn längs Z
- Projicera
- Rotera tillbaka
- Translatera tillbaka

Rotation runt godtycklig axel



Transform to align axis with the Z axis, rotate, and transform back.



Total transformation:

$$R(\theta) = T(p_1) * R^T * R_z(\theta) * R * T(-p_1)$$

Kod för projektion

```
float d = - p * n;
float f = n * l + d; // Distance light source to plane = focal distance
vec3 lp = l - n * (n * l + d); // lp = l - n(n dot l + d)

mat4 toz = AxisToZ(n);
mat4 fromz = Transpose(toz);
mat4 shadowProjectionMatrix = CreateShadowProjectionMatrix(f);

mat4 sm = T(lp) * fromz * shadowProjectionMatrix * toz * T(-lp);
```

Total transformation:

$$Proj = T(lp) * R^T * P * R * T(-lp)$$

Rita golvet i stencil, framebuffern och Z-buffer

```
// Inline geometry
GLfloat floorVertices[] =
     {100.0f, 0,-100.0f,
     -100.0f, 0,-100.0f,
     -100.0f, 0,100.0f,
     100.0f, 0,100.0f};
GLfloat floorTex[] = { 4.0f, 4.0f, 0.0f, 4.0f,
                       0.0f, 0.0f, 4.0f, 0.0f};
GLuint floorIndices[] = {0, 1, 2, 0, 2, 3};
```

Inlinegeometri laddas upp
(till ”model” i LittleOBJLoader)

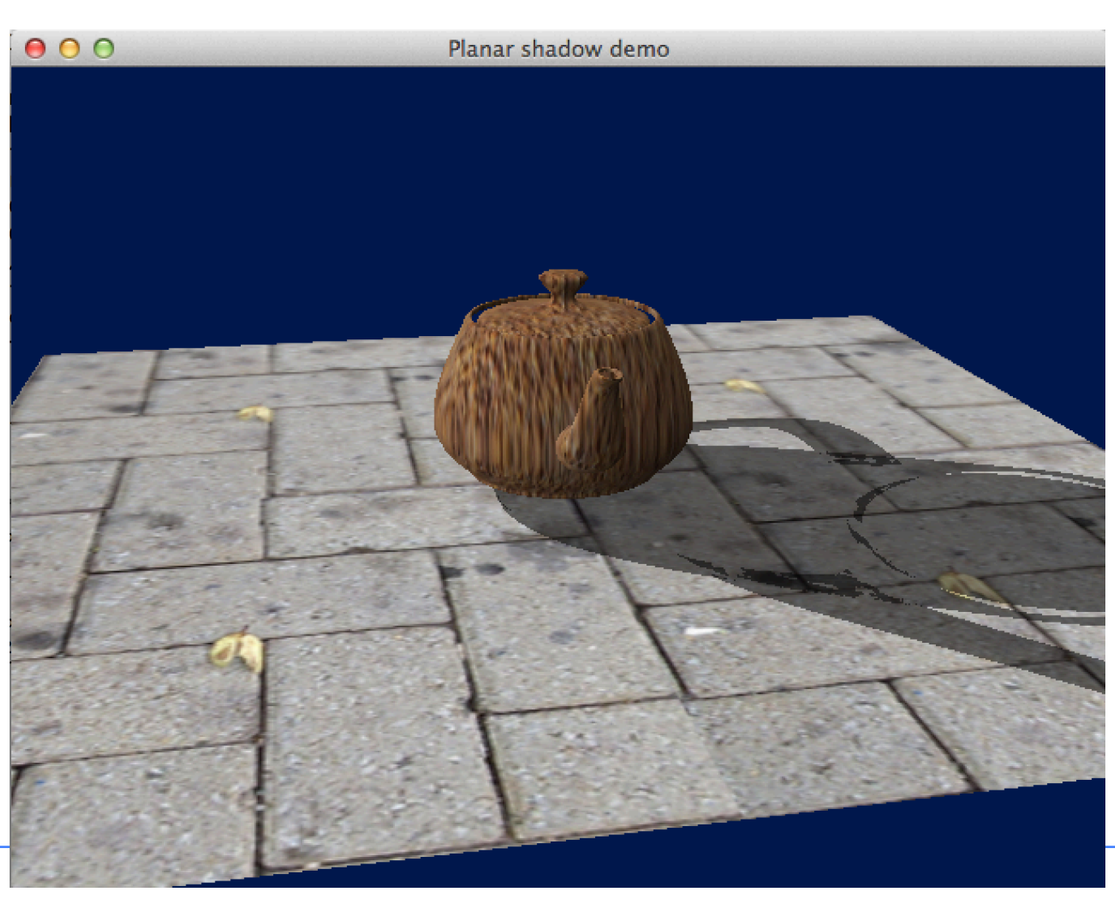
Skriv 1 om pass!
Annars keep!
Vi vill inte rita i skymd yta!

```
glStencilFunc(GL_ALWAYS, 1, 0xFFFFFFFF);
glStencilOp(GL_KEEP, GL_KEEP, GL_REPLACE); // 1 if pass
// Draw the floor
glBindTexture(GL_TEXTURE_2D, TextureArray[0]);
// Upload any shader variables here
DrawModel(floorModel);
```

Rita på vanligt sätt

Multipel blending

Ett fall för stencilbuffern!



Multipel blending

Vi använde bara stencilbuffern som binär mask. Den anger om vi skrivit en bakgrund i den eller ej. Men vi vill ju helst testa om vi skrivit skugga också!

Och detta visar sig vara ganska lätt!

Vi hade

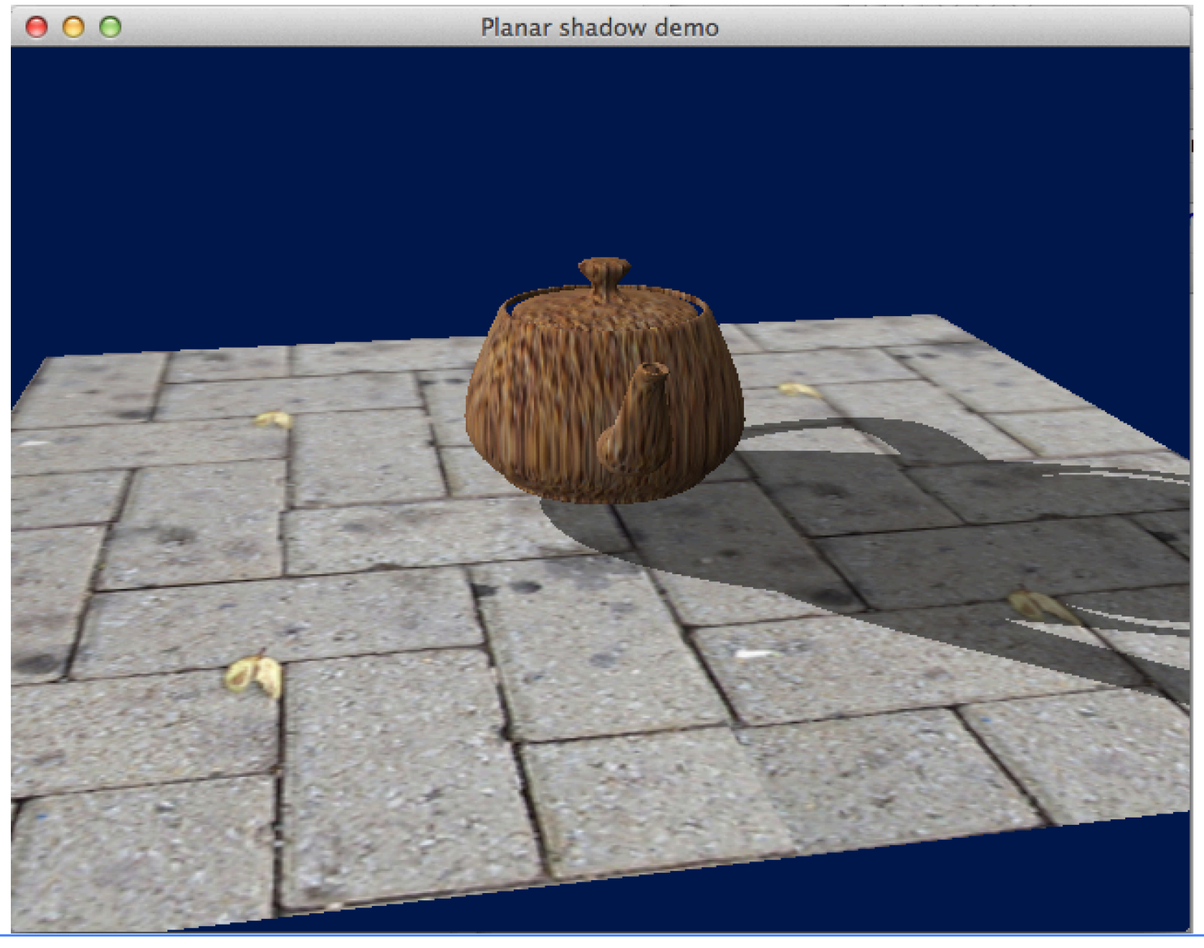
```
glStencilOp(GL_KEEP, GL_KEEP, GL_KEEP);
```

men vad sägs om denna?

```
glStencilOp(GL_KEEP, GL_KEEP, GL_INCR);
```

Dvs rita om stencil = 1 (enl glStencilOp) men öka med 1 om det går igenom!

Multipel blending korrigerat, resultat



Projektion på plana ytor

- Enkelt att göra med projektionsmatris och stencilbuffern

- Bara plana ytor! Bökigt och långsamt för flera ytor, kräver att stencilbuffern raderas och ritas om för varje yta!

- Problem vid blendning av komplexa objekt

Bättre kan vi! Shadow maps och shadow volumes ger bättre resultat!