Procedural Texturing

Steve Rotenberg

CSE168: Rendering Algorithms

UCSD, Spring 2014

Procedural Texturing

- Procedural texturing is the process of generating texture data algorithmically at render time, rather than getting texture data from a previously stored image
- This allows texture maps to be defined as procedural functions
- This is nice for things like organic, bumpy, irregular surfaces that can be defined by non-repeating functions
- This reduces visual tiling artifacts that can be visible when texture maps are repeated over and over
- Procedural texturing also allows for effectively unlimited resolution, as details are described by mathematical functions rather than pixels of a fixed size

Solid Textures

- It is also possible to use volumetric or *solid textures*
- Instead of a 2D image representing a colored surface, a solid texture represents the coloring of a 3D space
- Solid textures are often procedural, because they would require a lot of memory to explicitly store a 3D bitmap
- A common example of solid texture is marble or granite. These can be described by various mathematical functions that take a position in space and return a color
- Texturing a model with a marble solid texture can achieve a similar effect to having carved the model out of a solid piece of marble

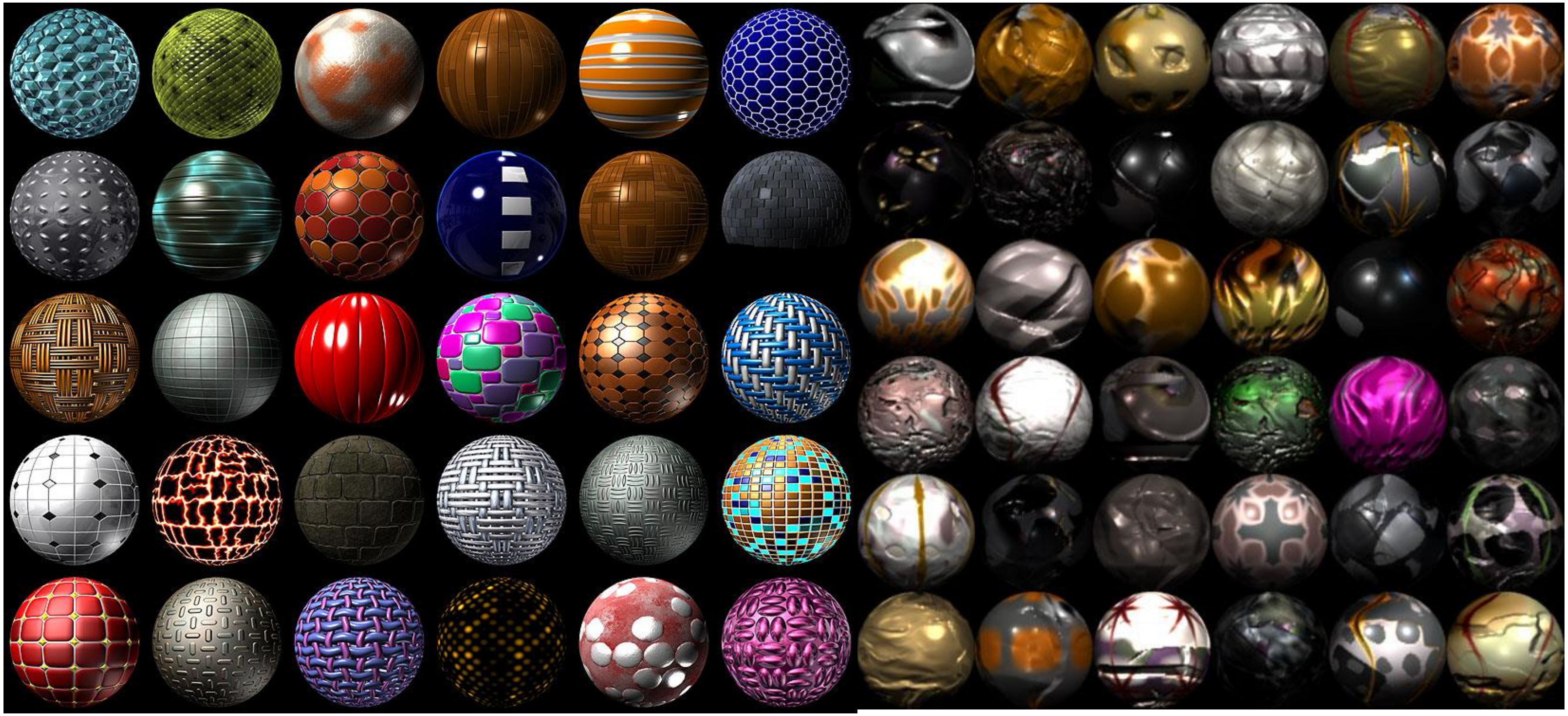
Procedural Texturing

- Procedural texturing is used throughout the movie industry and a lot of production time goes into writing procedural shaders for specific effects
- Procedural functions are used to generate colors, displacements, volumes, or can affect any other visual property or BRDF attribute
- There is no limit to the functions that could be used to generate the desired effects, but there are some standard functions and common approaches used across a variety of different tools and renderers

Basic Operations

- There are several basic features of a procedural texture function
- In general, they take a 2D or 3D texture coordinate as input and produce a vector of values as output (typically a color, displacement, etc.)
- Some common tools include:
  - Grids, bricks, etc.
  - Color ramps
  - Lerp, spline, & smooth step functions
  - BRDF selection & blending

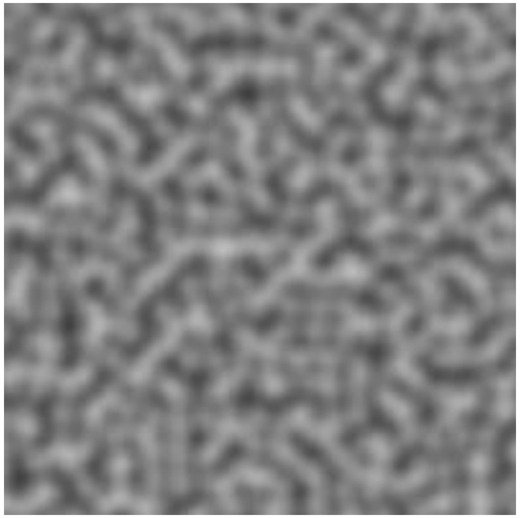
Procedural Shaders



Noise

Noise

- Many procedural textures are built up from *noise* functions
- A *noise* function is an n-dimensional function that generates a random pattern with some adjustable properties
- There are many popular noise functions that generate different patterns, but generally, noise functions are designed to generate variation at a specific spatial frequency



Noise Functions

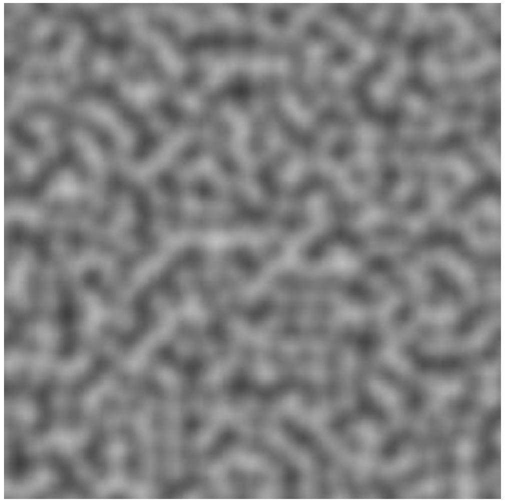
- There are many popular methods for implementing noise functions
- Many of them are based on starting with some lattice of random numbers
- *Value noise* uses a lattice of random values that represent the value of the noise field at the lattice points, and then uses cubic interpolation to determine the value in between the lattice points
- *Gradient noise* uses values of zero at the lattice points combined with random gradient vectors, and is sometimes combined with value noise
- *Fourier synthesis* adds up several sine waves of varying amplitudes, frequencies, and phase shifts
- *Sparse convolution* is not a lattice method, and instead uses a Poisson distribution of filter functions. It is a more expensive technique that avoids the XY gridding artifacts of many other noise techniques

Perlin Noise

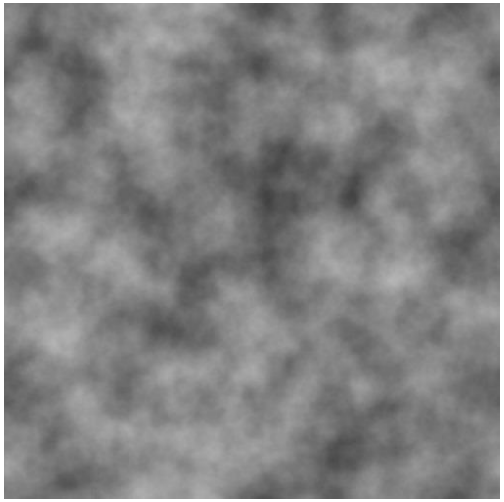
- Ken Perlin introduced the concept of procedural noise to the computer graphics community in 1985, and showed a wide variety of applications
- Modern variations of the classic Perlin noise are still popular today and are included in most 3D rendering tools

Turbulence

- *Turbulence* is a general concept that refers to combining multiple noise patterns at different frequencies
- This gives variation over a range of scales
- Typically, turbulence is a sum of N noise functions, where each function is roughly half the amplitude and twice the frequency of the previous function

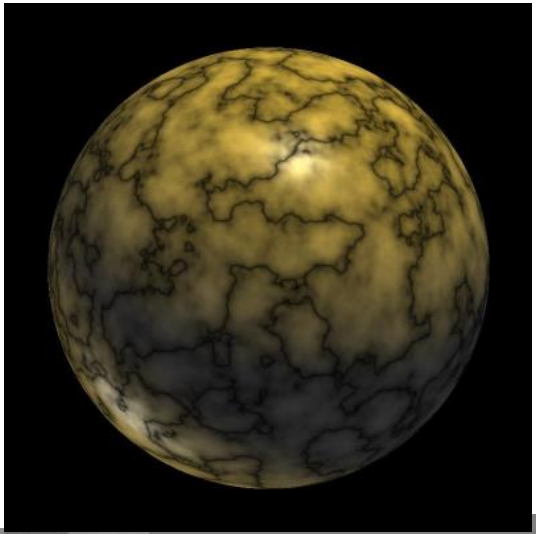


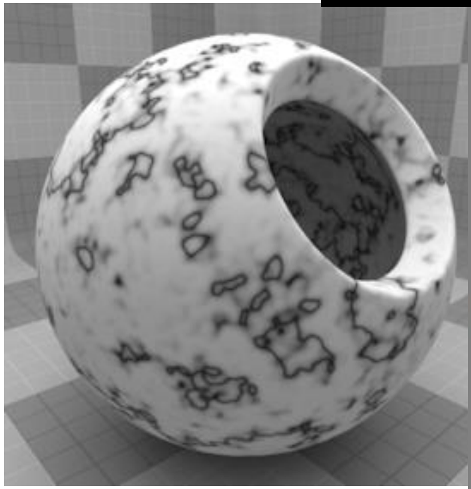
Noise



Turbulence

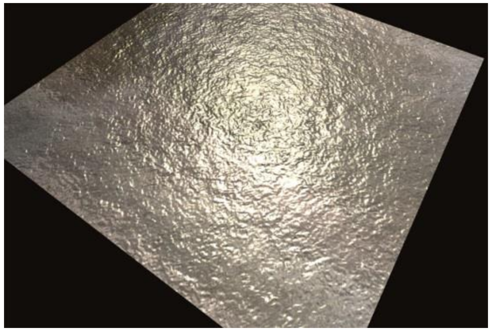
Using Noise





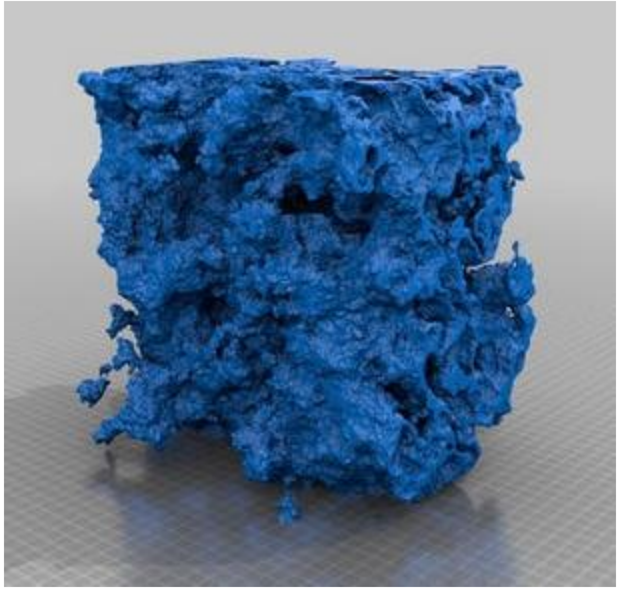
Displacement Noise

- Noise (and other procedural techniques) are also very useful for displacement maps and terrain generation as well



3D Noise

- 3D noise functions can be used to model solid or gaseous volumes



Noise Animation

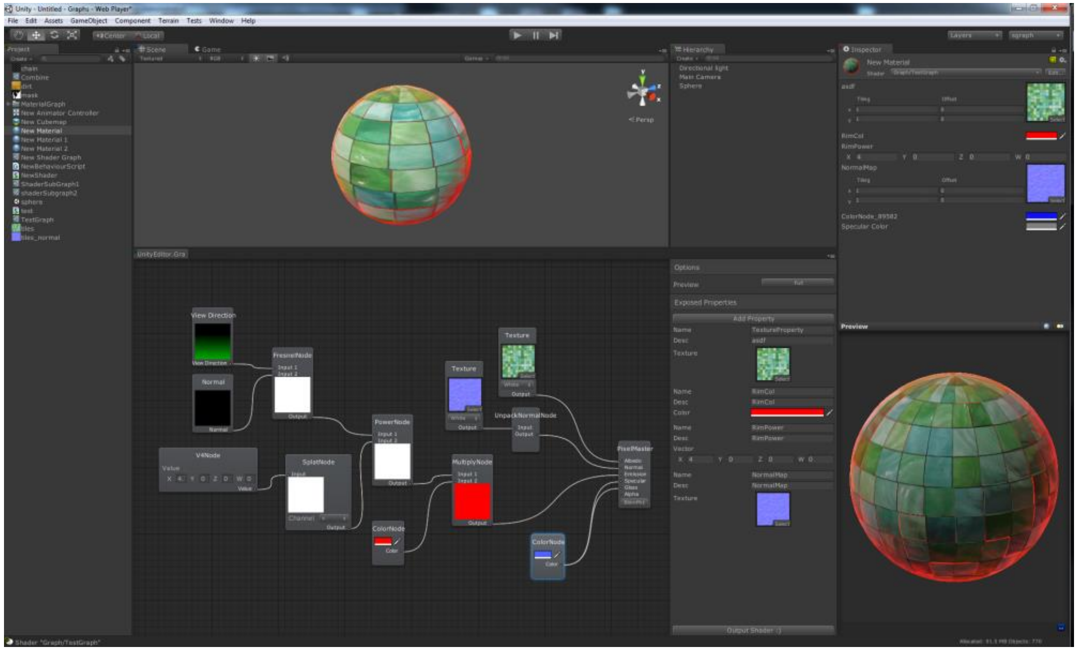
- If we use a 2D plane moving through a 3D noise function, we can generate animated effects such as fire
- It is also possible to use a 3D sub-volume of a 4D noise field to animate 3D fire

Worley Noise

- Another popular choice is the *Worley noise* function
- This one generates a cellular pattern that can be applied to a variety of applications

Shader Editors

- Many rendering systems (Maya, 3D Studio, etc.) have interactive editors that allow artists to create procedural shaders by connecting up components from a set of data flow tools


Unity - Untitled - Graphs - Web Player
File Edit Assets GameObject Component Terrain Tests Window Help
Project
Scene
Game
Hierarchy
Directional light
Main Camera
Sphere
Inspector
New Material
UnityEditor.Gra
View Direction
Normal
FresnelNode
Input 1
Input 2
Output
V4Node
Value
X 4 Y 0 Z 0 W 0
SplatNode
Input
Channel
PowerNode
ColorNode
Color
Texture
UnpackNormalNode
Input
Output
MultiplyNode
PixelMaster
Albedo
Normal
Emission
Specular
Gloss
Alpha
Options
Preview
Exposed Properties
Add Property
Name
Desc
Texture
Color
Vector
Output Shader :)
Preview