ECE4893A/CS4803MPG: MULTICORE AND GPU PROGRAMMING FOR VIDEO GAMES

Lecture 13: Environment and Bump Mapping

Prof. Aaron Lanterman
School of Electrical and Computer Engineering
Georgia Institute of Technology

Real-time graphics has come a long way

Virtual Fighter (SEGA Corporation)
NV1 50K triangles/sec 1M pixel ops/sec
1995

Dead or Alive 3 (Tecmo Corporation)
Xbox (NV2A) 100M triangles/sec 1G pixel ops/sec
2001

Dawn (NVIDIA Corporation)
GeForce FX (NV30) 200M triangles/sec 2G pixel ops/sec
2003

The Cg Tutorial

- Can get Cg Toolkit, example code, etc. here:

Image from “Teaching Cg” Powerpoint presentation:
developer.nvidia.com/object/cg_tutorial_teaching.html

Image from “Teaching Cg” Powerpoint presentation:
developer.nvidia.com/object/cg_tutorial_teaching.html

Nice framework for experimentation
**Cube maps**


**Reflection and refraction**

\[ \eta_1 \sin(\theta_i) = \eta_2 \sin(\theta_t) \]

\[ T = \text{refract}(I, N, \eta_{\text{Ratio}}) \]

\[ R = \text{reflect}(I, N) \]

\[ \theta_i = \theta_s \]

**Cg vertex shader for reflective mapping**

```c
void C7E1v_reflection(float4 position : POSITION,
float2 texCoord : TEXCOORD0, float3 normal : NORMAL,
out float4 oPosition : POSITION,
out float2 oTexCoord : TEXCOORD0,
out float3 R : TEXCOORD1,
uniform float3 eyePositionW,
uniform float4x4 modelViewProj,
uniform float4x4 modelToWorld)
{
    oPosition = mul(modelViewProj, position);
    oTexCoord = texCoord;
    // Compute position and normal in model space
    float3 positionW = mul(modelToWorld, position).xyz;
    float3 N = normalize(N);
    // Compute the incident and reflected vectors
    float3 I = positionW - eyePositionW; R = reflect(I, N);
}
```

From “The Cg Tutorial,” p. 177

**Cg pixel shader for reflective mapping**

```c
void C7E2f_reflection(float2 texCoord : TEXCOORD0,
float3 R : TEXCOORD1,
out float4 color : COLOR,
uniform float reflectivity,
uniform sampler2D decalMap,
uniform samplerCUBE environmentMap)
{
    // Fetch reflected environment color
    float4 reflectedColor = texCUBE(environmentMap, R);
    // Fetch the decal base color
    float4 decalColor = tex2D(decalMap, texCoord);
    color = lerp(decalColor, reflectedColor, reflectivity);
}
```

From “The Cg Tutorial,” p. 180
Different indices of refraction

<table>
<thead>
<tr>
<th>Material</th>
<th>Index of Refraction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vacuum</td>
<td>1.0</td>
</tr>
<tr>
<td>Air</td>
<td>1.0003</td>
</tr>
<tr>
<td>Water</td>
<td>1.333</td>
</tr>
<tr>
<td>Glass</td>
<td>1.5 (ordinary window glass)</td>
</tr>
<tr>
<td>Plastic</td>
<td>1.5</td>
</tr>
<tr>
<td>Diamond</td>
<td>2.417</td>
</tr>
</tbody>
</table>

Images from Thomas Kerwin, “Refraction in OpenGL,” www.cse.ohio-state.edu/~kerwin/refraction.html

Cg vertex shader for refractive mapping

```cpp
void C7E3v_refraction(float4 position : POSITION, float2 texCoord : TEXCOORD0, float3 normal : NORMAL, out float4 oPosition : POSITION, out float2 oTexCoord : TEXCOORD0, out float3 T : TEXCOORD1, uniform float etaRatio, uniform float3 eyePositionW, uniform float4x4 modelViewProj, uniform float4x4 modelToWorld)
{
    oPosition = mul(modelViewProj, position);
    oTexCoord = texCoord;
    // Compute position and normal in world space
    float3 positionW = mul((float3x3)modelToWorld, position).xyz;
    float3 N = mul((float3x3)modelToWorld, normal);
    N = normalize(N);
    // Compute the incident and refracted vectors
    float3 I = positionW - eyePositionW;
    T = refract(I, N, etaRatio);
}
```

From “The Cg Tutorial,” p. 187

Cg pixel shader for refractive mapping

```cpp
void C7E4f_refraction(float2 texCoord : TEXCOORD0, float3 T : TEXCOORD1, out float4 color : COLOR, uniform float transmittance, uniform sampler2D decalMap, uniform samplerCUBE environmentMap)
{
    // Fetch the decal base color
    float4 decalColor = tex2D(decalMap, texCoord);
    // Fetch refracted environment color
    float4 refractedColor = texCUBE(environmentMap, T);
    // Compute the final color
    color = lerp(decalColor, refractedColor, transmittance);
}
```

From “The Cg Tutorial,” p. 188

Chromatic dispersion

Images from Thomas Kerwin, “Refraction in OpenGL,” www.cse.ohio-state.edu/~kerwin/refraction.html
Fresnel effect

- Some light reflects and some refracts
- Think about looking into water
  - At shallow angles, a lot of reflection and little refraction
  - Looking straight in, a lot of refraction and a little reflection
- Empirical approximation:
  \[ \text{reflectCoeff} = \max(0, \min(1, \text{bias} + \text{scale}(1 + I \cdot N)\text{power})) \]
  \[ C_{\text{final}} = \text{reflectCoeff} \times C_{\text{reflected}} + (1 - \text{reflectCoeff})C_{\text{refracted}} \]

From “The Cg Tutorial,” p. 189

Bump mapping

Bump mapping examples

- Height map
- Normal map

Top row from Wikipedia entry on “bump mapping”

Bottom row from Søren Dreijer, “Bump Mapping Using Cg (3rd Edition),”

Images from Paul Baker, “Simple Bumpmapping,”
www.paulsprojects.net/tutorials/simplebump/simplebump.html

Shader effect movies

- Bump mapping demo with the Cimg library
  http://video.google.com/videoplay?docid=1570416667092534064

- Bump mapping and reflective textures
  - (HLEH - Half Life mod???)
  http://www.youtube.com/watch?v=FrmpyHc8hXc4

- Bump mapping on the Nintendo DS
  http://www.youtube.com/watch?v=Syte5IE-ofg
Storing normals in textures

- Textures don’t have to store color; we can store other things as well, like normals
  - Use r, g, b components to store, x, y, z of normal
- Problem: Textures take [0,1] values; normals need [-1,1] values
- Easy solution: “Range Compression”

```
colorComponent = 0.5 * normalComponent + 0.5;
normalComponent = 2 * (colorComponent - 0.5);
```

From “The Cg Tutorial,” p. 202

Creating normal map from height field

- Height field $H(u,v)$
  
  $\begin{bmatrix}
  H_x - H_r & H_y - H_s & H_z - H_e & 1 \\
  H_x - H_r & H_y - H_s & H_z - H_e & 1
  \end{bmatrix}
  
- In flat regions, normal is (0,0,1), i.e. pointing “up”

From “The Cg Tutorial,” p. 203

Cg vertex shader for bump mapping

```
void C8E1v_bumpWall(float4 position : POSITION,
float2 texCoord : TEXCOORD0,
out float4 oPosition : POSITION,
out float2 oTexCoord : TEXCOORD0,
out float3 lightDirection : TEXCOORD1,
uniform float3 lightPosition, // Objectspace
uniform float4x4 modelViewProj)
|
  cPosition = mul(modelViewProj, position);
  cTexCoord = texCoord;
  // Difference vectors for object-space light direction
  lightDirection = lightPosition - position.xyz;
}
```

From “The Cg Tutorial,” p. 205

Cg pixel shader for bump mapping

```
float3 expand(float3 v) { return (v-0.5)*2; }
void C8E2f_bumpSurf(float2 normalMapTexCoord : TEXCOORD0,
float3 lightDir : TEXCOORD1,
out float4 color : COLOR,
uniform sampler2D normalMap,
uniform samplerCUBE normalizeCube)
|
  // Normalizes light vector with normalization cube map
  float3 lightTex = texCUBE(normalizeCube, lightDir).xyz;
  float3 light = expand(lightTex);
  // Sample and expand the normal map texture
  float3 normalTex = tex2D(normalMap, normalMapTexCoord).xyz;
  float3 normal = expand(normalTex);
  // Diffuse lighting
  color = dot(normal, light);
```

From “The Cg Tutorial,” p. 206