Teaching Graphics with the OpenGL Shading Language

Jerry Talton & Darren Fitzpatrick
Teaching Graphics

- Realtime, interactive?
- Offline, rendering?
- High-level overview?
- Low-level details?
- New, relevant technology?
- Old, tried-and-true techniques?
Popular Approaches

- **Top-down**: students use an API to build complex graphics applications
  - Easy to become embroiled in technical idiosyncrasies

- **Bottom-up**: students develop toy implementations of the graphics pipeline
  - Students may never produce non-trivial applications
Our Approach

• Traditional, top-down survey structure firmly rooted in OpenGL

• Augment material via extensive integration of the **programmable pipeline** via the OpenGL Shading Language (GLSL)

• Give students the tools to create complex, high-level software

• Expose relevant low-level technical details of the graphics pipeline
OpenGL Shading Language

- Part of the OpenGL 2.0 standard
- High level, procedural language
- Used to process vertices and fragments
- Programs are called “shaders”
- Based on C/C++ syntax
- Very similar to Cg, HLSL, RSL, etc.
- Ubiquitous in research and industry
Programmable Pipeline
GLSL Example

```glsl
varying vec3 N;

void main()
{
    gl_Position = gl_ProjectionMatrix * 
        gl_ModelViewMatrix * gl_Vertex;
    N = gl_ModelViewMatrixInverseTranspose * 
        gl_Normal;
    N = normalize(N);
}
```
Educational Benefits

Shaders expose functionality the traditional API obscures

- Students have a direct connection with the mathematical equations governing their applications
- Students can implement a wide variety of lighting and shading models
- Students have state-of-the-art tools to produce course projects
gl_Enable(GL_TEXTURE_CUBE_MAP);

varying vec3 N;
varying vec3 E;

uniform samplerCube envMap;

void main()
{
  vec3 R = 2 * dot(N, E) * N - E;
  vec3 color = vec3(textureCube(envMap, R));
  gl_FragColor = vec4(color, 1.0);
}
Integrating GLSL into an existing curriculum is easy

- One early lecture covering GLSL syntax and setup
- One late lecture covering advanced rendering techniques
- Other lecture topics supplemented with shader-level implementation details
Recommended Texts

- *OpenGL® Shading Language*
  - Second Edition
  - Randi J. Rost

- *Interactive Computer Graphics*
  - Fourth Edition
  - Edward Angel

- *OpenGL® Programming Guide*
  - Fifth Edition
  - OpenGL Architecture Review Board
Programming Assignments

• First assignment on geometry processing, getting something on the screen
• Second assignment on lighting and shading algorithms
• Final project on student-selected topics, mostly focused on animation
Student Projects
"The quality of the course assignments was:"

<table>
<thead>
<tr>
<th></th>
<th>Poor</th>
<th>Fair</th>
<th>Satisfactory</th>
<th>Very Good</th>
<th>Excellent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Percent</td>
<td>0%</td>
<td>0%</td>
<td>16%</td>
<td>26%</td>
<td>58%</td>
</tr>
</tbody>
</table>

"I put considerable effort into the course:"

<table>
<thead>
<tr>
<th></th>
<th>Strongly Disagree</th>
<th>Disagree</th>
<th>Neutral</th>
<th>Agree</th>
<th>Strongly Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Percent</td>
<td>0%</td>
<td>0%</td>
<td>16%</td>
<td>26%</td>
<td>58%</td>
</tr>
</tbody>
</table>
Technological Considerations

- Windows only supports OpenGL 1.4
- Linux support highly vendor dependent
- Can use Macintosh OS X or the OpenGL Extension Wrangler Library (GLEW)
- Need NVIDIA FX or ATI 9700 series hardware (or later)
Future Work

- Course will be offered at Stanford this summer and further refined
- General purpose computing on graphics hardware (GPGPU) component will be added to curriculum
- Textbook?
Thanks!

Keenan Crane
Jesse Hall
Michael Garland
Suresh Lodha
Krystle de Mesa
Lynn Maxwell
Ron Fedkiw
James Davis
Tim Babb
Cynthia Lee
John Holsworth
David Luebke
Jacqui Talton