Introduction to Computer Graphics and OpenGL

Texture Mapping

Sung-Eui Yoon
(윤성의)

Course URL:
http://sglab.kaist.ac.kr/~sungeui/ETRI_CG/
Outline

- Texture mapping overview
- Perspective-correct interpolation
- Texture filtering
Texture Mapping

- Requires lots of geometry to fully represent complex shapes of models
- Add details with image representations

Excerpted from MIT EECS 6.837, Durand and Cutler
The Quest for Visual Realism

At what point do things start looking real?

For more info on the computer artwork of Jeremy Birn see [http://www.3drender.com/jbirn/productions.html](http://www.3drender.com/jbirn/productions.html)
Photo-Textures

For each triangle in the model establish a corresponding region in the phototexture.

During rasterization interpolate the coordinate indices into the texture map.

Excerpted from MIT EECS 6.837, Durand and Cutler
Texture Maps in OpenGL

- Specify normalized texture coordinates at each of the vertices \((u, v)\)
- Texel indices \((s,t) = (u, v) \cdot (width, height)\)

```c
glBindTexture(GL_TEXTURE_2D, texID)
glBegin(GL_POLYGON)
    glTexCoord2d(0,1); glVertex2d(-1,-1);
    glTexCoord2d(1,1); glVertex2d( 1,-1);
    glTexCoord2d(1,0); glVertex2d( 1, 1);
    glTexCoord2d(0,0); glVertex2d(-1, 1);
glEnd()
```
Wrapping

- The behavior of texture coordinates outside of the range [0,1) is determined by the texture wrap options.

```c
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, wrap_mode )
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, wrap_mode )
```

![GL_CLAMP](image1.png) ![GL_REPEAT](image2.png)
The uniform sampling pattern in screen space corresponds to some sampling pattern in texture space that is not necessarily uniform.
Sampling Density Mismatch

- Sampling density in texture space rarely matches the sample density of the texture itself.

Oversampling (Magnification)  Undersampling (Minification)
Handling Oversampling

- How do we compute the color to assign to this sample?
Handling Oversampling

- How do we compute the color to assign to this sample?
- Nearest neighbor - take the color of the closest texel
How do we compute the color to assign to this sample?

- Nearest neighbor – take the color of the closest texel

- Bilinear interpolation

\[
\alpha = \frac{x - x_0}{x_1 - x_0} \quad \beta = \frac{y - y_0}{y_1 - y_0}
\]

\[
c = ((1- \alpha)c_0 + \alpha c_1)(1- \beta) + ((1- \alpha)c_2 + \alpha c_3)\beta
\]
Undersampling

- Details in the texture tend to pop (disappear and reappear)
  - Mortar (white substances) in the brick
- High-frequency details lead to strange patterns
  - Aliasing
Spatial Filtering

● To avoid aliasing we need to prefilter the texture to remove high frequencies
  ● Prefiltering is essentially a spatial integration over the texture
  ● Integrating on the fly is expensive: perform integration in a pre-process

![Samples and their extents](image1.png)

![Proper filtering removes aliasing](image2.png)
MIP Mapping

- MIP is an acronym for the Latin phrase \textit{multium in parvo}, which means "many in one place"
  - Constructs an \textit{image pyramid}
  - Each level is a prefiltered version of the level below resampled at half the frequency

- While rasterizing use the level with the sampling rate closest to the desired sampling rate
  - Can also interpolate between pyramid levels

- How much storage overhead is required?

$$\text{mip map size} = \sum_{i=0}^{\infty} \left( \frac{1}{4} \right)^i = \frac{1}{1 - \frac{1}{4}} = \frac{4}{3}$$
Finding the MIP Level

- Use the projection of a pixel in screen into texture space to figure out which level to use
Texture Filtering in OpenGL

- **Automatic creation**
  
  ```c
  gluBuild2DMipmaps(GL_TEXTURE_2D, GL_RGBA, width, height, GL_RGBA, GL_UNSIGNED_BYTE, data)
  ```

- **Filtering**
  
  ```c
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, filter)
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, filter)
  ```

  where filter is:

  ```
  GL_NEAREST
  GL_LINEAR
  ```

  ```
  GL_LINEAR_MIPMAP_LINEAR
  ```

  ```
  GL_NEAREST_MIPMAP_LINEAR
  ```

  ```
  GL_NEAREST_MIPMAP_NEAREST
  ```

  ```
  GL_LINEAR_MIPMAP_NEAREST
  ```

  inter-level    intra-level
Uses of Texture Maps

- Texture maps are used to add complexity to a scene
  - Easier to paint or capture an image than geometry
- Model light
- Model geometry, etc
Modeling Lighting

- **Light maps**
  - Supply the lighting directly
  - Good for static environments

- **Projective textures**
  - Can be used to simulate a spot light
  - Shadow maps

- **Environment maps**
  - A representation of the scene around an object
  - Good for reflection
Light Maps in Quake

- Light maps are used to store pre-computed illumination

<table>
<thead>
<tr>
<th>Data</th>
<th>Texture Maps</th>
<th>Light Maps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resolution</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td></td>
<td>RGB</td>
<td>Intensity</td>
</tr>
</tbody>
</table>

Textures Only vs. Textures & Light Maps

![Texture Maps Example]

Light map image by Nick Chirkov
Projective Textures

- Treat the texture as a slide in a projector
  - A good model for shading variations due to illumination (cool spotlights)

- Projectors work like cameras in reverse
  - Camera: color of point in scene $\rightarrow$ color of corresponding pixel
  - Projector: color of pixel $\rightarrow$ color of corresponding point in the scene
Shadow Maps

Use the depth map in the light view to determine if sample point is visible

Point in shadow visible to the eye, but not visible to the light
Environment Maps

- Simulate complex mirror-like objects
  - Use textures to capture environment of objects
  - Use surface normal to compute texture coordinates
Environment Maps - Example

T1000 in Terminator 2 from Industrial Light and Magic
Cube Maps

- Maps a viewing direction $b$ and returns an RGB color
  - Use stored texture maps

[Diagram of cube maps with images pointing to different directions]
Cube Maps

- Maps a viewing direction \( b \) and returns an RGB color
- Assume \( b = (x, y, z) \),

- Identify a face based on magnitude of \( x, y, z \)

- For the right face, compute texture coord. \((u,v)\)

\[
\begin{align*}
(u,v) &= (1,1) \\
x &= y = z \\
(u,v) &= (0,0) \\
x &= -y = -z \\
u &= (y+x)/(2x) \\
v &= (z+x)/(2x)
\end{align*}
\]
Environment Maps - Problems

- Expensive to update dynamically
- Not completely accurate

Reflection of swimming pool is wrong
Modeling Geometry

- Store complex surface details in a texture rather than modeling them explicitly
  - Bump maps
    - Modify the existing normal
  - Normal maps
    - Replace the existing normal
  - Displacement maps
    - Modify the geometry
  - Opacity maps and billboards
    - Knock-out portions of a polygon using the alpha channel
Bump Mapping

- Modifies the normal not the actual geometry
  - Texture treated as a heightfield
  - Partial derivatives used to change the normal
  - Causes surface to appear deformed by the heightfield
More Bump Map Examples

Note that silhouette edge of the object not affected!
Normal Mapping

- Replaces the normal rather than tweaking it
Displacement Mapping

- Texture maps can be used to actually move surface points
Opacity Maps

Use the alpha channel to make portions of the texture transparent.
Billboards

Replace complex geometry with polygons texture mapped with transparent textures
3D or Solid Textures

- Solid textures are three-dimensional assigning values to points in 3 space
  - Very effective at representing some types of materials such as marble and wood
  - The object is “carved” out of the solid texture

- Generally, solid textures are defined procedural functions rather than tabularized or sampled functions as used in 2D
Next Time

- Visibility