Texture Mapping

- Increase Object Detail
  - Paint
  - Decal

- Material
  - Wood Grain
  - Marble
  - Non-Plastic

- Geometry
  - Surface Normal

- Surrounding Environment
  - Reflection
  - Transparency (Clouds)

- Increased Computation Time

---

I. Paint

II. Parametric Surface

III. Two Pass Method
Texture Mapping

II. Map to Texture (s,t) → (u,v) → Project to Surface (i,j)

- Project 4 Pixel Corners Onto Parametric Surface (u,v)
- Transform Parametric Surface Values to Texel Space (s,t)
- Compute Weighted Sum of Texels in Bounding Quadrilateral
- Set Image Pixel Value

Texture Mapping

- Shrinkwrapping Image Onto a Parametric Surface

Viewing Pixel (i,j) → Object Coords (u,v) → Texture Coords (s,t) → Pixel Value (r,g,b)

View Plane

Texture Image

Parametric Surface
Texture Mapping

Viewing Geometry → Cartesian Coord (x,y,z) → Parametric Coord (u,v) → Texture Coord (s,t)

\[ x = r \cos \theta \]
\[ y = r \sin \theta \]
\[ z = v \]

\[ u = \theta \quad 0 \leq \theta \leq \frac{\pi}{2} \]
\[ v = z \quad 0 \leq z \leq 1 \]

Texturing Image

1. "Vacuum-Form" Image Onto a Polygonal Surface
2. Assign Texture Coordinates to Polygon Vertices
Texture Mapping

- Polygon Mesh Texture Mapping
- Two-Part Mapping
  - Map Texture onto an intermediate surface
    - Generally non-planar
    - Possesses an simple parametric definition
  - Map intermediate surface onto object
    - Correspondence between object point and
      texture point is a 3-D to 3-D transform
  - Handles “global” mapping of ploygon meshes
  - Texture may be distorted by double mapping

Two Pass Texture Mapping

- Map from 2-D Texture Space to a
  - simple 3-D intermediate surface
    - Cylinder
    - Sphere
    - Box
    - Plane
  \[ T(s, t) \rightarrow T'(x_i, y_i, z_i) \quad \text{“S” mapping} \]
- Determine mapping from 3-D intermediate
  surface to 3-D object surface
  \[ T'(x_i, y_i, z_i) \rightarrow O(x, y, z) \quad \text{“O” mapping} \]
Two Pass Texture Mapping

1. Map the four pixel corners to the surface of the object \((x,y,z)\)

2. Apply the "O" mapping \((\theta, h)\)

3. Apply "S" mapping to find texture point \((s,t)\)

\[
s = \frac{1}{c}(\theta - \theta_0) \\
t = \frac{1}{d}(h - h_0)
\]

\(c, d\) are scaling factors \(\theta_0, h_0\) position the texture on the cylinder
OpenGL Implementation

• Texture Mapping in OpenGL
  • Create a Texture Object
  • Specify Texture for the Object
  • Define How the Texture is to be Applied
    • GL_DECAL, GL_REPLACE
  • Enable Texture Mapping
  • Draw the Scene
    • Supply Texture Coordinates
    • Supply Geometric Coordinates

• Size of Textures Must be a Power of 2
  • Minimum Size : 64x64

Bump Mapping

• Supply Texture Coordinates
Bump Mapping

- Bump Mapping
  - Texture map image shadows won’t change if scene lighting changes
  - Modify surface itself instead of surface color
  - Simulate wrinkled/rough surfaces [Blinn ‘78]

\[ \vec{N} = \vec{P}_u \times \vec{P}_v \]

\[ \vec{P}_u = \frac{\partial \vec{P}}{\partial u} \]
\[ \vec{P}_v = \frac{\partial \vec{P}}{\partial v} \]

Wrinkled Surface

P (u,v)

"Simulate wrinkled/rough surfaces [Blinn ‘78]"

Bump Mapping

- Modify surface itself instead of
- Texture map image shadows won’t change if scene lighting changes

\[ \vec{P} = \vec{P} + F(u,v)\vec{n} \]

\[ \vec{P} = \vec{P} + F(u,v)\vec{n} \]

where \( F \) is the wrinkle function

\[ \vec{N}' = \vec{P}_u' \times \vec{P}_v' \]

Wrinkled Surface

\[ P(u,v) \]

Smooth Surface

\[ P(u,v) \]

Wrinkle Function

\[ u \]

\[ F \]
### Bump Mapping

\[ \vec{N}' = \vec{P}'_u \times \vec{P}'_v \]

\[ \vec{P}'_u = \frac{\partial}{\partial u} (\vec{P} + F\hat{n}) = \vec{P}_u + F\hat{n} + F\hat{n}_u \]

\[ \vec{P}'_v = \frac{\partial}{\partial v} (\vec{P} + F\hat{n}) = \vec{P}_v + F\hat{n} + F\hat{n}_v \]

Let \( F \) be small, so that the last term can be neglected.

\[ \vec{N}' = \vec{N} + \vec{D} = \vec{N} + \{ F_u(\hat{n} \times \vec{P}_u) - F_v(\hat{n} \times \vec{P}_v) \} \]

#### Bump maps

- Modify surface normal
- Produce normals per pixel
  - Store normal maps in textures
  - Generate from height maps
    - Grayscale image stores height
    - Compute
      \[ Nx = (H[s+1,t] - H[s-1,t])/ds \]
      \[ Ny = (H[s,t+1] - H[s,t-1])/dt \]
      \[ Nz = 1 \]
      \( (R, G, B) = \) normalized \( \{Nx, Ny, Nz\} \)
      then packed into bytes
Advanced Texturing

Mip Mapping
Environment Mapping
Procedural Textures

MIP Maps

MIP from Latin: multim in parvo: “Many thing a small place”

• Provides anti-aliasing
• Pre-filtered, offline
• Multiple levels of detail
• mip maps accessed through u,v,d
• gluBuild2DMipmaps
Environment Maps

• Simulates specular reflected scene information
• Provides a virtual 1-ray ray-trace
• Only provides far field reflection correctly
• Can be mapped to a cube for easy access
Environment Maps

Procedural Textures

- Textures Based on functions not images
- Simulate the natural world through simplified physics
- Noise generators and fractals
- 3-D textures
- Not only colors but materials, motion ….
Procedural Textures

Advantages
• Very compact when compared to images.
• No fixed resolution
• No fixed area
• Can be parameterized for general classes

Disadvantages
• Difficult to code and debug
• Not always predictable
• Can be slow
• Aliasing can be a problem

Wood Example

Simulate wood in 3-D
  Growth rings
  Color
  Position
Procedural Textures

Lecture
10

Procedural Textures

Lecture
10

13
Procedural Textures

Lecture 10

Procedural Textures

Lecture 10

Procedural Textures

Lecture 10
Shadows

- Important for spatial relations
  - Where the lights are
- Light
  - Point
- Occluders
- Receivers
  - Occluders cast shadows onto receivers
- Easier if something is planar
  - Point and plane projection

\[ S = P - \alpha L \]

Since \( z_r = 0 \),
\[ 0 = z_p - \alpha z_i \]
\[ \alpha = \frac{z_r}{z_i} \]

\[
\begin{bmatrix}
  x_s \\
  y_s \\
  z_s
\end{bmatrix} = \begin{bmatrix}
  1 & 0 & -x_i/z_i & 0 & x_p \\
  0 & 1 & -y_i/z_i & 0 & y_p \\
  0 & 0 & 0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
  x_p \\
  y_p \\
  z_p \\
  1
\end{bmatrix}
\]
Projected shadows

- Simple algorithm
  - Draw projected occluders on top of receivers
    - In dark color
      - Artifacts are usual
      - May try offsetting

- Proper projected shadows
  - First draw receiving planar polygon
  - Disable Z-buffering
    - Draw projected occluder
      - Only where receiver is drawn
      - Use stencil buffer for this

Projected shadows

- Handle simple situations
- Often faster to create shadow textures and move them
  - Changing texture coordinates
  - Or texture coords transform matrix

- To create softer shadows
  - Accumulate several point lights
    - Distributed over an area
  - Can render into a texture and then filter
    - Recompute texture each time light moves
Shadow textures

- Point light
  - No self-shadowing
  - Occluders
    - Rendered into shadow map
      - From the light’s point of view
  - Receivers
    - Per-pixel lookup of shadow texture
      - Modulate with that
    - Lookup based on generated coordinates
      - Like light map
  - Resolution needs to be high for sharp shadows

Shadows

- Zbuffer Transform
  - Simple two-step approach
    - Render depth map from illumination viewpoint
    - Render from camera viewpoint
      - If point is visible...
        - Transform \((x',y',z')\) screen coords from camera view
        - To \((X,y',z')\) screen coords from light view
        - If \((z' > z_{\text{light}})\), then a surface is between the light source and the point, so shadow the point
Shadow maps

- Williams 1978
  - Like shadow texture
  - But render DEPTH from light
  - Use it when shading pixels

Shadow volumes

- Arbitrary occluders
- Self-shadowing
Shadow volumes

- Several stages
  - Clear stencil buffer
  - Render scene in ambient and depth
  - Z-buffer updates and color off
    - Z-test still on
  - Draw front facing polygons of shadow volume
    - Increment stencil values
  - Draw back facing
    - Decrement stencil values
  - Render full scene diffuse and specular
    - Where stencil is zero

Color Models
Color Models

- Electromagnetic Spectrum

- Visible Wavelengths (ROY G BIV)
  - Red ~ 670 nm ~ 4.3x10^{14} Hz
  - Violet ~ 420 nm ~ 7.5x10^{14} Hz

Color Models

- Color of an Object Determined by Reflected Wavelengths

- White = All Wavelengths/Frequencies

\[ c = \lambda \cdot \nu \]

\[ c = 3 \times 10^{10} \text{ cm/sec} \]
Color Models

- Color Description
  - Hue
  - Dominant wavelength/frequency
- Saturation
  - “Purity” of the color
- Brightness
  - Perceived intensity

- Colors Not in a Rainbow?
  - Color combination from multiple sources
  - Human perception
  - Tristimulus Theory
Color Models

• Color Matching
  • Choose weights for three sources (primaries)
  • Combine sources to produce sample color
  • If no match can be obtained, add a primary to sample
  • Negative weight

\[ C_\lambda = XX + YY + ZZ \]

International Commission on Illumination (CIE)

• Define three primaries (imaginary colors)
• Combine primaries with positive weights
• Normalize against Luminance \((X+Y+Z)\)

\[
x = \frac{X}{X+Y+Z} \quad y = \frac{Y}{X+Y+Z} \quad z = \frac{Z}{X+Y+Z}
\]

\((x + y + z = 1)\)

• Specify a color with chromaticity coordinates \((x,y)\)

\[
X = \frac{x}{y} \\
Y = y \\
Z = \frac{(1-x-y)}{y}Y
\]
Color Models

- Chromaticity Diagram
  - Compare color gamuts for different primaries
  - Identify complementary colors
  - Determine dominant wavelength and purity of a given color

![Color Models Diagram]

Color Models

- Color Gamut ➞ C₁, C₂, C₃
- Complementary Colors ➞ C₄, C₆
- Dominant Wavelength
  - C₅ ➞ C₆
  - C₇ ➞ C₇
- Purity ➞ Distance from C
  - C₄ very pure
  - C₆ not very pure
Color Models

- RGB Color Model
  - Additive Primaries

\[ C_x = R R + G G + B B \]

Chromaticity Coordinates

- \( R \) (0.735, 0.265)
- \( G \) (0.274, 0.717)
- \( B \) (0.167, 0.009)

Color Models
• CMY Color Model
  • Cyan, Magenta, Yellow
  • Complements of RGB
  • Subtractive Primaries
  • Hardcopy Devices
  • Combining Pigments

• YIQ Color Model (there is also YUV)
  • Re-coded RGB for NTSC transmission efficiency
  • Y component is luminance (CIE)
  • Black and white TV displays only the Y component
    • Largest bandwidth in the NTSC video signal (4 MHz)
  • Chromaticity encoded in I and Q

\[
\begin{bmatrix}
  Y \\
  I \\
  Q
\end{bmatrix} = \begin{bmatrix}
  0.299 & 0.587 & 0.144 \\
  0.596 & -0.275 & -0.321 \\
  0.212 & -0.528 & 0.311
\end{bmatrix} \begin{bmatrix}
  R \\
  G \\
  B
\end{bmatrix}
\]
Color Models

- HSV Color Model
  - User intuitive
  - Shades
    - Addition of black pigment to a given hue
  - Tints
    - Addition of white pigment to a given hue
  - Tones
    - Addition of both black and white to a hue

- Vary H (Hue) —> Select Color
- Decrease S (Saturation) —> Add white
- Decrease V (Value) —> Add Black