Lite repetition

Vi kan väl...

Transformationer som 4x4-matriser

Rotation

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Translation

$$\begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Skalning

$$\begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Spegling

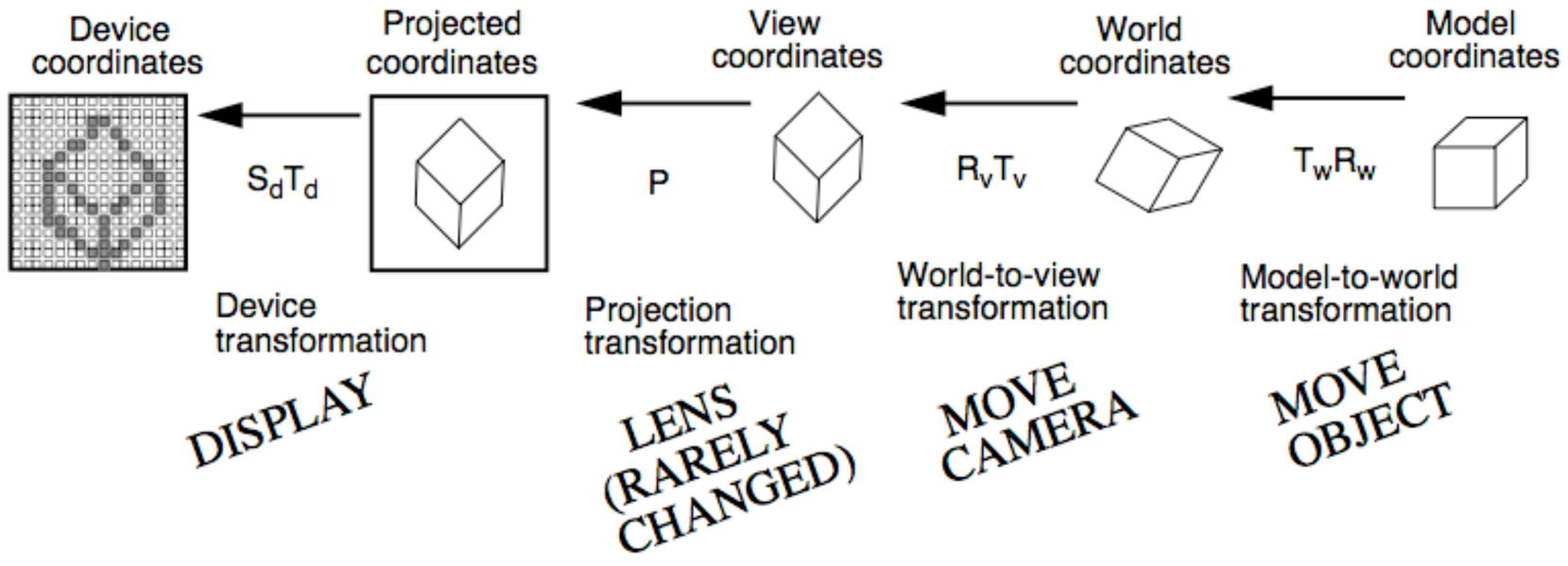
$$\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Skevning

$$\begin{bmatrix} 1 & 0 & a & 0 \\ 0 & 1 & b & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Transformation pipeline

Model coordinates
World coordinates
View coordinates
Projected coordinates
Device coordinates



Mer som ni kan som ett rinnande vatten

Ljussättning med
Phongs ljusmodell

Anti-aliasing

Phong shading

Brusgenerering

Texturering

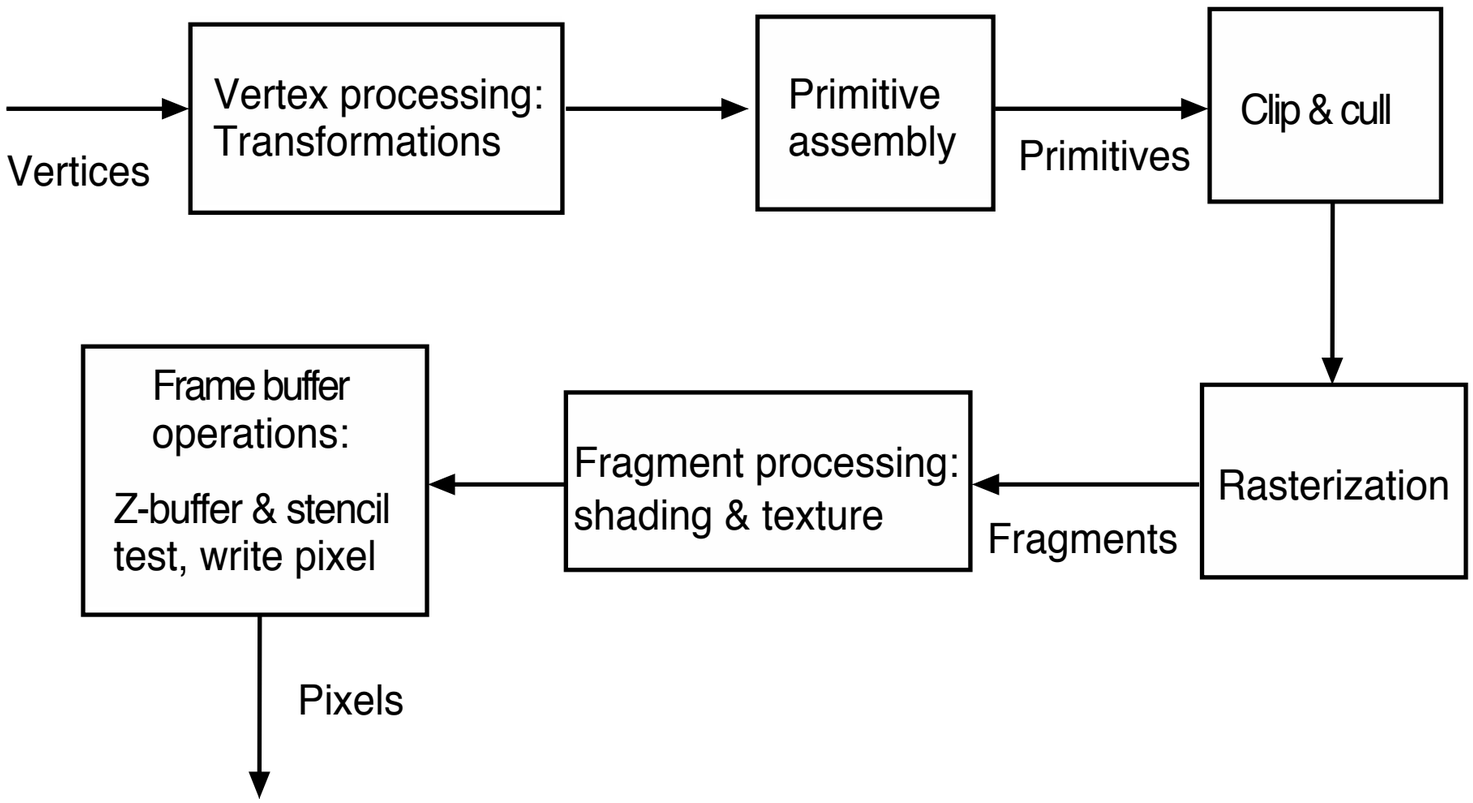
Skyboxes

Enkel kollisions-
detektering

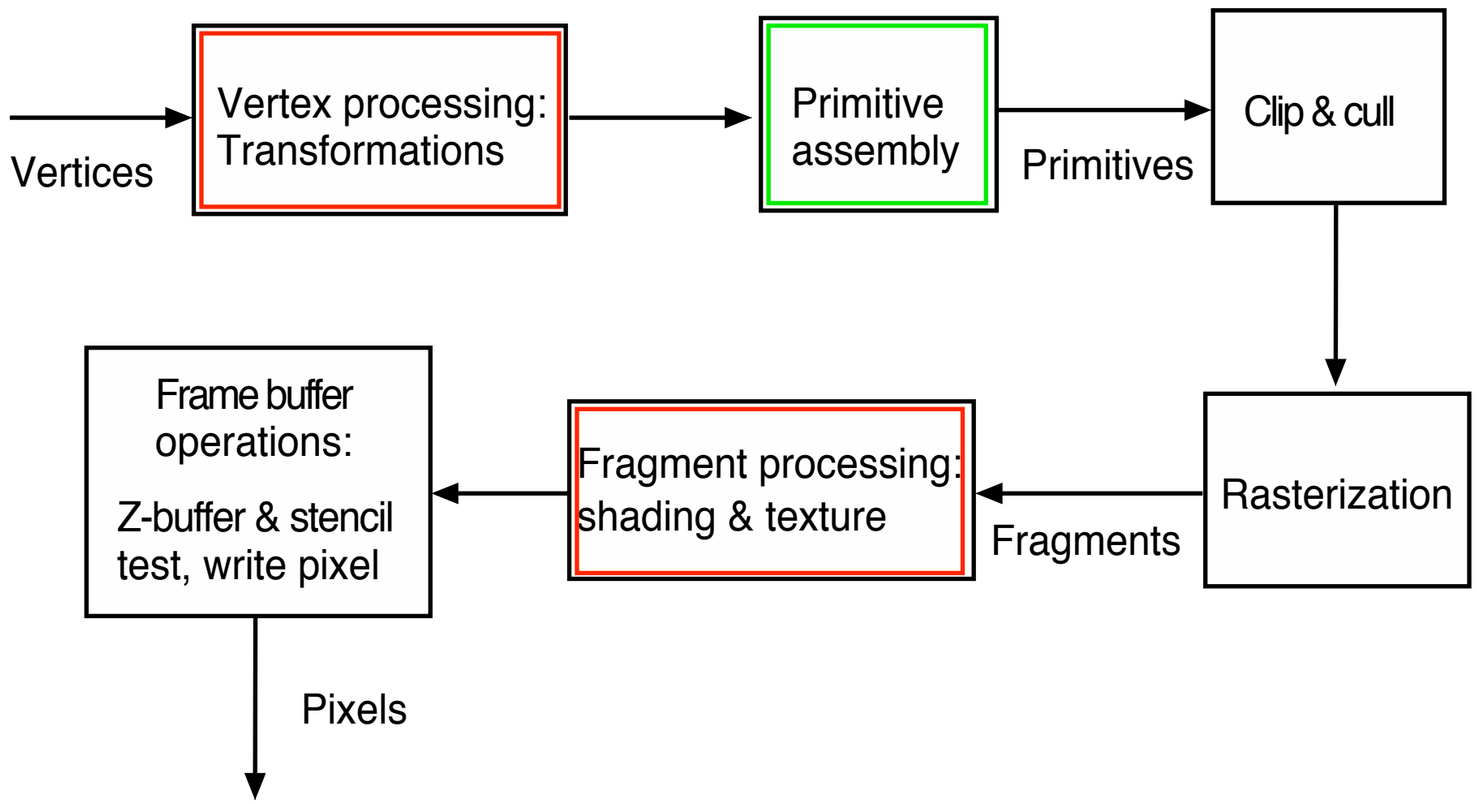
VSD

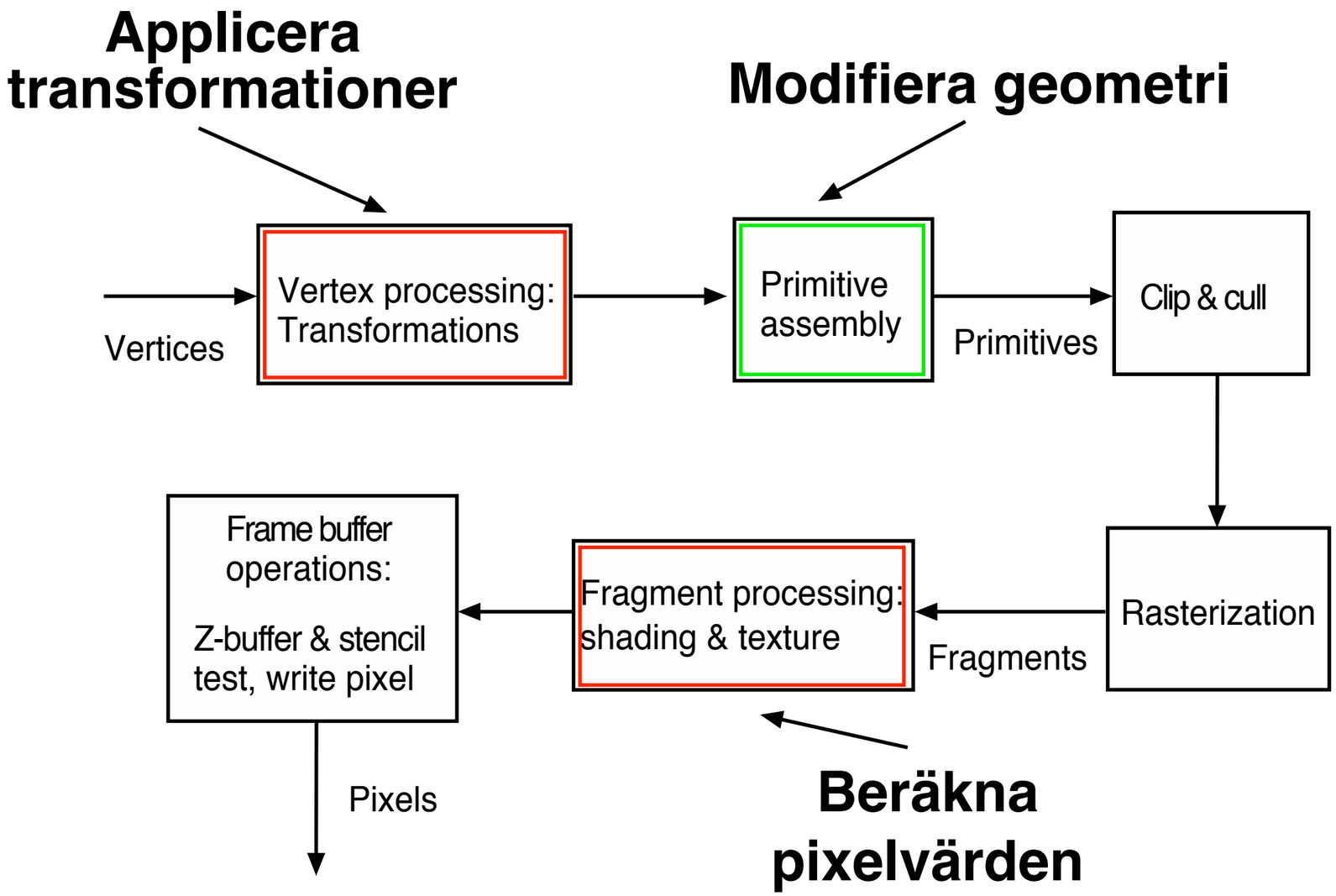
Transparens

OpenGL-pipelinen



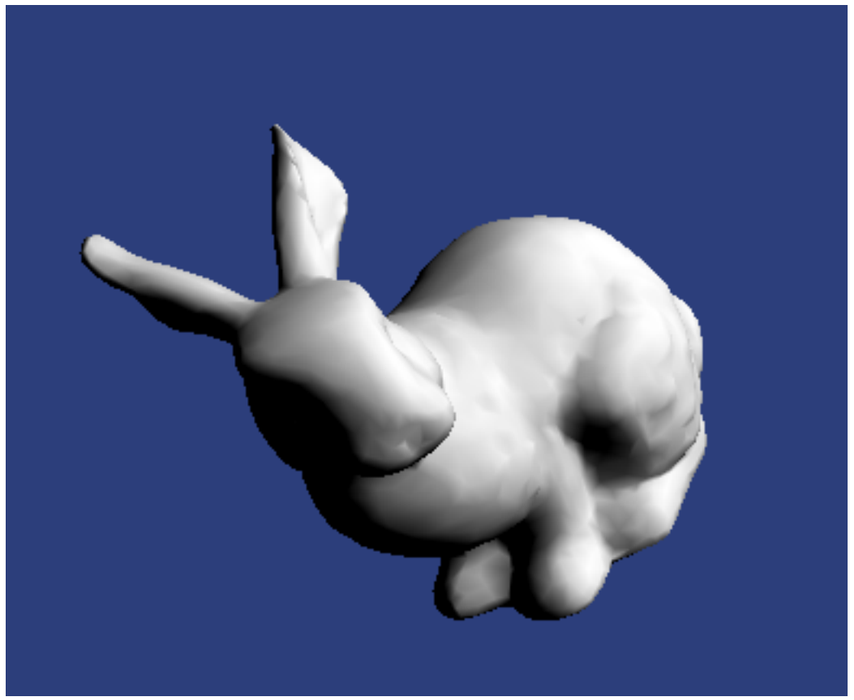
Steg som kan eller måste ha shaders




Applicera transformationer
Modifiera geometri
Vertex processing: Transformations
Vertices
Primitive assembly
Primitives
Clip & cull
Rasterization
Fragments
Fragment processing: shading & texture
Frame buffer operations:
Z-buffer & stencil test, write pixel
Pixels
Beräkna pixelvärden

Shader = beräkningskärna i OpenGL-pipelinen

- Körs när man ritar geometri
- Körs på GPUn
- Körs i massiv parallellprocessor
- Laddas, kompileras och aktiveras av huvudprogrammet (på CPU)
- Data skickas in från huvudprogrammet



Gouraud shader - vertex shader

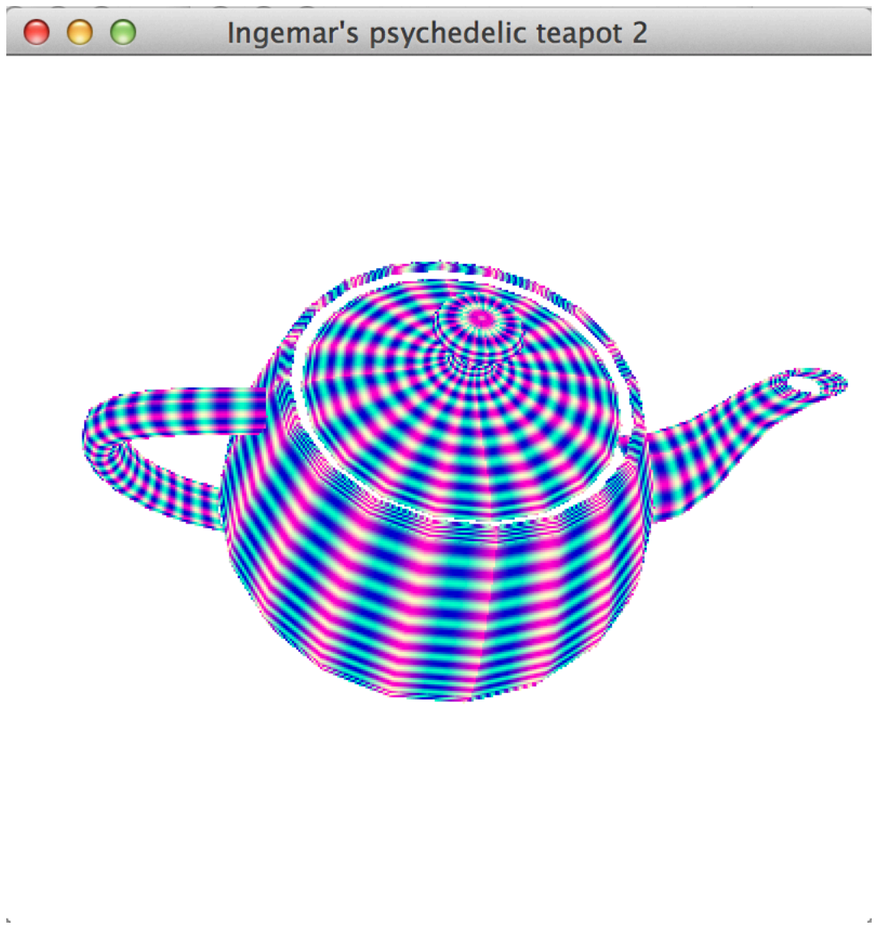
```
#version 150
in  vec3 inPosition;
in  vec3 inNormal;
out vec3 exColor;

void main(void)
{
  const vec3 light = vec3(0.58, 0.58, 0.58);
  float shade;
  shade = dot(normalize(inNormal), light);
  shade = clamp(shade, 0, 1);
  exColor = vec3(shade);
  gl_Position = vec4(inPosition, 1.0);
}
```

Gouraud shader - fragment shader

```
#version 150
in  vec3 exColor;
out vec4 outColor;
void main(void)
{
  outColor = vec4(exColor,1.0);
}
```

Procedurell textur




```
#version 150

in  vec3 inPosition;
in vec2 inTexCoord;
out vec2 texCoord;

uniform mat4 mdlMatrix;
uniform mat4 camMatrix;
uniform mat4 projMatrix;

void main(void)
{
    texCoord = inTexCoord;

    gl_Position = projMatrix * camMatrix * mdlMatrix * vec4(inPosition, 1.0);
}
```

```
#version 150

out vec4 outColor;
in vec2 texCoord;
uniform float t;

void main(void)
{
    float a = sin(texCoord.s * 30.0 + t)/2.0 + 0.5;
    float b = sin(texCoord.t * 30.0 * (1.0+sin(t/4.0)))/2.0 + 0.5;
    outColor = vec4(a, b, 0.8, 1.0); // inColor;
}
```

Vanliga shadereffekter

Multitexturering

Deformationer

Procedurella texturer

Ljussättning

Men vi kommer att gå vidare med

Bumpmappning

Projektiva texturer

Skinning

Rendering till textur,
multipassrendering

Skuggrendering

Har du några luckor?

Det är det labb 0 är till för att rätta till!

Helt frivillig labb för att repetera,
främst shaderprogrammering.

Frågor?