Project 5 Discussion

Robb T. Koether

Hampden-Sydney College

Wed, Nov 18, 2009
Outline

1. Introduction
2. The Interface
3. The Shaders
4. The OpenGL State
   - Built-in Vertex Attributes
   - Built-in Uniform Variables
   - Built-in Functions
5. Design
6. Due Dates
This program will implement the lighting model in a programmable GPU.

The four front machines in Bagby 120 have programmable GPUs.

We will use the language GLSL (GL Shading Language) to program them. (See Chapter 9.)

There is also the Cg language, developed by NVidia, which we will not use.
### The User Interface

<table>
<thead>
<tr>
<th>Event</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Press +</td>
<td>Zoom in</td>
</tr>
<tr>
<td>Press −</td>
<td>Zoom out</td>
</tr>
<tr>
<td>Press ←</td>
<td>Rotate the camera left</td>
</tr>
<tr>
<td>Press →</td>
<td>Rotate the camera right</td>
</tr>
<tr>
<td>Press ↑</td>
<td>Rotate the camera up</td>
</tr>
<tr>
<td>Press ↓</td>
<td>Rotate the camera down</td>
</tr>
<tr>
<td>Press &gt;</td>
<td>Increase fineness of mesh</td>
</tr>
<tr>
<td>Press &lt;</td>
<td>Decrease fineness of mesh</td>
</tr>
<tr>
<td>Press W</td>
<td>Toggle wireframe</td>
</tr>
<tr>
<td>Press P</td>
<td>Toggle shader/fixed pipeline</td>
</tr>
<tr>
<td>Right-click</td>
<td>Pop-up menu</td>
</tr>
</tbody>
</table>

**Event Key:**
- Press
- Right-click
The pop-up menu will contain four items:

- Point light source
- Directional light source
- Local viewer
- Infinite viewer
The vertex shader performs operations on the vertices as they are sent down the pipeline.

- It has access to most of the OpenGL state.
- In addition, we can pass it values from the application program.
- Its output is sent on to the fragment shader.
The Fragment Shader

- The fragment shader receives the rasterized fragments (pixels), along with any values sent to it by the vertex shader.
- It also has access to the OpenGL state, but only to those values that make sense for fragments.
- It computes the final shade for the pixel.
Built-in Vertex Attributes

- `vec4 gl_Color;`
- `vec3 gl_Normal;`
- `vec4 gl_Vertex;`
### Built-in Uniform Matrix Variables

- `mat4 gl_ModelViewMatrix;`
- `mat4 gl_ProjectionMatrix;`
- `mat4 gl_ModelViewProjectionMatrix;`
- `mat4 gl_NormalMatrix;`
Built-in Uniform Material Properties

```
struct gl_MaterialParameters {
    vec4 emission;
    vec4 ambient;
    vec4 diffuse;
    vec4 specular;
    float shininess;
}

gl_MaterialParameters gl_FrontMaterial,
    gl_BackMaterial;
```
Built-in Uniform Light Source Properties

```cpp
struct gl_LightSourceParameters {
    vec4 ambient;
    vec4 diffuse;
    vec4 specular;
    vec4 position;
}

gl_LightSourceParameters gl_LightSource[8];
```
Built-in Uniform Light Products

```c
struct gl_LightProducts
{
    vec4 ambient;
    vec4 diffuse;
    vec4 specular;
}

gl_LightProducts gl_FrontLightProduct[8],
    gl_BackLightProduct;
```
Built-in Uniform Light Model Properties

```
struct gl_LightModelParameters
{
    vec4 ambient;
}

gl_LightModelParameters gl_LightModel;
```
Built-in Uniform Light Model Products

```cpp
struct gl_LightModelProducts {
    vec4 sceneColor;
};

gl_LightModelProducts gl_FrontLightModelProduct, gl_BackLightModelProduct;
```
Built-in Operators

**Built-in Operators**

\[ \text{vec} + \text{vec} = \text{vec}. \]
\[ \text{vec} - \text{vec} = \text{vec} \]
\[ \text{float} \times \text{vec} = \text{vec}. \]
\[ \text{vec} \times \text{vec} = \text{vec}. \]
\[ \text{mat} \times \text{vec} = \text{vec}. \]
\[ \text{mat} \times \text{mat} = \text{mat}. \]
Built-in Functions

- `float length(vec);`
- `float dot(vec, vec);`
- `vec cross(vec, vec);`
- `vec normalize(vec);`
Your design should answer the following questions.

- What variables are needed in the vertex shader?
- What variables are needed in the fragment shader?
- Which variables, if any, are uniform?
- Which variables, if any, are attributes?
- Which variables, if any, are varying?
- Which lighting calculations should be done in the vertex shader and which in the fragment shader?
The Due Dates

<table>
<thead>
<tr>
<th>Assignment</th>
<th>Due Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design</td>
<td>Mon, Nov 30</td>
</tr>
<tr>
<td>Program</td>
<td>Mon, Dec 7</td>
</tr>
</tbody>
</table>