

Skuggor

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!

Dynamiska skuggor

Metoder för dynamiska skuggor:

- Projektion på plana ytor
- Shadow maps
- Shadow volumes

Ingemar
Ragnemalm
ingis@isy.liu.se

Projektion på plan yta

Projicera objektet på en yta!

Projektionsmatris postmultiplikeras på modelview.

Objektet renderas utan textur, i önskad skuggas färg, blendas multiplikativt (som light map).



Ingemar
Ragnemalm
ingis@isy.liu.se

Projektionsmatrisen

Enkla projektionmatriser längs Z:

$$\begin{array}{cccc} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & f & 0 \\ 0 & 0 & -1 & 0 \end{array} \quad \begin{array}{cccc} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & f \end{array}$$

Rotera och translatera så det stämmer med önskat plan.

En titt på ett populärt demo: “Apron tutorials”

Hur koden kan se ut hittar man ofta i demos. “Apron” visar hur det kan göras... men riktigt varför får vi inte veta, och allt är inte rätt.

- Rita golvet i stencil, framebuffer och Z-buffer
- Projicera från ljuskälla till plan
- Rita objektet projicerat, som skugga
- Rita objektet normalt

Rita golvet i stencil, framebuffern och Z-buffer

```
glStencilFunc(GL_ALWAYS, 1, 0xFFFFFFFF);  
glStencilOp(GL_KEEP, GL_KEEP, GL_REPLACE);
```

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

```
// Draw the floor  
glBindTexture(GL_TEXTURE_2D, TextureArray[0]);  
glBegin(GL_QUADS);  
    glNormal3f(0, 1.0f, 0);  
    glTexCoord2f(1.0f, 1.0f); glVertex3f( 100.0f, 0, -100.0f);  
    glTexCoord2f(0.0f, 1.0f); glVertex3f(-100.0f, 0, -100.0f);  
    glTexCoord2f(0.0f, 0.0f); glVertex3f(-100.0f, 0, 100.0f);  
    glTexCoord2f(1.0f, 0.0f); glVertex3f( 100.0f, 0, 100.0f);  
glEnd();
```

← Rita på vanligt
sätt

Ingemar
Ragnemalm
ingis@isy.liu.se

Rita objektet under projektionstransformation

```
glColor4f(0.0f, 0.0f, 0.0f, 0.5f); // Svart, 50%  
  
// Disable light  
glDisable(GL_TEXTURE_2D);  
glDisable(GL_LIGHTING);  
glDisable(GL_DEPTH_TEST);  
// Enable blending  
glEnable(GL_BLEND);  
  
glStencilOp(GL_KEEP, GL_KEEP, GL_KEEP);  
  
// Calculate the projected shadow  
shadowmatrix(floorShadow, groundplane, lightPosition);  
glMultMatrixf((float *)floorShadow);  
  
// Draw our model  
glRotatef(20.0f, 0, 1.0f, 0);  
Draw_Model();
```

Därefter ritas objektet utan projektion, men textur

Ingemar
Ragnemalm
ingis@isy.liu.se

Projektionstransformation görs i “shadowmatrix”. Hur får man fram den då?

```
void shadowmatrix(float shadowMat[4][4], float groundplane[4], float lightpos[4])
{
    // Find the dot product between the light position vector and the ground plane
    normal
```

```
    float dot = groundplane[X] * lightpos[X] + groundplane[Y] * lightpos[Y] +
    groundplane[Z] * lightpos[Z] + groundplane[W] * lightpos[W];
```

```
    shadowMat[0][0] = dot - lightpos[X] * groundplane[X];
    shadowMat[1][0] = 0.f - lightpos[X] * groundplane[Y];
    shadowMat[2][0] = 0.f - lightpos[X] * groundplane[Z];
    shadowMat[3][0] = 0.f - lightpos[X] * groundplane[W];
```

```
    shadowMat[X][1] = 0.f - lightpos[Y] * groundplane[X];
    shadowMat[1][1] = dot - lightpos[Y] * groundplane[Y];
    shadowMat[2][1] = 0.f - lightpos[Y] * groundplane[Z];
    shadowMat[3][1] = 0.f - lightpos[Y] * groundplane[W];
```

$$\begin{matrix} f-L_x * G_x & -L_x * G_y & -L_x * G_z & -L_x * G_h \\ -L_y * G_x & f-L_y * G_y & -L_y * G_z & -L_y * G_h \\ -L_z * G_x & -L_z * G_y & f-L_z * G_z & -L_z * G_h \\ -L_h * G_x & -L_h * G_y & -L_h * G_z & f-L_h * G_h \end{matrix}$$

```
    shadowMat[X][2] = 0.f - lightpos[Z] * groundplane[X];
    shadowMat[1][2] = 0.f - lightpos[Z] * groundplane[Y];
    shadowMat[2][2] = dot - lightpos[Z] * groundplane[Z];
    shadowMat[3][2] = 0.f - lightpos[Z] * groundplane[W];
```

```
    shadowMat[X][3] = 0.f - lightpos[W] * groundplane[X];
    shadowMat[1][3] = 0.f - lightpos[W] * groundplane[Y];
    shadowMat[2][3] = 0.f - lightpos[W] * groundplane[Z];
    shadowMat[3][3] = dot - lightpos[W] * groundplane[W];
```

**Jamen... vad
f*n kom den
från då????!!!**

Ingemar
Ragnemalm
ingis@isy.liu.se

Låt oss göra vår egen projektionsmatris!

Givet:

Ljuskälla var som helst, projektionsplan genom origo, normalvektor längs Y.

Projicera längs Y, välj varianten med projektionsplan genom origo

$$\begin{matrix} f & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & f & 0 \\ 0 & -1 & 0 & f \end{matrix}$$

Dessutom krävs translation i X- och Z-led för att placera ljuskällan (PRP) på Y-axeln! Enligt principen från TSBK05 borde det bli:

$$T(L_x, 0, L_z) * P * T(-L_x, 0, -L_z)$$

ger:

$$\begin{matrix} f & -L_x & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & -L_z & f & 0 \\ 0 & -1 & 0 & f \end{matrix}$$

**Funkar, och den
här gången vet
vi varför!**

Ingemar
Ragnemalm
ingis@isy.liu.se

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!

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

Shadow maps

Avancerad skuggningsmetod som kräver relativt mycket av GPU'n.

- Måste kunna rendera till textur effektivt
- Måste stödja avancerad texturaritmetik (fragment shader)

Fördel: Behöver ingen kunskap om scenens innehåll, klarar alla sorters former. "Self-shadowing" inget problem.

Shadow maps

Tvåpassalgoritm:

1: Rendera från ljuskällan. Endast Z-buffern är av intresse! Z-buffern är vår “shadow map”, en 2D-funktion som anger avståndet till ljuskällan.

2: Z-buffern används i andra passet:

Z-buffern används som projicerad textur, men inte för att rita med!

Ingemar
Ragnemalm
ingis@isy.liu.se

Shadow maps

Z-buffern är en “djuptextur”. När den projiceras över scenen så nås varje fragmentberäkning av ett Z-värde.

A: Om fragmentet (pixeln) är belyst så är Z-bufferns värde lika med avståndet till ljuskällan

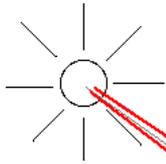
B: Om den är i skugga så är Z-värdet mindre än avståndet till ljuskällan.

Om vi kan testa denna likhet/olikhet så vet vi om pixeln är i skugga! Detta kan göras med fragment shader eller avancerad texturaritmetik.

Ingemar
Ragnemalm
ingis@isy.liu.se

Shadow maps - princip

Ljuskälla



Bildplan för
djupbilden

Kamera



Bildplan

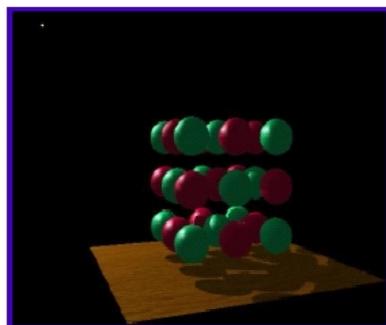
Via projicerad textur kan
djupbilden avläsas för
varje bildpunkt.

Jämförs med avstånd!

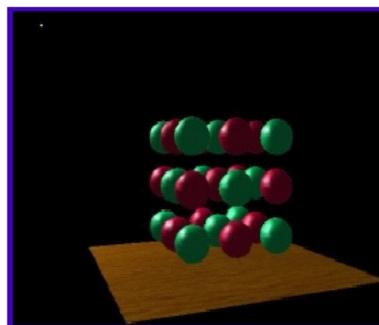
Ingemar
Ragnemalm
ingis@isy.liu.se

Shadow maps - exempel

Relativt komplicerad scen



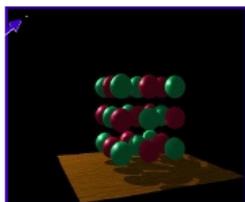
Med skuggor



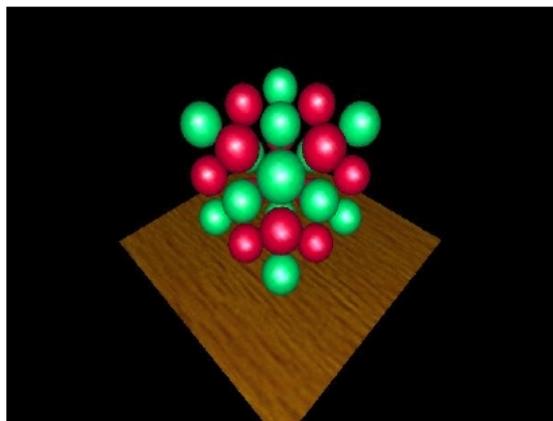
Utan skuggor

Ingemar
Ragnemalm
ingis@isy.liu.se

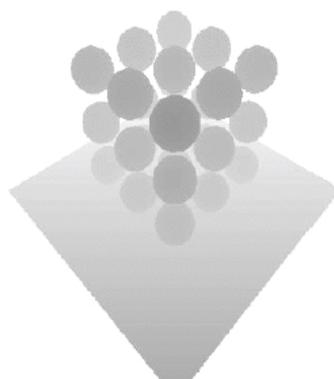
Shadow maps - exempel



Rendering av shadow map



Sedd från ljuskällan

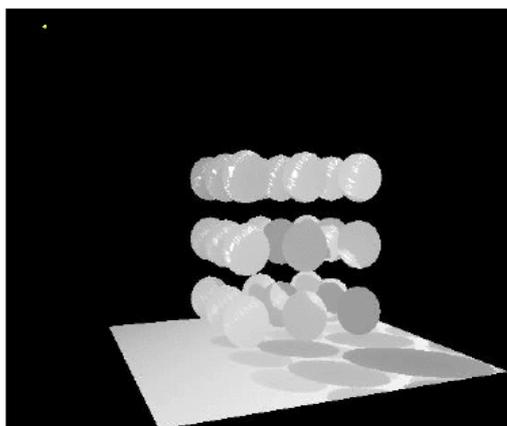


Z-buffer från ljuskällan

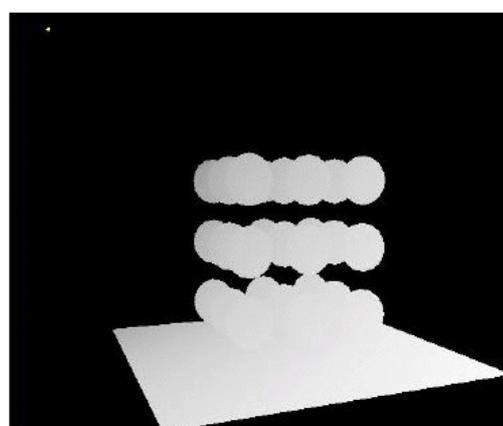
Ingemar
Ragnemalm
ingis@isy.liu.se

Shadow maps - exempel

Shadow map som projicerad textur



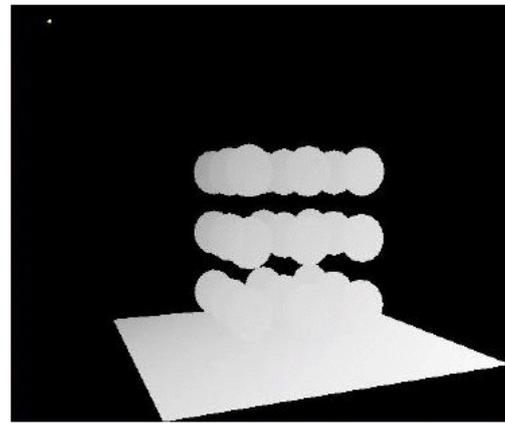
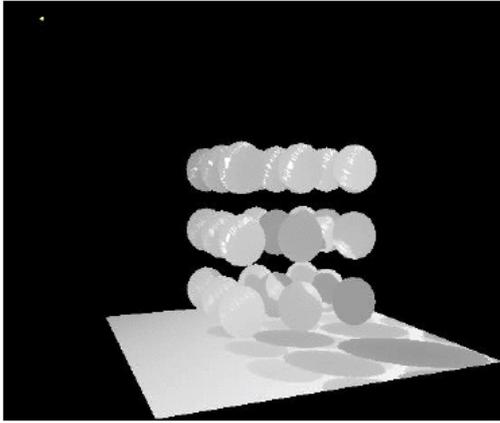
Z-buffer projicerad på
scenen



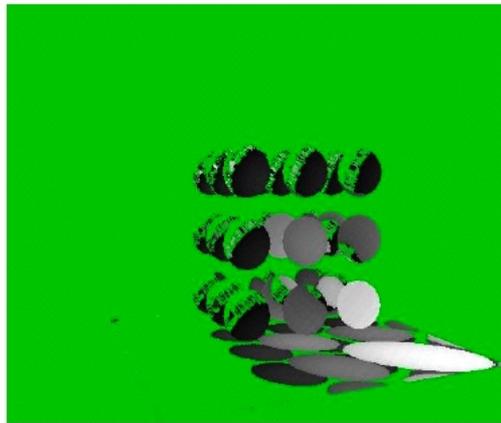
Avstånd till ljuskällan

När dessa är lika: ej i skugga!

Ingemar
Ragnemalm
ingis@isy.liu.se



Områden med avvikande värden:



Ingemar
Ragnemalm
ingis@isy.liu.se

Shadow maps - problem

Problem 1: “Lika” är ett dåligt villkor

Måste ha en viss marginal för att undvika artefakter.

För lite marginal: Nästan allting blir skugga (liknar Z-fightning)

För mycket marginal: Skuggor krymper

Måttligt problem!

Problem 2: Shadow buffer kräver hög upplösning för att inte ge kantartefakter

Ingemar
Ragnemalm
ingis@isy.liu.se

Shadow volumes (Stencil shadows)

En tredje skuggningsmetod.

Kräver mindre av GPU'n, men mer av CPU'n

Idé: "projicera" skuggande objekt till kantytor på en volym. Med hjälp av stencilbuffern kan vi avgöra om en viss punkt är inom eller utanför volymen.

Fördel: Undviker upplösningsberoendet som shadow buffer har.

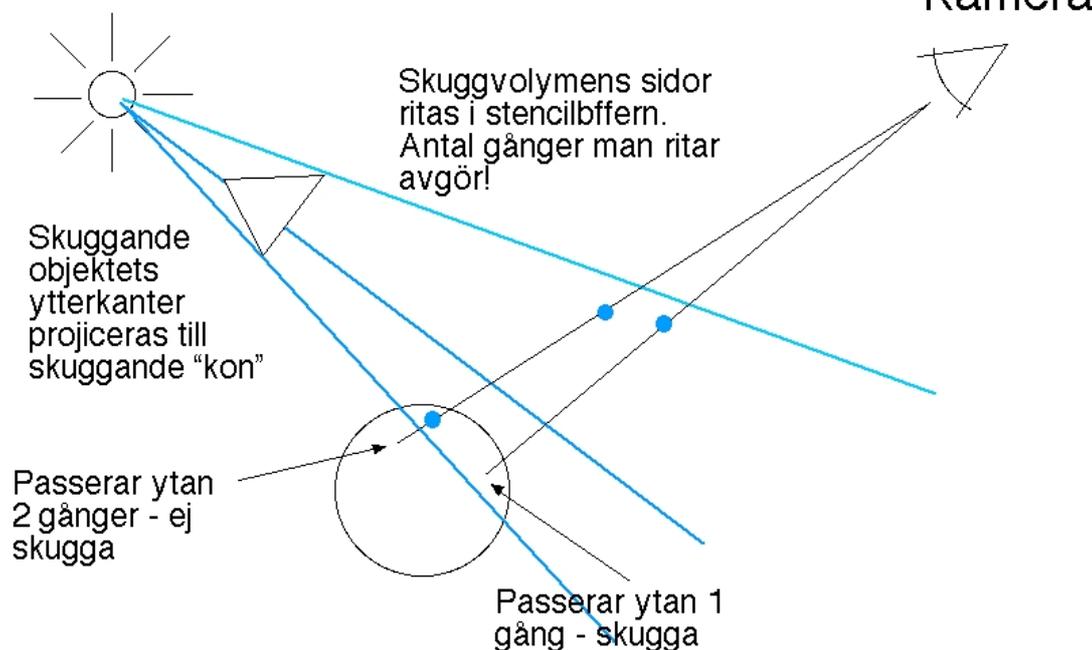
Ingemar
Ragnemalm
ingis@isy.liu.se

Shadow volumes

Princip

Ljuskälla

Kamera



Ingemar
Ragnemalm
ingis@isy.liu.se

Shadow volumes

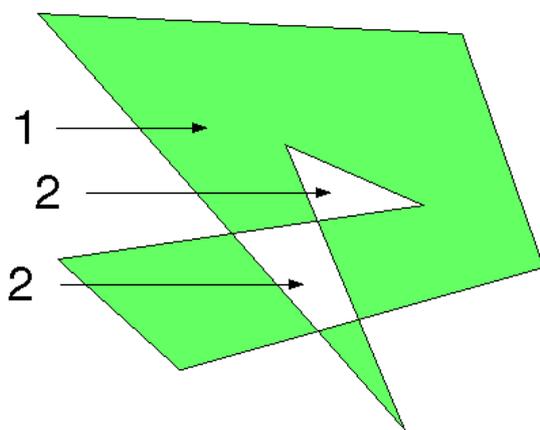
Algoritm:

- Bilda polygoner som utgör sidorna på skuggvolymen
- Initiera stencilbuffern till 1 om kameran är i skuggvolymen, annars 0
- Rita scenen med full belysning och Z-buffer
- Rita alla framsidor i stencilbuffern, med inkrement
- Rita alla baksidor i stencilbuffern, med dekrement
- Rita skuggor i alla pixlar där stencilbuffern > 0

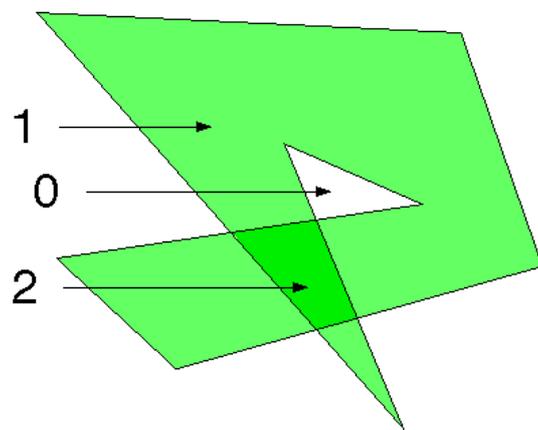
Ingemar
Ragnemalm
ingis@isy.liu.se

Non-zero winding number

Shadow volumes är samma princip i 3D!



Odd-even rule, udda antal kanter är "innanför" (fel)

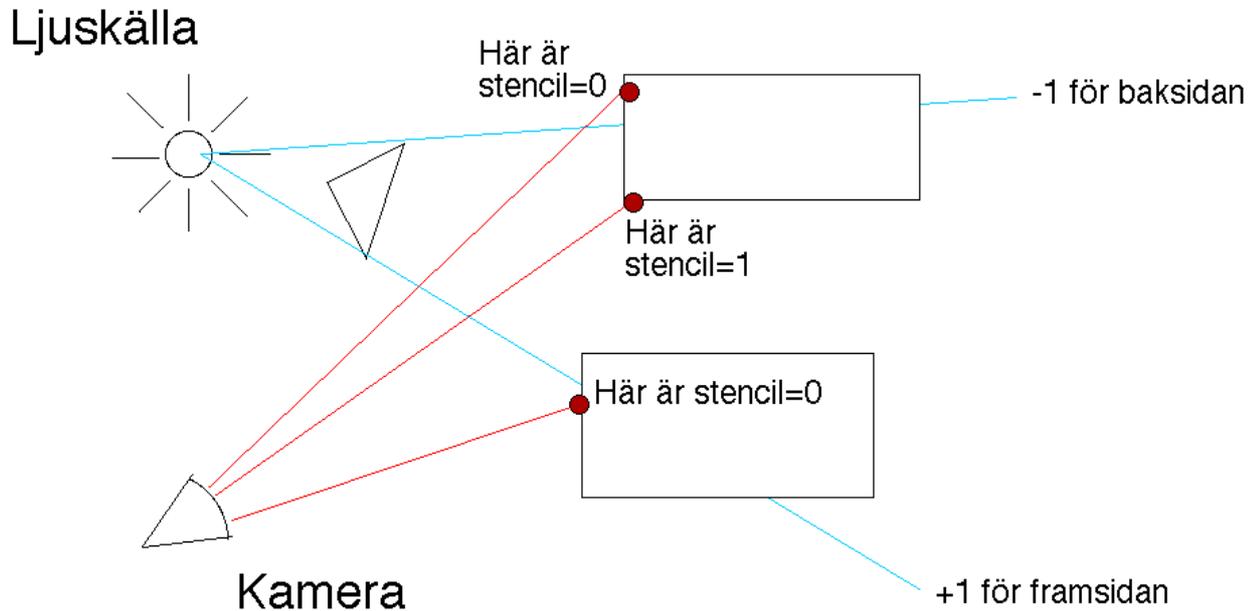


Non-zero winding number rule, summan av "uppkanter" och "nerkanter" är noll när vi är utanför (rätt)

Ingemar
Ragnemalm
ingis@isy.liu.se

Shadow volumes

Stencilbufferns uppdatering och beslut



Ingemar
Ragnemalm
ingis@isy.liu.se

Shadow volumes

Fördelar:

- Inga upplösningsproblem
- Låga krav på GPU, stencilbuffer räcker

Nackdelar:

- Bökigt att beräkna volymen
- Självsuggande objekt måste brytas isär

Ingemar
Ragnemalm
ingis@isy.liu.se

Skuggor

Sammanfattning

- Projektionsskuggor lätta att göra, men fungerar bara på plana ytor
- Shadow maps fungerar mycket bra på moderna GPU'er, klarar de flesta scener fint, men är minneskrävande och fungerar inte på enklare GPU'er
- Shadow volumes är beräkningskrävande och något begränsade, men fungerar på de flesta GPU'er

Ett tämligen svårt problem som börjar få sin lösning.
En självklarhet för högklassig grafik!

Ingemar
Ragnemalm
ingis@isy.liu.se

Mer skuggor

Vad mer kan man göra?

Shadow volume BSP trees: En metod för att hantera många skuggande objekt i samma scen

Shadow penumbras: Metoder för att få “mjuka skuggor” i realtid

Det finns mycket att göra. Ändå bygger det mesta på resultat från 70- och 80-talet.

Ingemar
Ragnemalm
ingis@isy.liu.se