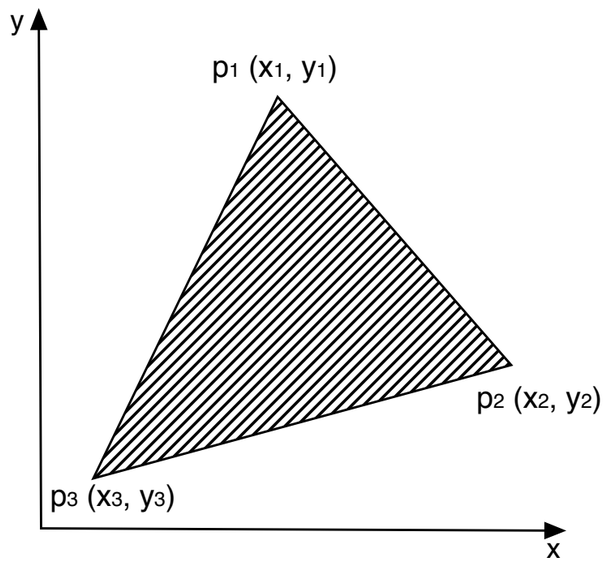
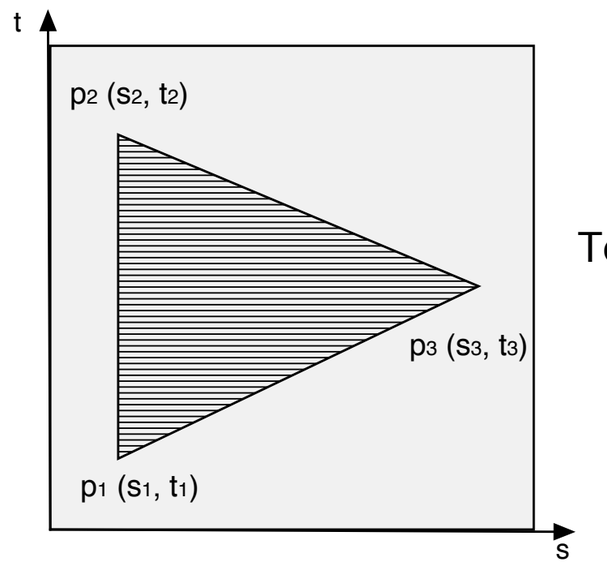
Texturmappning

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

Renderad polygon





Textur

- Mappningen djupberoende för rätt perspektiv
- Texturer läses in i VRAM, på GPU
- Flera texturer kan ritas på samma gång (multitexturering)
- Inte bara materialavbildningar, även ljus, reflektioner…

Vi kan väl redan:

Generera texturkoordinater:
• plan (linjär) mappning
• cylindrisk mappning
• sfärisk mappning

Texturinställningar: repeat/clamp

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

Mipmappning

Multitexturering, splatting

Texturer i 1, 2, 3 dimensioner

1D-texturer: rad av texlar

2D-texturer: som vanligt, yta, bild

3D-texturer: volymdata

Texturer i 3 dimensioner

3D-texturer:

- 3D-visualisering
- Volymetriska data (rök, vatten, eld...)

Multitexturering

Standardmetod, lätt att utföra i fragment shader.

- Tona mellan olika texturer (splatting)
- Använd vissa texturer för annan information (gloss mapping, bump mapping)
- Scrolling textures
- Detaljtexturer

Scrolling textures

Variant på multitexturering

2 eller flera texturer

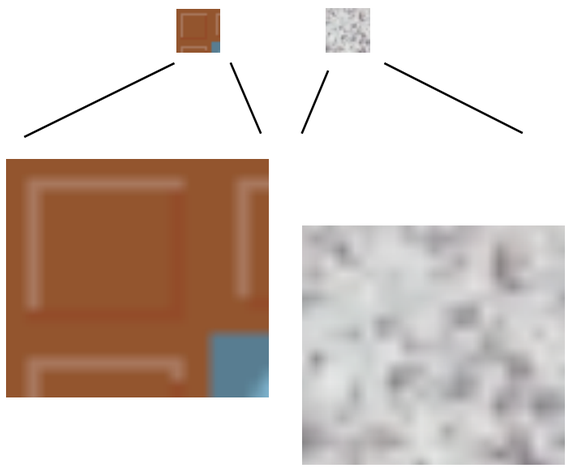
Vissa rör på sig, glider över ytan

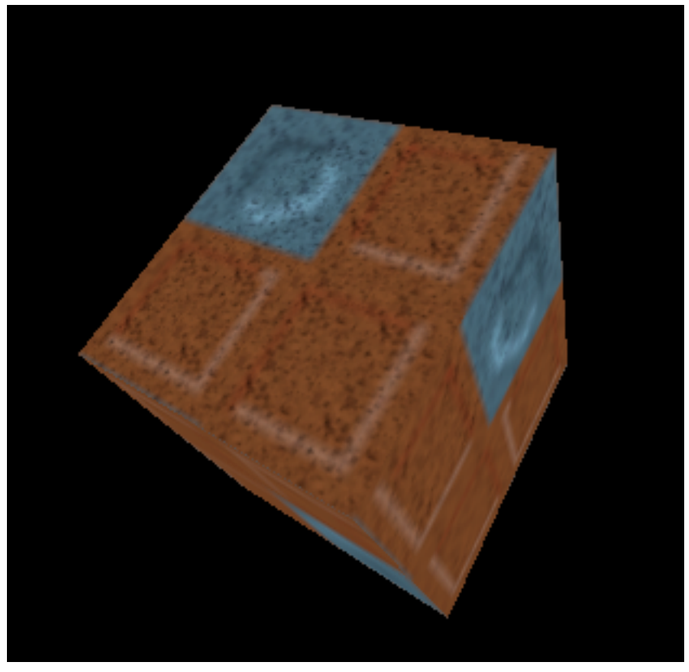
Texturer kan vara bilder eller offsets

Detaljtexturer

Kombinera högfrekvent och lågfrekvent liten textur till en stor!

Tillämpning av multitexturering.

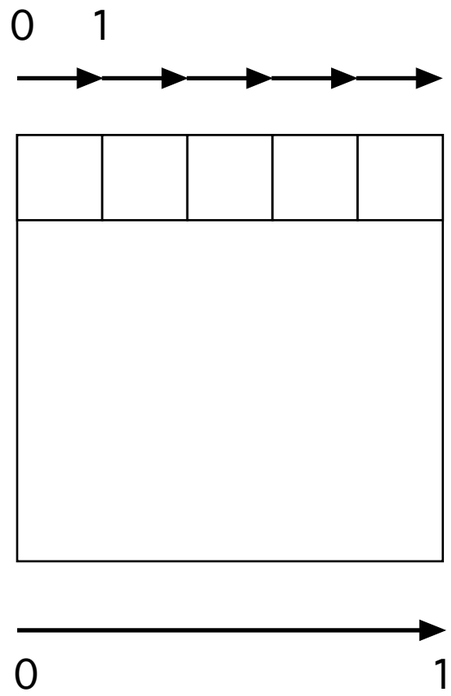




Koordinater

Multitexturering görs med olika koordinater per textur:

Lågfrekvent textur: 0 till 1
Högfrekvent textur: 0 till detailLevel
(med repeterande textur)

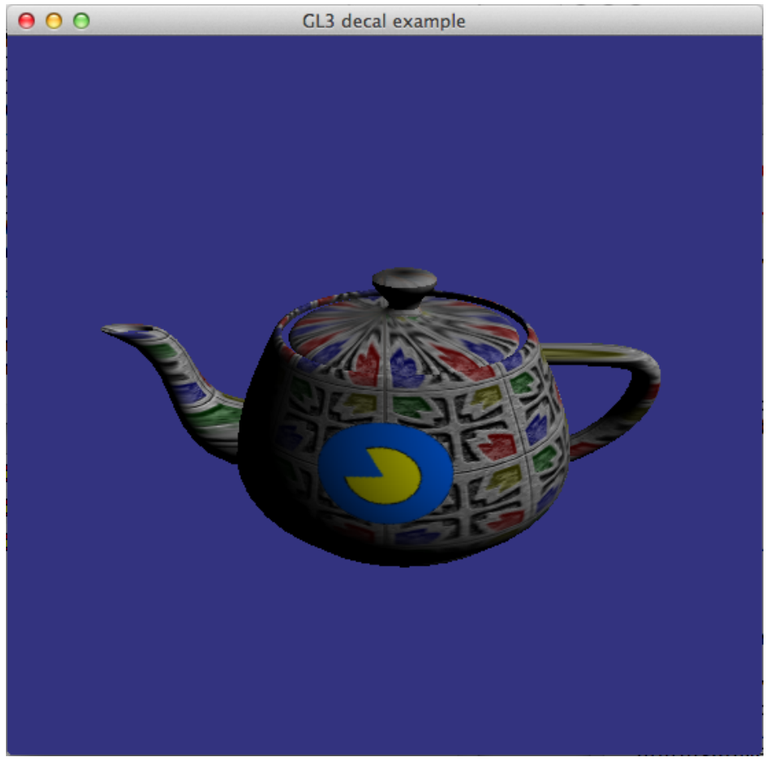


Texturkoordinater måste inte vara 1:1 med texturen!
Olika texturer kan mappas olika!

Exempel: Dekal

- Utah Teapot
- Sfärmappad yttextur
- Linjärmappad dekal med GL_CLAMP_TO_EDGE

Placera efter behov



Texturplacering

Normalt har objekt förgenererade texturkoordinater.

Man kan ganska lätt göra variationer som att skala eller translatera.

Men varför inte göra detta med matris?

Rotationsmatris för att snurra en textur!

Applicera matrisen på texturkoordinater!

Texturmatris

Enkelt exempel med texturmatris och multipasstexturering

Fast textur på hela objektet

En rörlig ”dekal”, transparent kant, GL_CLAMP för att bara ha ett varv av texturen.

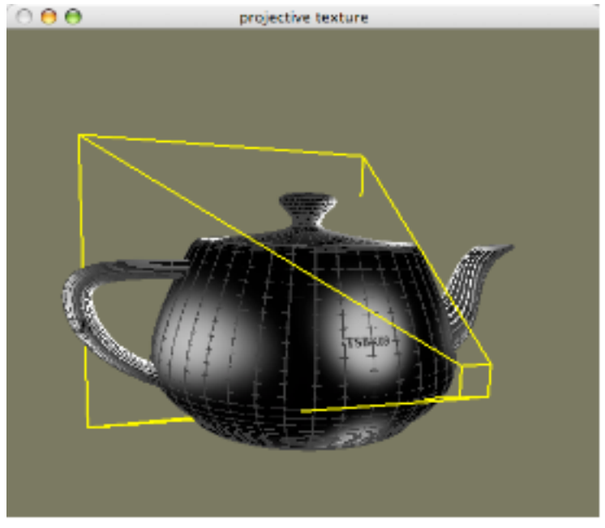
Dekalen kan flyttas och roteras med transformationer.

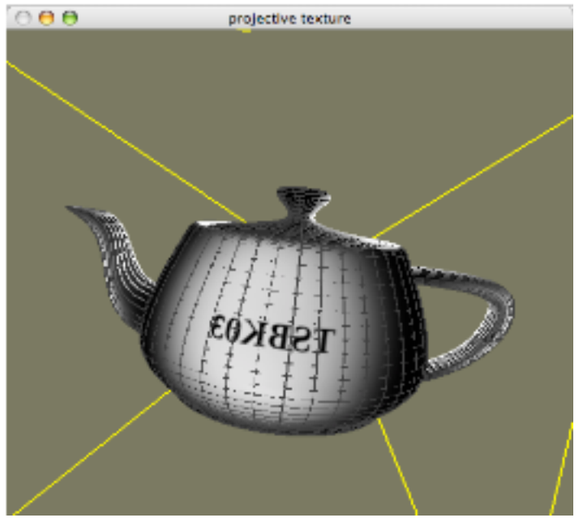
Projicerade texturer

Specialfall av texturplacering: Projektionsmatris!

Kan appliceras både i texturmatrisen eller i objektplanen.

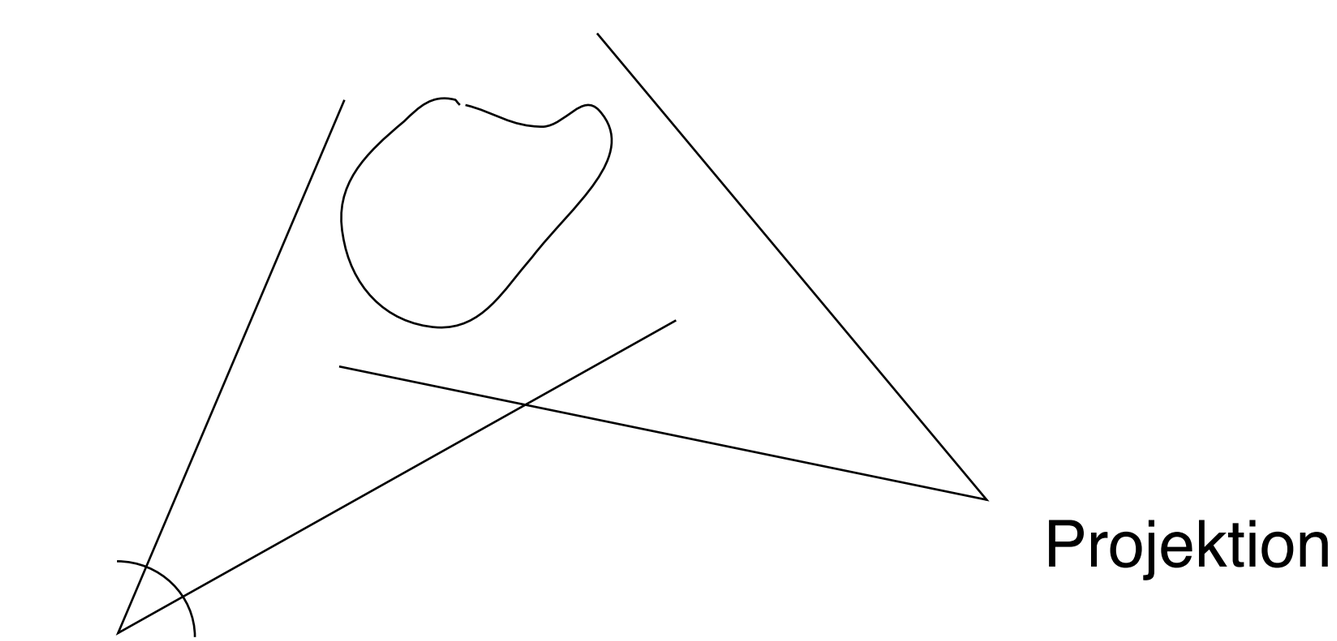
Viktiga för skuggenerering!





Projicerade texturer

En projektionsmatris skall in - men var?



Projicerade texturer

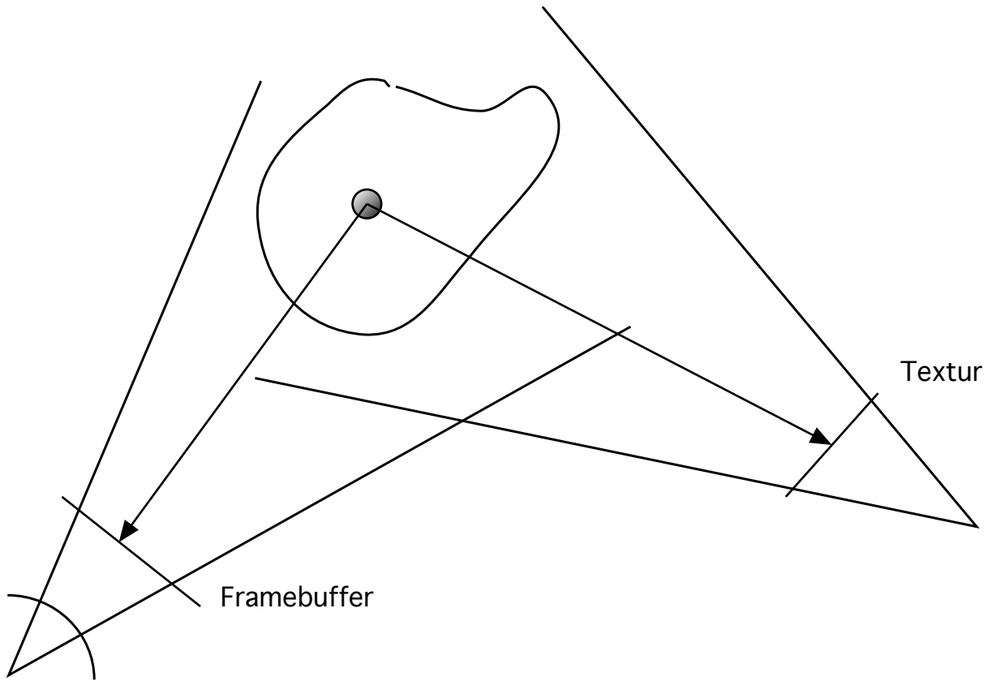
Princip:

Tag en punkt i rymden

Denna punkt skall vi projicera en textur på - dvs frågan är vilken texel den motsvarar!

"Kameran" (projektorn) är given

Transformera punkten som till en kamera, men kameran är nu en "projektor" som projicerar texturen!



Kamera

Projektion

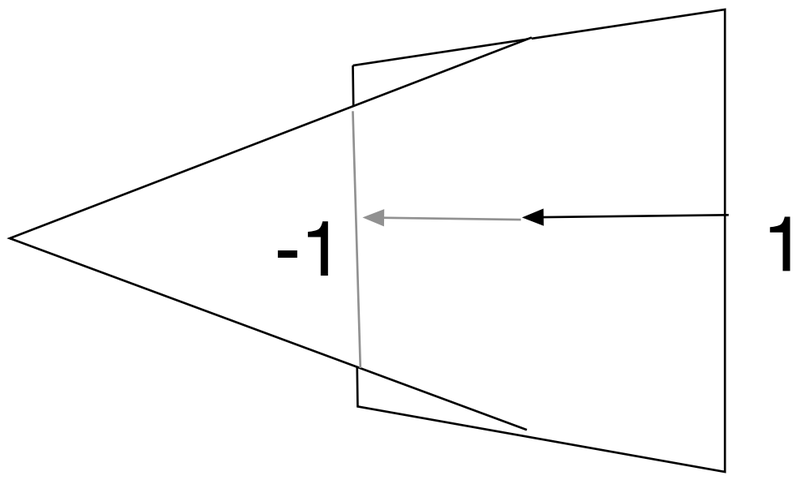
En punkt i rymden projiceras på båda planen.

1) Destination, kamera

2) Texel, "projektorn"

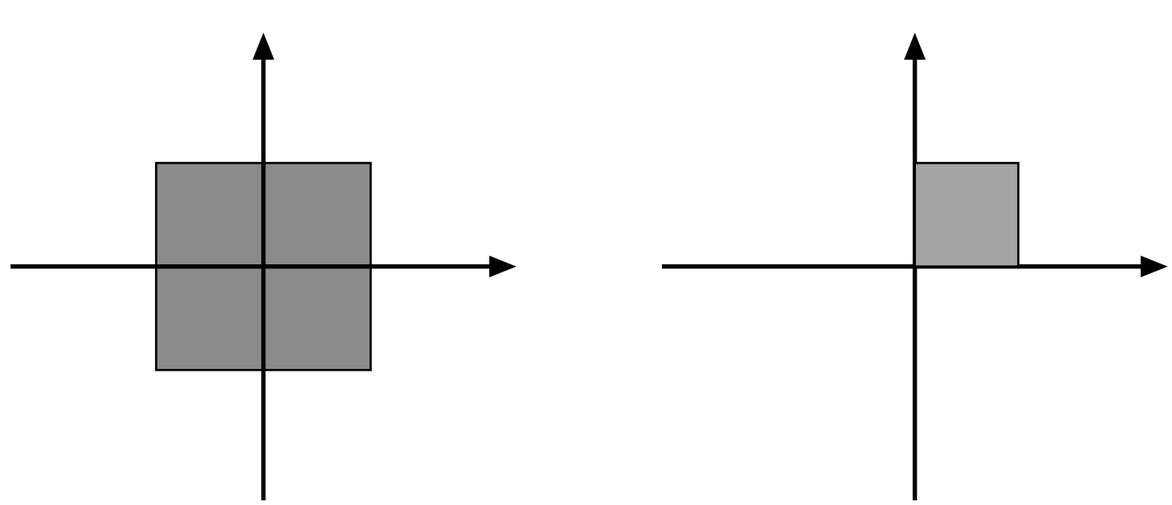
Texturkoordinaterna passar inte...

Vanlig projektionsmatris ger projektion till normerade skärmkoordinater. (-1 till 1)
Passar inte texturer! (0 till 1)



Lösning: Scale and Bias

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

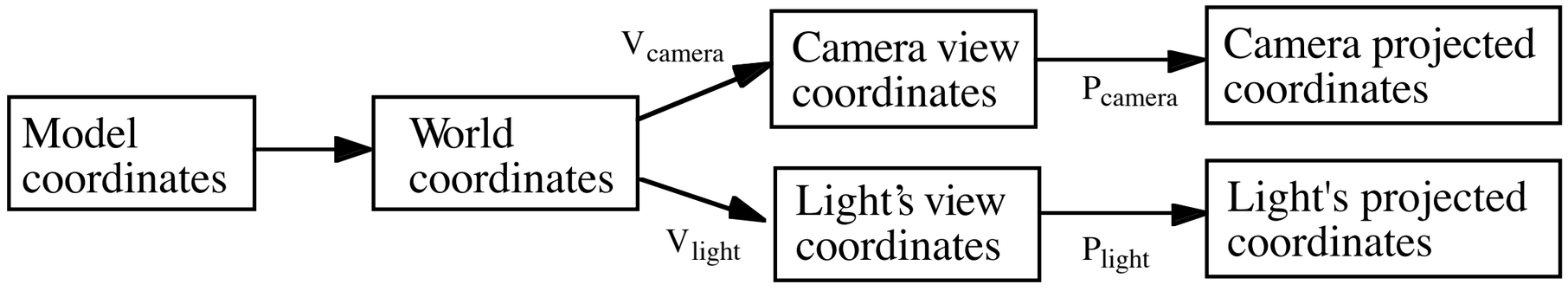


T(0.5, 0.5, 0.5);
S(0.5, 0.5, 0.5);

Projicerade texturer

Dubbla transformationskedjor, nu en till "projektorkoordinater".

Kan göras genom att backa från vykooordinater - opraktiskt.



Projicerade texturer från modellkoordinater

Bygg första steget likadant, både på modelview och texturmatrisen. Därefter unika transformationer för kameraplacering och projektion.

Projicerade texturer från modellkoordinater

Exempel

1) Sätt kameran till projektorn.
2) Multiplicera projektion*modelview, spara som texturmatrisen.
3) Sätt kameran till normal
4) Rita. Använd texturmatrisen för texturaccess.

1. Gå till projektorn

Vanlig ”look-at” men från annan punkt!

```
// Setup the modelview from the light source
modelViewMatrix = lookAtv(p_light, l_light, 0,1,0);
```

2. Skapa texturmatris

```
// Build the transformation sequence for the light source path,
// by copying from the ordinary camera matrices.
void setTextureMatrix(void)
{
   mat4 trans;
   mat4 scale;

   textureMatrix = IdentityMatrix();

// Scale and bias transform, moving from unit cube [-1,1] to [0,1]
   trans = T(0.5, 0.5, 0.0);
   scale = S(0.5, 0.5, 1.0);
   textureMatrix = trans * scale;
   textureMatrix = textureMatrix * projectionMatrix;
   textureMatrix = textureMatrix * modelViewMatrix;
}
```

3. Gå till kameran

Skapa den ”riktiga” model-to-view-matrisen!

```
// Setup the modelview from the light source
modelViewMatrix = lookAtv(p_camera, l_camera, 0,1,0,);
```

4. Alla model-to-world-transformationer läggs på BÅDA matriserna! Rita!

```
// One torus
  trans = T(0,4,-16);
  scale = S(2.0, 2.0, 2.0);
  trans = trans * scale;
  mv2 = modelViewMatrix * trans; // Apply on both
  tx2 = textureMatrix * trans; // Apply on both
  // Upload both!
  glUniformMatrix4fv(glGetUniformLocation(projTexShaderId,
"modelViewMatrix"), 1, GL_TRUE, mv2.m);
  glUniformMatrix4fv(glGetUniformLocation(projTexShaderId,
"textureMatrix"), 1, GL_TRUE, tx2.m);
  DrawModel(torusModel, ...);
```

Vertex shader:

```
out vec4 lightSourceCoord;

in vec3 in_Position;
uniform mat4 textureMatrix;
uniform mat4 modelViewMatrix;
uniform mat4 projectionMatrix;

void main()
{
  lightSourceCoord = textureMatrix * vec4(in_Position, 1.0); // Transform
vertex to light source coordinates
  gl_Position = projectionMatrix*modelViewMatrix * vec4(in_Position, 1.0);
}
```

Transformerar vertexkoordinater med texturmatrisen och skickar till en varying!

Fragment shader:

```
uniform sampler2D texture;
in vec4 lightSourceCoord;
out vec4 out_Color;
uniform float shade;

void main()
{
  vec4 lightSourceCoordinateWdivide = lightSourceCoord / lightSourceCoord.w;
  lightSourceCoordinateWdivide.t = 1.0 - lightSourceCoordinateWdivide.t;
  vec4 texel = texture(texture,lightSourceCoordinateWdivide.st);

  out_Color = texel * shade;
}
```

Utför homogena-koordinater-division. Applicerar textur som vanligt.

Sammanfattning projicerade texturer

Texturmappning genom projektionsmatris

Texturmatrisen transformerar texturkoordinater, modellerar en "projektor" för texturen

Glöm inte scale and bias!

Tillämpning: Skuggmappning! (Nästa föreläsning.)