Texture Mapping: Solid Texturing

Texture Mapping
Visual complexity on demand
Vary display properties over object
Visible pixel maps to location on object
Location on object
used to lookup display attributes
Or
as function parameters to generate attributes

Solid Texture Mapping
Object is ‘carved’ out of textured volume
Use x,y,z location of pixel
Use location in simple procedure to generate, e.g.
• Material color to be used in shading calculation
• Ambient, diffuse, or specular reflection coefficient
• Opacity
• Final color
World space coordinates vs. object space coordinates?

Solid Texture Map Coordinates
If world space
Ok in static scenes
Object moves through texture if object animated
If object space
Texture is ‘fixed’ to object
need to inverse transform intersection
or need to trace inverse ray in object space
Solid Texture Map Coordinates

Object Space $\rightarrow$ World Space $\rightarrow$

Space Filling Stripes

Uses: modulo divisor %

$\text{jump}(x,y,z) = ((\text{int})(x)) \% 2$

if (jump == 0) color = yellow
else if (jump == 1) color = red

$\text{jump}(x,y,z) = ((\text{int})(A + x/s.x) \% 2$

if (jump == 0) color = yellow
else if (jump == 1) color = red

Space Filling 2D Checkerboard

$\text{jump}(x,y,z) = ((\text{int})(A+x/s.x)+(\text{int})(A+y/s.y)) \% 2$

if (jump == 0)
    color = yellow
else if (jump == 1)
    color = red

$\text{jump}(x,y,z) = ((\text{int})(A + x/s.x)+(\text{int})(A+y/s.y)+(\text{int})(A+z/s.z)) \% 2$

if (jump == 0)
    color = yellow
else if (jump == 1)
    color = red

Space Filling 3D Checkerboard
Cube of Smoothly Varying Colors

Uses \( \text{fract}(x) = x - \text{floor}(x) \)

Texture\((x,y,z) = (1 - |2*\text{fract}(x)-1|, 1-|2*\text{fract}(y) - 1|, 1-|2*\text{fract}(z)-1|) \)

Rings

\[
\text{rings}(r) = (\text{int}(r)) \mod 2 \\
\text{rings}(x,y,z) = D + A \ast \text{rings}(r/M)
\]

\( D \) & \( A \) scale and translate into arbitrary values

Or, as before, map 0 & 1 into yellow and red

Wood Grain

Twist:

Rotate texture around y-axis by \( \theta \)

Implement by rotating point by \( -\theta \) around y-axis

Similarly, rotate \((x,y,z)\) point around z-axis
Noise, Turbulence, Marble

- Define function of random values which is
  - A function of 3D point
  - continuous
  - repeatable
- Use 3D point to retrieve random value
- 3D volume has frequency determined by spacing of random values
- Scale point first to change frequency of noise function

1D Noise Example

- Deposit random values at integer locations
- Interpolate through values to get continuous function
- Sample function at intersection points of object with ray

1D Noise Example

- Sample too frequently - no randomness
- Sample too sparsely - no continuity
  (Nyquist limit)
Turbulence

Add multiple frequencies together
As frequency goes up, amplitude goes down
Each component similar under scale
Fractal
e.g. coastline

1D Turbulence Example

1D Turbulence Example

1D Turbulence Example

1D Turbulence Example
1D Turbulence Example

1D Turbulence Example
1D Turbulence Example

Visible point from surface of object

3D Noise

Texture value from 3D table or procedure

Need controlled randomness => varying but continuous function

Integer Lattice

Use 256x256x256 volume

Deposit random values at integer grid points

Interpolate values within cube

Use tri-linear interpolation

\[ d_0 = d_{00} + f_x (d_{10} - d_{00}) \]
\[ d_1 = d_{01} + f_y (d_{11} - d_{01}) \]
\[ d = d_0 + f_z (d_1 - d_0) \]

\[ f_x = \text{FRACT}(x) \]
\[ f_y = \text{FRACT}(y) \]
\[ f_z = \text{FRACT}(z) \]
Implementation notes

NoiseTable[256]: random values [0, 1]

Index[256]: random permutation of values 0:255

```
#define PERM(x)      index[x & 255]
#define INDEX(ix,iy,iz) PERM( ix + PERM(iy + PREM(iz)))
```

```
Float latticeNoise(i,j,k)
    Return NoiseTable[INDEX(i,j,k)]
```

Turbulence implementation

```
Noise(s,x,y,z)
Scale point by s, add 1000 to each coordinate
Get integer (ix,iy,iz) and fractional parts (fx,fy,fz)
Get cell lattice noise values
d000,d001,d010,d011, d100,d101,d110,d111
Do the trilinear interpolation by fx,fy,fz

Turb(s,x,y,z,k) = (1/2) Σ (1/2^k) noise(2^k,x,y,z)
```

Where k is the number of frequencies

Marble Texture

```
Undulate(x) - basic ripple in x

Marble(x,y,z) = undulate(sin(2πxyz + A*turb(s,x,y,z,k)))
```

Parameters: amplitude, scale, number of frequencies

Marble Texture

See examples

www.cse ohio-state.edu/~parent/classes/681/Labs/lab3/noise.html