Procedural shading and texturing
Local shading is complex

- Assume we know diffuse, specular, transmitted, ambient components
- Must apply
  - texture
    - from map
  - procedural
  - volume
  - bump
  - displacement
  - opacity
  - etc
- Shaders
  - device for managing this complexity
Texturing

• Makes materials look more interesting
  – Color - e.g. decals
  – Opacity - e.g. swiss cheese, wire
  – Wear & tear - e.g. dirt, rust
• Provides additional depth cue to human visual system
Texture Mapping

- Maps image onto surface
- Depends on a surface parameterization $(s, t)$
  - Difficult for surfaces with many features
  - May include distortion
  - Not necessarily 1:1
Texture synthesis

- Use image as a source of probability model
- Choose pixel values by matching neighbourhood, then filling in
- Matching process
  - look at pixel differences
  - count only synthesized pixels
From “Image quilting for texture synthesis and transfer”, Efros and Freeman, SIGGRAPH 2001
From “Image quilting for texture synthesis and transfer”, Efros and Freeman, SIGGRAPH 2001
From “Image quilting for texture synthesis and transfer”, Efros and Freeman, SIGGRAPH 2001
From “Image quilting for texture synthesis and transfer”, Efros and Freeman, SIGGRAPH 2001
From “Image analogies”, Herzmann et al, SIGGRAPH 2001
From “Image analogies”, Herzmann et al, SIGGRAPH 2001
Solid Texturing

- Uses 3-D texture coordinates \((s, t, r)\)
- Can let \(s = x\), \(t = y\) and \(r = z\)
- No need to parameterize surface
- No worries about distortion
- Objects appear sculpted out of solid substance
Solid Texture Problems

- How can we deform an object without making it swim through texture?
- How can we efficiently store a procedural texture?
Procedural Texturing

- Texture map is a function
- Write a procedure to perform the function
  - input: texture coordinates - s, t, r
  - output: color, opacity, shading
- Example: Wood
  - Classification of texture space into cylindrical shells
    \[ f(s, t, r) = s^2 + t^2 \]
  - Outer rings closer together, which simulates the growth rate of real trees
  - Wood colored color table
    - Woodmap(0) = brown “earlywood”
    - Woodmap(1) = tan “latewood”

\[ \text{Wood}(s, t, r) = \text{Woodmap}(f(s, t, r) \mod 1) \]
Noise Functions

• Add “noise” to make textures interesting
• Perlin noise function $N(x,y,z)$
  – Smooth
  – Correlated
  – Bandlimited
• $N(x,y,z)$ returns a single random number in $[-1,1]$
• Gradient noise
  – Like a random sine wave
    $N(x,y,z)=0$ for int $x,y,z$
• Value noise
  – Also like a random sine wave
    $N(x,y,z)=\text{random}$ for int $x,y,z$
Using Noise

• Add noise to cylinders to warp wood
  – \( \text{Wood}(s^2 + t^2 + N(s,t,r)) \)

• Controls
  – Amplitude: power of noise effect
    \[ a N(s, t, r) \]
  – Frequency: coarse v. fine detail
    \[ N(f_s s, f_t t, f_r r) \]
  – Phase: location of noise peaks
    \[ N(s + \phi_s, t + \phi_t, r + \phi_r) \]
Making Noise

• Good:
  – Create 3-D array of random values
  – Trilinearly interpolate

• Better
  – Create 3-D array of random 3-vectors
  – Hermite interpolate
Hermite Interpolation

- Some cubic \( h(t) = at^3 + bt^2 + ct + d \) s.t.
  - \( h(0) = 0 \) \( (d = 0) \)
  - \( h(1) = 0 \) \( (a + b + c = 0) \)
  - \( h'(0) = r_0 \) \( (c = r_0) \)
  - \( h'(1) = r_1 \) \( (3a + 2b + r_0 = r_1) \)

- Answer:
  - \( h(t) = (r_0 + r_1) \ t^3 - (2r_0 + r_1) \ t^2 + r_0 t \)

- Tricubic interpolation
  - Interpolate corners along edges
  - Interpolate edges into faces
  - Interpolate faces into interior
Colormap Donuts

- Spotted donut
  - Gray(N(40*x,40*y,40*z))
  - Gray() - ramp colormap
  - Single 40Hz frequency

- Bozo donut
  - Bozo(N(4*x,4*y,4*z))
  - Bozo() - banded colormap
  - Cubic interpolation means contours are smooth
Bump Mapped Donuts

\[ \text{DNoise}(s,t,r) = \nabla \text{Noise}(s,t,r) \]

- Bumpy donut
  - Same procedural texture as spotted donut
  - Noise replaced with DNoise

\[ \mathbf{n} \mathrel{+}= \text{DNoise}(x,y,z); \ \text{normalize}(\mathbf{n}); \]
Composite Donuts

• Stucco donut
  – Noise(x,y,z)*DNoise(x,y,z)
  – Noisy direction
  – Noisy amplitude

• Fleshy donut
  – Same texture
  – Different colormap
DNoise Bump-Mapped Refraction
Fractals

- Fractional dimension - not
- Fractal dimension exceeds topological dimension
- Self-similar
- Detail at all levels of magnification
- $1/f$ frequency distribution
How Can Dimension be Fractional?

- Point: $D = 0$, $N=1$, $s=1/2$
- Line: $D = 1$, $N=2$, $s=1/2$
- Square: $D = 2$, $N=4$, $s=1/2$
- Cube: $D = 3$, $N=8$, $s=1/2$

$N = (1/s)^D$

$\log N = D \log (1/s)$

$D = \log(N)/\log(1/s)$
Examples

N=2
s=1/3
D = log 2/log 3
D = .6...

N=4
s=1/3
D = log 4/log 3
D = 1.3...
Brownian Motion

- random paths
- Integral of white noise
- $1/f^2$ distribution

![Diagram showing white and brown noise with log power graphs]
Fractional Brownian Motion

- $1/f^\beta$ distribution
- Roughness parameter $\beta$
  - Ranges from 1 to 3
  - $\beta = 3$ - smooth, not flat, still random
  - $\beta = 1$ - rough, not space filling, but thick
- Construct using spectral synthesis
  - $f(s) = \sum_{i=1}^{4} 2^{-i\beta} n(2^i s)$
  - Add several octaves of noise function
  - Scale amplitude appropriately
Fractal Bump-Mapped Donut

```cpp
fbm(beta) {
  val = 0; vec = (0,0,0);
  for (i = 0; i < octaves; i++) {
    val += Noise(2^i*x, 2^i*y, 2^i*z)/pow(2,i*beta);
    vec += DNoise(2^i*x, 2^i*y, 2^i*z)/pow(2,i*beta);
  }
  return vec or val;
}
```
Fractal Mountains

- Displacement map of meshed plane
- Can also be formed using midpoint displacement

Ken Musgrave

Gunther Berkus via Mojoworld
Clouds
Water

\[ f(s) = \sum_{i=1}^{4} 2^{-i} n(2^i s) \]
Marble

\[ f(s, t, r) = r + \sum_{i=1}^{4} 2^{-i} n(2^i s, 2^i t, 2^i r) \]
Fire

\[ f(s, t, r) = r + \sum_{i=1}^{4} 2^{-i} n(2^i s, 0, 2^i r + \phi)). \]
Planets

\[ f(s) = \sum_{i=1}^{4} 2^{-i} n(2^i s) \]

Ken Musgrave
Moonrise

Ken Musgrave