An Introduction to the OpenGL Shading Language

Keith O'Conor

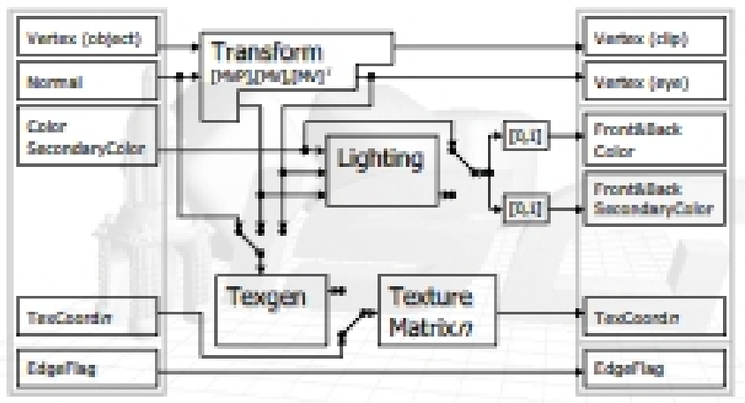
Outline

- How the fixed function pipeline works
- How it's replaced by GLSL
- Structure & syntax nitty-gritty
- How to integrate GLSL into OpenGL apps
- Some simple examples
- Resources

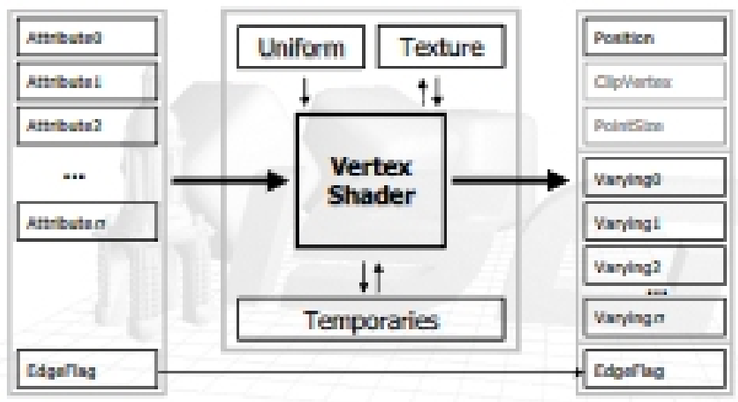
Who? When? Why?!

- ARB GL2 workgroup
  - Included in OpenGL 2.0
- February 27, 2003
  - OpenGL Shading Language draft released
- Advances in hardware
  - Just not feasible before now
    - Specific operations in specific order = fast hardware

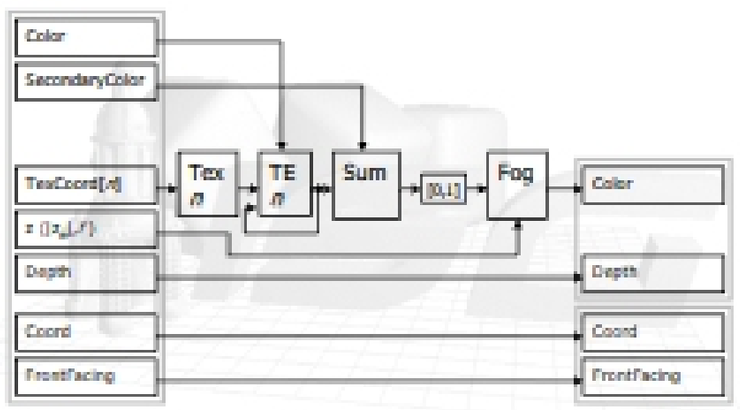
Fixed Function Vertex Processor



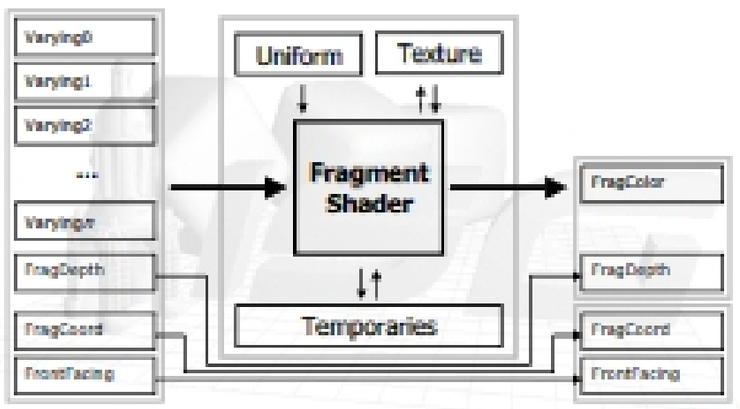
GL2 Vertex Processor



Fixed Function Fragment Processor



GL2 Fragment Processor



In General...

- Vertex processes programmed
  - Vertex Transformation
  - Normal Transformation, Normalization
  - Lighting
  - Texture Coordinate Generation and Transformation
- Fragment processes programmed
  - Texture accesses & application
  - Fog

Previous programmability

- Texture Shaders
- Register Combiners
- Assembly programs
  - ARB_vertex_program
  - ARB_fragment_program
  - Messy!
- Needed general, readable & maintainable language

Types

```
void
float  vec2  vec3  vec4
mat2  mat3  mat4
int  ivec2  ivec3  ivec4
bool  bvec2  bvec3  bvec4
samplernD, samplerCube,
  samplerShadownD
```

Types

- Structs
- Arrays
  - One dimensional
  - Constant size (ie `float array[4];`)
- Reserved types
  - `half  hvec2  hvec3  hvec4`
  - `fixed  fvec2  fvec3  fvec4`
  - `double  dvec2  dvec3  dvec4`

Type qualifiers

- attribute
  - Changes per-vertex
    - eg. position, normal etc.
- uniform
  - Does not change between vertices of a batch
    - eg light position, texture unit, other constants
- varying
  - Passed from VS to FS, interpolated
    - eg texture coordinates, vertex color

Operators

- grouping: ()
- array subscript: []
- function call and constructor: ()
- field selector and swizzle: .
- postfix: ++ --
- prefix: ++ -- + - !

Operators

- binary: * / + -
- relational: < <= > >=
- equality: == !=
- logical: && ^^ ||
- selection: ?:
- assignment: = *= /= += -=

Reserved Operators

- prefix: ~
- binary: %
- bitwise: << >> & ^ |
- assignment: %= <<= >>= &= ^= |=

Scalar/Vector Constructors

- **No casting**

```
float f; int i; bool b;
vec2 v2; vec3 v3; vec4 v4;

vec2(1.0 ,2.0)
vec3(0.0 ,0.0 ,1.0)
vec4(1.0 ,0.5 ,0.0 ,1.0)
vec4(1.0)                       // all 1.0
vec4(v2 ,v2)
vec4(v3 ,1.0)

float(i)
int(b)
```

Matrix Constructors

```
vec4 v4; mat4 m4;

mat4( 1.0, 2.0, 3.0, 4.0,
      5.0, 6.0, 7.0, 8.0,
      9.0, 10., 11., 12.,
      13., 14., 15., 16.)    // row major

mat4( v4, v4, v4, v4)
mat4( 1.0)                   // identity matrix
mat3( m4)                    // upper 3x3
vec4( m4)                    // 1st column
float( m4)                   // upper 1x1
```

Accessing components

- component accessor for vectors
  - xyzw rgba stpq [i]
- component accessor for matrices
  - [i] [i][j]