Noise and Procedural Techniques

John Spitzer
Simon Green
NVIDIA Corporation
Overview

- What is procedural texturing?
- Advantages and disadvantages
- When to use procedural texturing
- What is noise?
- Ideal noise characteristics
- Spectral synthesis
- Demos...
What is Procedural Texturing?

- Code vs. tables
- Classic time vs. storage space trade-off
Advantages of Procedural Texturing

• Compact
  - code is small (compared to textures)

• No fixed resolution
  - "infinite" detail, limited only by precision

• Unlimited extent
  - can cover arbitrarily large areas, no repeating

• Parameterized
  - can easily generate a large no. of variations on a theme

• Solid texturing (avoids 2D mapping problem)
Disadvantages of Procedural Texturing

- Computation time (big ouch!)
- Hard to code and debug
- Aliasing
When to use Procedural Textures

- Don’t use them just for the hell of it!
- Procedurals are good for animating effects – fire, water, clouds, explosions...
- ..or anywhere where a repeating texture would be obvious
- Combine the best aspects of both techniques – e.g. painted maps + noise to add variation
Procedural Noise

• Noise is an important part of many procedural textures
• Used *everywhere* in production rendering
• Procedural noise provides a controlled method of adding randomness to:
  – Color, texture
  – Bump map / displacement maps
  – Animation
  – Terrains, anything else...
Ideal Noise Characteristics

- Can’t just use rand()!
- An ideal noise function has:
  - *repeatable* pseudorandom values
  - specific range (typically \([-1,1]\) or \([0,1]\))
  - band-limited frequency \(\sim 1\)
  - no obvious repeating patterns
  - invariance under rotation and translation
- “Random yet smooth”
What does Noise look like?

• Imagine creating a big block of random numbers and blurring them:
What does Noise look like?

- Random values at integer positions
- Varies smoothly in-between. In 1D:
Spectral Synthesis

- Narrow-band noise by itself is not very exciting
- Summations of multiple frequencies are!
- Like Fourier synthesis (summing sine waves)
- Each layer is known as an “octave” since the frequency typically doubles each time
- Increase in frequency known as “lacunarity” (gap)
- Change in amplitude/weight known as “gain”
Fractal Sum

Fractal Sum

Fractal Sum

Fractal Sum

Fractal Sum

Fractal Sum
Turbulence

- Ken Perlin’s trick – assumes noise is signed \([-1,1]\)
- Exactly like fBm, but take absolute value of noise
- Introduces discontinuities that make the image more “billowy”

```c
float turbulence(float3 p, int octaves, float lacunarity, float gain)
{
    float sum = 0;
    float amp = 1;
    for(int i=0; i<octaves; i++) {
        sum += amp * abs(noise(p));
        p *= lacunarity;
        amp *= gain;
    }
    return sum;
}
```
Turbulence
Pixel Shader Noise

- Implementation of Perlin’s original (Academy-award winning) algorithm

- Gradient noise over $R^3$, scalar output

- Uses 2 1D textures as look-up tables

- Compiles to around 40 instructions
Pixel Shader Noise using 3D Textures

• Pre-compute 3D texture containing random values

• Pre-filtering with tri-cubic filter helps avoid linear interpolation artifacts

• 4 lookups into a single 64x64x64 3D texture produces reasonable looking turbulence
Applying Colour Tables to Noise
Using Noise to Perturb Patterns
Questions, comments, feedback?

- John Spitzer, jspitzer@nvidia.com
- Simon Green, sgreen@nvidia.com
- www.nvidia.com/developer