Texture Mapping

CPSC 314

Texture Mapping

- Real life objects have nonuniform colors, normals
- To generate realistic objects, reproduce coloring & normal variations = texture
- Can often replace complex geometric details

Texture Mapping

Introduced to increase realism
- Lighting/shading models not enough

Hide geometric simplicity
- Images convey illusion of geometry
- Map a brick wall texture on a flat polygon
- Create bumpy effect on surface

Associate 2D information with 3D surface
- Point on surface corresponds to a point in texture
- “Paint” image onto polygon

Color Texture Mapping

Define color (RGB) for each point on object surface

Two approaches
- Surface texture map (2D)
- Volumetric texture (3D)

Surface (2D) Textures: Texture Coordinates

Texture Image: 2D array of color values (texels)

Assigning texture coordinates \((s,t)\) at vertex with object coordinates \((x,y,z,w)\)
- Use interpolated \((s,t)\) for texel lookup at each pixel
- Use value to modify a polygon's color
  - Or other surface property
- Specified by programmer or artist

\(\text{glTexCoord2f}(s,t)\)
\(\text{glVertexf}(x,y,z,w)\)
Texturing Example

Example Texture Map

Texture Lookup: Tiling and Clamping

What if s or t is outside the interval [0...1]?

- Multiple choices
  - Cyclic repetition of texture to tile whole surface
    ```
    glTexParameteri( ... GL_TEXTURE_WRAP_S, GL_REPEAT, GL_TEXTURE_WRAP_T, GL_REPEAT, ... )
    ```
  - Clamp every component to range [0...1]
    ```
    glTexParameteri( ... GL_TEXTURE_WRAP_S, GL_CLAMP, GL_TEXTURE_WRAP_T, GL_CLAMP, ... )
    ```

Texture Coordinate Transformation

Motivation
- Change scale, orientation of texture on an object

Approach
- Texture matrix stack
- Transforms specified (or generated) tex coords
  ```
  glMatrixMode( GL_TEXTURE ):
  loadIdentity();
  glRotate();
  ...
  ```
- More flexible than changing (s,t) coordinates
Texture Functions

Once you have value from the texture map, can:
- Directly use as surface color: GL_REPLACE
  - Throw away old color, lose lighting effects
- Modulate surface color: GL_MODULATE
  - Multiply old color by new value, keep lighting info
  - Texturing happens after lighting, not refl!
- Use as surface color, modulate alpha: GL_DECAL
  - Like replace, but supports texture transparency
- Blend surface color with another: GL_BLEND
  - New values controls which of 2 colors to use

Specify desired behavior with glTexEnvi (GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, <mode>)

Texture Pipeline

\[
\begin{align*}
(x, y, z) & \quad \rightarrow \quad \{x, f\} \\
\text{Object position} & \quad \rightarrow \quad \text{Parameter space} \\
(-2.3, 7.1, 17.7) & \quad \rightarrow \quad (0.32, 0.29) \\
\text{Transformed} & \quad \rightarrow \quad \text{Parameter space} \\
\text{parameter space} & \quad \rightarrow \quad (0.52, 0.49) \\
\text{Texel space} & \quad \rightarrow \quad (0.46, 0.4, 0.36) \\
\text{Texel color} & \quad \rightarrow \quad (0.5, 0.5, 0.5) \\
\text{Final color} & \quad \rightarrow \quad \text{Object color}
\end{align*}
\]

Basic OpenGL Texturing

Create a texture object and fill it with texture data:
- glEnable(GL_TEXTURE_2D)
- glBindTexture(GL_TEXTURE_2D, identifier) to bind
  - Following texture commands refer to the bound texture
- glTexImage2D(GL_TEXTURE_2D, ..., ...) to specify parameters for use when applying the texture
- glTexImage2D(GL_TEXTURE_2D, ..., ...) to specify the texture data (the image itself)

Texture Objects and Binding

Texture object
- An OpenGL data type that keeps textures resident in memory and provides identifiers to easily access them
- Provides efficiency gains over having to repeatedly load and reload a texture
- You can prioritize textures to keep in memory
- OpenGL uses least recently used (LRU) if no priority is assigned

Texture binding
- Which texture to use right now
- Switch between preloaded textures

Basic OpenGL Texturing (cont.)

Enable texturing:
- glEnable(GL_TEXTURE_2D)

State how the texture will be used:
- glTexEnv(...)

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

Low-Level Details

Large range of functions for controlling layout of texture data
- State how the data in your image is arranged
- e.g., glTexImage2D(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 into memory: how many bits per “pixel”, which channels,...

Textures must have a size of power of 2
- Common sizes are 32x32, 64x64, 256x256
- But don’t need to be square, i.e., 32x64 is fine
- Smaller uses less memory, and there is a finite amount of texture memory on graphics cards
Texture Mapping

Texture coordinate interpolation
- Perspective foreshortening problem

Interpolation: Screen vs. World Space

Screen space interpolation incorrect
- Problem ignored with shading, but artifacts more visible with texturing

Texture Coordinate Interpolation

Perspective correct interpolation
- $\alpha, \beta, \gamma$:
  - Barycentric coordinates of a point $P$ in a triangle
- $s_0, s_1, s_2$:
  - Texture coordinates of vertices
- $w_0, w_1, w_2$:
  - Homogeneous coordinates of vertices

$$s = \frac{\alpha \cdot s_0 + \beta \cdot s_1 + \gamma \cdot s_2}{\alpha \cdot w_0 + \beta \cdot w_1 + \gamma \cdot w_2}$$

Reconstruction

- How to deal with:
  - Pixels that are much larger than texels?
    - Apply filtering, “averaging”
    - “Magnification”
  - Pixels that are much smaller than texels?
    - Interpolate
    - “Magnification”

Magnification: Interpolating Textures

- Nearest neighbor
- Bilinear
- Hermite (cubic)
**Minification: MIP-mapping**

Use “image pyramid” to precompute averaged versions of the texture.

- Without MIP-mapping
- With MIP-mapping

Store whole pyramid in single block of memory.

**MIPmaps**

*Multum in parvo — many things in a small place*

- Preserves a series of prefiltered texture maps of decreasing resolutions
- Requires more texture storage
- Avoid shimmering and flashing as objects move

Auto-creates a family of textures from original texture size down to 1x1 without
with

**Texture Parameters**

*In addition to color can control other material/object properties*

- Surface normal (bump mapping)
- Reflected color (environment mapping)

**Bump Mapping: Normals As Texture**

Object surface often not smooth — to recreate correctly need complex geometry model

Can control shape “effect” by locally perturbing surface normal

- Random perturbation
- Directional change over region

**Bump Mapping**

Original surface

\[ B(u) \]

A bump map
**Bump Mapping**

- Lengthening or shortening $O(u)$ using $B(u)$
- $N'(u)$
- The vectors to the 'new' surface

**Displacement Mapping**

- **Bump mapping gets silhouettes wrong**
  - Shadows wrong too
- **Change surface geometry instead**
  - Need to subdivide surface
- **GPU support**
  - Bump and displacement mapping not directly supported: require per-pixel lighting
  - However: modern GPUs allow for programming both yourself

**Environment Mapping**

- **Cheap way to achieve reflective effect**
  - Generate image of surrounding
  - Map to object as texture

**Sphere Mapping**

- **Texture is distorted fish-eye view**
  - Point camera at mirrored sphere
  - Spherical texture mapping creates texture coordinates that correctly index into this texture map

**Cube Mapping**

- **6 planar textures, sides of cube**
  - Point camera in 6 different directions, facing out from origin
Cube Mapping

Direction of reflection vector r selects the face of the cube to be indexed
- Co-ordinate with largest magnitude
  - e.g., the vector (-0.2, 0.5, -0.84) selects the –Z face
- Remaining two coordinates (normalized by the 3rd coordinate) selects the pixel from the face.
  - E.g., (-0.2, 0.3) gets mapped to (0.38, 0.80).

Difficulty in interpolating across faces

Procedural Textures

Generate “image” on the fly, instead of loading from disk
- Also called shader
- Often saves space
- Allows arbitrary level of detail
  - “magnification” not an issue
  - “minification” less than for sampled representation
- But can be quite slow for complicated shaders

Volumetric Bump Mapping

Marble

Volumetric Texture Mapping

In Hardware:
- Sampled 3D textures supported very much analogously to 2D textures:
  - gTexCoord3f, gTexImage3f...
- Procedural textures supported with modern GPUs
  - More in upcoming lectures

Volumetric (3D) Texture

Define texture pattern over 3D domain - 3D space containing the object
- Texture function can be sampled
  - 3D table of texels
- Or procedural
  - A function describes the color at each point
  - Implemented in special shading language

Common for natural material/irregular textures (stone, wood, etc...)

Coming Up...

Thursday:
- Sampling
- A2 due...

Tuesday:
- Quiz 2