CSE167
Introduction to
Computer Graphics

Matthias Zwicker
University of California, San Diego
Spring 2008

Cubic Bézier patch
• 4x4 mesh of control points

\[ q_0 = \text{Bez}(u, p_0, p_1, p_4) \]
\[ q_1 = \text{Bez}(u, p_2, p_3, p_5) \]
\[ q_2 = \text{Bez}(u, p_0, p_4, p_8) \]
\[ q_3 = \text{Bez}(u, p_2, p_6, p_10) \]
\[ r_0 = \text{Bez}(v, q_0, q_1, q_2) \]
\[ r_1 = \text{Bez}(v, q_1, q_2, q_3) \]

x(u, v) = \text{Bez}(u, r_0, r_1, r_2)

Tensor product formulation
• Corresponds to weighted average formulation
• Construct two-dimensional weighting function as product of two one-dimensional functions

\[ x(u, v) = \sum_i \sum_j p_{ij} B_i(u) B_j(v) \]
• Bernstein polynomials \( B_i \) \( B_j \) as for curves

Tessellating a Bézier patch
• Uniform tessellation is easiest
  - Evaluate points on a grid of \( u, v \) coordinates
  - Compute tangents at each point, take cross product to get per-vertex normal
  - Draw triangle strips (several choices of direction)

Piecewise Bézier surface
• \( C^0 \) continuity: share control points along edges
• \( C^1 \) continuity: parametric curves that cross each edge need to be \( C^1 \) continuous

\[ \text{C}^0 \text{ continuity} \hspace{2cm} \text{C}^1 \text{ continuity} \]

Today
Shader programming
• Environment mapping
• Toon shading
### More realistic illumination

- In real world, at each point in scene light arrives from all directions
  - Not just from point light sources
- Environment maps
  - Store “omni-directional” illumination as images
  - Each pixel corresponds to light from a certain direction

### Capturing environment maps

- “360 degrees” panoramic image
- Instead of 360 degrees panoramic image, take picture of mirror ball (light probe)

<table>
<thead>
<tr>
<th>Light probes</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="http://www.debevec.org/Probes/" alt="Light probes" /></td>
</tr>
</tbody>
</table>

### Environment maps as light sources

**Simplifying assumption**

- Assume light captured by environment map is emitted from infinitely far away
- Environment map consists of directional light sources
  - Value of environment map is defined for each direction, independent of position in scene
- Use single environment map at each point in the scene
- Approximation!

### Environment maps applications

- Use environment map as “light source”

<table>
<thead>
<tr>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="http://www.debevec.org/Probes/" alt="Global illumination" /></td>
<td>Global illumination</td>
</tr>
<tr>
<td><img src="http://www.debevec.org/Probes/" alt="Reflection mapping" /></td>
<td>Reflection mapping</td>
</tr>
</tbody>
</table>

### Environment maps applications

- Use environment map as “light source”

<table>
<thead>
<tr>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="http://www.debevec.org/Probes/" alt="Global illumination" /></td>
<td>Global illumination</td>
</tr>
</tbody>
</table>

### Cube environment maps

- Store incident light on six faces of a cube instead of on sphere

<table>
<thead>
<tr>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="http://www.debevec.org/Probes/" alt="Spherical map" /></td>
<td>Spherical map</td>
</tr>
<tr>
<td><img src="http://www.debevec.org/Probes/" alt="Cube map" /></td>
<td>Cube map</td>
</tr>
</tbody>
</table>
### Cube environment maps

**Cube map look-up**
- Given direction \((x, y, z)\)
- Largest coordinate component determines cube map face
- Dividing by magnitude of largest component yields coordinates within face
- In GLSL
  - Use \((x,y,z)\) direction as texture coordinates to `samplerCube`

### Reflection mapping

- Simulate mirror reflection
- Compute reflection vector at each pixel
- Use reflection vector to look up cube map
- Rendering cube map itself is optional

![Reflection mapping](image)

### Reflection mapping in GLSL

**Application setup**

#### Load, bind a cube environment map
```glsl
glBindTexture(GL_TEXTURE_CUBE_MAP, ...);
glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_X, ...);
glTexImage2D(GL_TEXTURE_CUBE_MAP_NEGATIVE_X, ...);
glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_Y, ...);
...
glEnable(GL_TEXTURE_CUBE_MAP);
```

### Reflection mapping in GLSL

**Vertex shader**
- Compute viewing direction
- Reflection direction
  - Use `reflect` function
- Pass reflection direction to fragment shader

**Fragment shader**
- Look-up cube map using interpolated reflection direction
  ```glsl
  varying float3 refl;
  uniform samplerCube envMap;
  textureCube(envMap, refl);
  ```

### Reflection mapping examples

- Approximation, reflections are not accurate

![Reflection mapping examples](image)

### Demo

- NVidia CG shading language
- Cg c7 reflection
**Today**

Shader programming
- Environment mapping
- Toon shading

**Toon shading**
- Simple cartoon style shader
- Emphasize silhouettes
- Discrete steps for diffuse shading, highlights

![Off-line toon shader](image1)
![GLSL toon shader](image2)

**Toon shading**
- Silhouette edge detection
  - Compute dot product of viewing direction \( \mathbf{v} \) and normal \( \mathbf{n} \)
    
    \[
    \text{edge} = \max(0, \mathbf{n} \cdot \mathbf{v})
    \]
  - Use 1D texture to define edge ramp
    
    \[
    \text{edge} = \max(0, \mathbf{n} \cdot \mathbf{v})
    \]

![edge detection diagram](image3)

**Toon shading**
- Compute diffuse and specular shading
  
  \[
  \text{diffuse} = \mathbf{n} \cdot \mathbf{L} \quad \text{specular} = (\mathbf{n} \cdot \mathbf{h})^\alpha
  \]
  
  - Use 1D textures diffuseramp, specularramp to map diffuse and specular shading to colors
  
  **Final color**

  \[
  \text{uniform sampler1D diffuseramp;}
  \text{uniform sampler1D specularramp;}
  \text{c} = \text{e} \cdot (\text{texture1D(diffuse, diffuseramp)} + \text{texture1D(specular, specularramp)});
  \]

**More shaders**
- OpenGL shading language book
- NVidia shading library
  - Most shaders are in HLSL
- NVidia Cg toolkit
  - Predecessor of GLSL
  - Lots of example shaders
    - [http://developer.nvidia.com/object/cg_toolkit_1_1.html](http://developer.nvidia.com/object/cg_toolkit_1_1.html)
<table>
<thead>
<tr>
<th>Next time</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Shadows</td>
</tr>
</tbody>
</table>