Texture Mapping

Thanks to Ed Angel, UNM

Texture Mapping

- The soup tin is easily described by pasting a label on the plain cylinder
- Texture mapping associates the color of a point with the color in an image: the texture
  - Soup tin: Each point on the cylinder gets the label's color
  - Question to address: Which point of the texture do we use for a given point on the surface?
  - Establish a mapping from surface points to image points
    - Different mappings are common for different shapes
    - We will, for now, just look at triangles (polygons)

Example Mappings

- Apply a 1D, 2D, or 3D image to geometric primitives
- Uses of Texturing
  - simulating materials
  - reducing geometric complexity
  - image warping
  - reflections

Texture Mapping

- Apply a 1D, 2D, or 3D image to geometric primitives
- Uses of Texturing
  - simulating materials
  - reducing geometric complexity
  - image warping
  - reflections

Texture Interpolation

- Specify where the vertices in world space are mapped to in texture space
- Linearly interpolate the mapping for other points in world space
  - Straight lines in world space go to straight lines in texture space
Interpolating Coordinates

\[ s_1 = \left( \frac{x_1 - x_2}{x_3 - x_2} \right) s + \frac{x_3}{x_3} \]
\[ s_2 = \left( \frac{x_1 - x_3}{x_2 - x_3} \right) s + \frac{x_2}{x_2} \]
\[ s = \left[ \begin{array}{c} s_1 + s_2 \end{array} \right] \]

Pipelines and Texture Mapping

- Texture mapping is done at rasterization
  - in canonical screen space
- When describing a scene, you assume that texture interpolation will be done in world space
- Which property of perspective projection means that the “wrong thing” will happen if we apply our simple interpolations from the previous slide?
  - Perspective correct texture mapping does the right thing, but at a cost
- Is it a problem with orthographic viewing?

Linear Interpolation of Textures

Texture Index Interpolation

Perspective Correction Hint

- Texture coordinate and color interpolation
  - either linearly in screen space
  - or using depth/perspective values (slower)
- Noticeable for polygons “on edge”

```glsl
glHint(GL_PERSPECTIVE_CORRECTION_HINT, hint) where hint is one of
- GL_DONT_CARE
- GL_NICEST
- GL_FASTEST```

Texture Interpolation Problem

Notice that uniform steps on the image plane do not correspond to uniform strips along the edge.
Texture Mapping and the OpenGL Pipeline

- Images and geometry flow through separate pipelines that join at the rasterizer
  - "complex" textures do not affect geometric complexity

Applying Textures I

- Three steps
  1. specify texture
     - read or generate image
     - assign to texture
  2. assign texture coordinates to vertices
  3. specify texture parameters
     - wrapping, filtering

Applying Textures II

- specify textures in texture objects
- set texture filter
- set texture function
- set texture wrap mode
- set optional perspective correction hint
- bind texture object
- enable texturing
- supply texture coordinates for vertex
  - coordinates can also be generated

Texture Objects

- Like display lists for texture images
  - one image per texture object
  - may be shared by several graphics contexts
- Generate texture names
  
  ```
  glGenTextures( n, *texIds );
  ```

Texture Objects (cont.)

- Create texture objects with texture data and state
  
  ```
  glBindTexture( target, id );
  ```
- Bind textures before using
  
  ```
  glBindTexture( target, id );
  ```

Texture Example

- The texture (below) is a 256 x 256 image that has been mapped to a rectangular polygon which is viewed in perspective
An Interactive Introduction to OpenGL Programming

**Specify Texture Image**

- Define a texture image from an array of texels in CPU memory
  
  ```
  glTexImage2D(target, level, components, w, h, border, format, type, *texels);
  ```
  
  - dimensions of image must be powers of 2
  - Texel colors are processed by pixel pipeline
    - pixel scales, biases and lookups can be done

**Converting A Texture Image**

- If dimensions of image are not power of 2
  
  ```
  gluScaleImage(format, w_in, h_in, type_in, *data_in, w_out, h_out, type_out, *data_out);
  ```
  
  - *in is for source image
  - *out is for destination image
  - Image interpolated and filtered during scaling

**Specifying a Texture: Other Methods**

- Use frame buffer as source of texture image
  
  ```
  glCopyTexImage2D(...) glCopyTexImage1D(...)
  ```
  
- Modify part of a defined texture
  
  ```
  glTexSubImage2D(...) glTexSubImage1D(...)
  ```
  
- Do both with `glCopyTexSubImage2D(...)` etc.

**Mapping a Texture**

- Based on parametric texture coordinates
  
  ```
  glTexCoord*(s, t) specified at each vertex
  ```

**Basic OpenGL Texturing**

- Specify texture coordinates for the polygon:
  - Use `glTexCoord2f(s, t)` before each vertex:
    - Eg: `glTexCoord2f(0, 0); glVertex3f(x, y, z);`
  
  **Create a texture object and fill it with texture data:**
  
  ```
  glGenTextures(num, &indices);
  glBindTexture(GL_TEXTURE_2D, identifier);
  glTexParameteri(GL_TEXTURE_2D, …, …);
  glTexImage2D(GL_TEXTURE_2D, …., ... to specify the texture data (the image itself)
  ```
An Interactive Introduction to OpenGL Programming

Basic OpenGL Texturing (cont)

- Enable texturing: `glEnable(GL_TEXTURE_2D)`
- State how the texture will be used:
  - `glTexEnvf(...)`
- Texturing is done after lighting

Nasty Details

- There are a large range of functions for controlling the layout of texture data:
  - You must state how the data in your image is arranged
  - E.g. `glTexParameteri(GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE, 1)` tells OpenGL not to skip bytes at the end of a row
  - You must state how you want the texture to be put in memory: how many bits per “pixel”, which channels...
  - Textures typically are square with width/height a power of 2
  - Common sizes are 32x32, 64x64, 256x256
  - Smaller uses less memory, and there is a finite amount of texture memory on graphics cards

Example Texture Map

Wrapping Mode

- Example:
  ```
  glTexParameteri( GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP/>
  glTexParameteri( GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT )
  ```

Texture Functions

- Controls how texture is applied
  ```
  glTexEnv{f|i}[v] (GL_TEXTURE_ENV, prop, param )
  ```
- GL_TEXTURE_ENV_MODE modes
  - `GL_MODULATE`
  - `GL_BLEND`
  - `GL_REPLACE`
- Set blend color with
  ```
  GL_TEXTURE_ENV_COLOR
  ```

Texture Functions

- Set blend color with
  ```
  GL_TEXTURE_ENV_COLOR
  ```

Basic OpenGL Texturing

- Set blend color with
  ```
  GL_TEXTURE_ENV_COLOR
  ```

Example Texture Map

- Example:
  ```
  glTexParameteri( GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP/>
  glTexParameteri( GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT )
  ```

Wrapping Mode

- Example:
  ```
  glTexParameteri( GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP/>
  glTexParameteri( GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT )
  ```

Texture Functions

- Controls how texture is applied
  ```
  glTexEnv{f|i}[v] (GL_TEXTURE_ENV, prop, param )
  ```
- GL_TEXTURE_ENV_MODE modes
  - `GL_MODULATE`
  - `GL_BLEND`
  - `GL_REPLACE`
Basic OpenGL Texturing

• Specify texture coordinates for the polygon:
  • Use glTexCoord2f(s,t) before each vertex:
    • Eg: glTexCoord2f(0,0); glVertex3f(x,y,z);

• Create a texture object and fill it with texture data:
  • glGenTextures(num, &indices)
  • glBindTexture(GL_TEXTURE_2D, identifier)
  • Following texture commands refer to the bound texture
  • glTexParameteri(GL_TEXTURE_2D, …, …) to specify parameters for use when applying the texture
  • glTexImage2D(GL_TEXTURE_2D, …) to specify the texture data (the image itself)

Basic OpenGL Texturing (cont)

• Enable texturing: glEnable(GL_TEXTURE_2D)
• State how the texture will be used:
  • glTexEnvf(…)
• Texturing is done after lighting

Nasty Details

• There are a large range of functions for controlling the layout of texture data:
  • You must state how the data in your image is arranged
    • Eg: glPixelStorei(GL_UNPACK_ALIGNMENT, 1) tells OpenGL not to skip bytes at the end of a row
  • You must state how you want the texture to be put in memory: how many bits per “pixel”, which channels,…

• Textures typically are square with width/height a power of 2
  • Common sizes are 32x32, 64x64, 256x256
  • Smaller uses less memory, and there is a finite amount of texture memory on graphics cards

Example Texture Map

• Query largest dimension of texture image
  • Typically largest square texture
  • Doesn’t consider internal format size
  • glGetIntegerv( GL_MAX_TEXTURE_SIZE, &size )

• Texture proxy
  • Will memory accommodate requested texture size?
    • No image specified; placeholder
    • If texture won’t fit, texture state variables set to 0
      • Doesn’t know about other textures
      • Only considers whether this one texture will fit all of memory

Is There Room for a Texture?

Texture Residency

• Working set of textures
  • High-performance, usually hardware accelerated
  • Textures must be in texture objects
  • A texture in the working set is resident
    • For residency of current texture, check GL_TEXTURE_RESIDENT state

• If too many textures, not all are resident
  • Can set priority to have some kicked out first
  • Establish 0.0 to 1.0 priorities for texture objects
An Interactive Introduction to OpenGL Programming

Tutorial: Texture

The Art of 3D Computer Animation and Effects
Isaac Kerlow