Textures and Shaders

CS 148, Summer 2012
Introduction to Computer Graphics and Imaging
Justin Solomon
Updated Staff/Office Hours

Instructor: Justin Solomon  
Email: justin.solomon@stanford.edu  
Office: Clark S297  
Office hours: Tuesdays, 9am-11am;  
Fridays, 2:15pm-4:15pm

CA: Blake Carpenter  
Email: blakec@stanford.edu  
Office hours: Wednesdays, 2pm-4pm;  
Thursdays, 1pm-2pm (Gates B24)

CA: Taesung Park  
Email: taesung@stanford.edu  
Office hours: Mondays, 2:15pm-4:15pm (Gates B24)
July 18, in class
TWO PAGES OF NOTES

WARNING

MIDTERM APPROACHING

July 18, in class
<leftover>
Hierarchical Modeling

glTranslatef(0,1.5,0);
drawTorso();
glPushMatrix();
    glTranslatef(0,5,0);
drawShoulder();
glPushMatrix();
    glRotatef(neck_y,0,1,0);
    glRotatef(neck_x,1,0,0);
drawHead();
glPopMatrix();
glPushMatrix();
    glTranslatef(1.5,0,0);
    glRotatef(l_shoulder_x);
drawUpperArm(); ...
GL Transformation Model

\[ T = M_{\text{projection}} \cdot M_{\text{camera}} \cdot M_{\text{model}} \]

- Camera coordinates to normalizes device coordinates
- Local coordinates to camera coordinates

```c
glMatrixMode(GL_PROJECTION);
glMatrixMode(GL_MODELVIEW);
```

Applied to all geometry
Camera coordinates to
normalized device
coordinates

Local coordinates to
camera coordinates

\[ T = M_{\text{projection}} \cdot M_{\text{camera}} \cdot M_{\text{model}} \]

Can do whatever you want with these.

```c
glMatrixMode(GL_PROJECTION);
glMatrixMode(GL_MODELVIEW);
```
</leftover>
Per-Vertex Coloring
Per-Vertex Coloring
Problem

Texture and geometry resolution might not coincide.
Problem

Texture and geometry resolution might not coincide.

Hi-res texture, Lo-res geometry

http://www.ten24.info/upload/pages/Head%20Texture.jpg
Texture Coordinates

$\mathbf{X}(u,v)$

$\Omega$

$\mathbb{R}^2$

$\mathbb{R}^3$

$U(x,y,z)$
Texture Coordinates

Images store hi-res texture

\[ U(x, y, z) \]

\[ X(u, v) \]

\[ \mathbb{R}^3 \rightarrow \mathbb{R}^2 \]

http://graphics.stanford.edu/courses/cs468-10-fall/LectureSlides/12_Parameterization1.pdf
Topological Issues: Texture Atlas

http://www.blender.org/development/release-logs/blender-2.46/uv-editing/
(aside!)
Research in Parameterization

Trade off between area and angle preservation

(aside!) Research in Parameterization


Deal with high genus, distortion
Each vertex in each triangle stores \((u,v)\) texture coordinates

Same vertex in adjacent triangles might have different \((u,v)\)
Barycentric Parameterization

Goal: Assign \((u,v)\) coordinate to each mesh vertex.
Barycentric Parameterization

Goal: Assign \((u,v)\) coordinate to each mesh vertex.

1. Fix \((u,v)\) coordinates of boundary.
2. Want interior vertices to be at the (bary)center of their neighbors:

\[
    v_i = \frac{1}{\text{valence}(i)} \sum_{(i,j) \text{ neighbors}} v_j
\]
Rasterizing Textured Triangles

“Texture Lookup”
Rasterizing Textured Triangles

Texture Memory

“Texture Lookup”

http://raweb.inria.fr/rapportsactivite/RA2006/iparla/uid33.html
http://runfatgirl.files.wordpress.com/2008/05/bigstockphoto_skin_texture_308750.jpg
Perspective-Correct Texturing

Flat

Affine

Correct

Can’t use image plane barycentric coordinates!
Texture Interpolation

Unlikely to land at pixel center

Between four pixels!
Texture Interpolation

Most efficient: Rounding
Texture Interpolation

Smoothen: Bilinear Interpolation
Texture Interpolation

Interpolate linearly

Smoother: Bilinear Interpolation
Texture Interpolation

Smoothen: Bilinear Interpolation
Texture Interpolation

Smooother: Bilinear Interpolation
Texture Interpolation

Smoother: Bilinear Interpolation
Texture Interpolation

Smother: Bilinear Interpolation

Same formula!
Texture Can Be *Too* Detailed

Small image

Large texture

Adjacent rendered pixels are far apart in texture
MIP Maps

Precompute small images

http://glasnost.itcarlow.ie/~powerk/GeneralGraphicsNotes/texturemapping/texturemapping.htm
http://www.ithinkibrokeit.co.uk/articles/graphics06/page6.php
MIP Maps

“Multum in Parvo”
(“much in little”)
MIP Maps

More when we discuss sampling!

Precompute small images
We’re asking a lot of the GPU!
What’s in common?

- Apply transformations... to **millions** of triangles
- Look up texture... for **millions** of pixels
- Compute lighting... for **millions** of vertices
- Pick closest... of **millions** of fragments

**Repetitive operations**
What’s in common?

Apply transformations ... to **millions** of triangles
Look up texture ... for **millions** of pixels
Compute lighting ... for **millions** of vertices
Pick closest ... of **millions** of fragments

Repetitive operations

Single instruction,
Multiple data
What’s in common?

Apply transformations ... to millions of triangles

Look up texture... for millions of pixels

Compute lighting... for millions of vertices

Pick closest... of millions of fragments

SIMD

Repetitive operations
Modern GPU Throughput

\[ \text{fragment rate} = |\text{Image}| \times \text{frames/sec} \times \text{depth complexity} \]
\[ = (1440 \times 900) \times 60 \times 4 \]
\[ \approx 32 \text{ gigafragments/sec} \]

\[ \text{bandwidth} = \text{gigafragments/sec} \times \text{bytes/fragment} \]
\[ = 32 \times 4 \]
\[ = 96 \text{ GB/sec} \]

\[ \text{triangle rate} = (\text{gigafragments/sec})/(\text{fragments/triangle}) \]
\[ = 32/16 \]
\[ = 2 \text{ gigatriangles/sec} \]
<table>
<thead>
<tr>
<th>Year</th>
<th>Product</th>
<th>Triangles/sec</th>
<th>Fragments/sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>1998</td>
<td>Riva ZX</td>
<td>3m</td>
<td>100m</td>
</tr>
<tr>
<td>1999</td>
<td>Riva TNT2</td>
<td>9m</td>
<td>350m</td>
</tr>
<tr>
<td>2000</td>
<td>GeForce2 GTS</td>
<td>25m</td>
<td>664m</td>
</tr>
<tr>
<td>2001</td>
<td>GeForce3</td>
<td>30m</td>
<td>800m</td>
</tr>
<tr>
<td>2002</td>
<td>GeForce Ti 4600</td>
<td>60m</td>
<td>1200m</td>
</tr>
<tr>
<td>2003</td>
<td>GeForce FX</td>
<td>167m</td>
<td>2000m</td>
</tr>
<tr>
<td>2004</td>
<td>GeForce 6800 Ultra</td>
<td>170m</td>
<td>6800m</td>
</tr>
<tr>
<td>2005</td>
<td>GeForce 7800 GTX</td>
<td>940m</td>
<td>10300m</td>
</tr>
<tr>
<td>2006</td>
<td>GeForce 7900 GTX</td>
<td>1400</td>
<td>15600m</td>
</tr>
<tr>
<td>2007</td>
<td>GeForce 8800 GTX</td>
<td>1800m</td>
<td>36800m</td>
</tr>
<tr>
<td>2008</td>
<td>GeForce GTX 280</td>
<td></td>
<td>48160m</td>
</tr>
<tr>
<td>2010</td>
<td>GeForce 480</td>
<td></td>
<td>42000m</td>
</tr>
<tr>
<td>2011</td>
<td>GeForce GTX 580</td>
<td></td>
<td>49400m</td>
</tr>
</tbody>
</table>
Fixed-Function Pipeline
More Generic Pipeline

- Command
- Vertex
- Assembly
- Rasterization
- Fragment
- FB operations
- Display

Introducing “Shaders”

Short program customizing a part of the graphics pipeline.

Misnomer: They do more than shade!
Types of Shaders

Customizes:
- Color and lighting
- Texture coordinate
- Position and transformation
- Normal

Vertex shader or "Vertex program"
Types of Shaders

Vertex shader or "Vertex program"
Types of Shaders

Vertex shader or "Vertex program"
Types of Shaders

Customizes:
- Visibility and transparency
- Color
- Texturing and shading
- Composition
Types of Shaders

http://www.youtube.com/watch?v=7LvjWGoIzQk

Fragment shader
Types of Shaders

Fragment shader

http://www.youtube.com/watch?v=9ETfTD6L2l
Shader Considerations

- Single-precision arithmetic
- Branching, loops expensive
- No access to neighboring fragments/vertices
- Cannot bind output buffer as input
Shader Considerations

- Limited stack/instruction count
- Timeout possibility
- Support, debugging not consistent
Shader Language Options

- **Assembly**
  
  *Issue commands directly to GPU*

- **GLSL**
  
  *OpenGL*

- **Cg**
  
  *OpenGL/Direct3D*

- **HLSL**
  
  *Direct3D*
Shader Language Options

- **Assembly**
  
  *Issue commands directly to GPU*

- **GLSL**
  
  *OpenGL*

- **Cg**
  
  *OpenGL/Direct3D*

- **HLSL**
  
  *Direct3D*
void main() {
    gl_Position = gl_ProjectionMatrix
                  * gl_ModelViewMatrix
                  * gl_Vertex;
    gl_FrontColor = gl_Color;
    gl_BackColor = gl_Color;
}

http://graphics.stanford.edu/courses/cs148-10-summer/docs/08_rendering3.pdf
void main() {
    gl_Position = gl_ProjectionMatrix
                  * gl_ModelViewMatrix
                  * gl_Vertex;
    gl_FrontColor = gl_Color;
    gl_BackColor = gl_Color;
}

Replace for per-vertex lighting
void main() {
    gl_FragColor = gl_Color;
}

Input

Output

http://graphics.stanford.edu/courses/cs148-10-summer/docs/08_rendering3.pdf
void main() {
    gl_FragColor = gl_Color;
}

Replace for per-fragment lighting
GLSL Variable Types

- **Attribute**: Per-vertex property; input to vertex shader
  
  \( \text{gl\_Color}, \text{gl\_Normal}, \text{gl\_MultiTexCoord0} \)

- **Uniform**: Constant during shader execution
  
  \( \text{gl\_ModelViewMatrix}, \text{gl\_ProjectionMatrix} \)

- **Varying**: Output by vertex shader and interpolated before running fragment shader
  
  \( \text{gl\_FrontColor}, \text{height} \)
void main() {
    gl_Position = ftransform();
    gl_TexCoord[0] =
        gl_MultiTexCoord0;
}

http://graphics.stanford.edu/courses/cs148-10-summer/docs/08_rendering3.pdf
Texture Mapping

```cpp
void main() {
    gl_Position = ftransform();
    gl_TexCoord[0] = gl_MultiTexCoord0;
}
```

http://graphics.stanford.edu/courses/cs148-10-summer/docs/08_rendering3.pdf
uniform sampler2D myTexture;

void main() {
    gl_FragColor = texture2D(myTexture, gl_TexCoord[0].xy);
}

http://graphics.stanford.edu/courses/cs148-10-summer/docs/08_rendering3.pdf

texture.frag
Common Shader Tasks

4M triangles

500 triangles

Bump/normal mapping

http://upload.wikimedia.org/wikipedia/commons/3/36/Normal_map_example.png
Common Shader Tasks

Bump/normal mapping

http://www.kxcad.net/lightwave/lightwave_3d_g/normalmap_normal.png
http://charhut.info/files/cs280/CliffNormal.png
http://imageshack.us/photo/my-images/412/gahumanefacenorce9.png/sr=1
http://users.tkk.fi/~mliukka/textures/Brickwall_windows_01_pom_ddn.jpg
http://www.ericspitler.com/images/2d/normalmap.jpg
Common Shader Tasks

Bump/normal mapping

http://upload.wikimedia.org/wikipedia/commons/4/4e/Bump_map_vs_isosurface2.png
Common Shader Tasks

Original  Bump  Displacement

http://www.spot3d.com/vray/help/15oSP1/tutorials_displacement.htm

Displacement mapping
Common Shader Tasks

Use a smaller set of colors

Toon shading
Common Shader Tasks

Complex materials/lighting
Common Shader Tasks

Specialized computations

Common Shader Tasks


Image processing
Common Shader Tasks

Simulation

http://www.geeks3d.com/20080812/nvidia-physx-powerpack-download/
http://pcper.com/images/reviews/245/graw_physx_3.jpg
Common Shader Tasks

Geometry processing

http://web.media.mit.edu/~gordonw/OpenGL/images/teapot_vertexShader.jpg
Crowds

Common Shader Tasks

http://developer.amd.com/gpu_images/ManagingLargeCrowds.png
“General purpose” GPU
“General purpose” GPU
“General purpose” GPU
Textures and Shaders

CS 148, Summer 2012
Introduction to Computer Graphics and Imaging
Justin Solomon