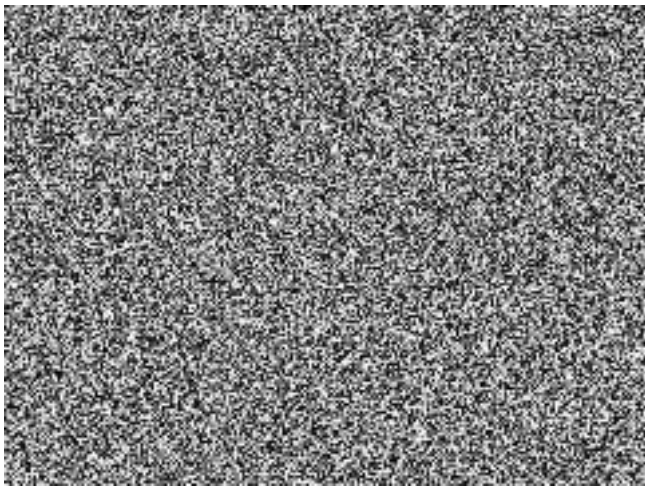
CMSC 498W Lecture 5
"Generating Virtual Worlds I"
2016-02-11
Derek Juba, Amitabh Varshney

Procedural content is content which is not specified explicitly by an artist, but which is, instead, generated by a procedure or program designed by the artist.

As an example, consider noise functions. The familiar White noise is a signal in which all frequencies are present at equal amplitudes, analogous to white light. Such a signal can be easily modeled by simply generating a series of IID (Independent and Identically Distributed) random values.

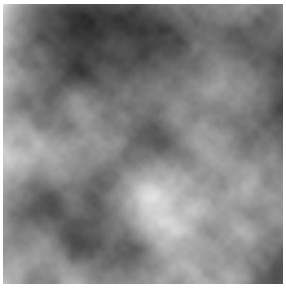


White noise [1]

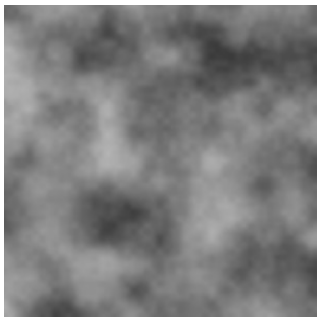
However, many natural phenomena can be modeled by signals in which higher frequencies have smaller amplitudes than lower frequencies. One such noise function was developed by Ken Perlin around 1983, for which he won an Academy Award in Technical Achievement in 1997.

Perlin noise [2]

Perlin's original noise function was defined in terms of generating random vectors at lattice points and then interpolating between them, but a simple approximation can be created by instead generating White noise values at the lattice points and then interpolating. The fewer interpolated points there are, the higher the frequencies in the noise will be. You can then generate fractal noise by taking a weighted sum of a series of different frequency noises (Perlin or other), generally with larger weights being used for the lower frequencies.



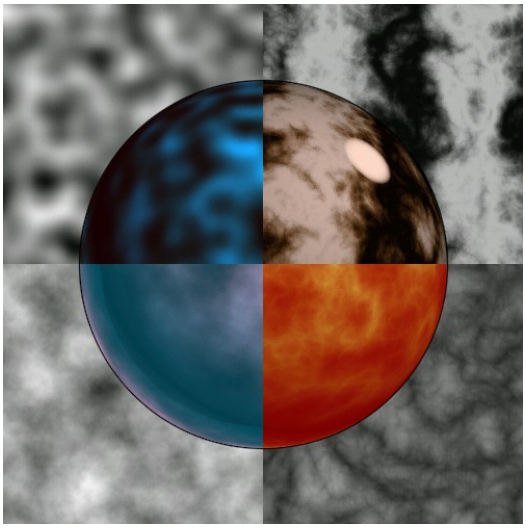
Fractal Value noise [3]



Fractal Perlin noise [2]

Noise functions, such as Perlin's, can be used to generate a variety of content. Perhaps the most obvious application is 2D textures- these can be created by using the noise values as indexes into a look-up table of colors. However, 2D noise functions can also be used to generate terrain by interpreting the noise values as heights. This could be further improved by, for example, making high heights snow-covered and low heights submerged under water.

Noise

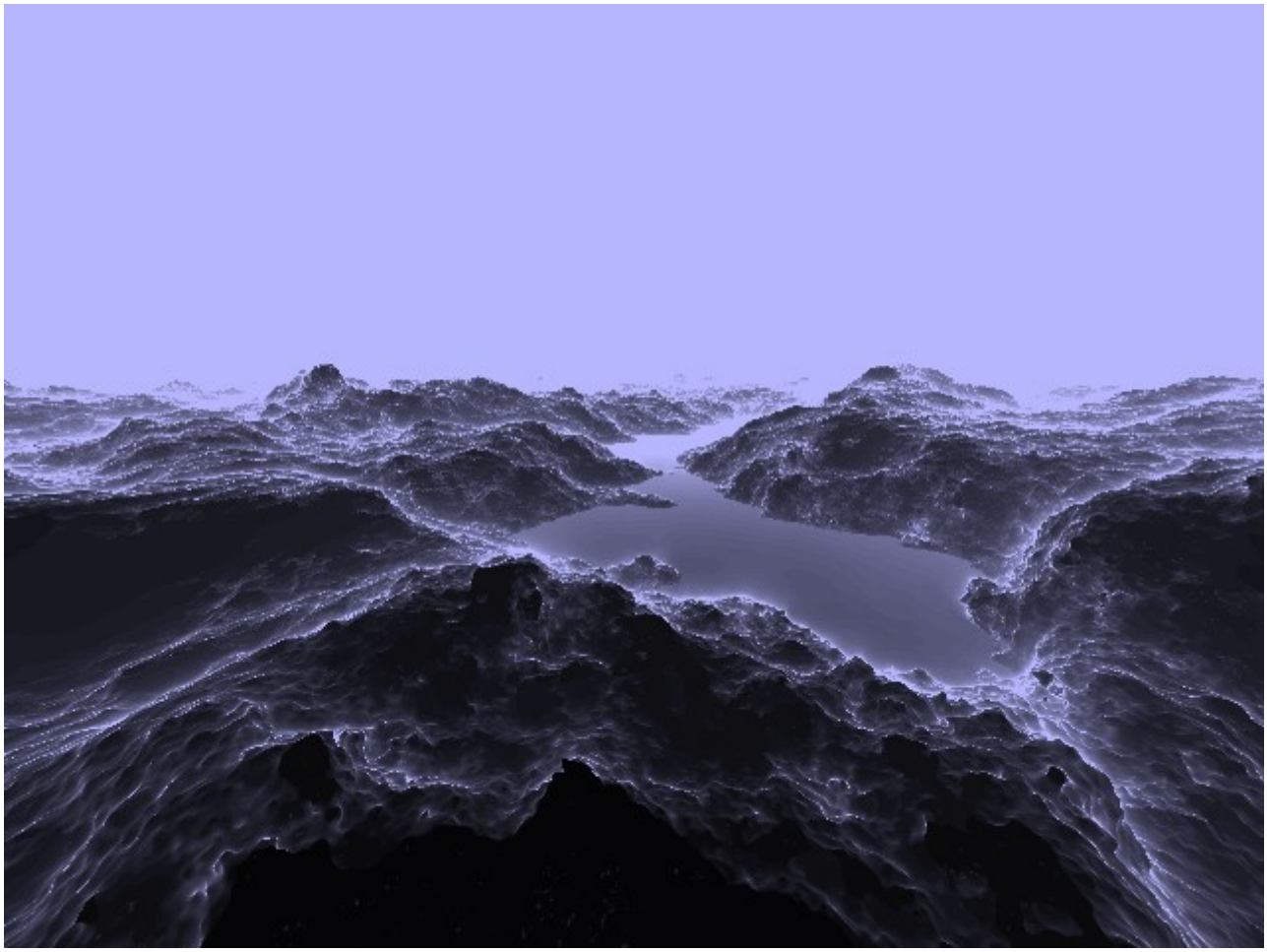


sin(x + Fractal |Noise|)

Fractal Noise

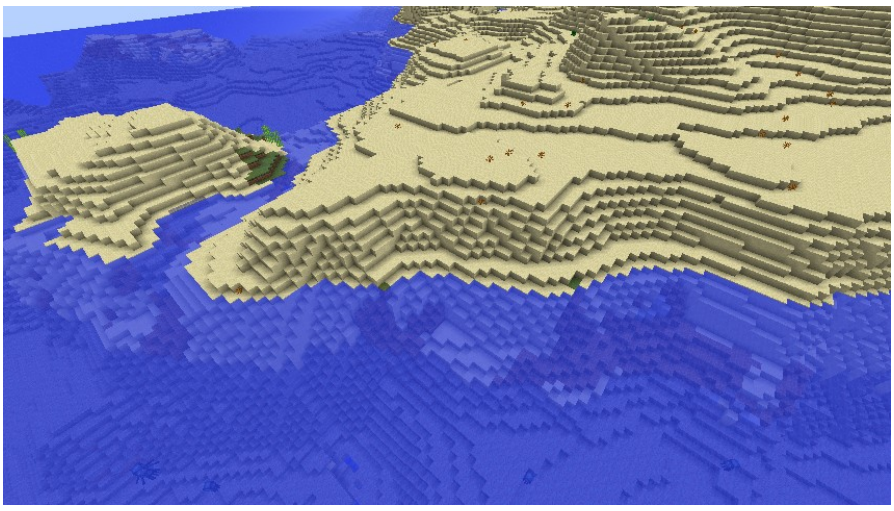
Fractal |Noise|

Variations on Perlin noise [4]



Terrain generated using Perlin noise [5]

One recent example of the use of Perlin noise to generate terrain is in the game Minecraft. At one point in the game's development its terrain was generated using 3D Perlin noise to allow for tunnels and overhangs [8], but the engine was later changed to instead use multiple levels of 2D noise [9]. Terrain types are assigned to blocks generated by the noise functions using a Biome system [10], in which regions of the world are assigned Biomes such as Beach or Forest, which then influence the interpretation of the noise values, as well as other factors such as weather, creature types, etc..



Minecraft Beach Biome [10]

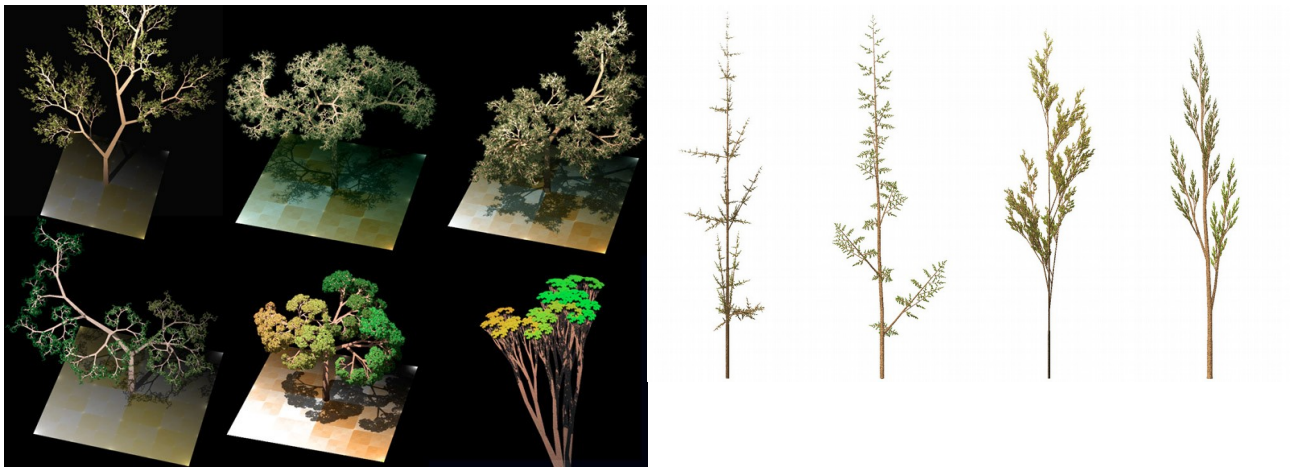
Minecraft Forest Biome [10]

3D noise functions can also be used as solid textures. Regular 2D textures must be mapped onto the surface of a 3D object by assigning a 2D texture coordinate to every point on the object's surface. Solid textures avoid this problem by using the 3D coordinates of the surface points as indexes into the texture.

3D marble texture generated using Perlin noise [6]

1D noise functions can be used as well, for example to generate joint angle changes for idle or fidgeting animation of a character. Furthermore, any of the previous examples can be made to vary over time by adding an extra dimension to the noise function.

There are many other types of procedural content besides noise functions. For example, physics-based animations, such as rag dolls or fluid simulations, are types of procedural animation. Another example would be using a grammar, such as an L-system, to generate procedural models of plants.



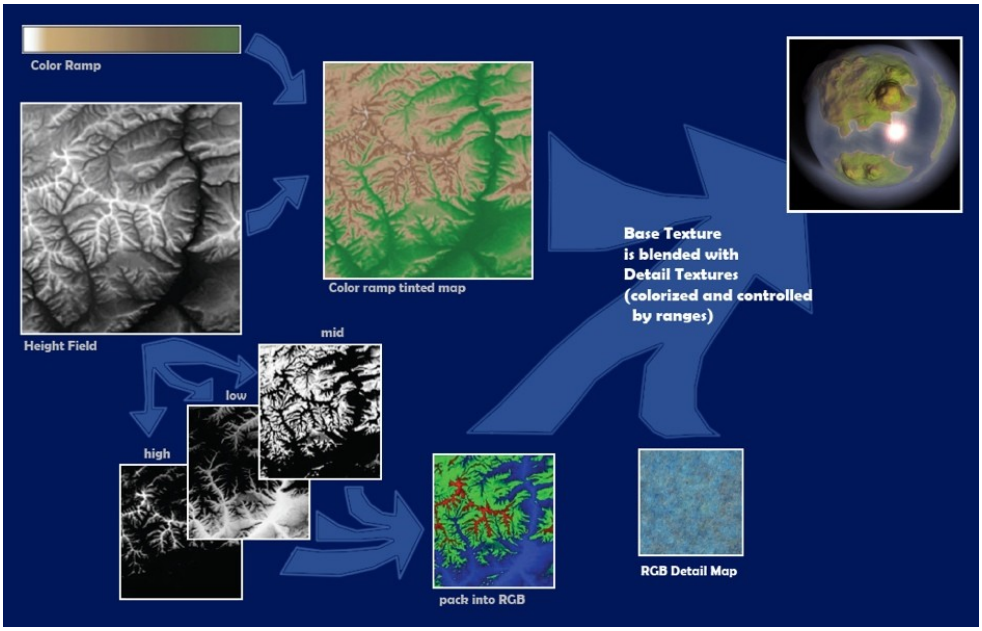
Trees generated using an L-System [7]

0	1	2
3	4	7

Plant generated by the simple L-System rules:

- Leaf → Stem + 2 Leaf
- Stem → 2 Stem

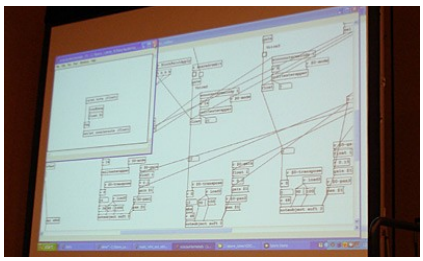
The video game Spore contains examples of several types of procedural content, including procedurally-generated planets [11] and procedurally-generated textures and animations for player-created creatures [12]. This content is generated using a variety of technologies (particle systems, etc.).



Procedurally generating planets in Spore [11]


Procedurally-textured played-generated creature from Spore [11]

Another interesting type of procedural content in Spore is the music [13], which is procedurally generated by combining a variety of samples using a language called Pure Data [14]. This allows effects such as creating custom theme music for player-generated creatures based on their physical attributes or changing the music from more energetic to more relaxing over long play-times.


Pure Data script used for music in Spore, shown at GDC 2008 [13]

In general, procedural content has several benefits over explicit. Because a relatively small procedure can generate a large (or potentially infinite) amount of content, it can be used to generate much more content than an artist would be able to create manually. The relatively small amount of

data required to generate the content can also provide performance gains, since it can be much quicker to read from memory, send to the GPU, stream over a network, etc.. This is an example of the general principle of trading memory access for computation.

Another benefit of procedural content is that it can be reactive to user input. For example, depending on the user's actions, generated terrain could gradually become more mountainous, or a texture could take on a progressively more aged appearance over time.

The downside to procedural content, of course, is that you must find a way to design an algorithm that generates the content you want.

The Unity 3D engine has support for several types of procedural content. It supports procedural animation in the form of physical simulation of rigid bodies, cloth, and rag dolls. It also supports procedural generation of Materials (textures) and Meshes, including special support for procedural generation of trees and their animation in wind. Procedural height-mapped terrain is also possible.

http://docs.unity3d.com/ScriptReference/Mathf.PerlinNoise.html
http://docs.unity3d.com/Manual/ProceduralMaterials.html
http://docs.unity3d.com/Manual/GeneratingMeshGeometryProcedurally.html

[1] https://en.wikipedia.org/wiki/White_noise
[2] https://en.wikipedia.org/wiki/Perlin_noise
[3] https://en.wikipedia.org/wiki/Value_noise
[4] http://www.noisemachine.com/talk1/19.html
[5] https://digitalerr0r.wordpress.com/2011/05/25/procedural-landscapes-using-perlin-noise
[6] http://mrl.nyu.edu/~perlin/experiments/hypertexture/vase.html
[7] https://en.wikipedia.org/wiki/L-system
[8] http://notch.tumblr.com/post/3746989361/terrain-generation-part-1
[9] http://minecraft.gamepedia.com/Customized
[10] http://minecraft.gamepedia.com/Biome
[11] http://www.andrewwillmott.com/s2007
[12] http://chrishecker.com/My_Liner_Notes_for_Spore
[13] http://uk.pc.gamespy.com/pc/spore/853810p1.html
[14] http://puredata.info/Members/hans/spore/