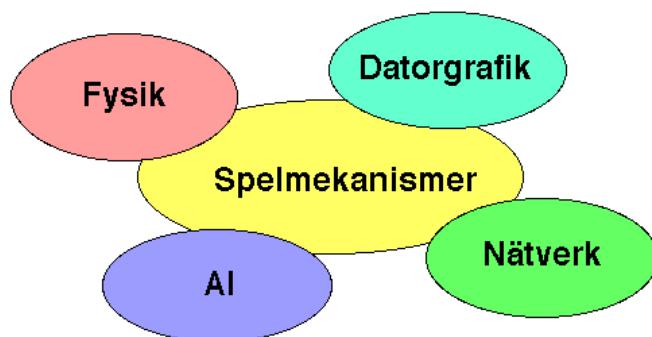


TSBK 10

Teknik för avancerade datorspel

Ingemar Ragnemalm, ISY



Ingemar
Ragnemalm
ingis@isy.liu.se

Föreläsning 3

- **Skuggor, definitioner**
- **Plana skuggor**
- **Skuggmappning**
- **Skuggvolymer**
- **Mjuka skuggor**

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!

Ingemar
Ragnemalm
ingis@isy.liu.se

Skuggor

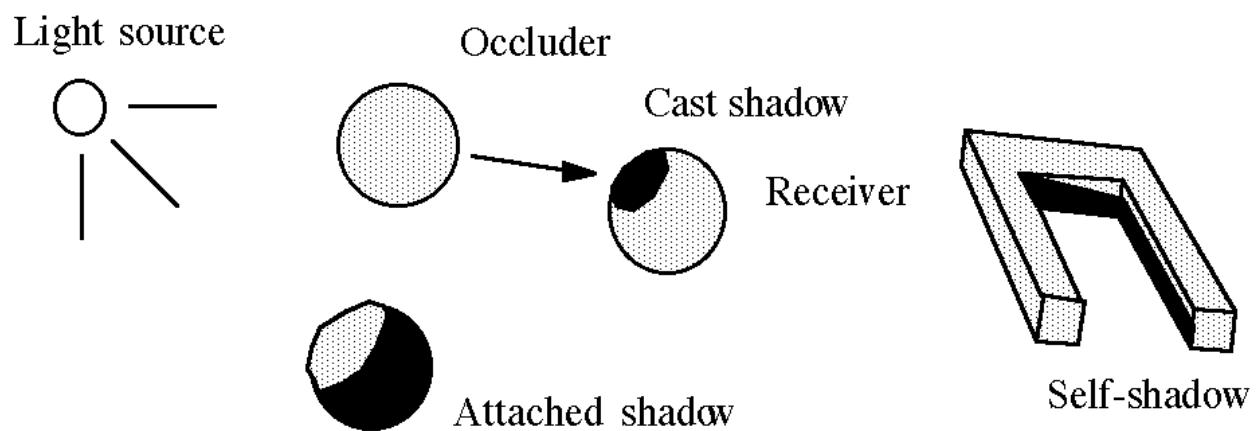
Vi vet redan att vi kan göra skuggor på flera sätt:

- Strålföljning**
- Radiosity**
- Light mapping baserad på dessa**

men ingen av dem ger dynamiska skuggor!

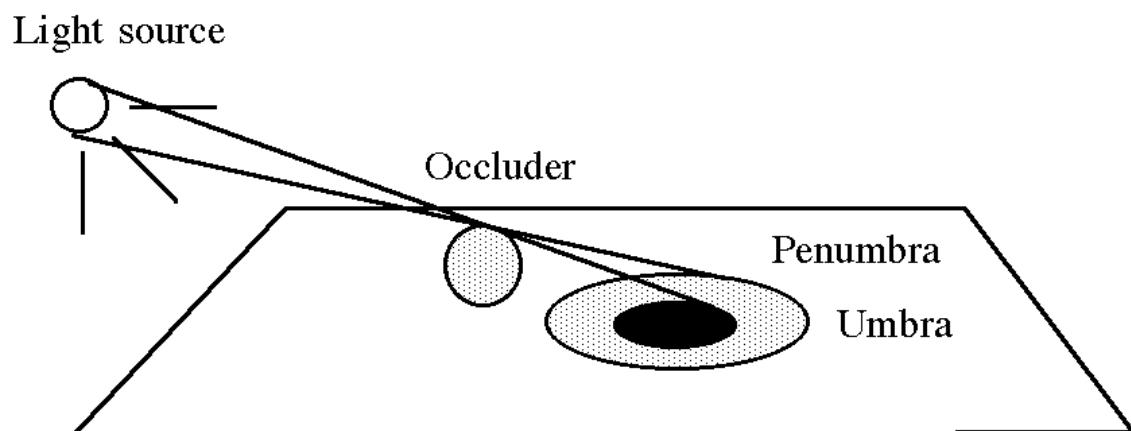
Definitioner

Occluder
Receiver
Attached shadow
Self-shadow



Definitioner

Umbra
Penumbra (yttre & inre)



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 postmultipleras
på modelview.**

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



Ingemar
Ragnemalm
ingis@isy.liu.se

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 multipass

Ingemar
Ragnhjemalm
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.

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



Vi väljer denna

Ingemar
Ragnhjemalm
ingis@isy.liu.se

Projektionstransform

Samma princip som många liknande fall i grundkursen.

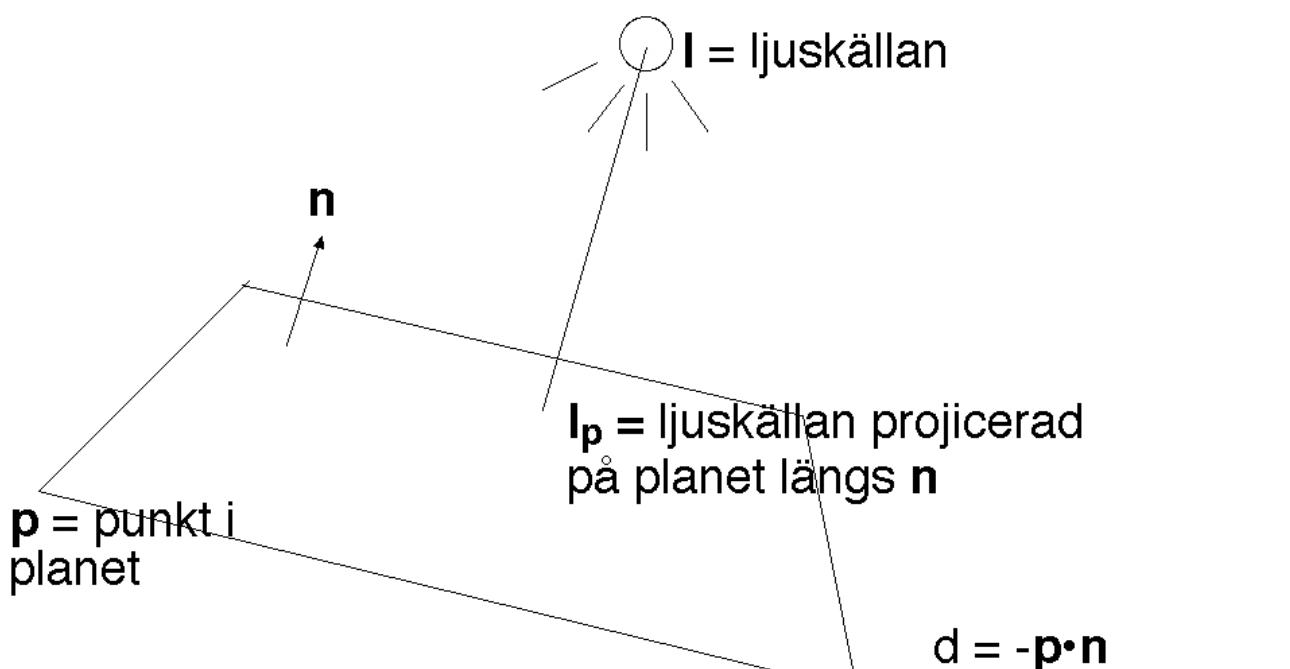
Låt I_p vara ljuskällan projicerad på planet.

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

OBS! Fel i boken sid 38. $T(-I)$ skall vara $T(-I_p)$

Ingemar
Ragnemalm
ingis@isy.liu.se

Variabler i projektionen



Kod för projektion

```

function shadowmatrix2(groundplane, lightpos: Point3D; d: GLfloat): Matrix4D;
var
  f: GLfloat;
  M, t1, r, ir, t2, p: Matrix4D;
  v: Point3D;
begin
  Normalize(groundplane);
  f := groundplane * lightpos + d; // Distance light source to plane

  // Calculate the projection of the light source onto the plane
  // to translate it to origin
  v := lightpos - groundplane * f; // l - n*(n•l+d)

  t1 := T(v);
  r := AxisToZ(groundplane);
  p := ProjZ(f);
  ir := Transpose(r);
  t2 := T(-v);

  M := t1 * ir * p * r * t2;
  return M;
end;

```

(Nej det är inte C.)

Alternativ kod från McReynolds demo
“projshadow.c”. “Svart låda”-kod!

```

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];

    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???!!!

Två olika lösningar på samma problem.

**Min och McReynolds lösning resulterar i exakt
samma matris!**

(Att bevisa detta känns dock mindre viktigt.)

**Liten visdomsgrej: McReynolds kod funkar bra
om man vet vad den gör och antar, som normerad
normalvektor.**

***Svarta lådor är värdelösa om man inte vet vad
som finns i dem!***

Rita golvet i stencil, framebuffern och Z-buffer

```
glStencilFunc(GL_ALWAYS, 1, 0xFFFFFFFF);
glStencilOp(GL_KEEP, GL_KEEP, GL_REPLACE); ← Skriv 1 om
                                                bass! Annars
                                                keep! Vi vill inte
                                                rita i skymd 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); ← Rita på vanligt
    glTexCoord2f(0.0f, 1.0f); glVertex3f(-100.0f, 0,-100.0f); ← sätt
    glTexCoord2f(0.0f, 0.0f); glVertex3f(-100.0f, 0, 100.0f);
    glTexCoord2f(1.0f, 0.0f); glVertex3f( 100.0f, 0, 100.0f);
glEnd();
```

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, 0.1f, 0);  
Draw_Model();
```

Ingen
textur!

Därefter ritas objektet utan projektion, med textur och shading

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!
- Problem vid blendning av komplexa objekt

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

Ingemar
Ragnemalm
ingis@isy.liu.se

Shadow maps/Skuggmappning

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.

Ingemar
Ragnemalm
ingis@isy.liu.se

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

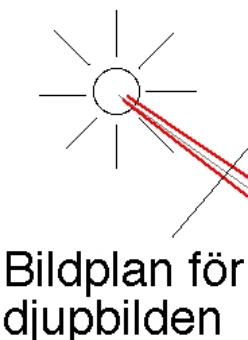
B: Om den är i skugga så är Z-värdet mindre än avståndet till ljuskällan.

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



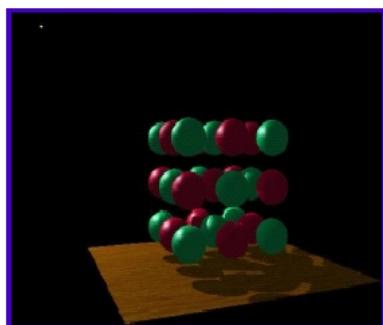
Bildplan

Via projicerad textur kan
djupbilden avläsas för
varje bildpunkt.

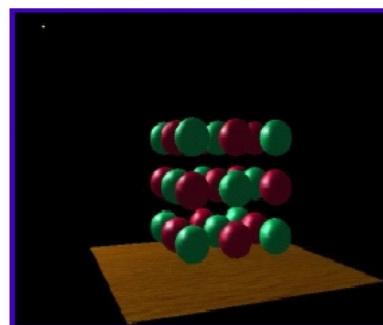
Jämförs med avstånd!

Shadow maps - exempel

Relativt komplicerad scen



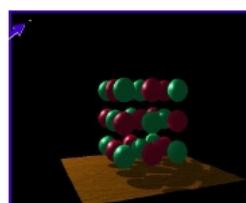
Med skuggor



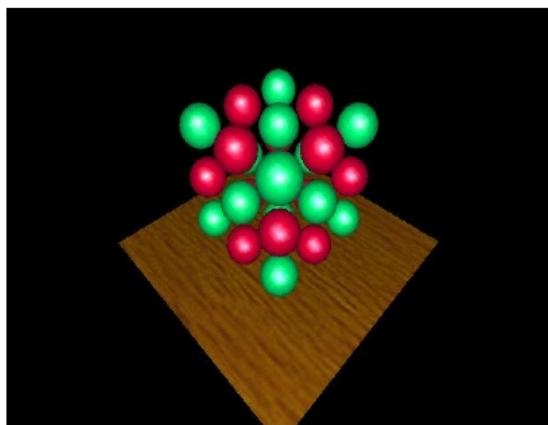
Utan skuggor

Ingemar
Ragnemalm
ingis@isy.liu.se

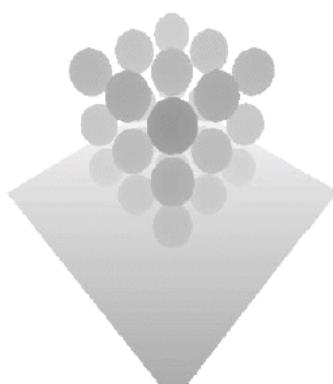
Shadow maps - exempel



Rendering av shadow map



Sedd från ljuskällan

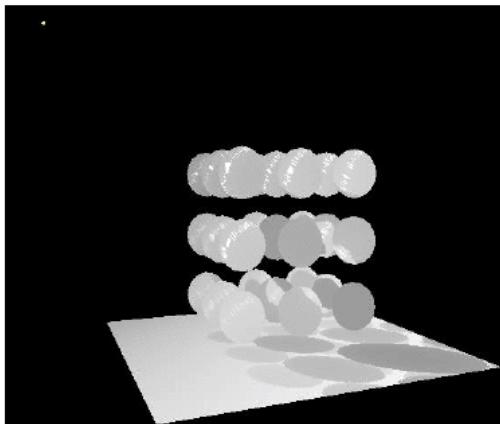


Z-buffer från ljuskällan

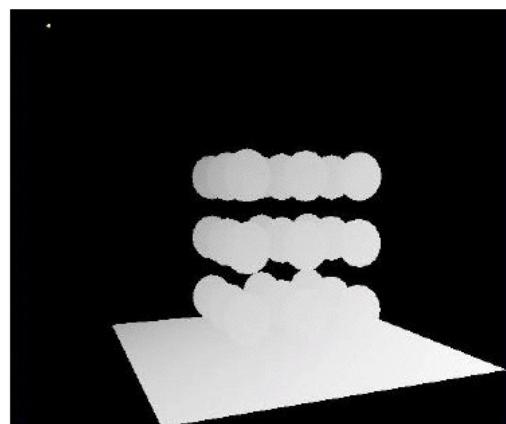
Ingemar
Ragnemalm
ingis@isy.liu.se

Shadow maps - exempel

Shadow map som projicerad textur



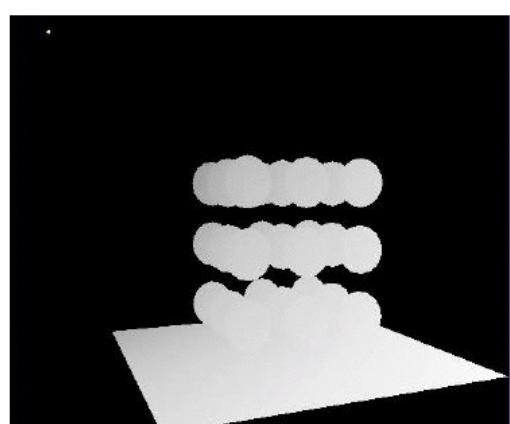
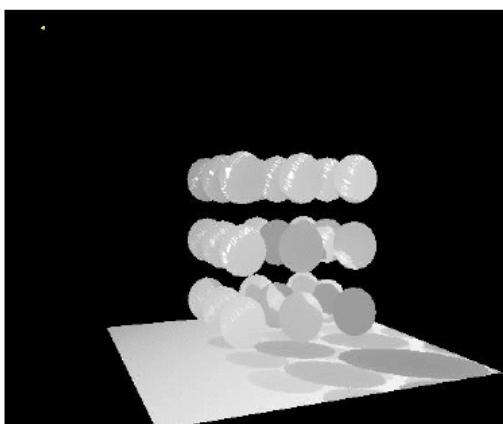
Z-buffer projicerad på scenen



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-fighting)

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 buffer-exempel

**Baserat på SGI-demo, förenklat,
omskrivet för att undvika special-
extensions.**

Ingemar
Ragnemalm
ingis@isy.liu.se

Shadow buffer-exempel

Viktiga detaljer

- Placera kameran i ljuskällan
 - Z-buffer till textur
 - Projicera textur
- Utför avståndsjämförelse

Ingemar
Ragnemalm
ingis@isy.liu.se

Z-buffer till textur

```
glCopyTexImage2D(GL_TEXTURE_2
D, 0, GL_DEPTH_COMPONENT, 0,0,
width, height,0);
```

glCopyTexImage och glCopyTexSubImage är
användbara, pålitliga och snabba!

Ingemar
Ragnemalm
ingis@isy.liu.se

Koordinatsystem

- **Modell-världs-transformationer appliceras dubbelt! Både på MODELVIEW och TEXTURE!**
- **Följer i stort texturprojektion... nästan.**

Ingemar
Ragnemalm
ingis@isy.liu.se

Projicera textur

Följer i stort texturprojektion... nästan.

```
glGetFloatv(GL_PROJECTION_MATRIX, perspective_mat);  
// Get the projection matrix  
glMatrixMode(GL_TEXTURE);  
glLoadIdentity();  
glTranslatef(0.5, 0.5, 0.4994); // Scale and bias transformation  
glScalef(0.5, 0.5, 0.5);  
glMultMatrixf(perspective_mat); // Multiply it on the texture matrix  
glMultMatrixf(modelview_mat); // and the lightsource modelview too
```

Vart tog invers world-to-view vägen???

Ingemar
Ragnemalm
ingis@isy.liu.se

Utför jämförelsen

En fråga om inställningar!

```
// Set up fixed pipeline shadow mapping  
glTexParameteri(GL_TEXTURE_2D,  
                 GL_TEXTURE_COMPARE_MODE,  
                 GL_COMPARE_R_TO_TEXTURE);  
glTexParameteri(GL_TEXTURE_2D,  
                 GL_TEXTURE_COMPARE_FUNC,  
                 GL_EQUAL);
```

plus objektplan för TexGen

Ingemar
Ragnemalm
ingis@isy.liu.se

Shadow buffer

Obekväma detaljer...

GL_OBJECT_LINEAR eller GL_EYE_LINEAR?

Behövs inversmatris eller ej?

Dessutom skum texturhantering i SGI-demo

En del “magi” – varningssignal!
Risk för strul?

**Shaderimplementation behövs för avancerade
lösningar! (Dessutom kan dessa kringgå
egendomigheterna ovan.)**

Ingemar
Ragnemalm
ingis@isy.liu.se

Shadow buffer i shader

I shader kan man ”finlira” bättre med artefakterna, samt skapa mjuka skuggor.

Dessutom kommer man undan de störande ”fulhack” som OpenGL tyvärr innehåller i detta fall.

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.

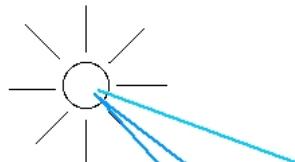
Fördel: Undviker upplösningsberoendet som shadow buffer har.

Ingemar
Ragnemalm
ingis@isy.liu.se

Shadow volumes

Princip

Ljuskälla



Kamera



Skuggvolymens sidor
ritas i stencilbuffern. Antal
gånger man ritar avgör!

Skuggande
objektets
ytterkanter
projiceras till
skuggande "kon"

Passerar ytan 2
gånger - ej
skugga

Passerar ytan 1
gång - skugga

Ingemar
Ragnemalm
ingis@isy.liu.se

Shadow volumes

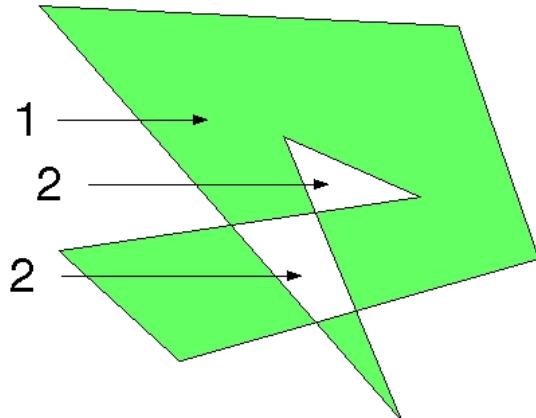
Algoritm:

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

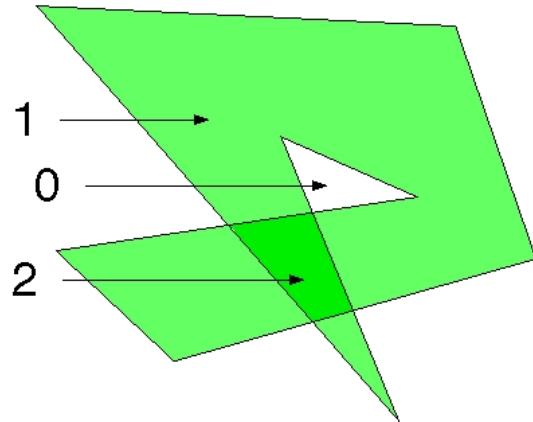
Ingemar
Ragnemalm
ingis@isy.liu.se

Non-zero winding number

Shadow volumes är samma princip i 3D!



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



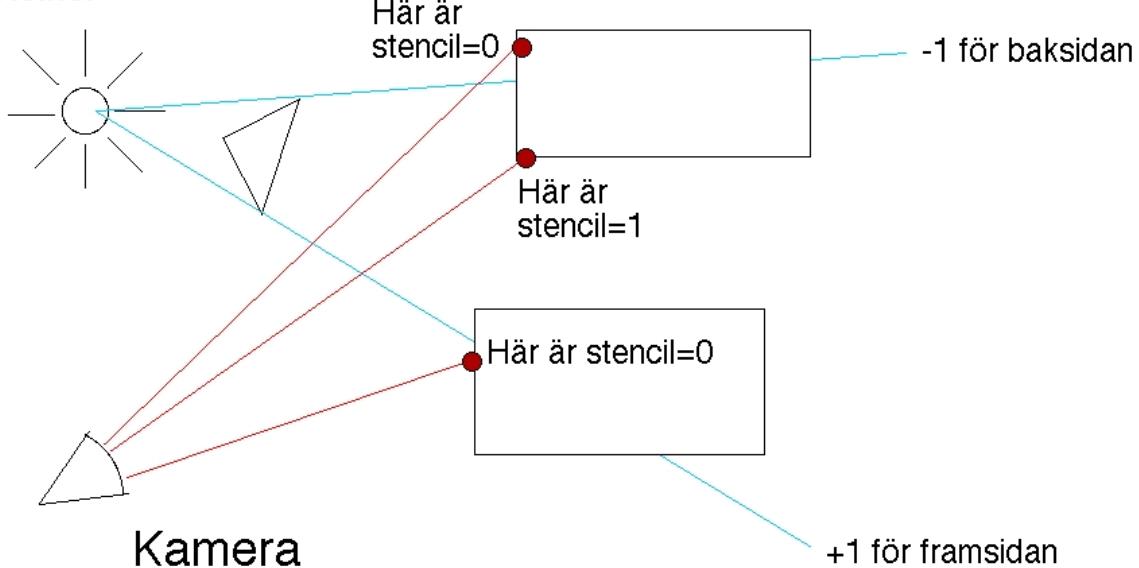
Non-zero winding number
rule, summan av "upp-
kanter" och "ner-kanter" är
noll när vi är utanför (rätt)

Ingemar
Ragnemalm
ingis@isy.liu.se

Shadow volumes

Stencilbufferns uppdatering och beslut

Ljuskälla



Ingemar
Ragnemalm
ingis@isy.liu.se

Shadow volumes

Fördelar:

- Inga upplösningsproblem, bra “hårda skuggor”
 - Låga krav på GPU, stencilbuffer räcker
 - Klarar självskuggning
 - Klarar alla riktningar

Nackdelar:

- Bökigt att beräkna volymen
- Många renderingspass över samma yta

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

Mjuka skuggor

Simulera skuggor från icke punktformiga ljuskällor

- Accumulator buffer
- Percentage Closer Filtering
 - Smoothies
 - Soft shadow volumes
- Single sample soft shadows

Ingemar
Ragnemalm
ingis@isy.liu.se

Mjuka skuggor = ljuskällor med signifikant utsträckning

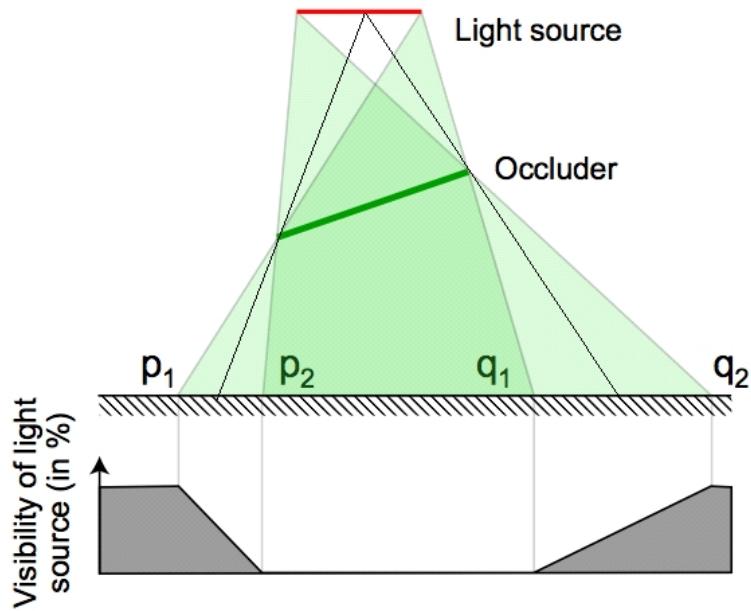


Figure 12: Percentage of a linear light source that is visible.

Mjuka skuggor med accumulator buffer

Enkel men ineffektiv lösning

Gör flera renderingar med ljuskällan i olika positioner.

Variant: Rendera enbart skuggor flera gånger.

Kräver många renderingar innan det blir snyggt.
Omöjlig balans för många scener.

Percentage Closer Filtering

Vanligaste? Den viktigaste för “allmänbildning” i ämnet.

Relativt enkel metod som beräknar skuggningsgrad ur bilden/skuggmappen.

Uniform filtrering i bildplanet -> penumbrastorleken dåligt avbildad, konstant storlek i bildplanet.

Ingermar
Ragnemalm
ingis@isy.liu.se

Percentage Closer Filtering

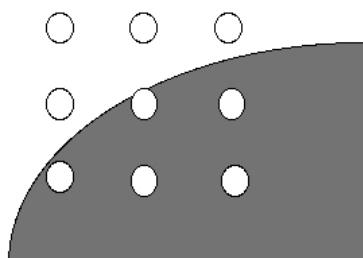
Testa n punkter kring samplingspunkten.

Om k av dessa är belysta

belysningsnivå k/n

OBS att bilddata inte filtreras! Räkning av skugga/ej skugga.
Materialtextur etc är orört, får enbart ljussättning.

Man kan INTE skapa mjuka skuggor med konventionella lågpassfilter!



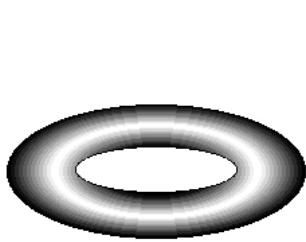
4 belysta av 9 -> ljusnivå 4/9

Smoothies

Smoothies “Fake” till och med enligt uppfinnaren!

Yttre penumbra simuleras med extrakanter i utkanten av occluderns projektion.

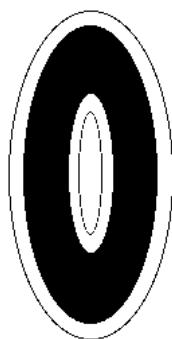
Shadow map + rendering av kanterna.



Objekt



Projicerad profil



“Smoothies” läggs till i kanterna

Ingemar
Ragnemalm
ingis@isy.liu.se

Soft Shadow Volumes

**Soft Shadow Volumes mer ambitiös, multipassmetod med både inre och yttre penumbra.
Besläktad med Smoothies.**

Single Sample Soft Shadows

Besläktad med PCF. Beräknar avstånd till närmaste skugga. Vissa problem med att närmaste skugga inte alltid är den som har närmast penumbra!

En djungel av algoritmer för mjuka skuggor finns!

Ingemar
Ragnemalm
ingis@isy.liu.se

Slutsatser om skuggor

**Mycket viktig aspekt i spel och annan modern
realtidsgrafik!**

**Skuggvolymer ger hög kvalitet men är krävande för
komplexa modeller - och komplexiteten ökar.**

**Skuggmappning har artefakter, men ger speciellt
bra resultat för mjuka skuggor.**

**Många algoritmer för mjuka skuggor. Skall man
nöja sig med PCF eller gå längre?**