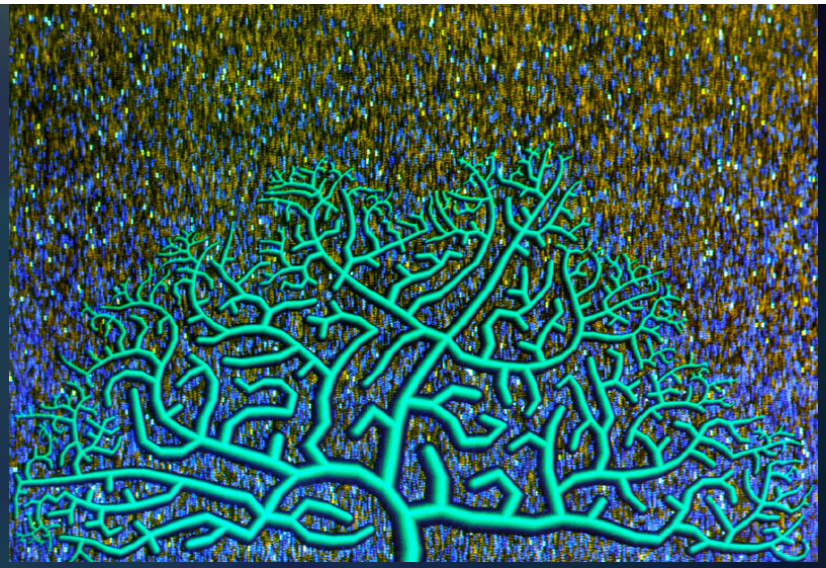


Jeff Ventrella, www.ventrella.com

CNM 190
Advanced Digital Animation
Lec 05 : Procedural Modeling Basics

Dan Garcia, EECS (co-instructor)

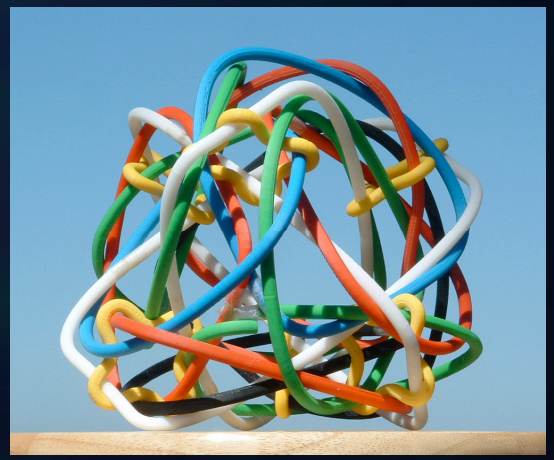
Greg Niemeyer, Art (co-instructor)

Jeremy Huddleston, EECS (TA)

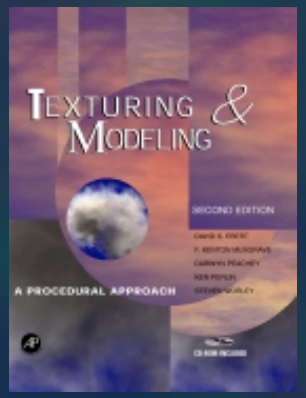
Carlo Séquin, EECS (guest lecturer)

Overview

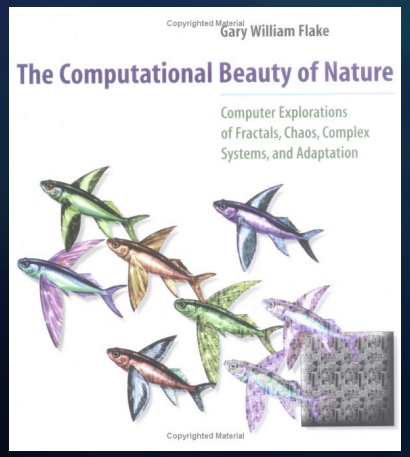
- Dan on Procedural Modeling Basics
  - Great references
  - A note before starting
  - What IS it?
  - Why would you use it?
  - What are its varied flavors?
  - Control vs Chaos
  - Randomness, Perlin Noise
  - Fractals & L-systems
  - Genetic algorithms
  - How do we do it?
  - Conclusion
- Carlo on Procedural Modeling of Abstract Geometric Sculptures
  - Trying to find the underlying paradigm and suitable parameterization for a new class of shapes



Great references



- "Texturing & Modeling : A procedural approach" (2/e)
  - David S. Ebert, F. Kenton Musgrave, Darwyn Peachey Ken Perlin & Steven Worley
  - Morgan Kaufman, 1998



- "The Computational Beauty of Nature"
  - Gary William Flake
  - MIT Press, 2000

A note before starting

- This is a mixed group of programmers and non-programmers
- I'm aiming this presentation at the non-programmers for whom much of this might be new, to bring us all up to the same level of discourse (roughly)
- I ask patience of the programmers, however, hopefully there is new information for all students

What IS it?

en.wikipedia.org/wiki/Procedural_modeling

- "Procedural modeling is an umbrella term for a number of techniques in computer graphics to create 3D models (and textures) from sets of rules. L-Systems, fractals, and generative modeling are procedural modeling techniques since they apply algorithms for producing scenes. The set of rules may either be embedded into the algorithm, configurable by parameters, or the set of rules is separate from the evaluation engine."

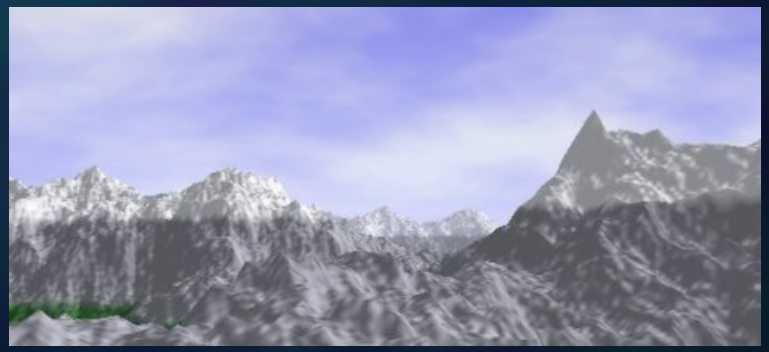
  "Although all modeling techniques on a computer require algorithms to manage & store data at some point…"

  "procedural modeling focuses on creating a model from a rule set, rather than editing the model by mouse."

Why would you use it?

en.wikipedia.org/wiki/Procedural_modeling

- "Procedural Modeling is applied when it would be too cumbersome (or impossible!) to create a model using general 3D modelers, or when more specialized tools are required. This is the case for plants & landscapes."



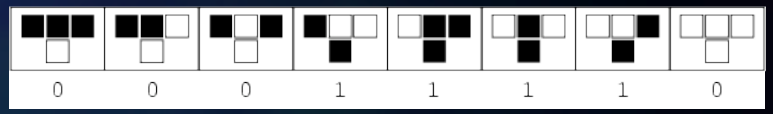
Paul Martz, martz@frii.com

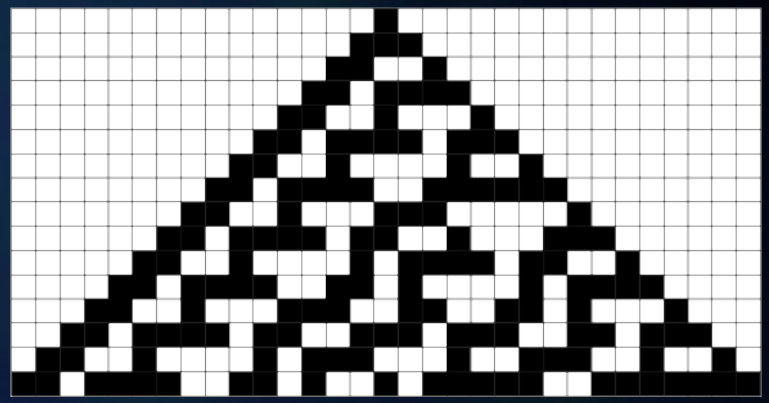
What are its varied flavors?

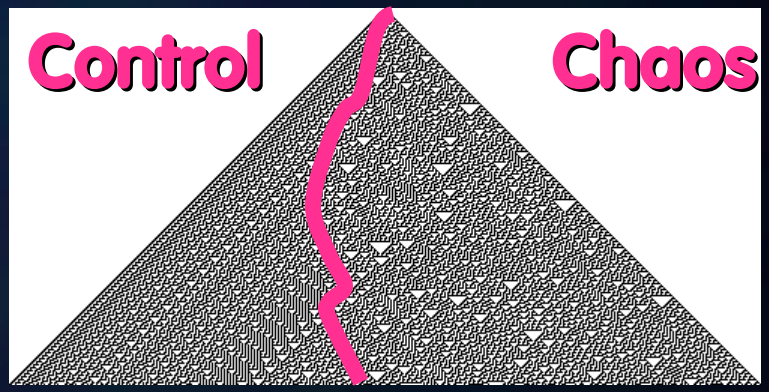
- Algorithm- or Data-driven geometry
  - …and animation (we'll talk about later)
- Fractals
  - Objects, landscapes, coastlines
- Particle Systems
- Simulations
  - Hair, fur, water, clouds, natural phenomena
- Genetic Algorithms
- More?

Control vs Chaos

- How much do you let randomness control your geometry?
  - None at all means you'll always get predictable results
  - You can constrain randomness
  - You can either randomize or set the initial seed, which can give repeatable randomness
  - Or you can give in to chaos, setting initial conditions and letting it run
- Sometimes deterministic rules generates chaotic sequence… emergent randomness!





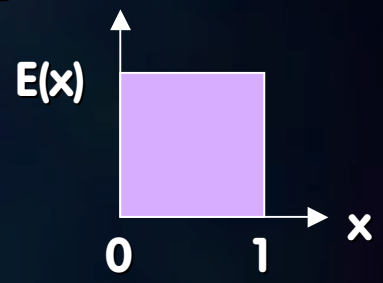


Life1D Rule 30

What is (pseudo) randomness?

en.wikipedia.org/wiki/Random_number_generator

- Almost every programming language has a built in `random()` and `seed()`
- They usually return a uniformly-distributed floating point number [0,1). E.g., in `Python`:
  - `>>> from random import *`
  - `>>> random()`
  - `0.72936670465656062`



- That's pretty good for almost every CG use
  - For other applications (e.g., crypto), often use "natural randomness" from physical processes.
  - See `www.lavarnd.org`

What is the seed?

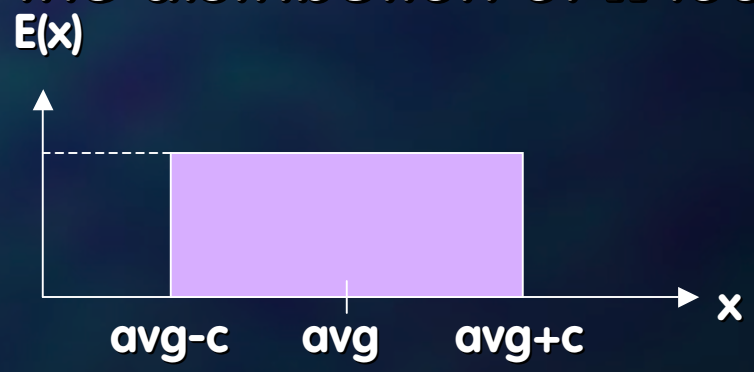
- The `seed()` allows for repeatability!
  - `>>> seed(0) # Arg => initialization`
  - `>>> random()`
  - `0.8444218515250481 2`
  - `>>> random()`
  - `0.7579544029403024 7`
  - `>>> random()`
  - `0.420571580830845`
  - `>>> seed(0)`
  - `>>> random()`
  - `0.8444218515250481 2`

What is the seed's argument?

- Seeding allows for multiple repeatable paths!
  - `>>> seed(0)`
  - `>>> random()`
  - `0.84442185152504812`
  - `>>> seed(1)`
  - `>>> random()`
  - `0.13436424411240122`
  - `>>> seed(0)`
  - `>>> random()`
  - `0.84442185152504812`
  - `>>> seed(1)`
  - `>>> random()`
  - `0.13436424411240122`

How do we constrain it?

- Scale the random variables down so its effect isn't felt much, and range changes
  - `>>> r = random()  # [0, 1)`
  - `>>> h = avg + 2*r*c - c # height`
- What does the distribution of `h` look like?



- If we want our noise to vary between -1 and 1, as we often do, how would we do that?

Perlin Noise Background

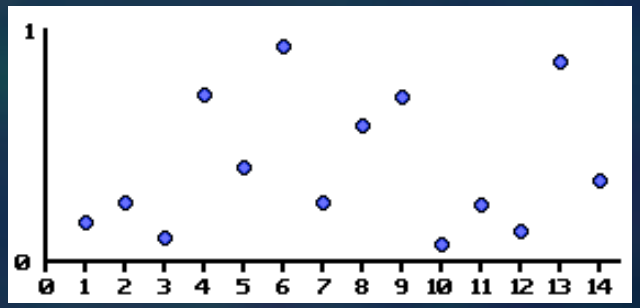
mrl.nyu.edu/~perlin/doc/oscar.html

- Ken Perlin, NYU Prof
- In 1983 he "invented" the idea of CG noise
- In 1985 he wrote the seminal SIGGRAPH paper (top 10 most influential in CG!)
- In 1986 it started being adopted by all!
- In 1997 he received an Oscar for his work

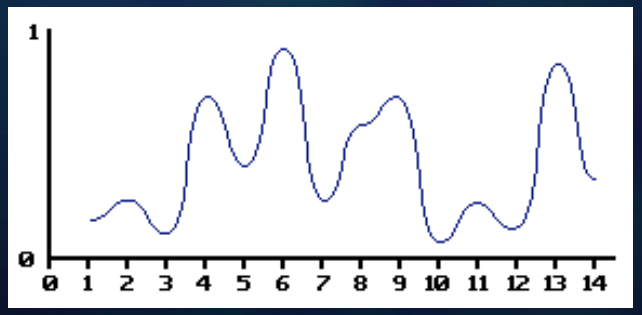
Perlin Noise Basics I

freespace.virgin.net/hugo.elias/models/m_perlin.htm

- Idea… start w/noise from `random()`



- Smoothly interpolate

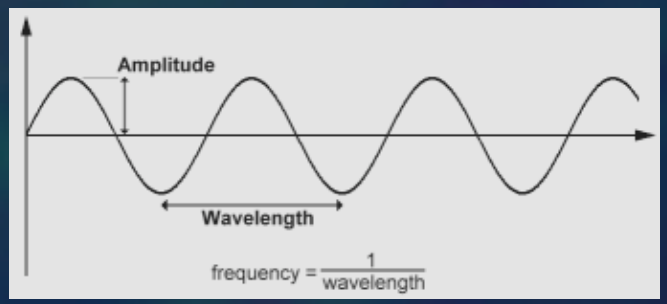


- …We have a continuous function of `F(t)`, yay!

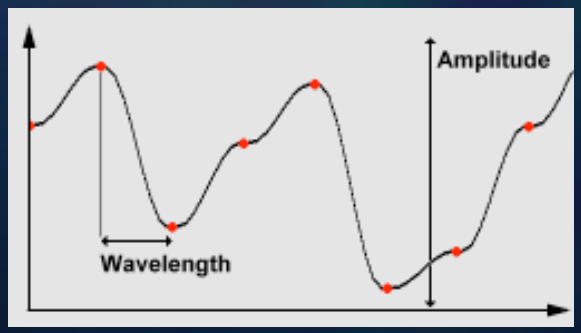
Perlin Noise Basics II

freespace.virgin.net/hugo.elias/models/m_perlin.htm

- Amplitude and frequency



- …which for a Noise wave

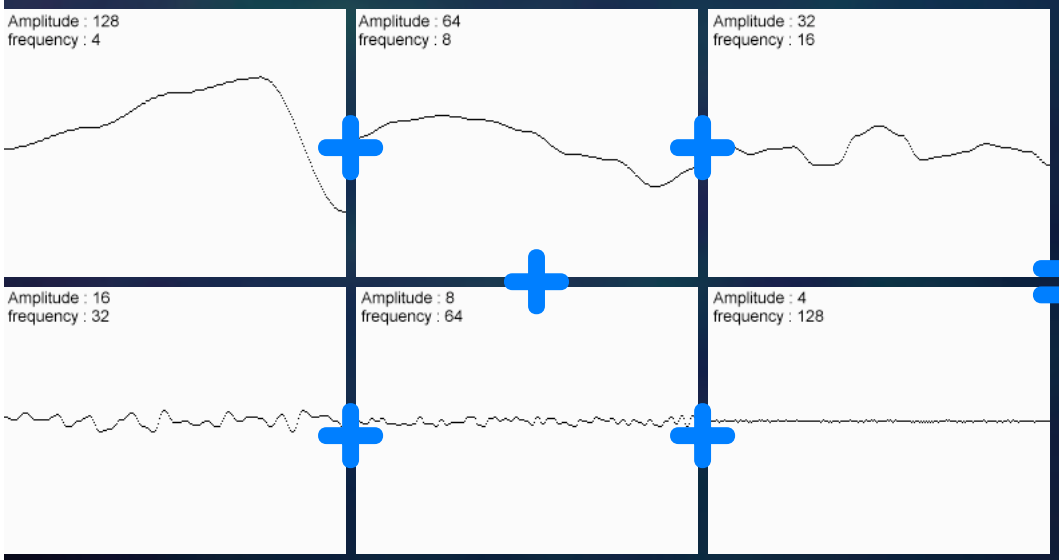


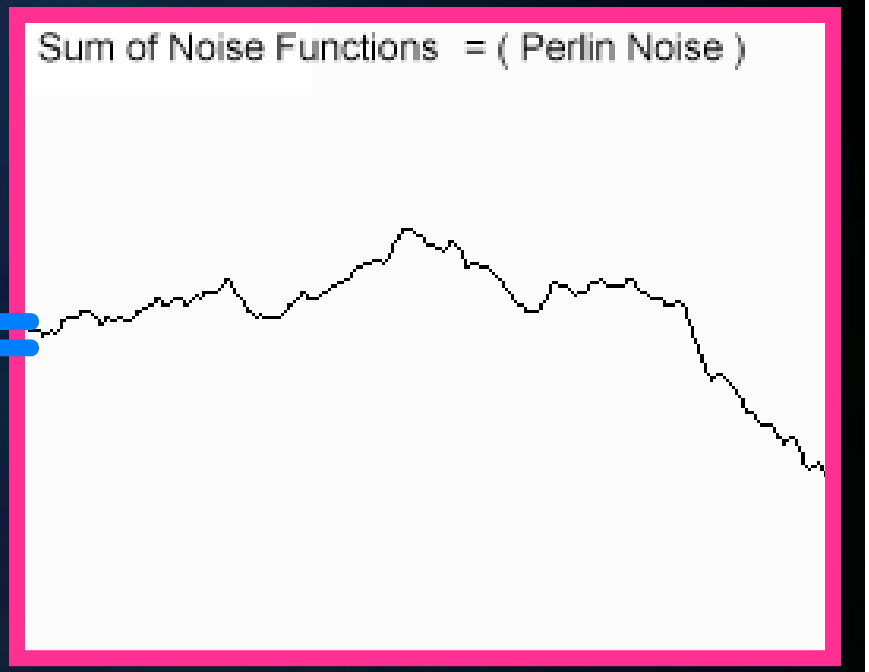
- …is controllable via:
  `AMP * F(FREQ * t)`

Perlin Noise Basics III

freespace.virgin.net/hugo.elias/models/m_perlin.htm

- Now, take combinations of these AMP and FREQ octaves and sum them together.





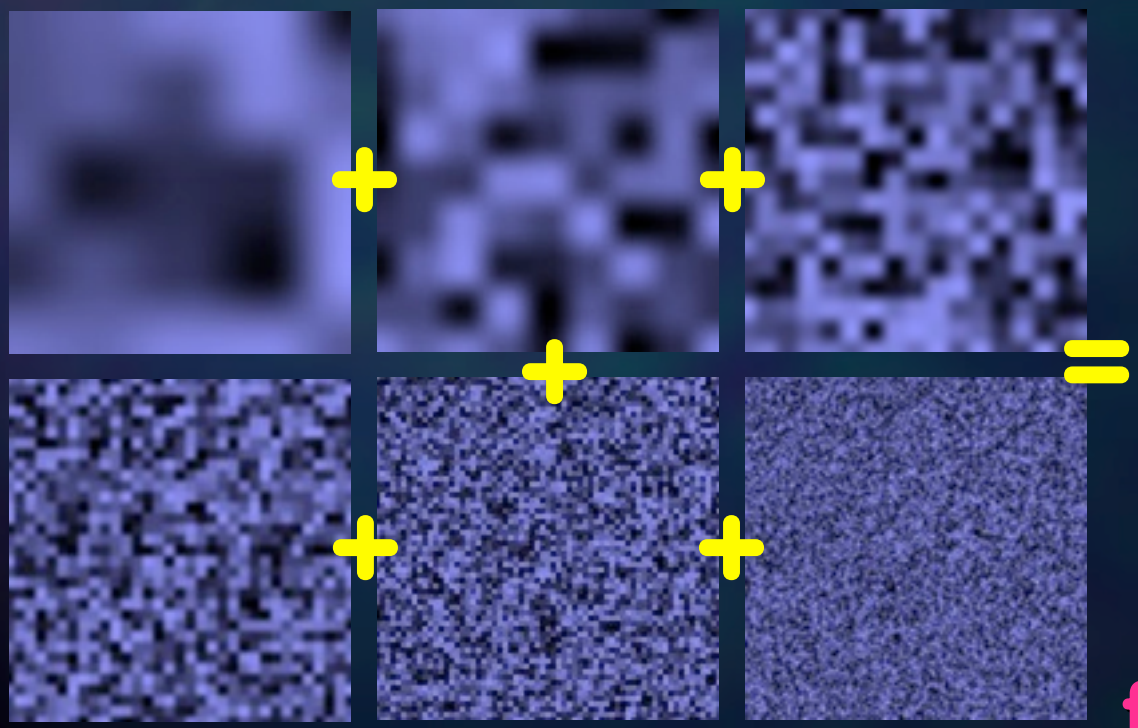
- Note that here we double the FREQ and halve the AMP, so the high frequencies affect the result much less than the low

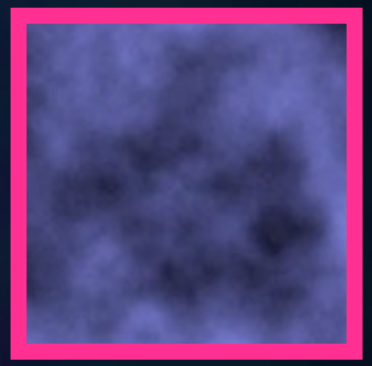
Perlin noise fractal 1D mountain

Perlin Noise Basics IV

freespace.virgin.net/hugo.elias/models/m_perlin.htm

- This also works in 2D





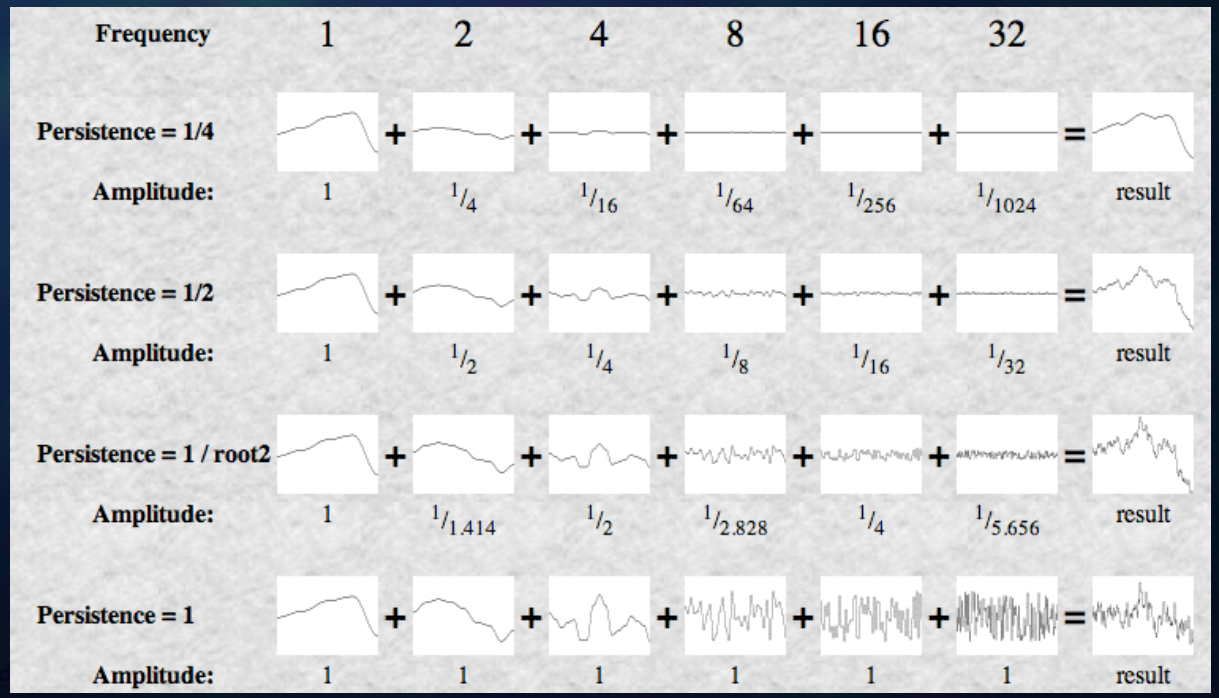
Perlin noise fractal 2D mountain (or texture!)

- …and for 3D, and for 4D!

Perlin Noise Basics V

freespace.virgin.net/hugo.elias/models/m_perlin.htm

- How should you attenuate the `AMP` as you're increasing `FREQ`? How rocky will your mountain be?
- Introduce Persistence
  - Frequency = 2^i
  - Amplitude = $persistence^i$



Perlin Noise Details

freespace.virgin.net/hugo.elias/models/m_perlin.htm

- How many octaves?
  - Up to you, not more than you can see
- How to interpolate?
  - Linear, cosine, cubic, many spline options
- Can smooth noise
  - Average with neighbors
- Center random @ [-1, 1]
  - You can turn noise up or down and it just "fuzzes out" to zero when the scale is small

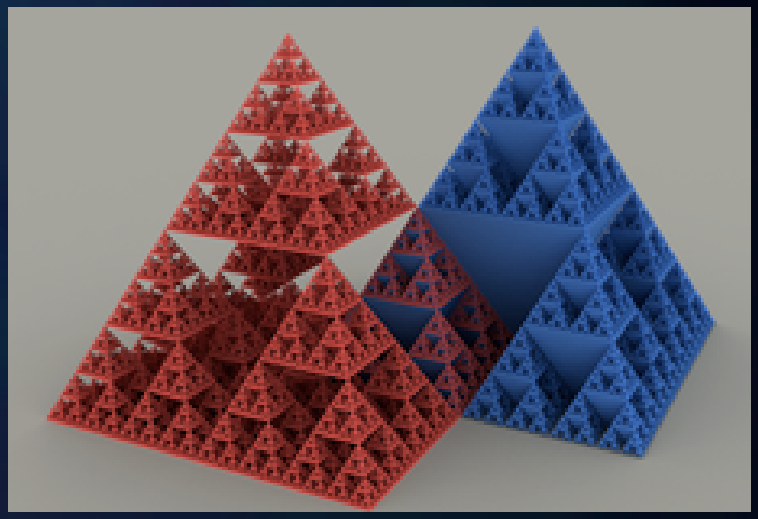
Perlin Noise Final Thoughts

www.noisemachine.com/talk1/

- Perlin summarizes history and context in an online talk (link shown above)
  - "Making Noise", 1999
- He thinks of noise like salt
  - Noise [salt]
    - is boring
  - F(blah) [food without salt]
    - can be boring
  - F(noise, blah)
    - can be really tasty
- We'll see this again in CNM191 when we cover procedural textures (I.e., shaders)

Fractal Geometry Overview

- A shape recursively constructed, or self-similar. How similar?
  - Exact
  - Quasi
  - Statistical
- Fractals are also found in nature, so can be used to model
  - Clouds, snow flakes, mountains, rivers



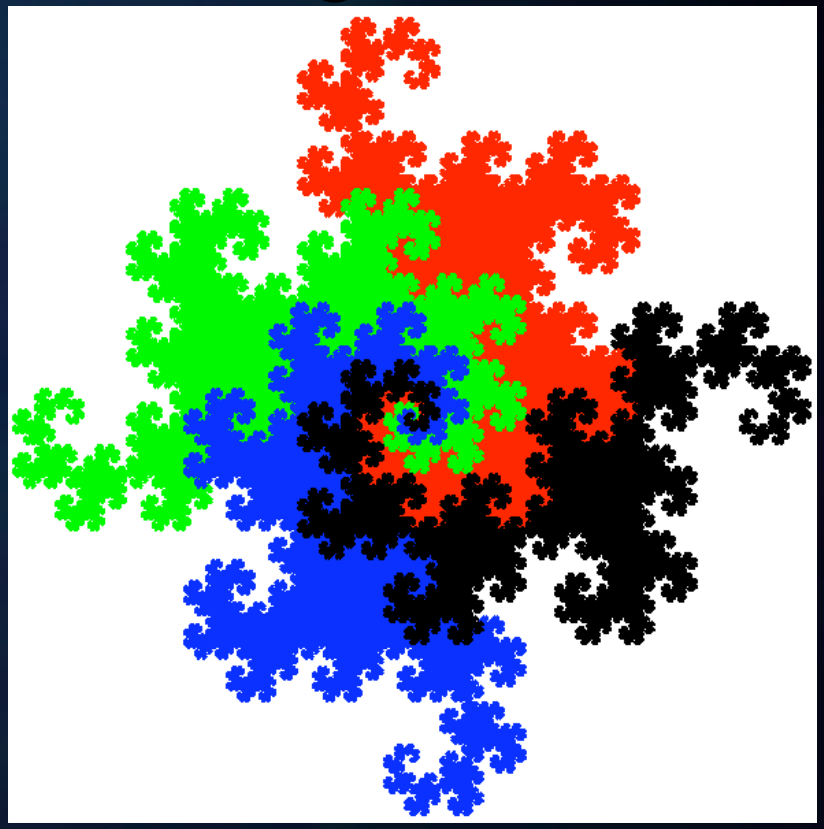
Sierpinski pyramids

Fractal Geometry Simplicity

en.wikipedia.org/wiki/Dragon_curve

- The beauty of them is also how **little code** is required to author!
- Base case
  - Draw a straight line and choose an "up"
- Recursive case
  - Break every line in 2, bend them to make a right angle, and set the "left" line up and the "right" line down.

Dragon curve

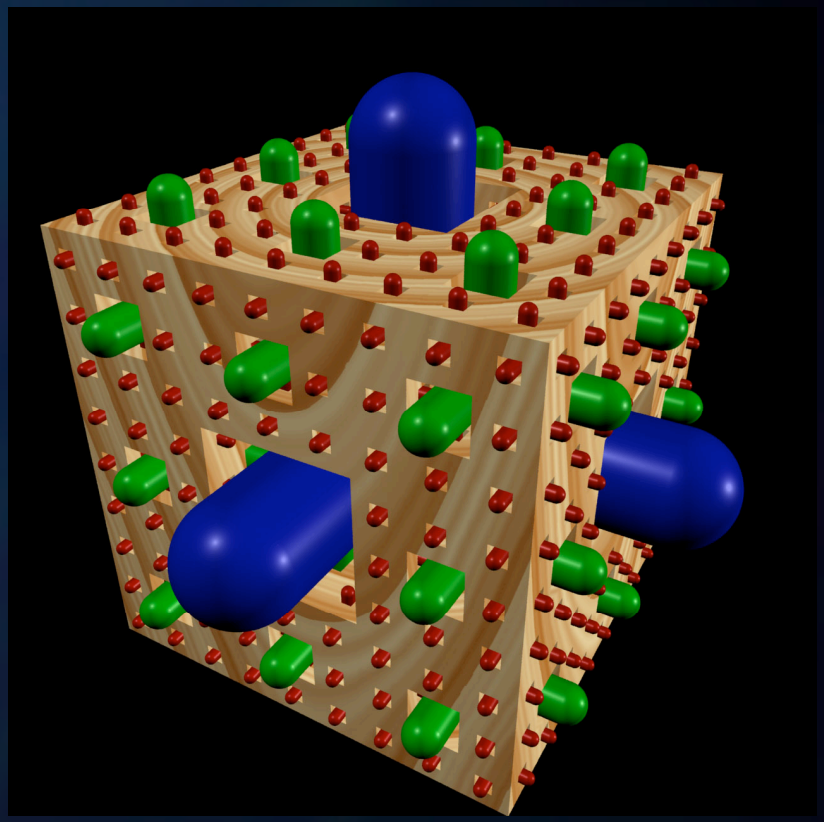


dragon, dan's fractals, ucbugg 2d

Fractal Geometry in 3D

en.wikipedia.org/wiki/Menger_sponge

- It's just as simple in 3D
- Dan's Menger cube:
  - Base case (n=0)
    - Spit out cube
  - Recursive case (n)
    - Scale down the world by a third
      - I.e., scale (1/3,1/3,1/3)
    - Move to the rims of the cube, leaving middles empty (20 in total)
      - I.e., Translate (blah)
    - Call yourself recursively with the n-1 case



hard*.tif, siercube, ucbugg 3d

Lindenmayer system (L-system)

en.wikipedia.org/wiki/L-system

- Formal grammar often used to model the growth of plant development
- These can also be used to specify / generate fractals
- Simply put, it's a series of rewriting rules that can succinctly describe fractal geometry using turtle graphics

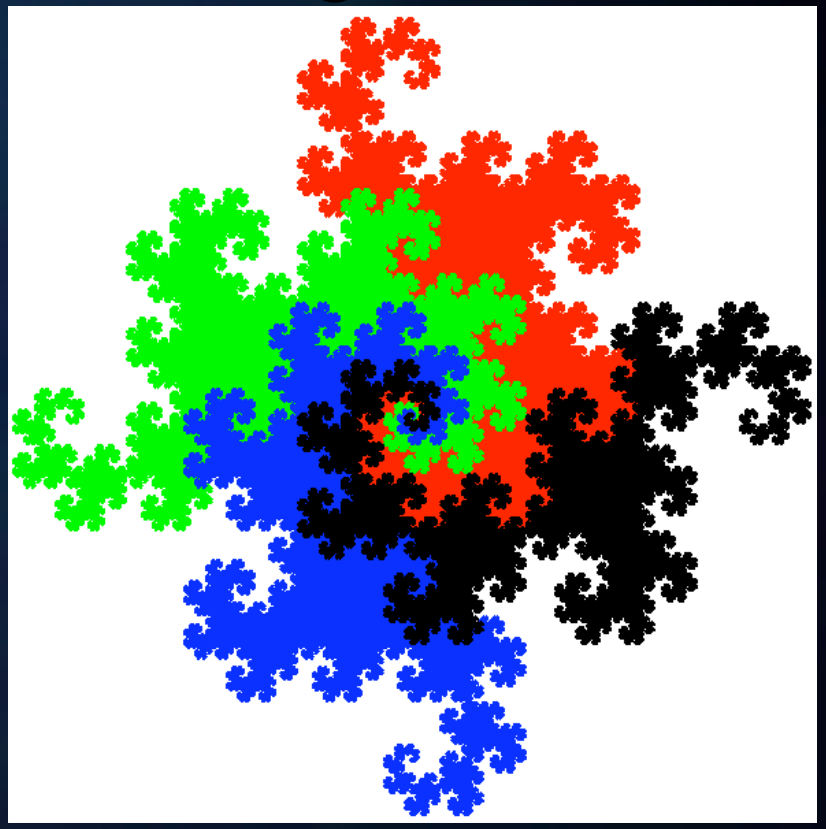
Weeds

L-system example : C-Curve

en.wikipedia.org/wiki/Dragon_curve

- Angle 90 degrees
- Initial string FX
  - F means move forward, in Turtle graphics mode
- String rewriting rules
  - X → X+YF+
  - Y → -FX-Y
- Example
  - 1 : FX → F
  - 2 : FX+YF+ → F+F+
  - 3 : FX+YF++-Fx-YF+ → F+F+F-F

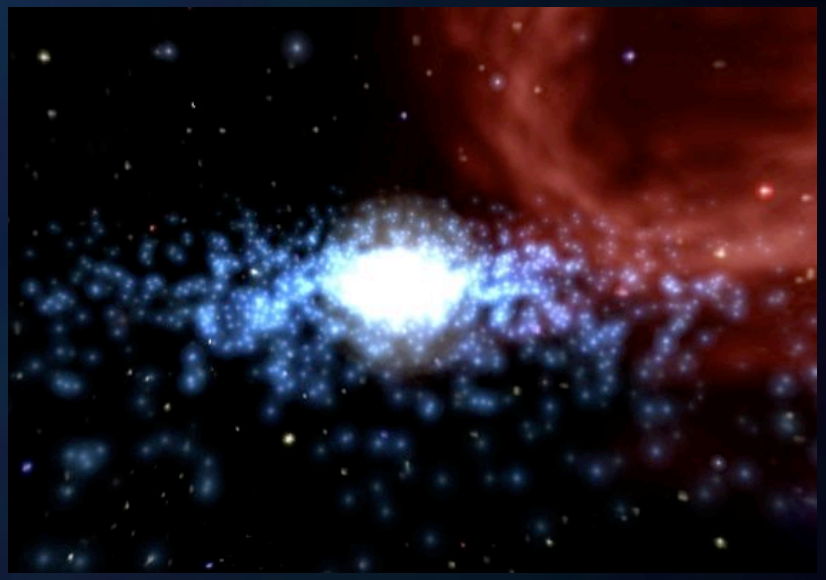
Dragon curve



Particle systems

en.wikipedia.org/wiki/Particle_systems

- Typically used to simulate fuzzy phonomena
  - Smoke, Explosions, Water, Sparks, Dust, Stars
- Emitter controls particle generation, simulation
- What is involved with the geometry?
  - Typically output as textured billboard quad
  - Or, single pixel
  - Or, metaball (for gooey materials)

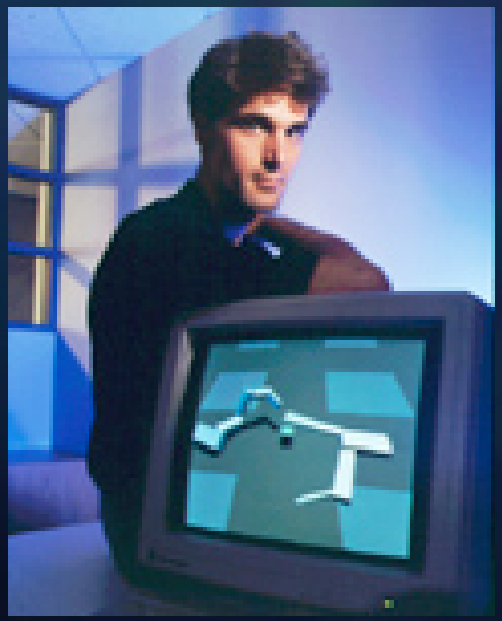


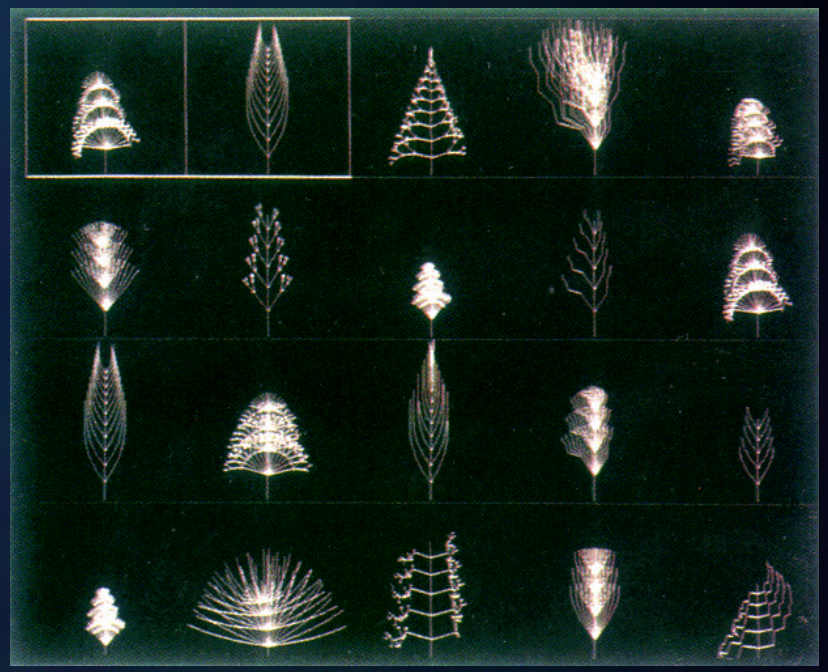
pixar cosmic voyage

Genetic Algorithms

web.genarts.com/karl/

- Karl Sims blew away his colleagues with his 1994 seminal work on evolved creatures



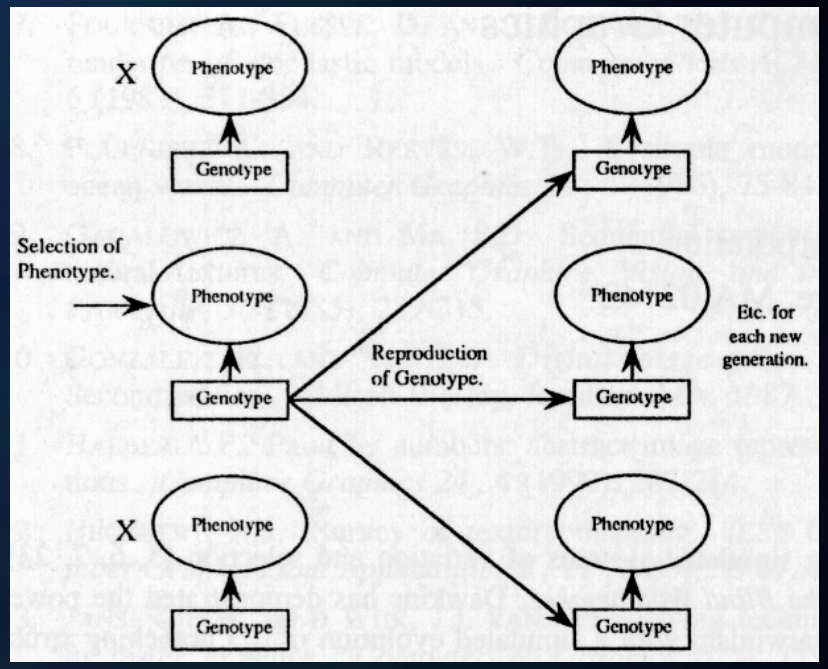


evolved virtual creatures

Genetic Algorithms

web.genarts.com/karl/papers/siggraph91.html

- **Genotype** is the genetic information that codes the creation of an individual
- **Phenotype** is the individual
- **Selection** is the process by which fitness of phenotypes is determined
- **Reproduction** is the process by which new genotypes are generated from existing ones
- There must be probabilistic **mutations** with some frequency



panspermia

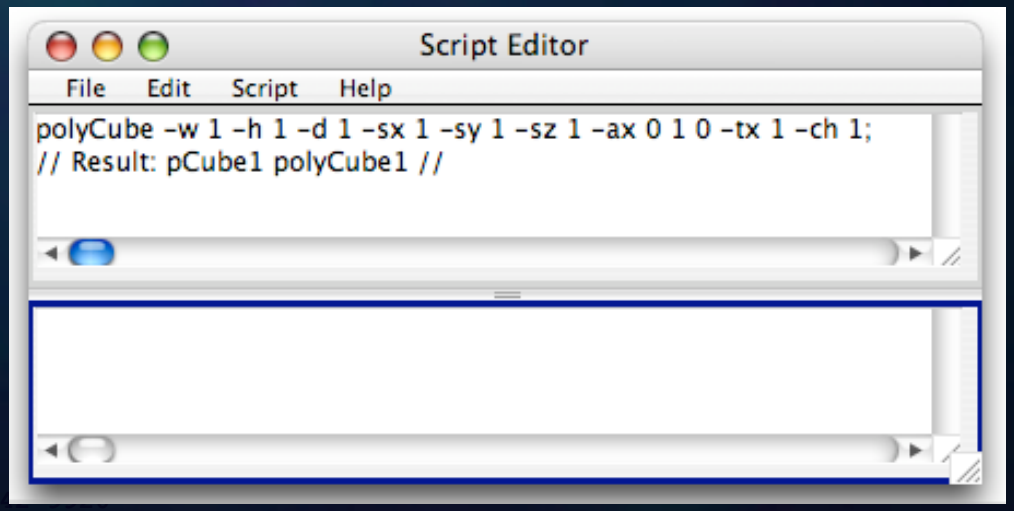
How do we do it in Renderman?

renderman.pixar.com/products/whatsrenderman/
www.cs.berkeley.edu/~ddgarcia/renderman/

- Easy!
- Renderman supports a library that you can compile with your C program that will help you output **RIB**
- RIB is the intermediate file that Renderman reads
- The result of a render is a tiff file (or multiple tiffs)

How do we do it in Maya?

- Easy!
- Every action you perform in Maya is available in the Script Editor
- You can generate this text **explicitly** through a program using the techniques we just discussed. E.g., Name it `Foo.mel`
- Or you can use MEL as a programming language (it looks like C) and do all your scripting there!
- Open this in Maya (double-click, or use the Script Editor)



Conclusion

- There's a wonderful world of procedural geometry waiting for you to discover!
- Let the mathematician in you run wild!
- Take a look at some of the wonderful sculptures by Prof Séquin on the 6th floor for inspiration
- Speak of the artist!!
- Next week:
  - Hair, fur, sets + Pixar guest!