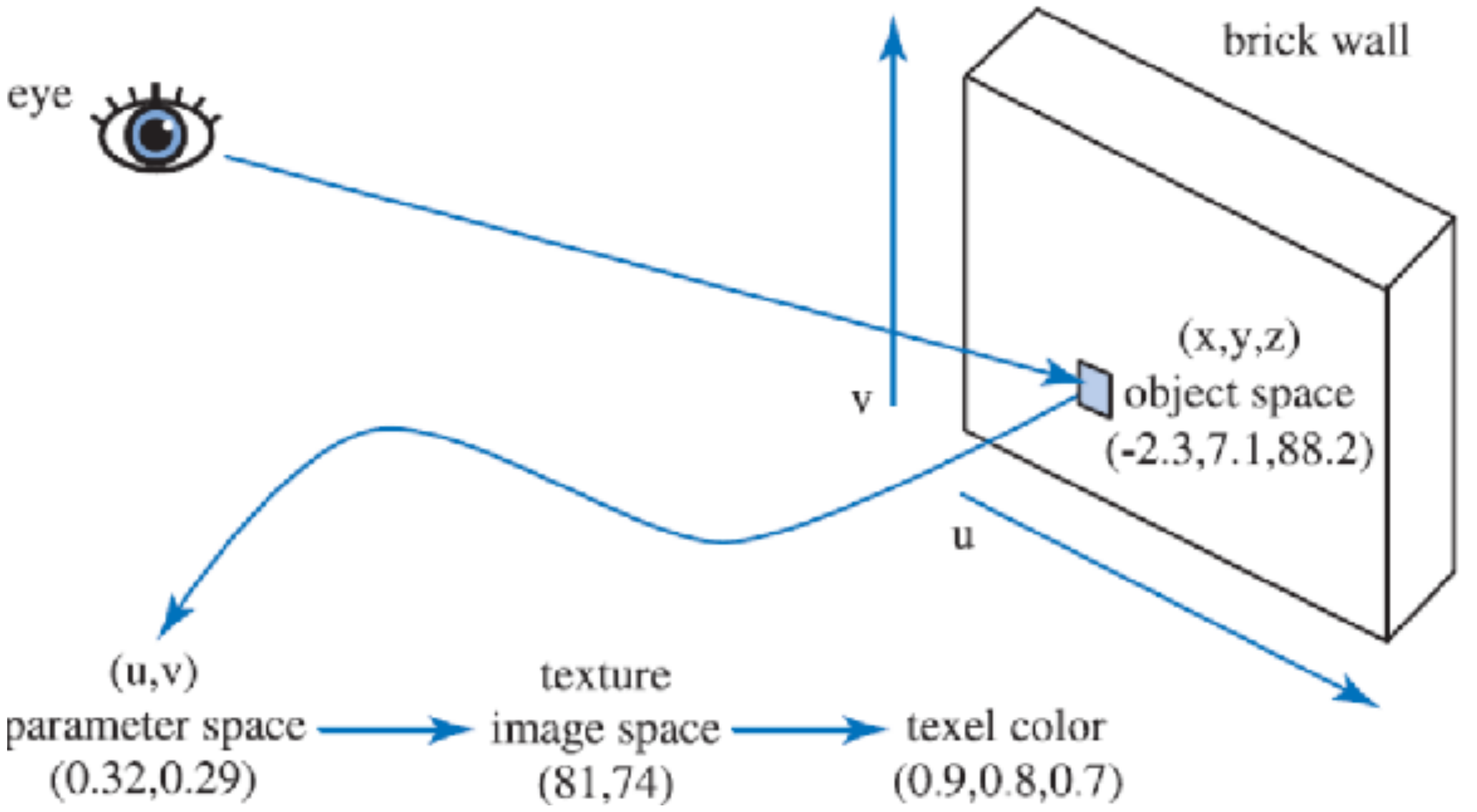
Noise Functions

Recall: Texture Mapping

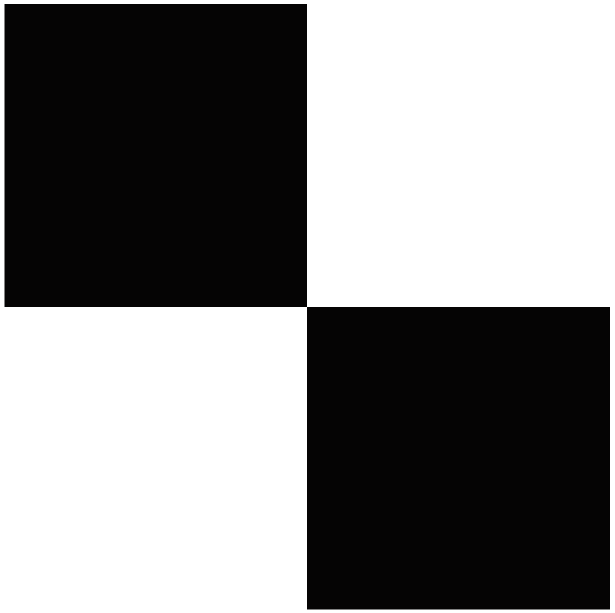


Procedural Texture

Main idea: determine color at (u,v) using **mathematical function**

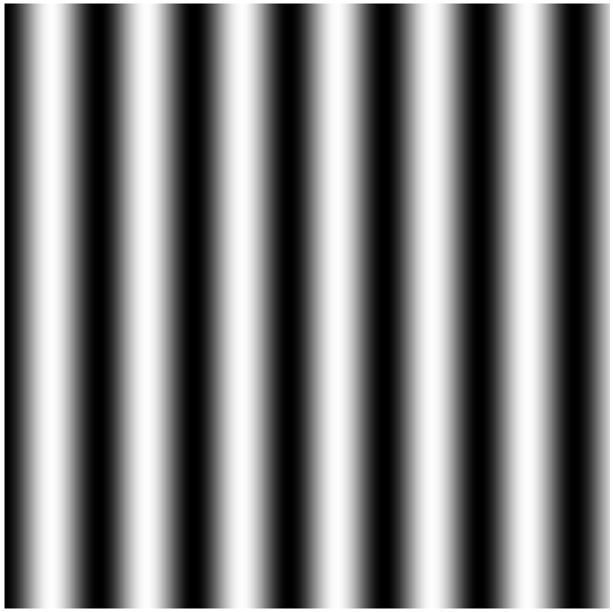
- no need for art assets
- computed on the fly: no memory cost
- can generate infinite amounts of data

Checkerboard

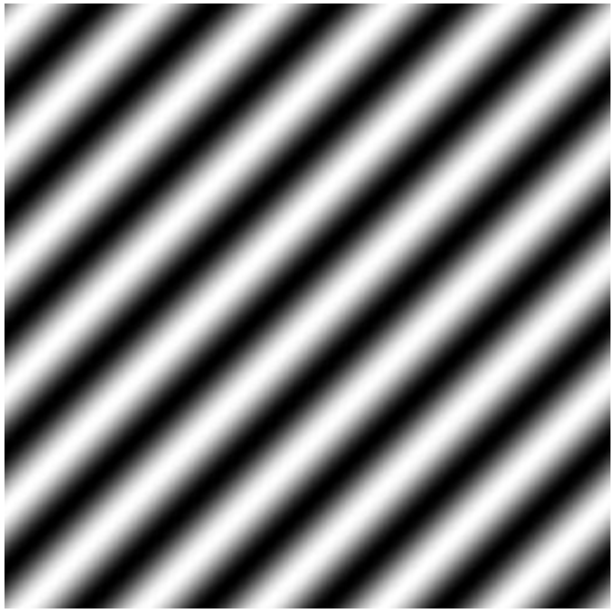


$$I(u, v) = (\lfloor 2u \rfloor + \lfloor 2v \rfloor) \mod 2$$

Stripes

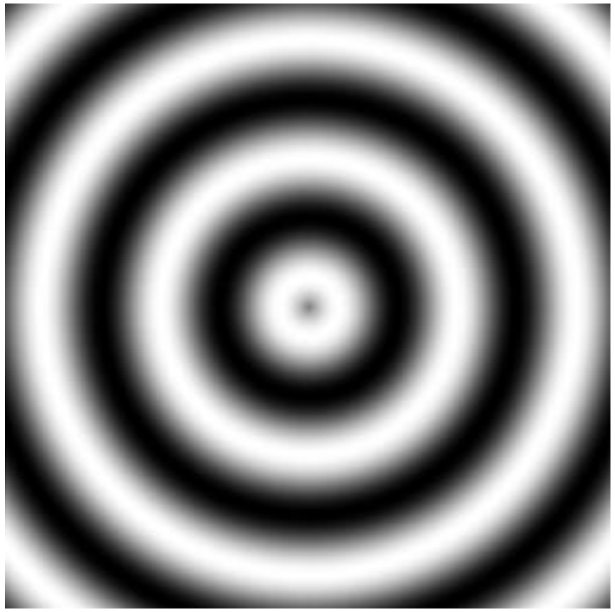


$$I(u,v) = \sin u$$



$$I(u,v) = \sin(u+v)$$

Stripes



$$I(u,v) = \sin\sqrt{(u-0.5)^2 + (v-0.5)^2}$$

What’s Missing?

Procedural textures provide some visual interest, but how can we make them look better?

Adding Noise

Problem: procedural textures are too regular

- Looks “fake” (i.e. inorganic)

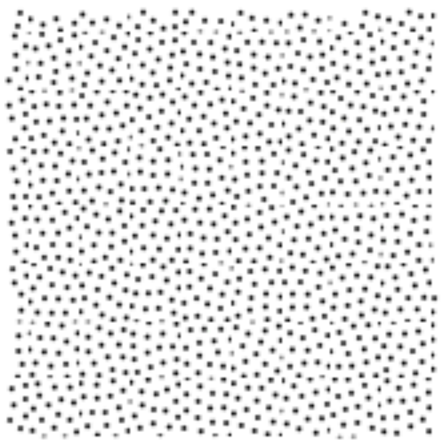
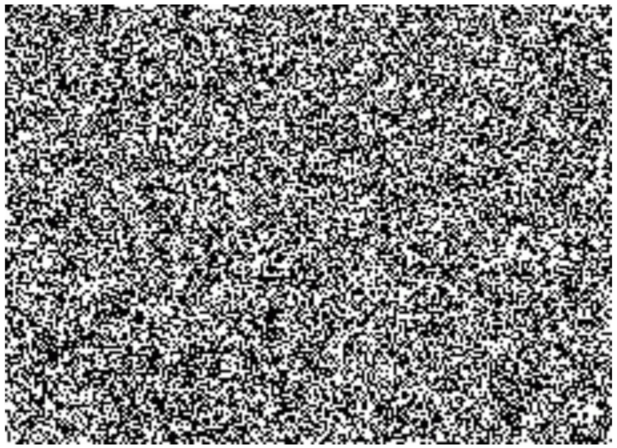
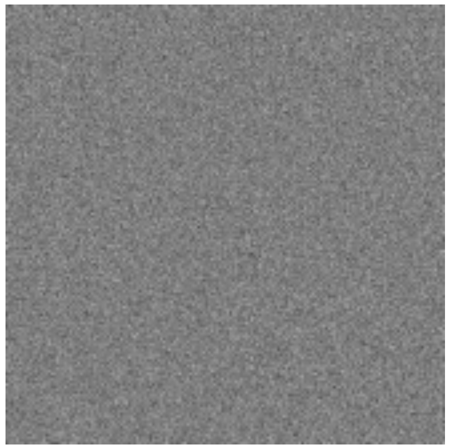
Real textures have noise

What is noise?

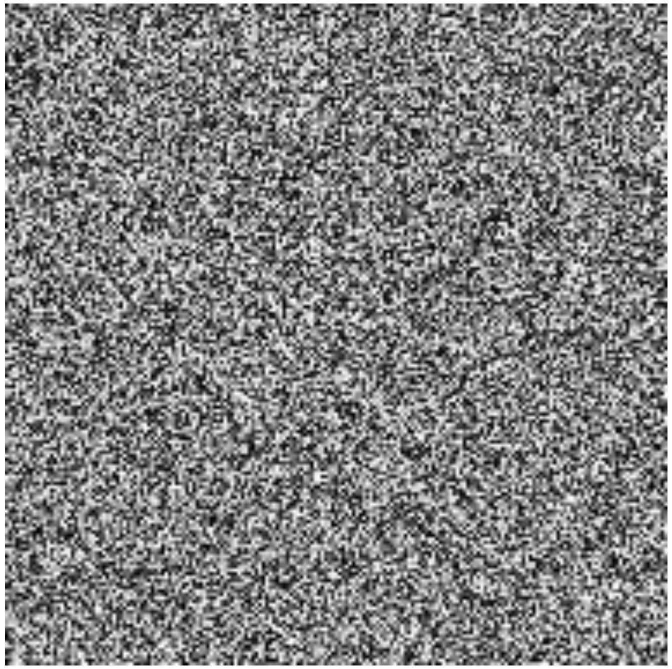
Noise

Random (stochastic) fluctuations in an expected signal

- Different concentrations of energy create different patterns

White Noise



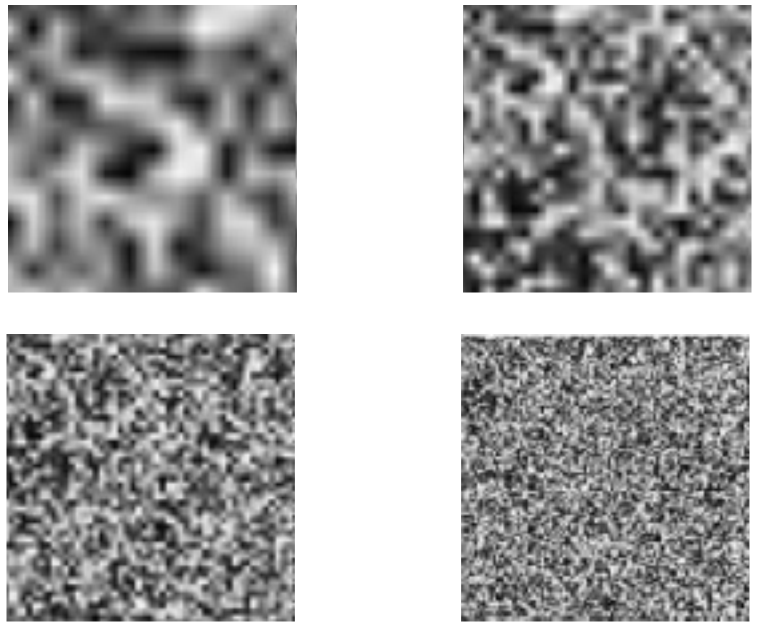
$$I(u, v) = \mathrm{rand}()$$

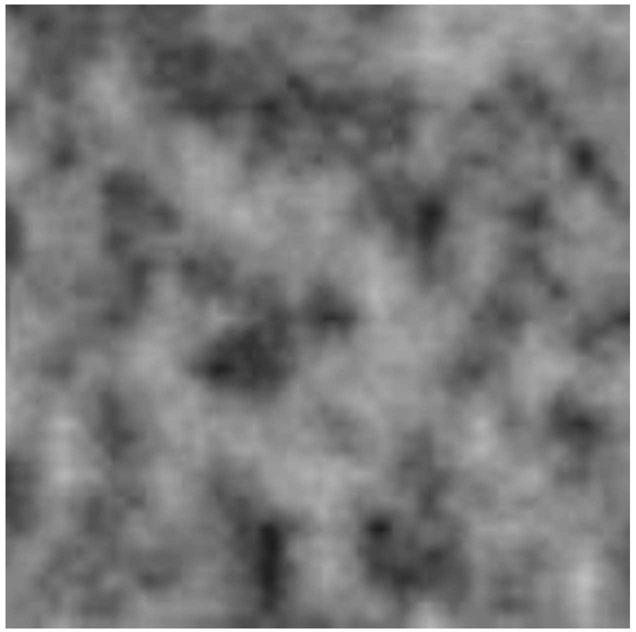
White noise problems:

- isn’t smooth
- isn’t correlated

Perlin Noise

Insight: A single noise function is unstructured but a combination of noise functions has structure



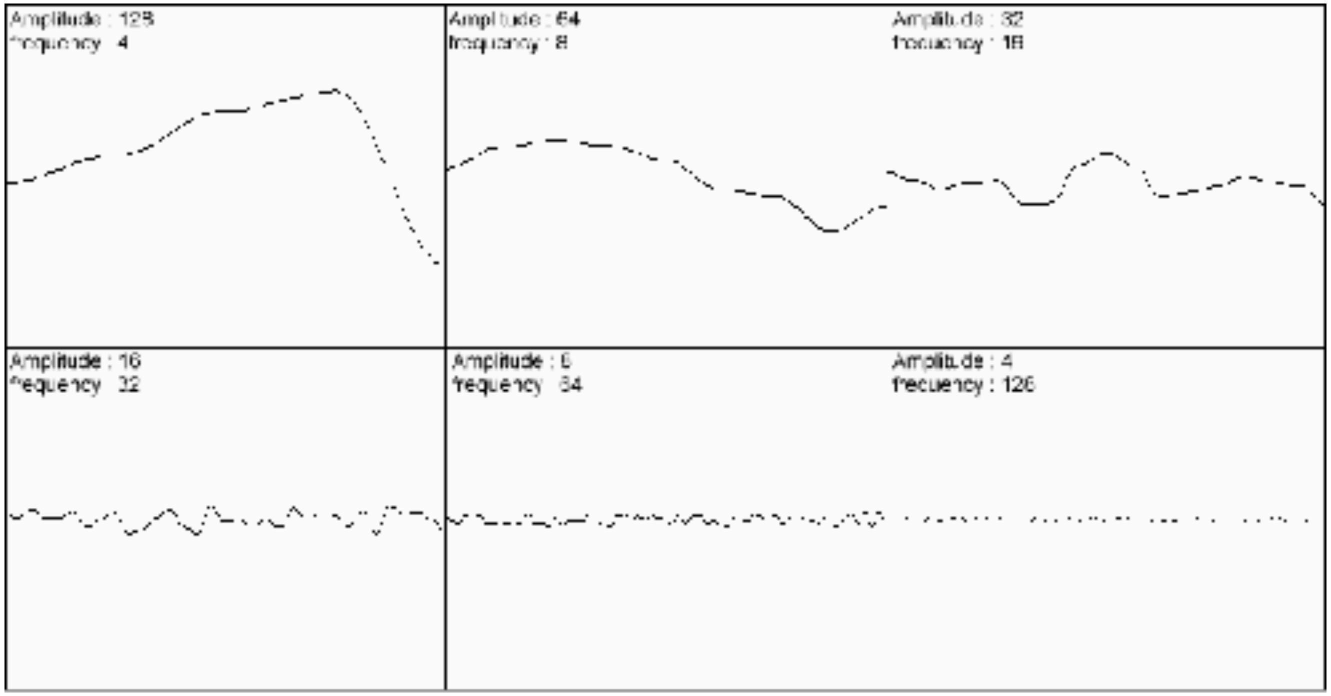


Generating Smooth Noise

1. Create grid of random gradient vectors
2. Compute points within grid using nearest nodes
3. Interpolate between node values to form continuous function
4. Combine smooth noise function with other smooth noise functions at different octaves

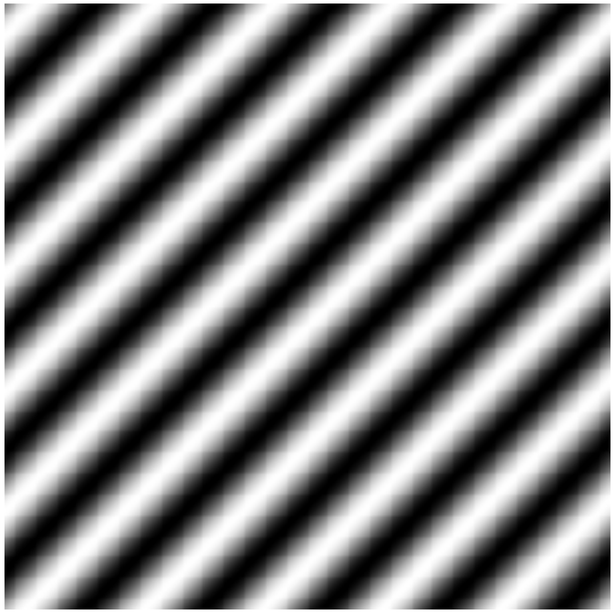
Noise Octaves

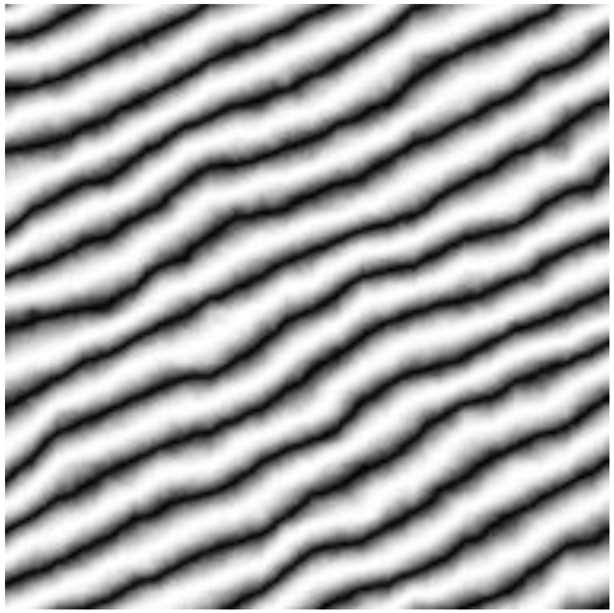
An octave represents a noise function with a particular frequency-amplitude



Perlin Noise Applications

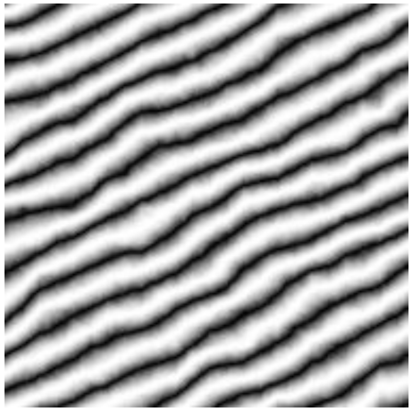
Perlin noise applied to procedural function…

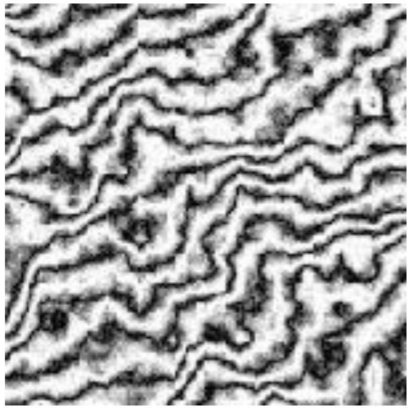




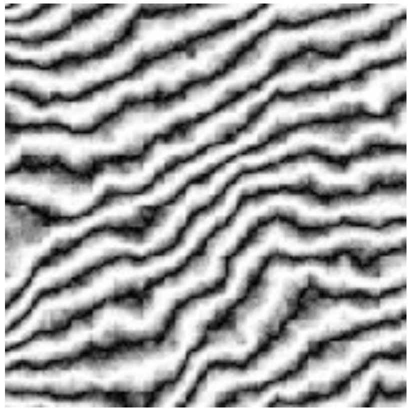
Manipulating Perlin Noise

- Control over:
  - Amplitude
  - Frequency
  - Number of octaves
  - Persistence (influence of amplitude on each successive octave)
- Parametrizable for artist controls



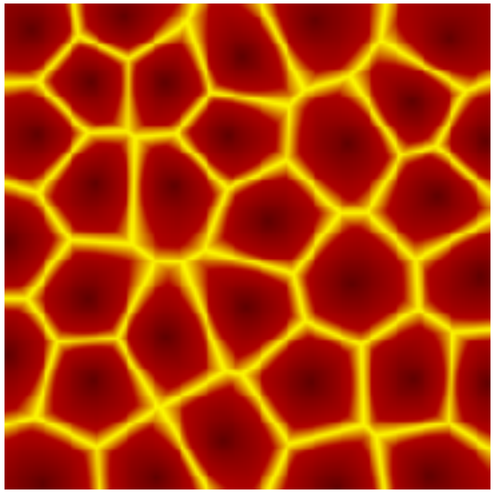
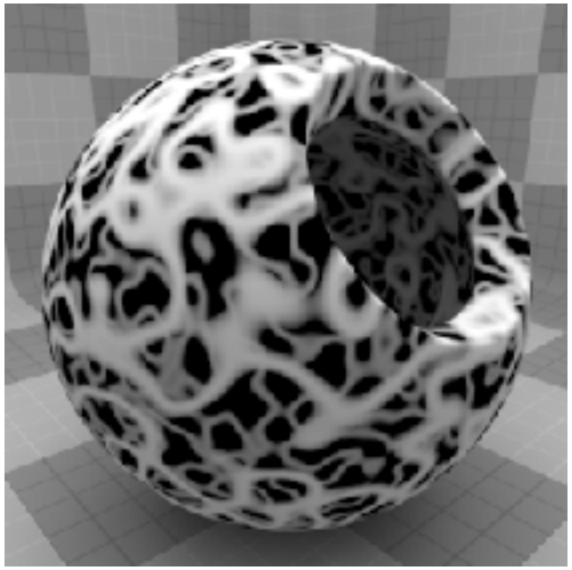
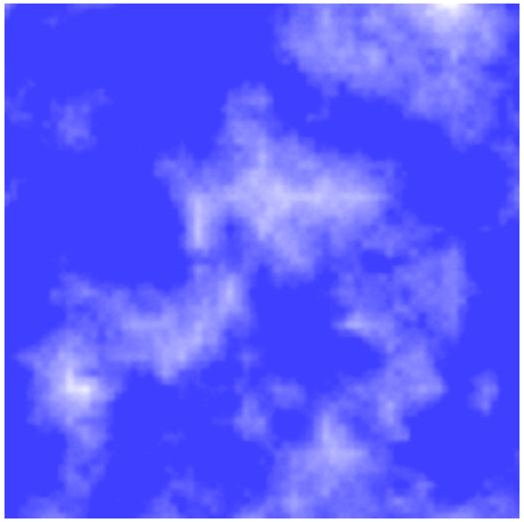
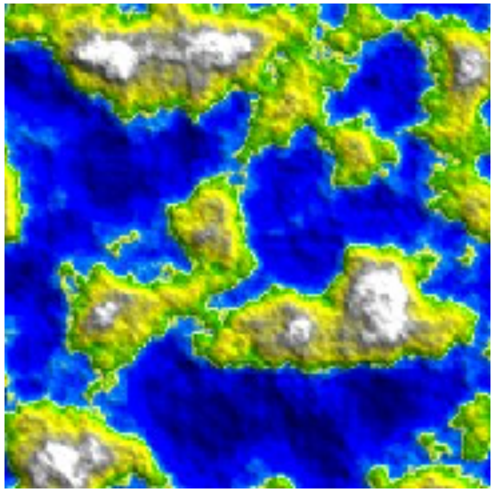


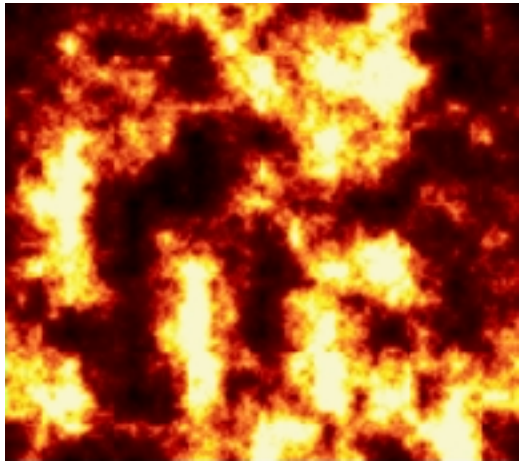
Increased amplitude



Increased frequency

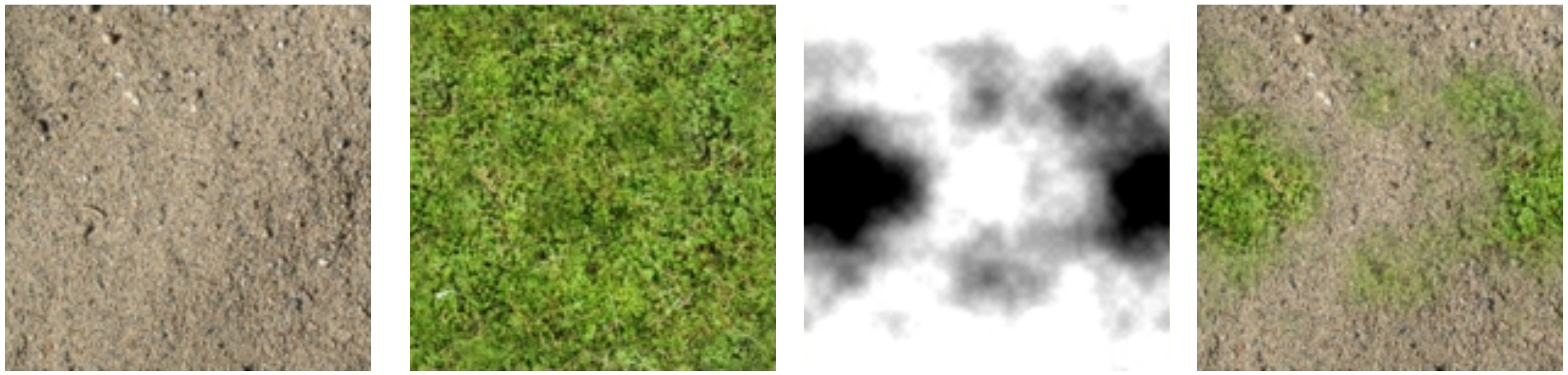
Perlin Noise Examples



Blending with Perlin Noise

Perlin Noise provides “organic” transition between textures by interpolating based on function value



http://devmag.org.za/2009/04/25/perlin-noise/

Animating with Perlin Noise

Generate Perlin noise in 3D and animate 2D frames using this 3D texture

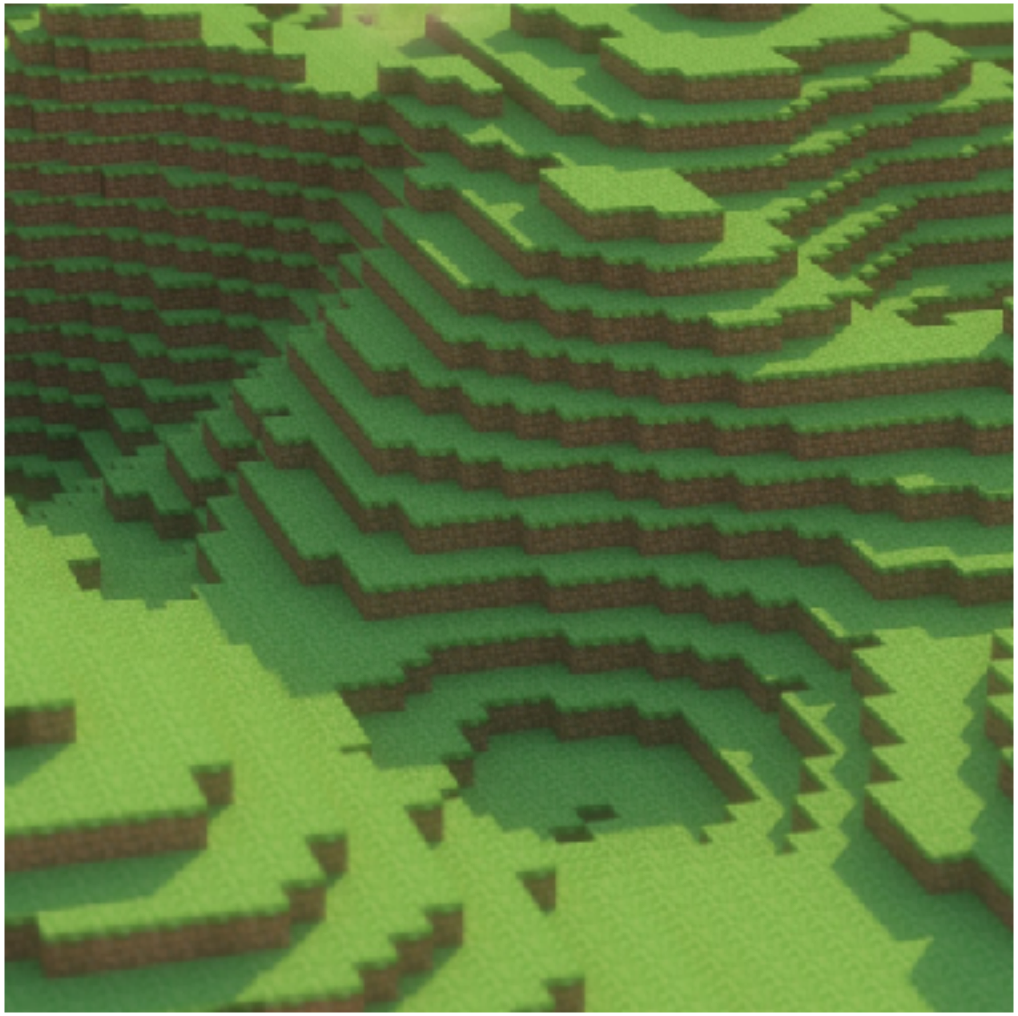
Or interpolate between Perlin noise functions to create smooth transitions over time

Simplex Noise

So actually use Simplex noise instead of Perlin noise…

- Fewer directional artifacts
- More computationally efficient
- Also by Perlin
- OpenSimplex is open source version of algorithm (many other implementations as well!)

Example: Minecraft



Minecraft Landscape

Key features:

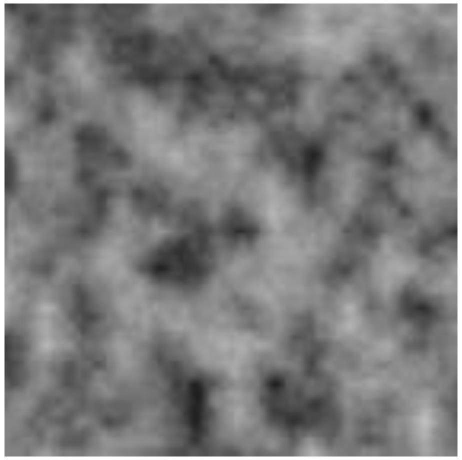
- Discrete (made of blocks / **voxels**)
- Random – no repeated features
- Extends indefinitely
- Persistent

Minecraft Landscape

Making a small piece of landscape easy:

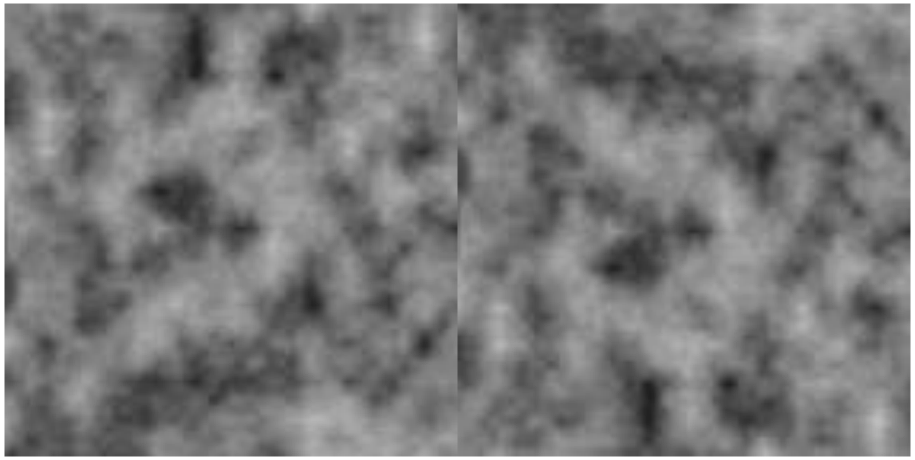
1. Generate Perlin noise patch
   - One pixel per block in (x,y) directions

2. Clamp heights to discrete steps



Problems…

How to handle seams?

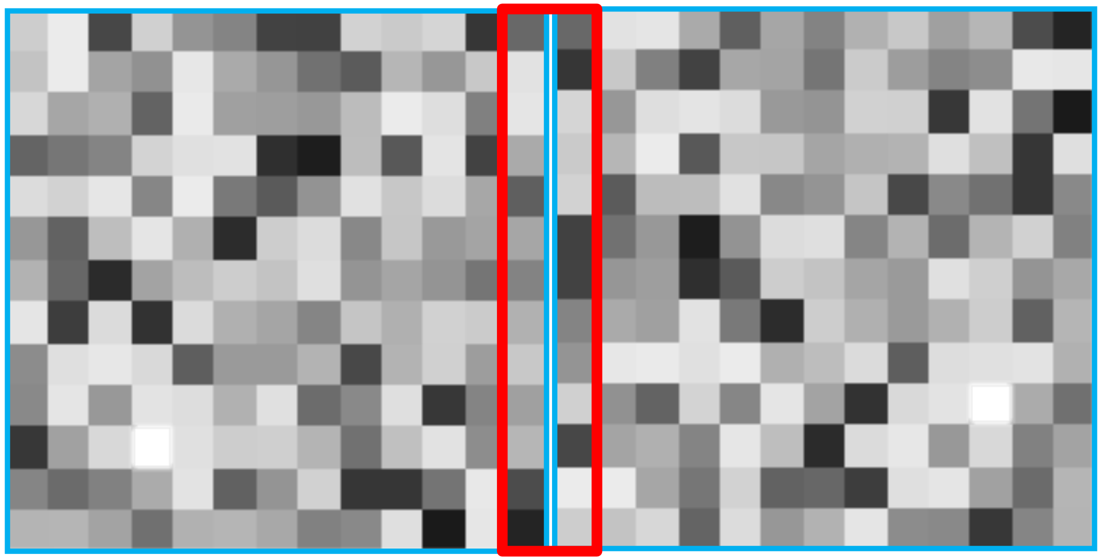


How to handle persistence?

Handling Seams

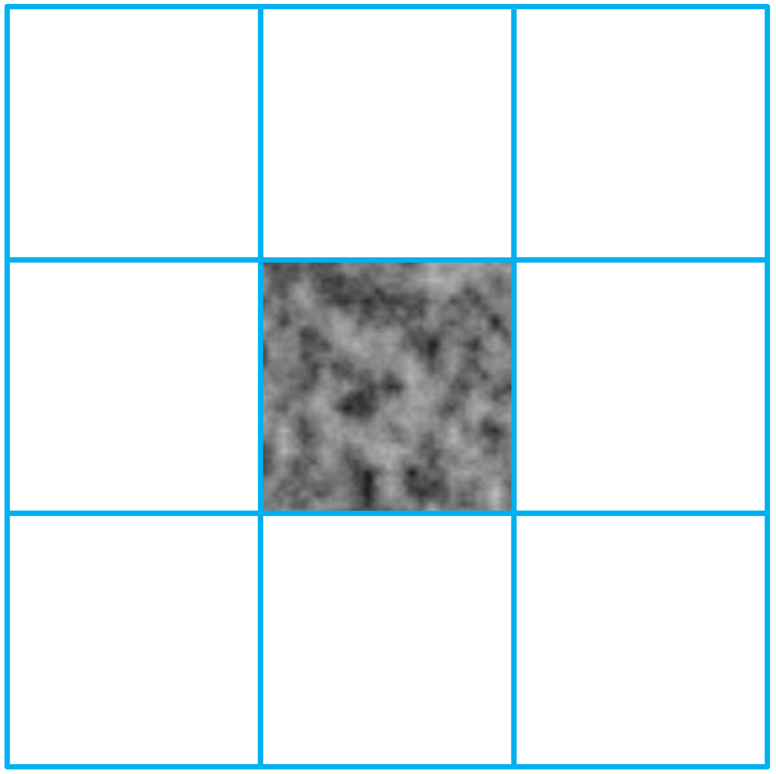
Interpolate with neighbor tiles at each level…

…perhaps using Perlin noise!



Handling Persistence

- World is made of tiles
- Each tile has a deterministic seed
- Must keep track of user activity with a change log



Same Idea in No Man's Sky…

Memory Efficient!

Tiles can be swapped in and out as needed

Keep an nxn buffer of tiles loaded around the player (same idea as Zelda, Megaman, etc on the NES)

Doesn’t solve popping (must implement some form of LOD)

Perlin Noise and Voronoi

- Possible to combine Perlin noise with other algorithms!
- Voronoi diagrams partition planes based on distance to provided points
- Commonly used together for terrain generation

Discuss: How can we generate terrain using Voronoi partitions? How do we combine this with a noise function?

Voronoi Sampling

Voronoi samples represent biome qualities:

- Altitude
- Latitude
- Rainfall

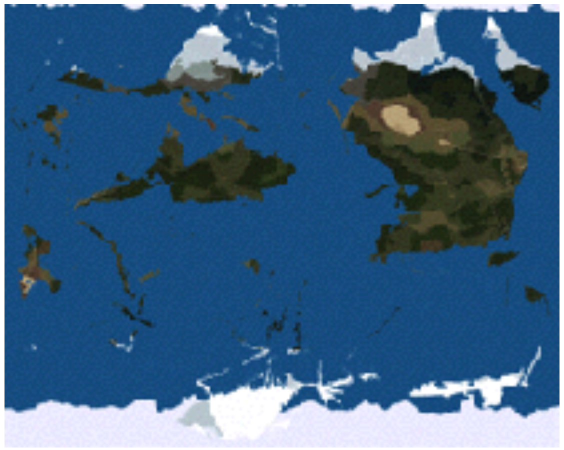
Generate biomes from samples then add noise between borders

- Noise represents biome weight or something similar

Going Further…

Simulation can be applied via hydraulic erosion, temperature, tectonics, flora and fauna etc

Can be as simple or as complex as desired (within system limitations!)



Perlin Worms

“Worms” have segments:

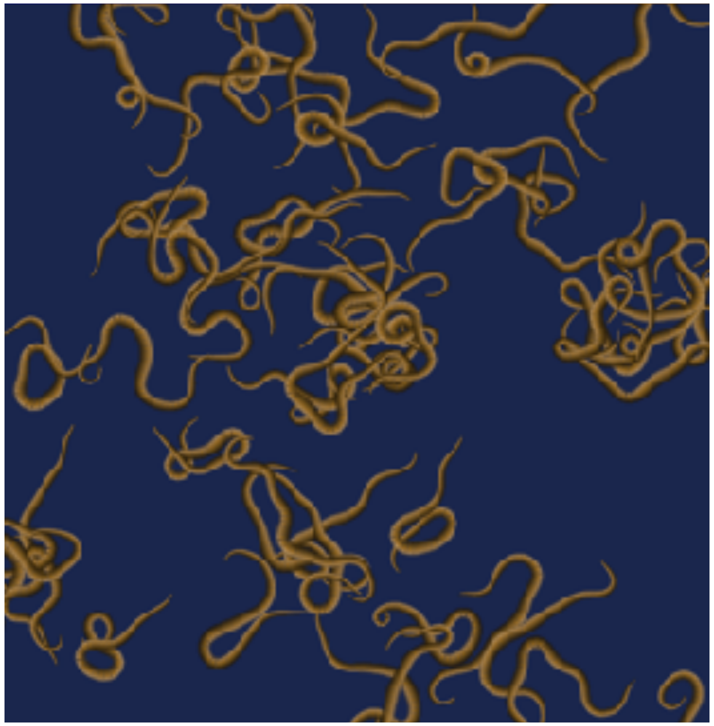
- Segments always connected
- Segments oriented based on noise function

Create “tunnels” in world to simulate caves

Guarantees on connectivity between tunnels
(unlike standard 3D Perlin noise)

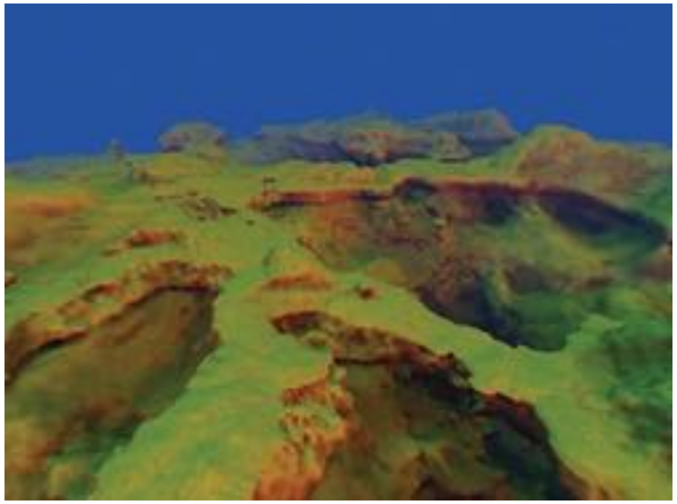
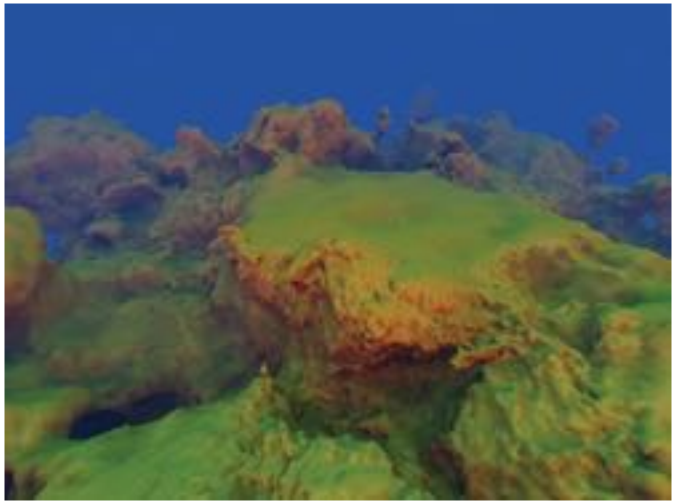
Perlin Worms

https://www.youtube.com/watch?v=aQ-bomm3lEc



NVIDIA Terrain Generation

http://http.developer.nvidia.com/GPUGems3/gpugems3_ch01.html

Additional Reading

http://mrl.nyu.edu/~perlin/paper445.pdf

http://flafla2.github.io/2014/08/09/perlinnoise.html

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

https://www.smashingmagazine.com/2016/03/procedural-content-generation-introduction/