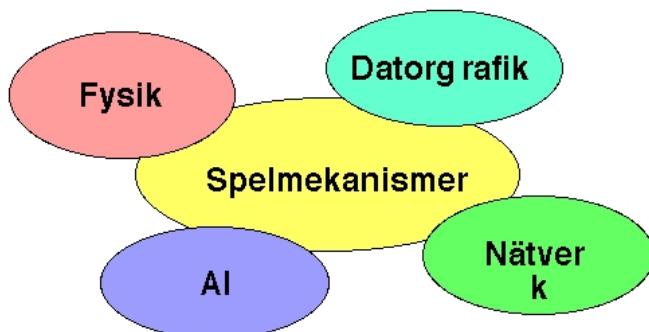


TSBK 10

Teknik för avancerade datorspel

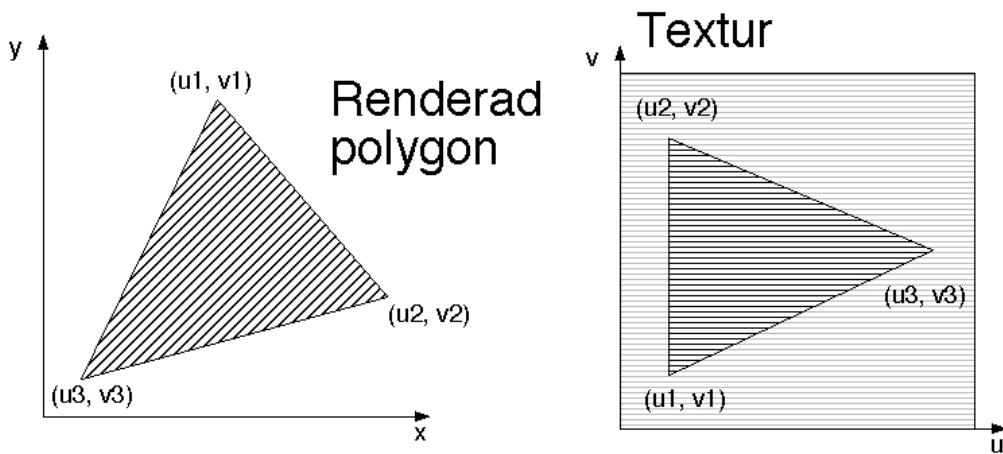
Ingemar Ragnemalm, ISY



Ingemar
Ragnemalm
ingis@isy.liu.se

Texturmappning

En pixelbild sträcks över en polygon och renderas med den, för detalj och realism.



- Mappningen djupberoende för rätt perspektiv
- Texturer läses in i VRAM, på GPU
- Flera texturer kan ritas på samma gång (mer om det senare i kursen)
- Inte bara materialavbildningar, även ljus, reflektioner...

Ingemar
Ragnemalm
ingis@isy.liu.se

Texturmappning

I TSBK05/07: 2D-texturer för materialytor, koordinater gavs manuellt eller med `glTexGen`.

Låt oss gå lite längre:

- Fler dimensioner
- Texturplacering
- Projicerade texturer
- Multipass
- Multitexturering
- Rendering till textur
- Optimering

Ingemar
Ragnemalm
ingis@isy.liu.se

Texturer i 1, 2, 3 dimensioner

1D-texturer: rad av texlar

2D-texturer: som vanligt

3D-texturer: volymdata

Ingemar
Ragnemalm
ingis@isy.liu.se

Texturer i 1 och 3 dimensioner

1D-texturer:

- Höjdfärgkodning
- Ljusdämpning

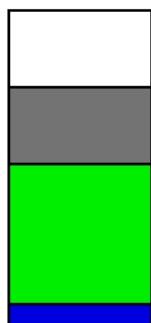
3D-texturer:

- 3D-visualisering

Ingemar
Ragnemalm
ingis@isy.liu.se

1-dimensionell textur för höjdfärgkodning

1D texture



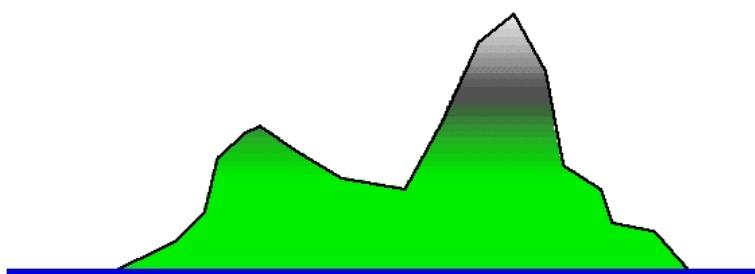
White (snow)

Grey (rock)

Green (grass)

Blue (water)

Terrain with 1D texture



Kombineras med andra texturer för detalj

Vi kan väl redan:

Generera texturkoordinater:

- linjär mappning
- cylindrisk mappning
- sfärisk mappning

Texturinställningar: repeat/clamp

Texturfilter, near/far, närmaste granne, linjär interpolation...

Mipmappning

Ingemar
Ragnemalm
ingis@isy.liu.se

Texturplacering

När vi använder glTexGen, hur får vi rätt täthet och rätt placering?

Placeringen justeras med spec av objektplan (en per texturkoordinat):

`glTexGenfv(GL_S, GL_OBJECT_PLANE, plane);`

`GLfloat plane[] = {1, 0, 0, 0.5}`

Ger möjlighet att sätta rotation, translation skalning, till och med skevning!

Ingemar
Ragnemalm
ingis@isy.liu.se

Texturmatrisen

Men objektplanen är ändå inte helt generella!

För ännu större flexibilitet: texturmatris!

```
glMatrixMode(GL_TEXTURE);
glRotatef();
glTranslatef();

-
glMatrixMode(GL_MODELVIEW); // Ställ tillbaka
```

Tillåter alla transformationer som en 4x4-matris kan representera - till och med projektion!

Kan använda de vanliga transformanropen - smidigt!

Ingemar
Ragnemalm
ingis@isy.liu.se

Multipass-texturering

Man kan rita samma yta flera gånger - och det kan vara meningsfullt!

Görs enklast - men inte snabbast - genom att helt enkelt rita geometrin flera gånger, med olika texturer.

Kräver att man ändrar Z-buffern till att rita på lika, LEQUAL i stället för LESS:

```
drawMyShape();
glDepthFunc(GL_LEQUAL); // Rrita även på lika
(Cändra texturinformation)
drawMyShape();
glDepthFunc(GL_LESS); // Ställ tillbaka
```

Ingemar
Ragnemalm
ingis@isy.liu.se

Exempel multipass och texturplacering

Två texturer på samma objekt.

Använder transparens, glBlendFunc specar hur den andra ritas på den ena.

GL_CLAMP kan begränsa till ett enda värde

Texturmatrisen används för rotation

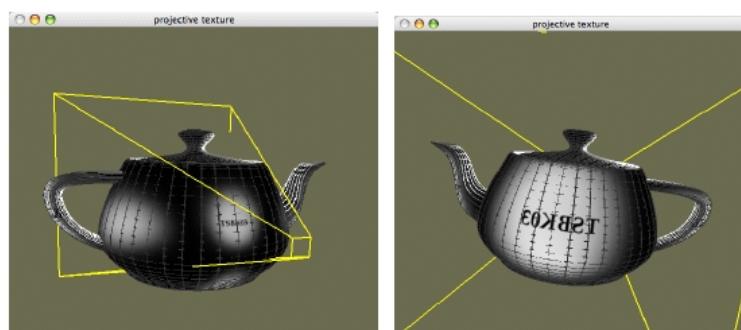
Ingemar
Ragnemalm
ingis@isy.liu.se

Projicerade texturer

**Specialfall av texturplacering:
Projiceringsmatris!**

Kan appliceras både i texturmatrisen eller i objektplanen.

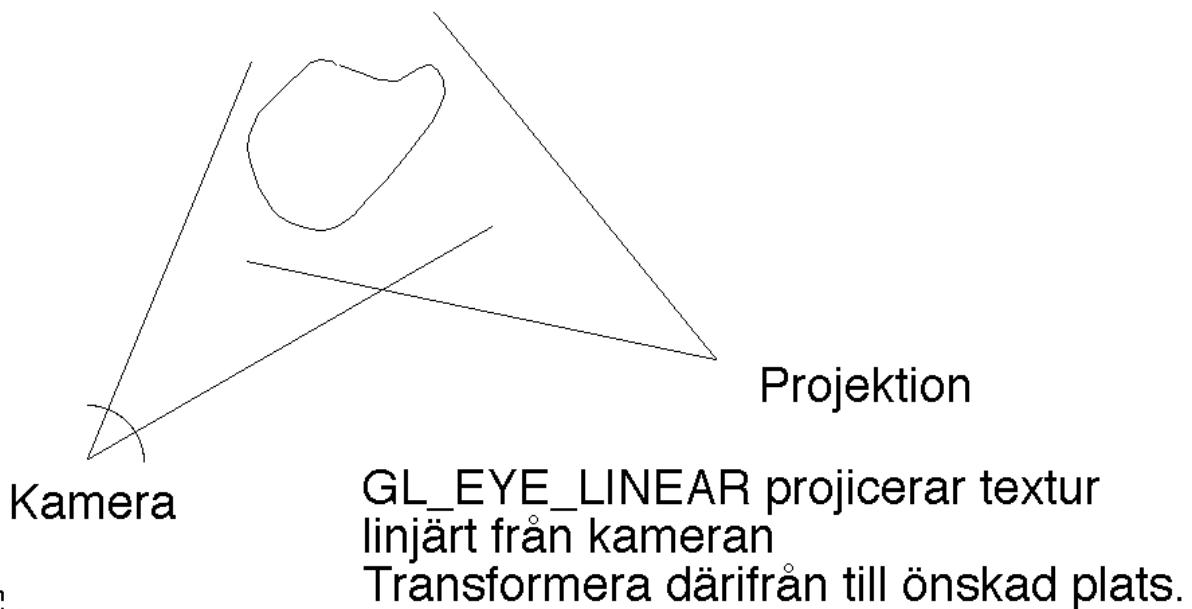
Viktiga för skugggenerering!



Ingemar
Ragnemalm
ingis@isy.liu.se

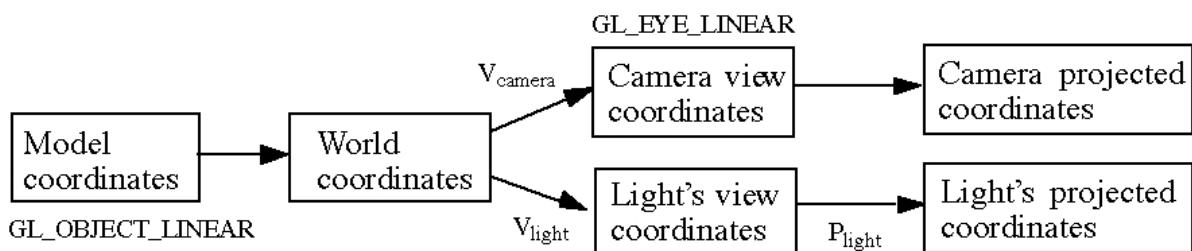
Projicerade texturer

En projektionsmatris skall in - men var?



Projicerade texturer

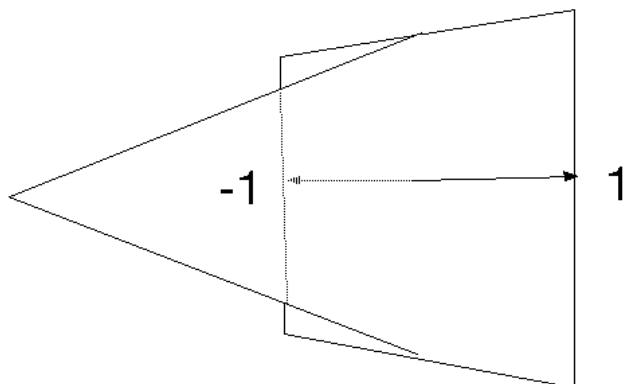
Backa från vykoordinater till
världskoordinater, sedan “framåt” till
önskad projektion.



$$B^* P_{light} * V_{light} * V_{camera}^{-1}$$

Projicerade texturer

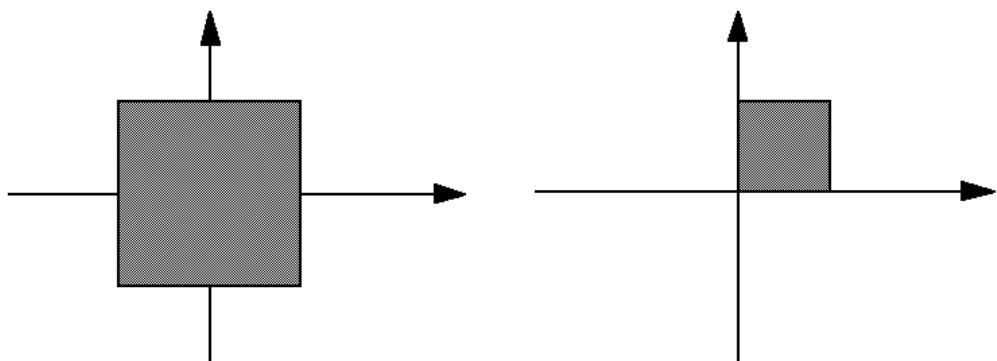
glFrustum och gluPerspective ger
projektion till normerade
skärmkoordinater. (-1 till 1)
Passar inte texturer! (0 till 1)



Ingemar
Ragnemalm
ingis@isy.liu.se

Scale and Bias

Anpassar (-1 till 1) till (0 till 1)



glTranslatef(0.5, 0.5, 0.0);
glScale(0.5, 0.5, 1.0);

$B^*P_{\text{light}}^*V_{\text{light}}^*V_{\text{camera}}^{-1}$

Ingemar
Ragnemalm
ingis@isy.liu.se

Projicerade texturer

Hela transformationen i OpenGL

```
glMatrixMode(GL_TEXTURE);  
glLoadIdentity();
```

```
glTranslatef(0.5, 0.5, 0.0);           B (Scale and bias)  
glScale(0.5, 0.5, 1.0);
```

```
glFrustum(left, right, bottom, top, near, far);    Plight
```

```
glMultMatrix(projectionPlacement);          Vlight  
glMultiMatrix(inverseCameraPlacement);      V-1camera
```

Ingemar
Ragnemalm
ingis@isy.liu.se

Multitexturering

Multipass är bra men långsamt!

Dagens GPU'er har flera texturenheter. Flera texturer kan ritas i en pipeline! Löser hastighetsproblemet!

Finns som ARB redan i OpenGL 1.2, senare som standard.

Nyckeln till mycket av modern grafikkvalitet! En självklarhet i "seriös" realtidsgrafik.

Ingemar
Ragnemalm
ingis@isy.liu.se

Multitexturering

Kan utföras antingen i shader eller med fix pipeline.

Med shader ingick i grundkursen.

Låt oss titta lite på hur det går till utan shaders.

(Mycket är samma.)

Ingemar
Ragnemalm
ingis@isy.liu.se

Multitexturering

Välj texturenhet:

```
glActiveTexture(GL_TEXTURE0);
```

Speca texturkoordinater för varje texturenhet:

```
glMultiTexCoord2f(...);
```

Varje texturenhet har sitt eget tillstånd, med:

- Textur (texturobjekt)
- Filterparametrar
- Texturmatris
- Texturkoordinatgenerering
- Texturkoordinat-array

Ingemar
Ragnemalm
ingis@isy.liu.se

Texturaritmetik

Viktig fråga: Hur kombineras texturerna vid multipass och multitexturering?

Multipass: glBlendFunc

Multitexture: Sätts med glTexEnv!

GL_MODULATE multiplicerar src med arg

GL_REPLACE kopierar

GL_DECAL kopierar med α -blend

GL_BLEND vald färg blendas in med
texturen som α -värde

GL_COMBINE texture combiner, mer
generell operation!

Specar en operation med två
operander och en operation.

Ingemar
Ragnemalm
ingis@isy.liu.se

Tillämpningar

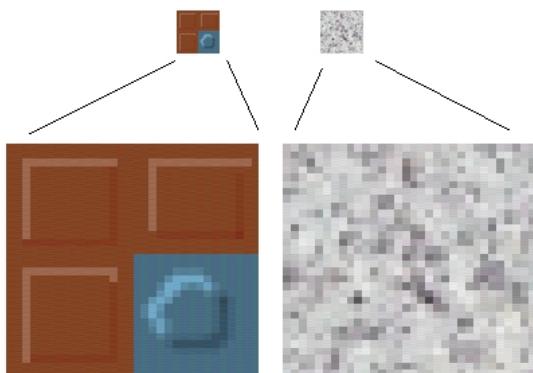
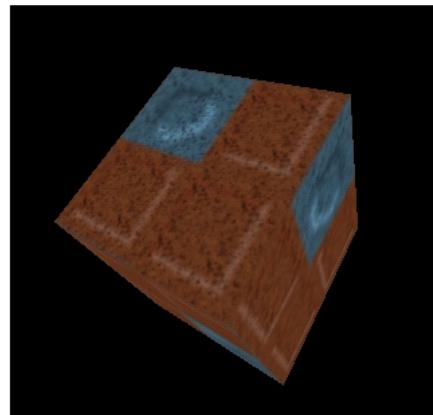
Multitexturering öppnar vägen till många
tillämpningar:

- Mjuka övergångar mellan terrängtexturer
- Färgskiftnings efter höjd (med 1D-textur)
- Detaljtexturer, lägg ihop en HF-textur med en LF-
textur, kan ge detaljkänsla med lite information
- Ljusmappning, med GL_MODULATE
- Ytor kan ha både materialtextur och andra effekter,
som ljus och reflektioner

Detaljtexturer

Kombinera högfrequent och lågfrequent liten textur till en stor!

Tillämpning av multitexturering.



Ingemar
Ragnemalm
ingis@isy.liu.se

Multitexturering

Multitexturering görs med olika koordinater per texturenhet:

```
void cube()
{
    glBegin(GL_QUADS);

    // I stället för glTexCoord2f(0,0);
    glMultiTexCoord2f(GL_TEXTURE0, 0, 0);
    glMultiTexCoord2f(GL_TEXTURE1, 0, 0);
    glVertex3f(-0.5, -0.5, -0.5);

    ...
    // I stället för glTexCoord2f(1,1);
    glMultiTexCoord2f(GL_TEXTURE1, detailLevel, detailLevel);
    glMultiTexCoord2f(GL_TEXTURE0, 1, 1);
    glVertex3f( 0.5, 0.5, -0.5);
}
```

Ingemar
Ragnemalm
ingis@isy.liu.se

Multitexturering

Inställningar per texturenhet:

```
// Texture 0, low frequency
    glEnable(GL_TEXTURE_2D);
    glBindTexture(GL_TEXTURE_2D, lfTex);
// Combiner - overwrite
    glTexEnvi(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_REPLACE);

// Texture 1, detail
    glEnable(GL_TEXTURE_2D);
    glBindTexture(GL_TEXTURE_2D, detailTex);
// Combiner - add detail (noise)
    glTexEnvi(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_ADD_SIGNED);
```

Ingemar
Ragnemalm
ingis@isy.liu.se

Multitexturering och shaders

Värdprogrammet måste speca texturer
och aktivera texturenheter,

Värden KAN speca texturkoordinater

Avancerad texturaritmetik görs bäst i
shadern

Ingemar
Ragnemalm
ingis@isy.liu.se

Rendera till textur

Man kan rendera scener som inte visas direkt, utan som läggs i en textur för att användas i senare renderingssteg.

Exempel på tillämpningar:

- Environment mapping på “riktigt”. Rendera omgivningen till textur.
- Filtreringar
- Simulera accumulation buffer
- Viktigt för skuggor

Ingemar
Ragnemalm
ingis@isy.liu.se

Rendera till textur

Kan göras på flera sätt:

glReadPixels + glTexImage

Dålig, går via CPU.

glCopyTexImage
glCopyTexSubImage

Kopiering inom VRAM

pBuffers (Pixel buffers)
Frame buffer objects (FBO) - preferred!

Skriv direkt till egen buffer.

Framebuffer objects, användning

Skapa referens, samma princip som för texturer:

```
glGenFramebuffersEXT(1, &fb);
```

Ange textur:

```
glFramebufferTexture2DEXT(GL_FRAMEBUFFER_EXT,  
GL_COLOR_ATTACHMENT0_EXT, GL_TEXTURE_2D, tex, 0);
```

Dito för renderbuffer:

```
glGenRenderbuffersEXT(1, &rb);  
glBindRenderbufferEXT(GL_RENDERBUFFER_EXT, rb);  
glRenderbufferStorageEXT(GL_RENDERBUFFER_EXT,  
GL_DEPTH_COMPONENT24, width, height);
```

Ingemar
Ragnemalm
ingis@isy.liu.se

Framebuffer objects, användning

Välj aktiv FBO:

```
glBindFramebufferEXT(GL_FRAMEBUFFER_EXT, fb)
```

Stäng av FBO, rendera till vanliga frame buffern:

```
glBindFramebufferEXT(GL_FRAMEBUFFER_EXT, 0)
```

Ingemar
Ragnemalm
ingis@isy.liu.se

Debugging av FBO

Under utveckling bör man fånga upp felmeddelanden från FBO. Det görs med glCheckFramebufferStatusEXT.

Typiskt problem: Om man inte har rätt parametrar till sin textur så kan den underkännaas som textur till en FBO! Destinationstexturen bör vara inställd på närmaste granne-interpolation.

Ingermar
Ragnemalm
ingis@isy.liu.se

Exempel med glCopyTexSubImage

```
void display()
{
    glBindTexture(GL_TEXTURE_2D, minitexid);
    glViewport(0, 0, width, height);

    // Draw what should go in the texture
    glClearColor(1,1,0.5,0);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    DrawTextureScene();
    glFlush();

    // Copy result to the texture
    glBindTexture(GL_TEXTURE_2D, tex);
    glCopyTexSubImage2D(GL_TEXTURE_2D, 0, 0, 0, 0, 0, width, height);

    glViewport(0, 0, lastw, lasth);

    // Render final image using the generated texture
    glClearColor(0.3, 0.3, 0.7,0);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    DrawMainScene();
    glutSwapBuffers();
}
```

Exempel med glCopyTexSubImage

```
void display()
{
    // render to the FBO
    glBindFramebufferEXT(GL_FRAMEBUFFER_EXT, fb);
    glBindTexture(GL_TEXTURE_2D, minitexid);

    glViewport(0, 0, width, height);

    // (draw something here, rendering to texture)
    glClearColor(1,1,0.5,0);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    DrawTextureScene();

    glViewport(0, 0, lastw, lasth);

    // render to the window, using the texture
    glBindFramebufferEXT(GL_FRAMEBUFFER_EXT, 0);
    glBindTexture(GL_TEXTURE_2D, tex);

    glClearColor(0,0,0,0);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    DrawMainScene();
    glutSwapBuffers();
}
```

Slutsatser

Rendering till textur är mycket viktigt - fler tillämpningar följer!

FrameBuffer Objects är bra - men undeskatta inte "gamla goda" glCopyTexSubImage! Den är snabb, bekväm och säker! FBO är ännu ung, icke standard före OpenGL 3.0.