Class Update

- Assignment 2 is due Thursday night.
- Questions?
- Assignment 0 & 1 graded
- Assignment 3 will be posted shortly.
Today’s Menu

- 2D texturing
- Environment mapping
- Procedural texturing
2D Texturing Mapping

Take a 2D texture and wrap it around an object
A Wall

![Brick Wall](image-url)
A Difficult Wall
A Square

\[ T(u,v) = I(x,y) \]
Texturing A Square

Object Space

Raster Space

Object Space
A Square

\[ R_d = T(u,v) \]
Transformations

$$R_d = T(\text{transform}(u,v))$$
What if the object is not a square?

Need to determine mapping from 3D coordinates to 2D pixel coordinates.

- Projection mapping
- Parametric surfaces
- Assigned UVs
Planar Projection

Henrik Wann Jensen
Cubic Projection

Wojciech Jarosz 2007
Spherical Projection

\[ u = \frac{\text{atan2}(y,x)}{2 \pi} \]

\[ v = \frac{\text{acos}(z / \sqrt{x^2+y^2+z^2})}{\pi} \]
Spherical Projection

\[ u = \frac{\text{atan2}(y, x)}{2\pi} \]
\[ v = \frac{\text{acos}(z / \sqrt{x^2 + y^2 + z^2})}{\pi} \]
Cylindrical Projection

\[ u = \frac{\text{atan2}(y, x)}{2\pi} \]

\[ v = z \]
Parametric Surfaces

• Parametric surfaces have a natural 2D coordinate system.
Bilinear patches
Bicubic patches

Wojciech Jarosz 2007
Spheres
Cylinders
TORUSES

Wojciech Jarosz 2007
UV Texturing

- Assign UV coordinates at vertices.
- Interpolate using bilinear or barycentric interpolation.

\[
(u,v) = \alpha(u_A,v_A) + \beta(u_B,v_B) + \gamma(u_C,v_C)
\]
Texture Coordinates

$p_A, (u_A, v_A)$  $p_B, (u_B, v_B)$  $p_C, (u_C, v_C)$  $p_D, (u_D, v_D)$

Object Space
Texture Coordinates

$p_A, (u_A, v_A)$

$p_B, (u_B, v_B)$

$p_D, (u_D, v_D)$

$p_C, (u_C, v_C)$

Object Space  Texture Space
**Texture Coordinates**

- \( p_A, (u_A, v_A) \)
- \( p_B, (u_B, v_B) \)
- \( p_C, (u_C, v_C) \)
- \( p_D, (u_D, v_D) \)

Object Space

Texture Space
Texture Coordinates

$p_A, (u_A, v_A)$  $p_B, (u_B, v_B)$

$p_D, (u_D, v_D)$  $p_C, (u_C, v_C)$

Object Space  Texture Space  Object Space
UV Texturing

Wojciech Jarosz 2007
Texture Filtering

- Magnification: Texel size larger than pixel size
- Minification: Text size smaller than pixel size (potential aliasing)
Texture Filtering

Magnification

Nearest Neighbor
Texture Filtering

Nearest Neighbor  Bilinear
Texture Filtering

Magnification

Nearest Neighbor  Bilinear  Bicubic
Texture Filtering

Magnification

Nearest Neighbor

Bilinear
ALIASING
Texture Filtering

Point Sample
Texture Filtering

Minification

- Point Sample
- Area Average
Texture Filtering

- Averaging over large texture areas at every texture lookup is costly.
- How can we make this more efficient?
MipMap
Summed Area Table

Image
Summed Area Table

Image

Integral Image

1 5 3
8 2 6
1 9 7

1 6 9
9 16 25
10 26 42

34
## Summed Area Table

![Summed Area Table Diagram]
Summed Area Table
Summed Area Table
Summed Area Table
Summed Area Table
Summed Area Table
Environment Mapping

- Use a texture to represent the surrounding scene.
Environment Mapping

Grace cathedral environment map
Environment Mapping
Environment Mapping
Implementation Hints

- If a ray doesn’t hit any objects, in Scene::raytraceImage, use environment mapping.
- To compute environment map pixel, use only the ray’s direction component.
Cubemap Projection

- Easy to produce with renderer
- Possible to produce with camera
Capturing Environment Maps
Mirror Sphere
Angular Map
Latitude/Longitude
Exploit Texturing!
QUESTIONS?
Procedural Texturing

• Instead of using image data, define texture procedurally.
• Simple example:
  • color = 0.5*sin(x) + 0.5
• Often called “solid texturing” because texture can vary in all 3 dimensions.
Procedurals vs Images

- Pixar almost exclusively uses procedural textures.
- Why?
Procedural Synthesis

created using Terragen
Procedural Synthesis

created using Terragen
Procedural Synthesis

created using MojoWorld
Digital matte painting for Pirates of the Caribbean 2 created using Vue Infinite
Procedural Synthesis

created using Vue Infinite
Procedural Textures

- Often want to add controlled variation to a texture.
- Just calling rand() is not that useful.
Random Noise

- $R_d = \text{rand}() / \text{RAND\_MAX}$;
- Not band-limited, white noise.
Noise Functions

- Function: $R^n \rightarrow [-1,1]$, where $n = 1,2,3,...$
- Desirable properties:
  - Band-limited
  - No obvious repetition
  - Fundamental building block of most procedural textures.
Value Noise

Implementation

- Values associated with integer lattice locations.
- Given arbitrary position, interpolate value from neighboring lattice points.
Value Noise Example
Value Noise Example

Random Values on Grid
Value Noise Example

Random Values on Grid

- Also called cell noise
- Not to be confused with cellular noise
Value Noise Example

Linearly Interpolated Values
Value Noise Example

Cubic Interpolation
Value Noise

Implementation Issues

- Not feasible to store values at all integer locations
- Hash function can be used to map lattice locations to a pre-computed set of pseudo-random values
Value Noise

Implementation Details

// randomly permuted array of 0...255, duplicated
cost unsigned char values[256*2] = [1, 234, ...];

float noise1D(float x)
{
    int xi = int(floor(x)) & 255;
    return lerp(values[xi], values[xi+1], x-xi)/128.0-1;
}

// 2D hashing:
// values[xi + values[yi]];
// 3D hashing:
// values[xi + values[yi + values[zi]]];
// etc.
Perlin Noise

- Generate random vectors at lattice locations.
- Use Hermite interpolation to compute noise value.
- Also called gradient noise.
Classic Perlin Noise
Classic Perlin Noise

Random Gradients on Grid
Classic Perlin Noise

Hermite-interpolated values
2D Perlin Noise

Wojciech Jarosz 2007
3D Perlin Noise
**Perlin Noise**

- Change amplitude: $10 \times \text{noise}(x)$
- Change frequency: $\text{noise}(10 \times x)$
Spectral Synthesis

- Representing a complex function $f_s(s)$ by a sum of weighted contributions from another function $f(x)$:

  $$f_s(x) = \sum_i w_i f(s_i x)$$

- Each term in the summation is called an octave.
FBm

- Fractional Brownian motion: Spectral synthesis of Perlin noise function.
- Progressively smaller frequency
- Progressively smaller amplitude
FBm - 1 Octave
FBm - 2 Octaves
FBm - 4 Octaves

Wojtech Jarosz 2007
Turbulence

- Same as FBm, but sum absolute value of noise function.
Turbulence - 1 Octave
Turbulence - 2 Octaves

Wojtech Jarosz 2007
Turbulence - 3 Octaves
Turbulence - 4 Octaves
Bump Mapping

FBm

Turbulence
Noise Demo
\[ \text{color} = \sin(x + \text{turbulence}(x,y,z)) \]
Wood

\[ \text{color} = \sin(\sqrt{x^2+y^2} + \text{FBm}(x,y,z)) \]
AND MORE...
and More...
In 1996, Steve Worley introduced a cellular texture based function.

- Randomly distribute “feature points” in space.
- \( F_n(x) = \text{distance to } n^{th} \text{ closest point to } x. \)
2D Worley Noise: F1
2D Worley Noise: F2-F1
Noise Demo
3D Worley Noise

F2-F1
Worley Noise

Fractal F1-F4 combinations
Worley Noise

Fractal F1, color and bump map

Steve Worley
Worley Noise

Fractal F1, bump map
Worley Noise

Fractal F1, bump map
Other Resources
Assignment 3

- Shading and Texturing
  - Reflections/refractions
  - HDR environment mapping
  - Procedural textures
  - Global illumination
  - and more!
QUESTIONS?