Debugging

There is no debugger! You must use other tricks!

- Compiler error messages
- Signals using vertex shader
- Signals using fragment shader
- Use simple geometry - easy to understand

InfoLog

glGetProgramInfoLog/glGetShaderInfoLog (glGetInfoLogARB)

Retrieves information about compilation and linking results

May include error messages, warnings... The exact contents varies depending on GPU brand.

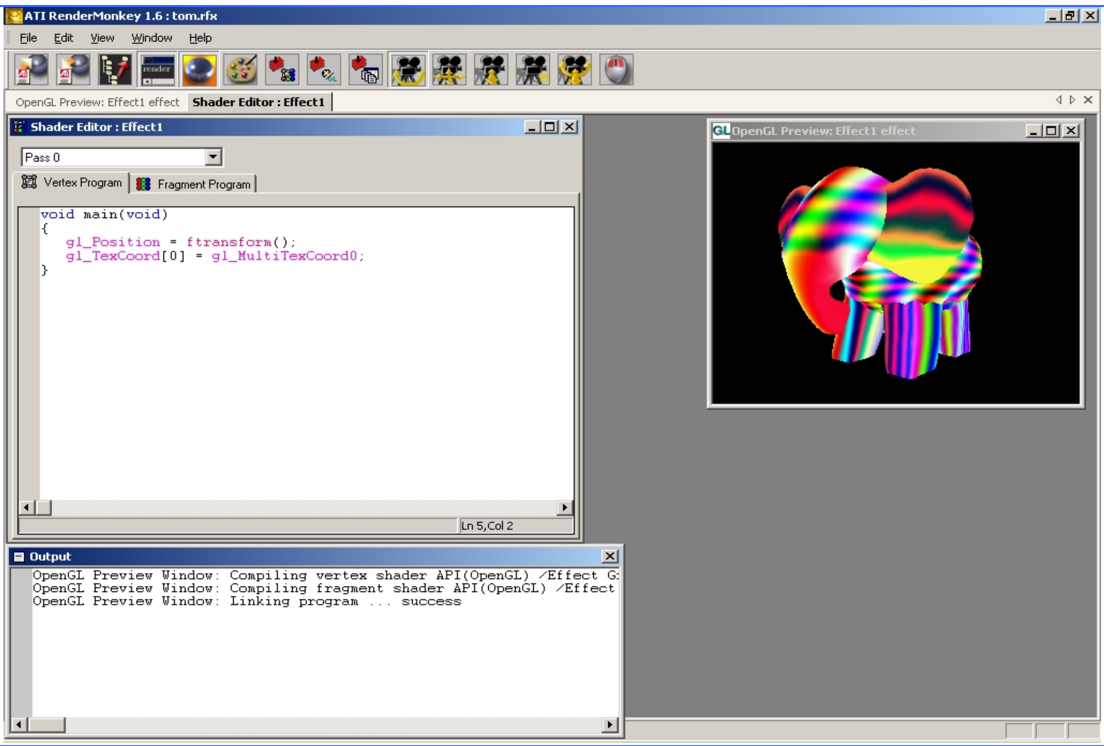
Development tools

Shader development directly in large application unreasonable

Simple development shells are used to:

- **Edit source code for vertex and fragment shader**
- **Recompile when desired**
- **Test the shader on a model**
- **Display compilation results**

I.e. Rendermonkey, OpenGL Shader Builder, and our lab shell for lab 3



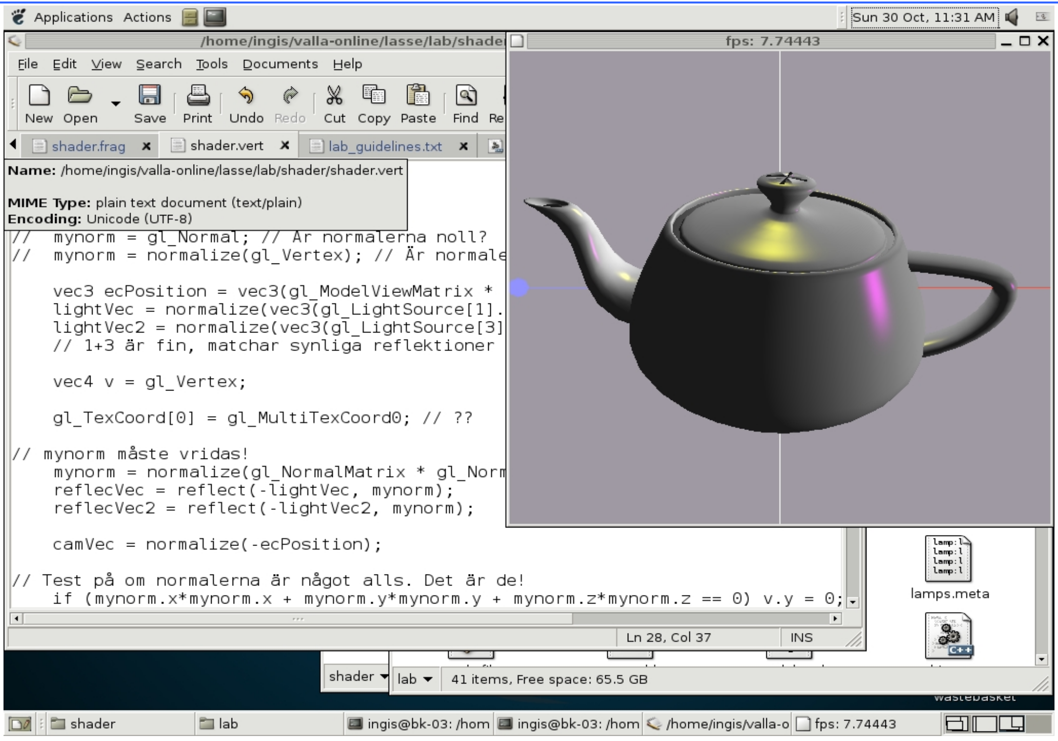
```
void main(void)
{
   gl_Position = ftransform();
   gl_TexCoord[0] = gl_MultiTexCoord0;
}
```

```
OpenGL Preview Window: Compiling vertex shader API(OpenGL) /Effect G:
OpenGL Preview Window: Compiling fragment shader API(OpenGL) /Effect
OpenGL Preview Window: Linking program ... success
```

The shader lab shell

Created by Lars Abrahamsson, more or less rewrittenthis year by Mikael Kalms.

- **Source code edited in separate editor, as two files put in a certain place.**
- **Compilation results in the shell window**
- **Two pre-installed textures**
- **Can switch between shader and fixed pipeline**



Name: /home/ingis/valla-online/lasse/lab/shader/shader.vert

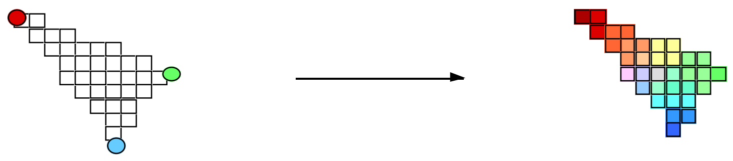
MIME Type: plain text document (text/plain)
Encoding: Unicode (UTF-8)

```
//   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;
```

The fragment processor

From pixel coordinates and interpolated data for color, texture etc, calculate a color for the fragment.

- Shading
- Texturing
- Fog
- Color calculations

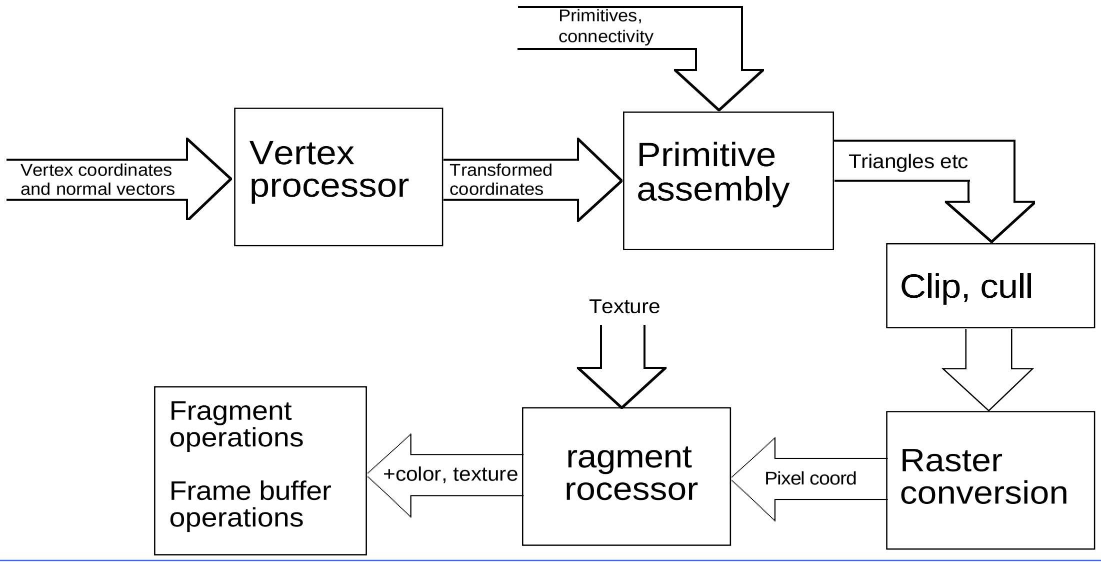


Fragmentoperationer

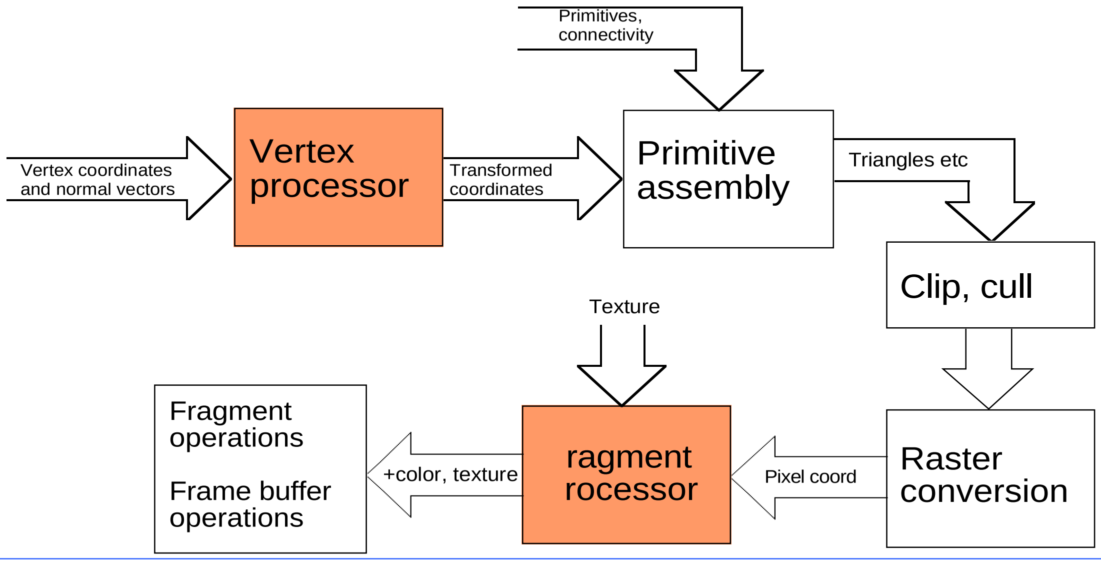
Final operations before the fragment is written to a frame buffer pixel

- Stencil test
- Z-buffer test
- The blend function (glBlendFunc mm)
- The alpha function (glAlphaFunc)

Out of these, two are programmable!



Out of these, two are programmable!



Shader programs

Program snippets that are executed per vertex or per fragment, on the GPU!

Two programs cooperate, one vertex program and one fragment program.

“Shader” implies that the goal is lighting, but that is only one of the goals!.

ertex transform
ertexcolor, vertex-level lighting

Can be done in a vertex shader

exturing
olor and light per pixel

Can be done in a fragment shader

Vertex shader

Replaces the fixed functionality of the vertex processor.

It can:
- **transform vertices, normals and texture coordinates**
- **generate texture coordinates**
- **calculate lighting per vertex**
- **set values for interpolation for use in a fragment shader**

It knows nothing about:
- **Perspective, viewport**
- **Frustum**
- **Primitives (!)**
- **Culling**

Fragment shader
(a.k.a pixel shader)

eplaces the fixed functionality of the fragment rocessor.

t can:
- set the fragment color
- get color values from textures
- calculate fog and other color calculations
- use any kind of interpolated data from the vertices

t can not
- change the fragment coordinates
- write into textures
- affect stencil, scissor, alpha, depth...

Shader languages

Four different:

Assembly language: Old solution, being phased out, no longer updated.

Cg: “C for graphics”, NVidia
HLSL: “High-level shading language”, Microsoft
GLSL: “OpenGL shading language”

Choce depends on flatform and needs (and aste).