Solid and procedural textures
Course announcements

• Programming assignment 1 is due on Friday 2/9.
  - Any issues with the homework?

• Take-home quiz 1 due tonight.

• Office hours on website and Slack.
Alan Lee (soohyun3@andrew.cmu.edu)

BS in Computer Science, (hopefully) starting MS in CS this Fall

Research Interests: Virtual reality, non-photorealistic rendering

My PBR Final Project: Photon mapping!

You won’t understand what’s going on here but hey it looks cool
Overview of today’s lecture

• 3D textures.
• Procedural textures.
• Generating “realistic noise”.

Slide credits

Most of these slides were directly adapted from:

• Wojciech Jarosz (Dartmouth).
3D textures

Texture is a function of (u, v, w)
- can evaluate texture at 3D point
- good for solid materials
- often defined procedurally
Procedural texturing

Instead of using rasterized image data, define texture procedurally.

Simple example:
- color = 0.5*\( \sin(x) \) + 0.5

Often called “solid texturing” because texture can easily vary in all 3 dimensions.
- but you can also do 2D or 1D procedural textures
Raster vs. procedural textures

Why use procedural textures?
- low memory usage
- infinite resolution
- solid texture: no need to parametrize surface
3D stripe texture
3D stripe texture

color stripe(point $p$):
    if $(\sin(p_x) > 0)$
        return $c_0$
    else
        return $c_1$
3D stripe texture
3D stripe texture

color stripe(point \( p \), real \( w \)):
    if \( \left( \sin\left(\frac{\pi p_x}{w}\right) > 0 \right) \)
        return \( c_0 \)
    else
        return \( c_1 \)
3D stripe texture

color stripe(point p, real w):
   t = (1 + sin(\pi p_x/w))/2
return lerp(c_0, c_1, t)
2D checkerboard texture

color checkerboard(point \( p \)):
   real a = floor(\( p_x \))
   real b = floor(\( p_y \))
   real val = a+b
   if (isEven(val))
      return \( c_0 \)
   else
      return \( c_1 \)
3D checkerboard texture

color checkerboard(point p):
    real a = floor(p_x)
    real b = floor(p_y)
    real c = floor(p_z)
    real val = a+b+c
    if (isEven(val))
        return c_0
    else
        return c_1
Procedural synthesis
Procedural synthesis
Procedural synthesis

Digital matte painting for Pirates of the Caribbean 2; created using Vue Infinite
Procedural synthesis
Procedural textures

Our procedurals are “too perfect”

Often want to add controlled variation to a texture
- Real textures have many imperfections

Just calling rand() is not that useful.
Random noise

\[ \textit{albedo} = \text{randf}() ; \]

Not band-limited, white noise.
Noise functions

Function: $\mathbb{R}^n \rightarrow [-1, 1]$, where $n = 1, 2, 3, ...$

Desirable properties:
- no obvious repetition
- rotation invariant
- band-limited (i.e., not scale-invariant)

Fundamental building block of most procedural textures
Value noise

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

Cell noise: use value of nearest point on grid
Value noise example

(Bi-) linearly interpolated values

Interpolate between $2^n$ nearest grid points
Value noise example

(Bi-) cubic interpolation

Interpolate between $4^n$ nearest grid points
Value noise - implementation issues

Not feasible to store values at all integer locations
- pre-compute an array of pseudo-random values
- use a randomized hash function to map lattice locations to pseudo-random values
Value noise - implementation details

// randomly permuted array of 0...255, duplicated
const 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.
Value noise - limitations
Value noise - limitations

Lattice structure apparent
- Minimal/maxima always on lattice

Slow/many lookups
- 8 values for trilinear
- 64 values for tricubic
  • $4^n$ for $n$ dimensions
Perlin noise

Perlin noise, invented by Ken Perlin in 1982
- First used in the movie Tron!

Store random vectors/gradients on lattice
- Use Hermite interp.
- a.k.a. “gradient noise”
Classic Perlin noise
Classic Perlin noise

Random gradients on grid
Classic Perlin noise

Hermite-interpolated values
Perlin noise vs. value noise

Perlin Noise
(gradient noise)

Cubic Value Noise

Why is Perlin noise better?
Perlin noise

Typically signed by default, \( \sim \) in \([-1, 1]\) with a mean of 0

offset/scale to put into \([0, 1]\) range

\[
\frac{\text{noise}(p) + 1}{2}
\]

take absolute value

\[
|\text{noise}(p)|
\]
3D Perlin noise
Absolute value of noise
Perlin noise

Change frequency: ?

Change amplitude: ?
Perlin noise

Change frequency: $\text{noise}(10\times x)$

Change amplitude: $10\times \text{noise}(x)$
Absolute value of noise
Absolute value of noise
Absolute value of noise
Absolute value of noise

\[ |\text{noise}(p)| \quad |\text{noise}(4p_x, p_y, p_z)| \quad |\text{noise}(p_x, 4p_y, p_z)| \]
Perlin noise - limitations
Perlin noise - limitations

Lattice structure apparent for |noise |
- all lattice locations have value 0

Lookups faster, but still slow:
- Perlin is $2^n$ for $n$ dimensions instead of $4^n$ for value noise
- other variations: simplex noise ($O(n)$)

Not quite rotation invariant
More reading

Fantastic explorable explanation by Andrew Kensler at Pixar
- eastfarthing.com/blog/2015-04-21-noise
Spectral synthesis

Representing a complex function $f_s(p)$ by a sum of weighted contributions from a scaled function $f(p)$:

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

Called a “fractal sum” if $w_i$ and $s_i$ are set so:

- increasing frequencies have decreasing amplitude, e.g.: $w_i = 2^{-i}$, $s_i = 2^i$
- when $s_i = 2^i$, each term in summation is called an “octave”

What function $f(p)$ should we use?
fBm - fractional Brownian motion

In graphics:
- Fractal sum of Perlin noise functions
- “Fractal noise”
fBm - 1 octave

Wojciech Jarosz 2007
fBm - 2 octaves
fBm - 3 octaves
fBm - 4 octaves

Wojciech Jarosz 2007
Turbulence

Same as fBm, but sum absolute value of noise function
Turbulence - 1 octave

Wojciech Jarosz 2007
Turbulence - 2 octaves
Turbulence - 4 octaves

Wojciech Jarosz 2007
fBm vs Turbulence
Bump mapping

fBm

Turbulence
2D fBm

A fractional Brownian motion (fBm) terrain patch of fractal dimension ~2.1.

source: Ken Musgrave
Fractal dimension

Fractals have fractional dimension, e.g. \( D = 1.2 \).
- under some appropriate definition of dimension...

Integer component indicates the underlying Euclidean dimension of the fractal, in this case a line ("1" in 1.2).

Fractional part is called the fractal increment (".2" in 1.2).

Fractal increment varies from .0 to .999...
- fractal goes from (locally) occupying only its underlying Euclidean dimension (the line), to filling some part of the next higher dimension (the plane).

Continuous "slider" for the visual complexity of a fractal
- “smoother” \( \leftrightarrow \) “rougner”

What determines the dimension of fBm?
Fractal dimension of fBm

Traces of fBm for H varying from 1.0 to 0.0 in increments of 0.2

source: Ken Musgrave
fBm

fBm is statistically homogeneous and isotropic.
- Homogeneous means "the same everywhere"
- Isotropic means "the same in all directions"

Fractal phenomena in nature are rarely so simple and well-behaved.
Multifractals

Fractal system which has a different fractal dimension in different regions

Heterogeneous fBm
- Scale higher frequencies in the summation by the value of the previous frequency.
- Many possibilities: hetero terrain, hybrid multifractal, ridged multifractal
2D fBm

A fractional Brownian motion (fBm) terrain patch of fractal dimension ~2.1.

source: Ken Musgrave
Heterogeneous fBm

A hybrid multifractal terrain patch made with a Perlin noise basis: the “alpine hills” Bryce 4 terrain model.

source: Ken Musgrave
Heterogeneous fBm

The “ridges” terrain model from Bryce 4: a hybrid multifractal made from one minus the absolute value of Perlin noise.

source: Ken Musgrave
Heterogeneous fBm

A hybrid multifractal made from Worley’s Voronoi distance-squared basis

source: Ken Musgrave
Heterogeneous fBm

A hybrid multifractal made from Worley’s Voronoi distance basis

source: Ken Musgrave
Domain Distortion

fBm distorted with fBm

source: Ken Musgrave
A sample of the “warped ridges” terrain model in Bryce 4: the “ridges” model distorted with fBm.

source: Ken Musgrave
A sample of the “warped slickrock” terrain model in Bryce 4: fBm constructed from one minus the absolute value of Perlin noise, distorted with fBm.

source: Ken Musgrave
Recall: 3D stripe texture

color stripe(point $p$, real $w$):
  $t = (1 + \sin(\pi \frac{p_x}{w}))/2$
return lerp($c_0$, $c_1$, $t$)

How can we make this less structured (less “boring”)?
Marble 

\[
\frac{1 + \sin(k_1 p_x + \text{turbulence}(k_2 p))}{w}/2
\]

from http://lodev.org/cgtutor/randomnoise.html
Marble

\[
\frac{1 + \sin(k_1 p_x + \text{turbulence}(k_2 p))}{w}/2
\]
Wood

$(1 + \sin(\sqrt{p_x^2 + p_y^2}) + fBm(p)))/2$

from http://lodev.org/cgtutor/randomnoise.html
Wood

\[
(1 + \sin(\sqrt{p_x^2 + p_y^2} + fBm(p)))/2
\]
and more...
and more...
Worley noise

“Cellular texture” function
- Introduced in 1996 by Steve Worley
- Different from cell texture!

Randomly distribute “feature points” in space
- $f_n(x) =$ distance to $n^{\text{th}}$ closest point to $x$
2D Worley noise: f1
2D Worley noise: f1
2D Worley noise: f1
2D Worley noise: $f_1$

What do we call this image in geometry?
2D Worley noise: $f_1$
Worley Noise

fractal F1, bump map
Worley Noise

fractal F1, bump map
2D Worley noise: $f_1$
2D Worley noise: $1-f_1$
Worley Noise

fractal $1-f_1$, color and bump map
2D Worley noise: $f_1$
2D Worley noise: $f_1$, thresholded

Wojdech Jarosz 2007
2D Worley noise: $f_1$, thresholded
2D Worley noise: $f_2 - f_1$
2D Worley noise: $f_2 - f_1$, thresholded
3D Worley noise

\[ f_2 - f_1 \]
Worley Noise

fractal f1-f4 combinations
Other Resources
Demos

Amazing realtime demos using fractal noise:

- [http://www.iquilezles.org/www/articles/morenoise/morenoise.html](http://www.iquilezles.org/www/articles/morenoise/morenoise.html)
- [https://www.shadertoy.com/view/4ttSWf](https://www.shadertoy.com/view/4ttSWf)
- [https://www.shadertoy.com/view/XttSz2](https://www.shadertoy.com/view/XttSz2)