Introduction to the OpenGL Shading Language (GLSL)

Mike Bailey
CS 519
Oregon State University

May 31, 2006

Cryptic Cryptic
Compiler messages
Driver does it for you
Must do yourself
Compile Shaders
Eventually built-in, Texture for now
Built-in Noise
Model (=OC), Eye (≈WC)
Shader, World, Object
Coordinate Systems

Fundamental Differences Between RenderMan Shaders and OpenGL Shaders

<table>
<thead>
<tr>
<th>Topic</th>
<th>RenderMan</th>
<th>GLSL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goals</td>
<td>1. Image quality, 2. Speed</td>
<td>1. Speed, 2. Image quality</td>
</tr>
<tr>
<td>Shader Types</td>
<td>Surface, Displacement (+3 others)</td>
<td>Vertex, Fragment</td>
</tr>
<tr>
<td>Surface Preprocessing</td>
<td>Microfacets</td>
<td>None</td>
</tr>
<tr>
<td>Recompute Normals</td>
<td>CalculateNormal</td>
<td>None</td>
</tr>
<tr>
<td>Get Rid of Pixels</td>
<td>Oi = 0.;</td>
<td>discard;</td>
</tr>
<tr>
<td>Surface/Fragment shader sets</td>
<td>R, G, B, or, og, ob</td>
<td>R, G, B, A, Z</td>
</tr>
<tr>
<td>Shader Variables</td>
<td>Uniform, Varying</td>
<td>Attribute, Uniform, Varying</td>
</tr>
<tr>
<td>Coordinate Systems</td>
<td>Shader, World, Object</td>
<td>Model (=OC), Eye (=WC)</td>
</tr>
<tr>
<td>Noise</td>
<td>Built-in</td>
<td>Eventually built-in, Texture for now</td>
</tr>
<tr>
<td>Compile Shaders</td>
<td>Must do yourself</td>
<td>Driver does it for you</td>
</tr>
<tr>
<td>Compiler messages</td>
<td>Cryptic</td>
<td>Cryptic</td>
</tr>
</tbody>
</table>
OpenGL Graphics Pipeline

GLSL Vertex Shader Inputs and Outputs

Courtesy of Randi Rost
A GLSL Vertex Shader Replaces These Operations:

- Vertex transformations
- Normal transformations
- Normal normalization
- Handling of per-vertex lighting
- Handling of texture coordinates

A GLSL Vertex Shader Does Not Replace These Operations:

- Frustum clipping
- Homogeneous division
- Viewport mapping
- Backface culling
- Polygon mode
- Polygon offset
Built-in Vertex Shader Variables You Will Use a Lot:

- `vec4 gl_Vertex`
- `vec4 gl_Normal`
- `vec4 gl_Color`
- `vec4 gl_MultiTexCoordi (i=0, 1, 2, ...)`
- `mat4 gl_ModelViewMatrix`
- `mat4 gl_ProjectionMatrix`
- `mat4 gl_ModelViewProjectionMatrix`
- `mat4 gl_NormalMatrix`

GLSL Vertex Shader Internal Names

Application calls to set standard vertex attributes

Current attribute value

Built-in attribute variables

Courtesy of Randi Rost
GLSL Shaders Are Like C With Extensions for Graphics:

- Types include int, ivec2, ivec3, ivec4
- Types include float, vec2, vec3, vec4
- Types include mat2, mat3, mat4
- Types include bool, bvec2, bvec3, bvec4
- Types include sampler to access textures
- Vector components are accessed with [index], .rgba, .xyzw, and .stpq
- Vector components can be "swizzled" (c1.rgba = c2.abgr)
- `discard` operator used in frag shaders to discard fragments
- Type qualifiers: const, attribute, uniform, varying
- Procedure type qualifiers: in, out, inout

GLSL Shaders Are Missing Some C-isms:

- No type casts (use constructors instead)
- No automatic promotion
- No switch statement
- No pointers
- No strings
- No bitwise operators
- No enums
- Can only use 1-D arrays (no bounds checking)
- No `file-based` pre-processor directives
A GLSL Fragment Shader Replaces These Operations:

- Color computation
- Texturing
- Color arithmetic
- Lighting
- Fog
- Blending
- Discarding fragments
A GLSL Fragment Shader Does Not Replace These Operations:

- Stencil test
- Z-buffer test
- Stippling

Built-in Fragment Shader Variables You Will Use a Lot:

vec4 gl_FragColor
vec4 gl_FragDepth