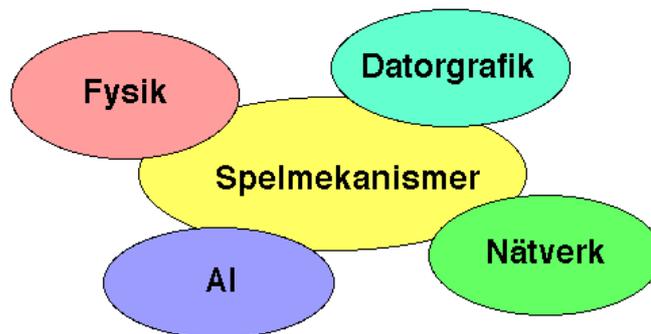


TSBK 10

Teknik för avancerade datorspel

Ingemar Ragnemalm, ISY



Ingemar
Ragnemalm
ingis@isy.liu.se

Föreläsning 5

- **GLSL och världen utanför**
 - Kommunikation med värdprogram
 - Kompilering och start
 - Design, debugging
 - Labbskalet
- **Lite mer GLSL**
 - Constructors
 - Swizzling
 - Inbyggda variabler
 - Inbyggda funktioner

Ingemar
Ragnemalm
ingis@isy.liu.se

Texturdata

**För att använda förgenererade
texturdata måste dessa
kommuniceras från OpenGL!**

**Detta görs med en “uniform”, en
variabel som inte kan ändras inom
ett primitiv.**

**Fördeklarerade typer för
texturreferens, “samplers”**

Ingemar
Ragnemalm
ingis@isy.liu.se

Texturdata

Exempel:

```
uniform sampler2D texture;
```

```
void main()  
{  
    gl_FragColor = texture2D(texture,  
                             gl_TexCoord[0].st);  
}
```

texture2D() gör texturuppslagning i texture.

Ingemar
Ragnemalm
ingis@isy.liu.se

Kommunikation med värdprogram

Viktigt! Vårdprogrammet måste kunna sätta uniform- och attribute-variabler som GLSL kan läsa.

GLSL kan inte skicka information till värdprogram annat än genom fragment.

OpenGL skickar address och namn till GLSL med speciella anrop.

Ingemar
Ragnemalm
ingis@isy.liu.se

Exempel: En uniform float:

```
float myFloat;  
GLint loc;
```

```
loc = glGetUniformLocationARB(p, "myFloat");  
glUniform1fARB(loc, myFloat);
```

p: Ref till shaderprogrammet, som installerats tidigare,

loc: adress till variabeln

Nu kan variabeln användas i GLSL:

```
uniform float myFloat;
```

Observera att det är strängen till glGetUniformLocation som avgör namnet i GLSL!

Ingemar
Ragnemalm
ingis@isy.liu.se

Exempel: En textur, uniform sampler:

```
GLuint tex;

glActiveTextureARB(GL_TEXTURE0_ARB);
glBindTexture(GL_TEXTURE_2D, tex);
loc = glGetUniformLocationARB(PROG, "tex");
glUniform1iARB(loc, 0);
```

Observera nollan till glUniform1iARB! Det är numret på texturenheten!

Används i shader:

```
uniform sampler2D tex;

vec3 texval = vec3(texture2D(texture,
    gl_TexCoord[0].st));
```

Ingemar
Ragnemalm
ingis@isy.liu.se

Exempel: Multitextur

**Bind en textur per texturenhet
Ge GLSL enhetsnummer och namn
Deklarera som samplers i GLSL**

Många möjligheter:

Väg samman texturdata med godtycklig funktion.

- **Gör en textur belysningsberoende och en annan inte.**
- **Använd en textur som t.ex. bump map**

Mitt exempel: Byt mellan två texturer beroende på belysningen

Ingemar
Ragnemalm
ingis@isy.liu.se

Exempel: Multitextur

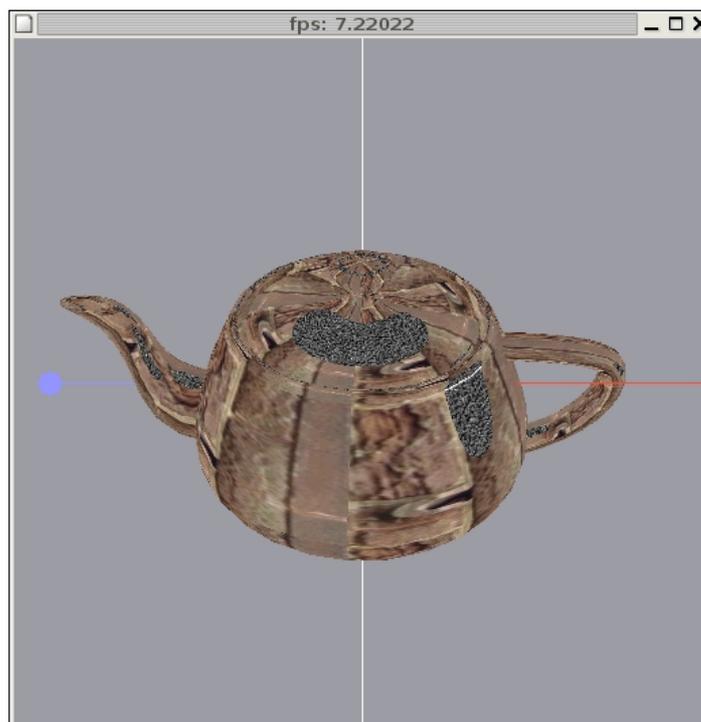
(Belysningskod överhoppad, beräknar ljusvärden för två ljuskällor, spec/spec2)

```
uniform sampler2D tex;  
uniform sampler2D bump;  
  
    vec3 texval = vec3(texture2D(tex, gl_TexCoord[0].st));  
if (spec+spec2 > kLimit)  
    texval = vec3(texture2D(bump, gl_TexCoord[0].st));
```

Ingemar
Ragnemalm
ingis@isy.liu.se

Exempel: Multitextur

Byter textur i “highlights”



Ingemar
Ragnemalm
ingis@isy.liu.se

Kompilering och exekvering

Detta görs i två steg:

1) Initiering och kompilering

- Skapa ett “programobjekt”
- Skapa “shaderobjekt” och knyt källkodsfiler till det
- Kompilera shaderprogrammen

2) Aktivering

- Aktivera programobjektet för rendering

Ingemar
Ragnemalm
ingis@isy.liu.se

Skapa ett “programobjekt”

`glCreateProgramObjectARB`

“Programobjektet” samlar all information om våra shaders. Skapa ett programobjekt per shaderpar i din tillämpning.

Ingemar
Ragnemalm
ingis@isy.liu.se

Skapa “shaderobjekt”

glCreateShaderObjectARB

Läs in källkoden och sänd till shaderobjektet:

glShaderSourceARB

Kompilera!

glCompileShaderARB

Ingemar
Ragnemalm
ingis@isy.liu.se

Koppla ihop och länka

För både vertex och fragment shader:

glAttachObjectARB

Länka:

glLinkProgramARB

Ingemar
Ragnemalm
ingis@isy.liu.se

Hela initieringen i kod

```
PROG = glCreateProgramObjectARB();
```

```
VERT = glCreateShaderObjectARB(GL_VERTEX_SHADER_ARB);  
text = readTextFile("shader.vert");  
glShaderSourceARB(VERT, 1, text, NULL);  
glCompileShaderARB(VERT);
```

Dito för fragment shader

```
glAttachObjectARB(PROG, VERT);  
glAttachObjectARB(PROG, FRAG);
```

```
glLinkProgramARB(PROG);
```

Ingemar
Ragnemalm
ingis@isy.liu.se

Aktivera programobjektet för rendering

Givet ett installerat och kompilerat programobjekt:

```
extern GLhandleARB PROG;
```

aktivera:

```
glUseProgramObjectARB(PROG);
```

avaktivera:

```
glUseProgramObjectARB(0);
```

Ingemar
Ragnemalm
ingis@isy.liu.se

Debugging

Det finns ingen debugger! Man måste ta till knep.

- Kompilatorns felmeddelanden
- Signalera med vertex shader
- Signalera med fragment shader
 - Använd enkel geometri

InfoLog

`glGetInfoLogARB`

Hämtar information om kompileringsresultatet och länkningsresultat.

Kan innehålla felmeddelanden, varningar. Exakta innehållet beror på tillverkaren.

Utvecklingsverktyg

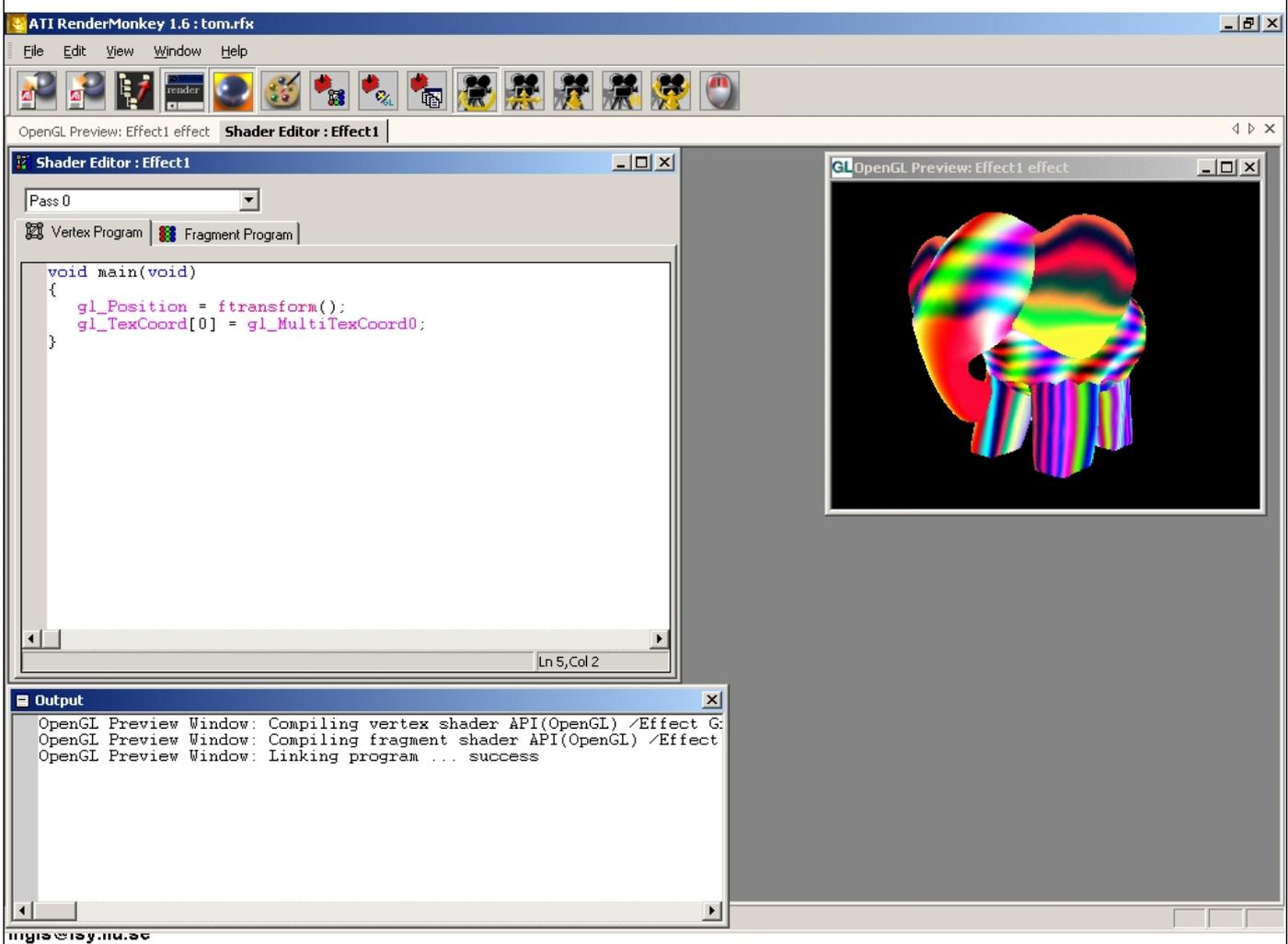
Shaderutveckling direkt i stor tillämpning är opraktiskt!

Enkla specialprogram används för att:

- Redigera källkod för vertex och fragment shader
 - Kompilera om på kommando
 - Köra shadern på en modell
 - Visa kompileringsresultat

T.ex. Rendermonkey, OpenGL Shader Builder, samt vårt labbskal för labb 1.

Ingemar
Ragnemalm
ingis@isy.liu.se



Labbskalet

Heter i skrivande stund bara “lab”. (Bättre namn önskas.) Utvecklat av Lars Abrahamsson.

- Källkod redigeras i separat editor, som två filer på förbestämd plats.
- Kompileringsresultat i shellfönstret.
- Två texturer förinstallerade.
- Kan växla mellan shaders och fasta pipelinen

Ingemar
Ragnemalm
ingis@isy.liu.se

The screenshot shows a Linux desktop environment. The top panel includes the menu bar (Applications, Actions) and the system tray (Sun 30 Oct, 11:31 AM). The main window is a text editor titled "/home/ingis/valla-online/lasse/lab/shader" with a toolbar and a file list containing "shader.frag", "shader.vert", and "lab_guidelines.txt". The editor displays GLSL code for a teapot shader, including comments and function calls like `normalize`, `reflect`, and `vec3`. The right side of the window shows a 3D rendering of a teapot with a yellow light source and a purple reflection. The bottom panel shows a taskbar with "shader" and "lab" windows, a system tray with "41 items, Free space: 65.5 GB", and a "wastebasket" icon.

```
Name: /home/ingis/valla-online/lasse/lab/shader/shader.vert
MIME Type: plain text document (text/plain)
Encoding: Unicode (UTF-8)
// mynorm = gl_Normal; // Är normalerna noll?
// mynorm = normalize(gl_Vertex); // Är normale

vec3 ecPosition = vec3(gl_ModelViewMatrix *
lightVec = normalize(vec3(gl_LightSource[1].
lightVec2 = normalize(vec3(gl_LightSource[3]
// 1+3 är fin, matchar synliga reflektioner

vec4 v = gl_Vertex;

gl_TexCoord[0] = gl_MultiTexCoord0; // ??

// mynorm måste vridas!
mynorm = normalize(gl_NormalMatrix * gl_Norm
reflecVec = reflect(-lightVec, mynorm);
reflecVec2 = reflect(-lightVec2, mynorm);

camVec = normalize(-ecPosition);

// Test på om normalerna är något alls. Det är de!
if (mynorm.x*mynorm.x + mynorm.y*mynorm.y + mynorm.z*mynorm.z == 0) v.y = 0;
```