Impact
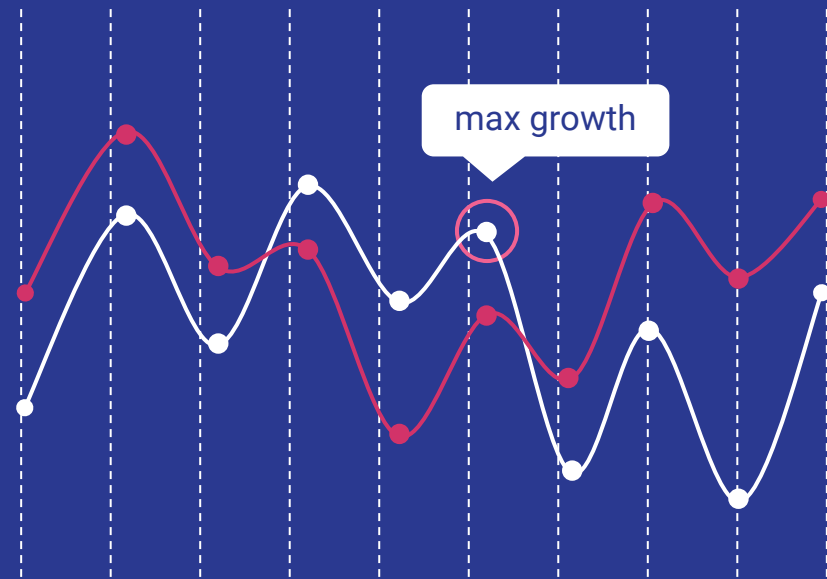
of procedurally-generated terrain

Zachariah Watt

Outline

1. History
2. Implementation
   a. Ontogenetic vs Teleological methods
   b. Value Noise
   c. Perlin Noise
   d. Fourier Synthesis
   e. Diamond Square
   f. Demo: My Program
3. Teleological Methods
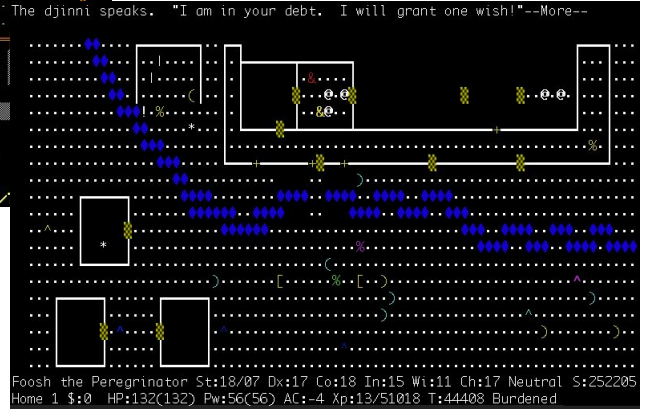   a. Overview
   b. Demo: PlaTec

History

Beneath Apple Manor (1978)

Rogue (1980)

NetHack (1987)

Rogue-like Dungeon Crawlers

Elite (1984)

The Elder Scrolls II: Daggerfall (1996)

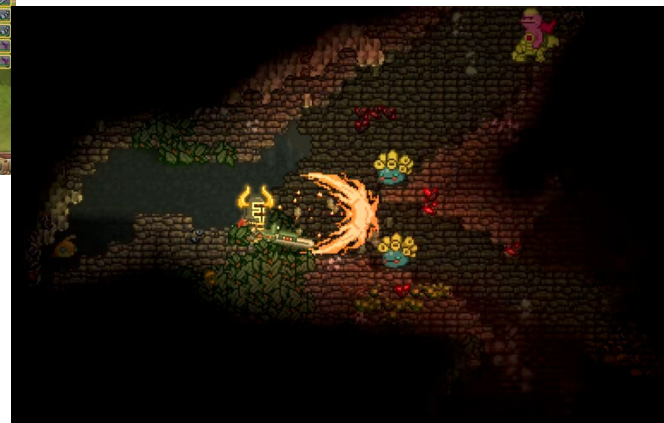Minecraft (2011)

Open World Concepts

LOTR: The Two Towers (2002)

Spore (2008)
Image Credit: WIRED

Starbound (2016)
Image Credit: GameTyrant

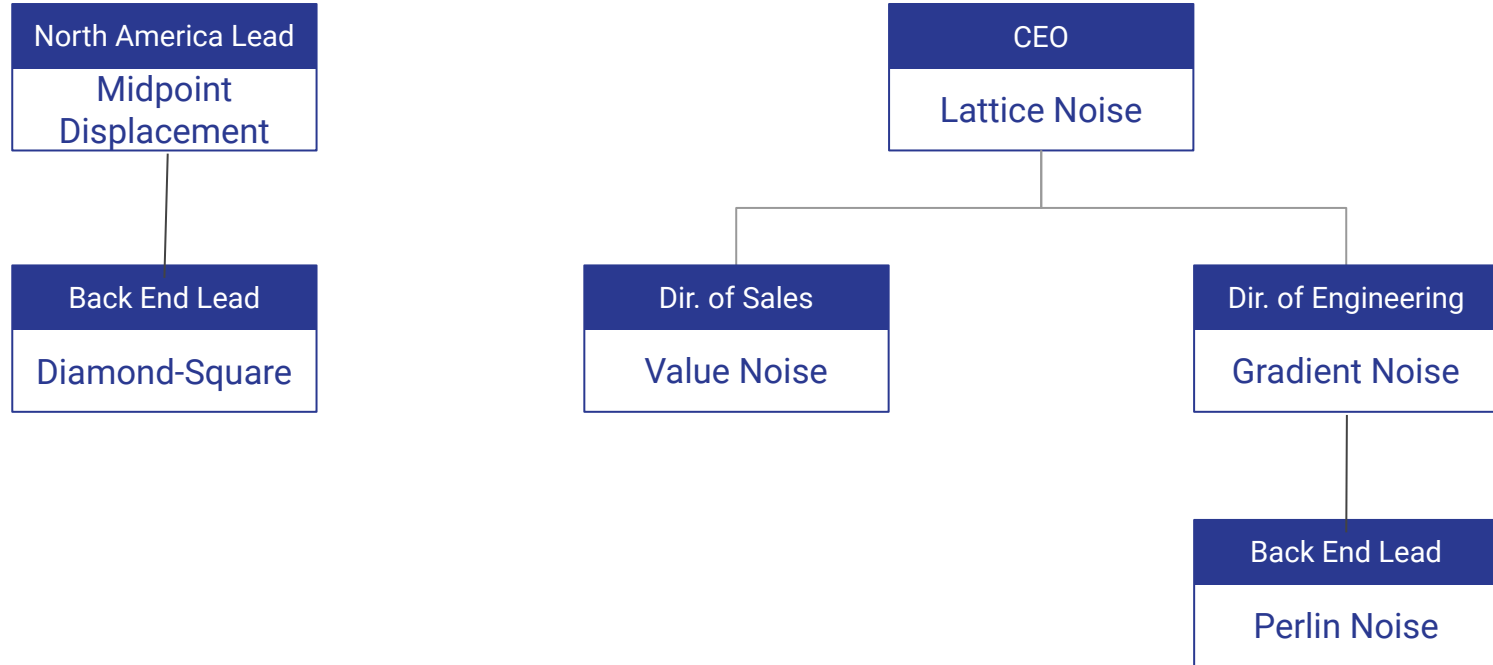More Creativity With Procedural Content Generation

Implementation

Ontogenetic

- Ad hoc: whatever works
- Simple
- Cheap
- Easy algorithms for computer scientists
- Results usually good enough
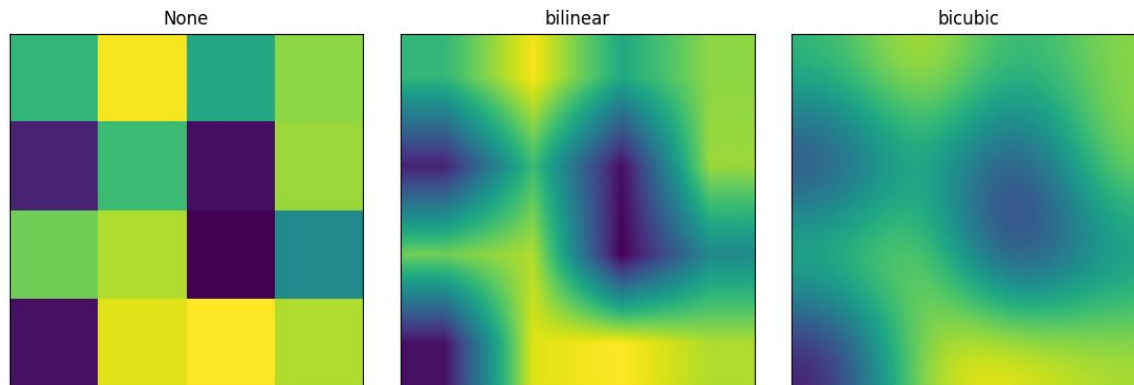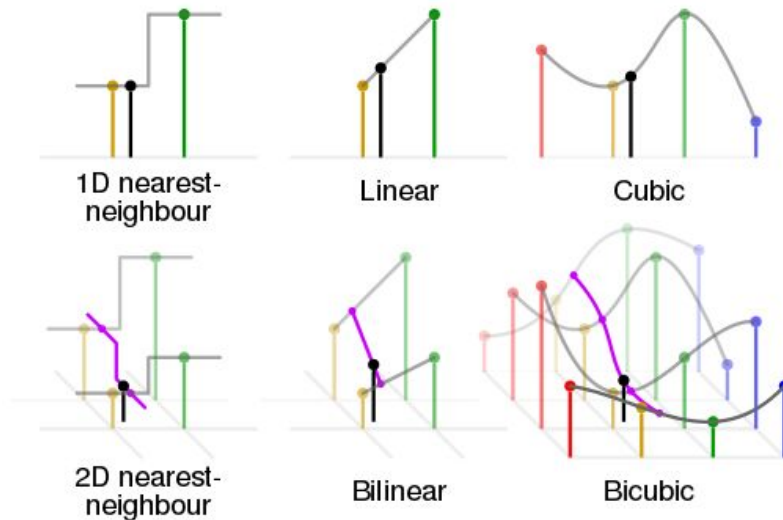- Unnatural artifacts

Teleological

- Based on real-world processes
- Complicated
- Expensive
- Requires research
- Results can be very lifelike

The team (of ontogenetic algorithms)

North America Lead

Midpoint Displacement

Back End Lead

Diamond-Square

CEO

Lattice Noise

Dir. of Sales

Value Noise

Dir. of Engineering

Gradient Noise

Back End Lead

Perlin Noise
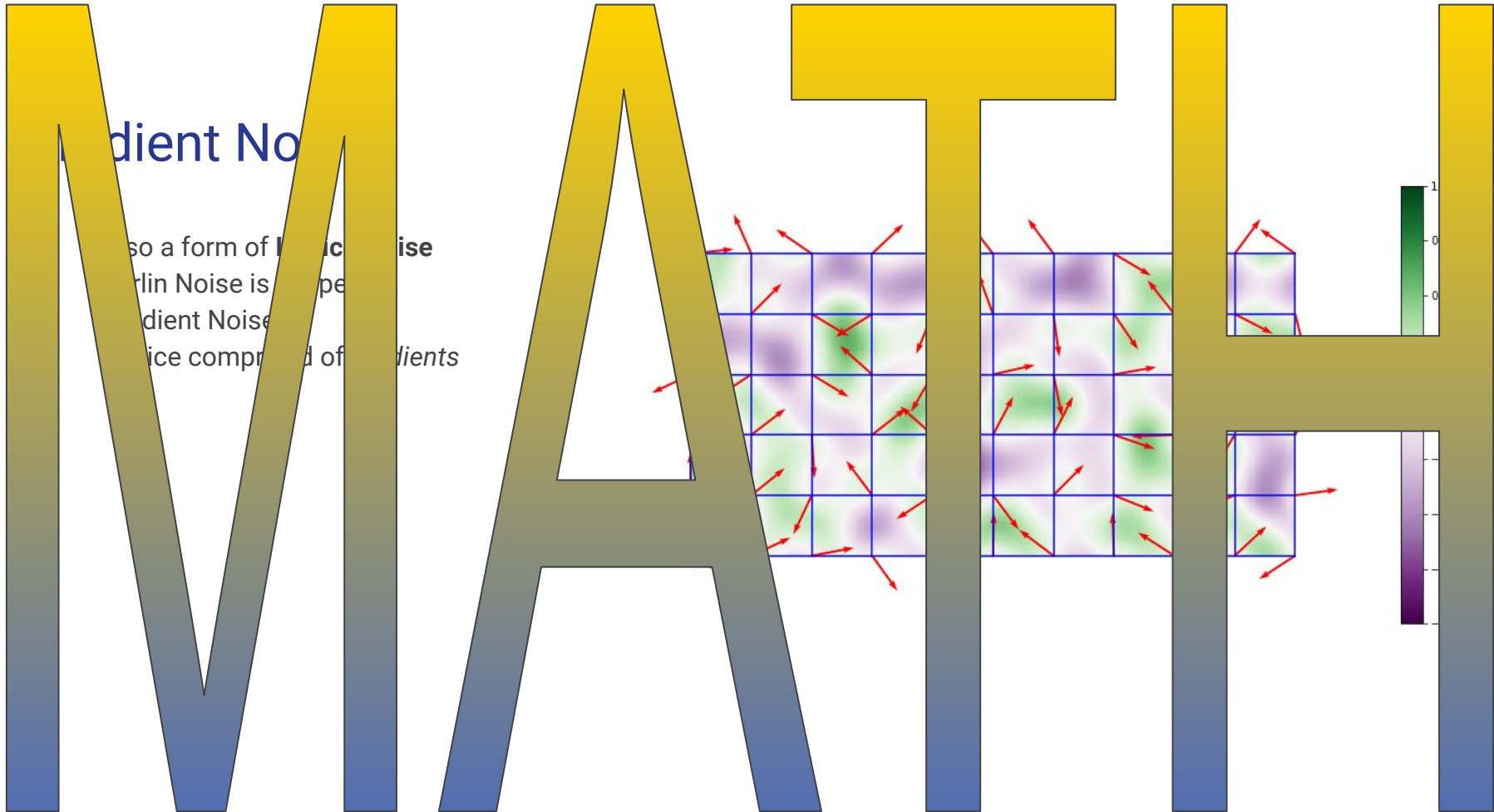
Value Noise

- A **Lattice Noise** algorithm
- Superimposes a lattice over the heightmap
- Random heights for each lattice point
- Simple interpolation for all pixels

1D nearest-neighbour

Linear

Cubic

2D nearest-neighbour

Bilinear

Bicubic

None

bilinear

bicubic

dient No

- so a form of **...ic ...ise**
- ...rlin Noise is ...pe
- ...dient Noise
- ...ice compr...d of *...dients*

Fourier Synthesis

- "Hey, this all decomposes to a linear combination of trigonometric functions anyway"
- Gives developer control
- Simple
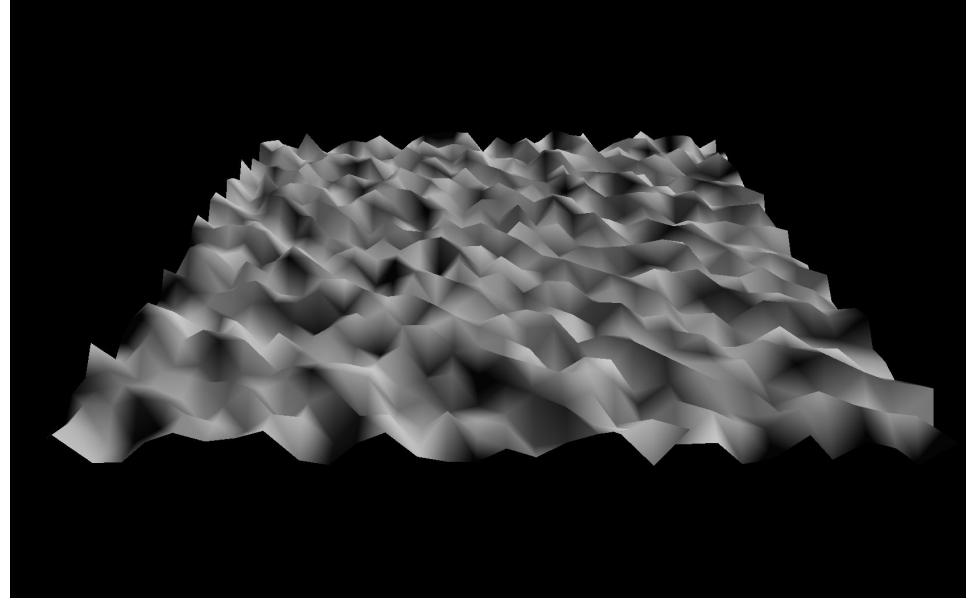
NOT Fractal

- D ≈ 2
- *Fractal Noise* is a composite of noises
- Approximation of a fractal

Freq=16
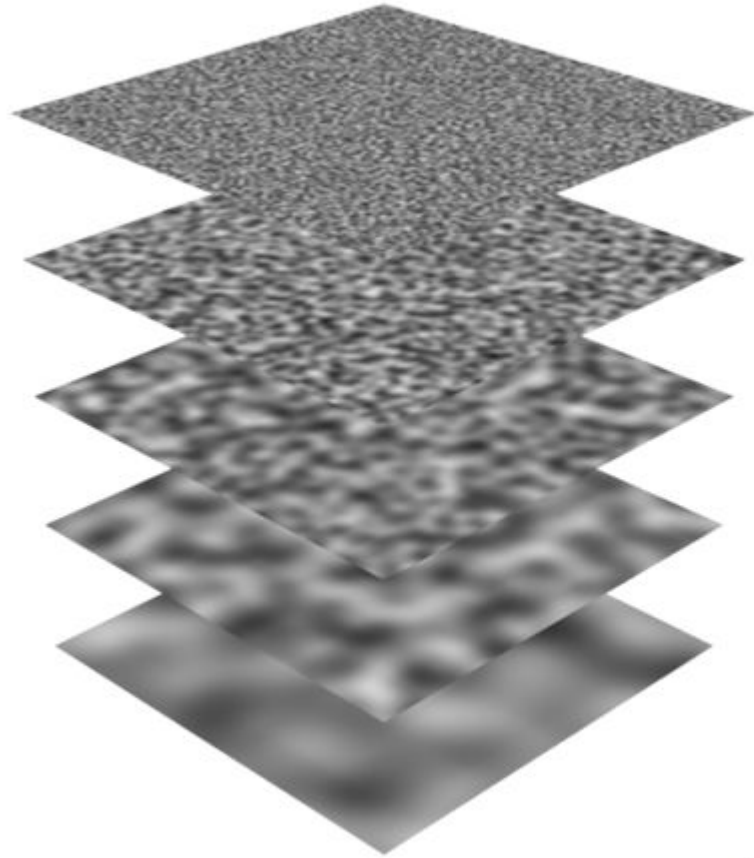Scale=0.0625
    +
Freq=8
Scale=0.125
    +
Freq=4
Scale=0.25
    +
Freq=2
Scale=0.5
    +
Freq=1
Scale=1

Diamond Square

- Based on the Midpoint Displacement Algorithm
- Advantage: fractal
- Disadvantage: fractal
- Advantage: Easy
- Disadvantage: Memory

y_0 y_n

$y_{n/2}$

$i = 0$

$y_{n/4}$

$i = 1$

$y_{3n/4}$

$y_{3n/8}$

$y_{n/8}$

$i = 2$

$y_{7n/8}$

$y_{5n/8}$

Initialize corner values

Perform diamond step

Perform square step

Perform diamond step

Perform square step

Image Credit: 14mul8.wordpress.com
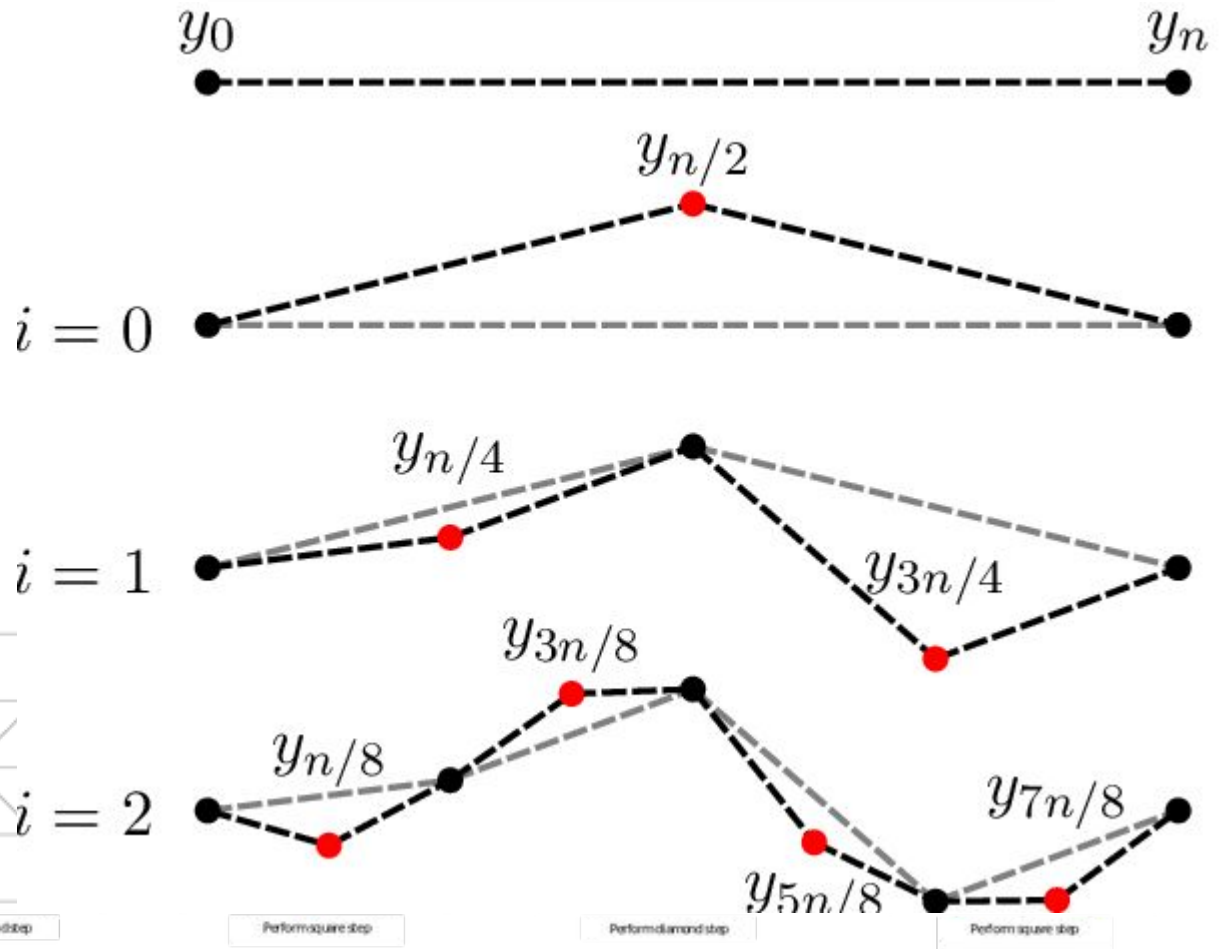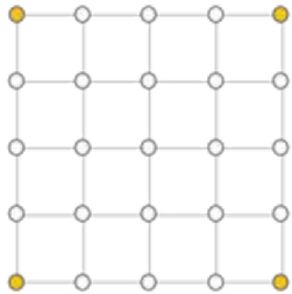
My Crappy Program

Teleological Methods

Artifacts

- Topographic maxima and minima placed strangely
- Most algorithms are uniform; real landscapes are not
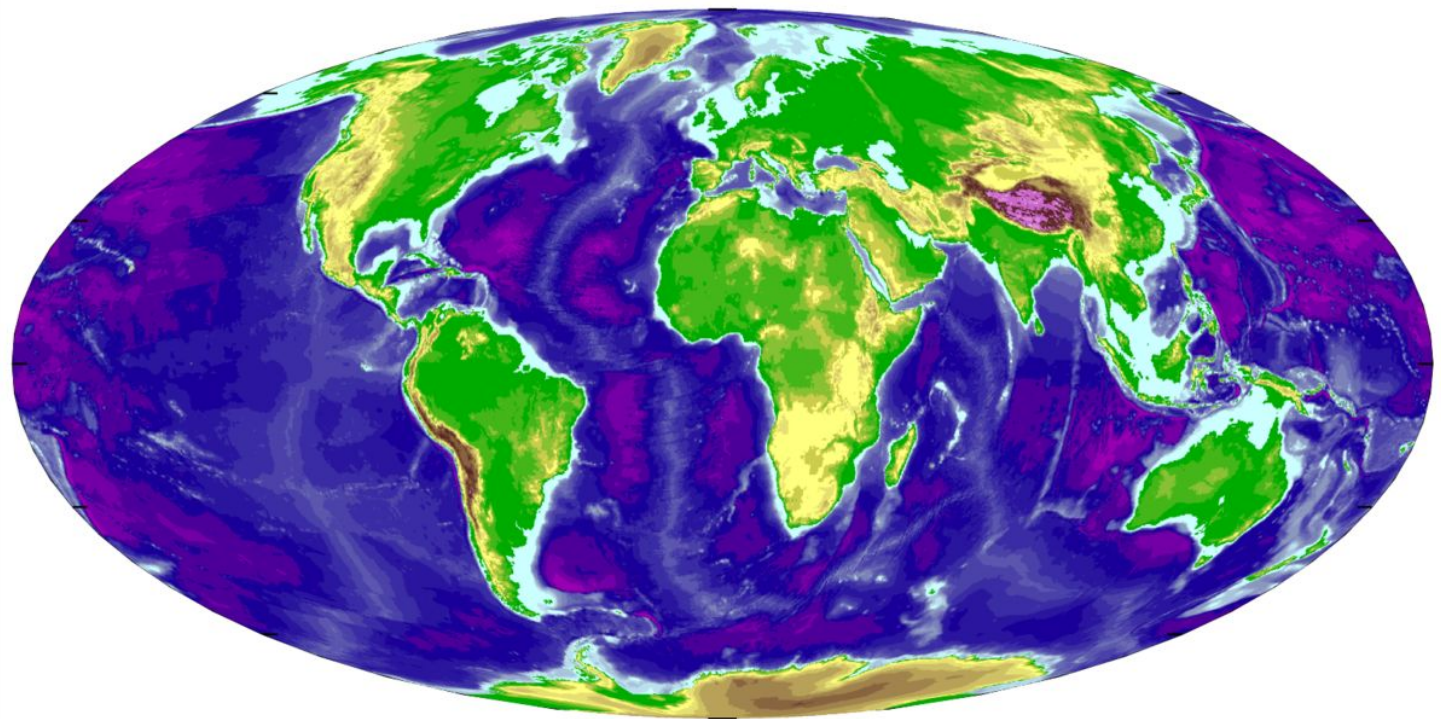- Creases, and other unnatural phenomena

Absence of Natural Features

- Hydraulic (drainage basins, lakes, alluvial fans)
- Diachronic (mesas, canyons)
- Eolian (arches)
- Karst (caves)

桂林山水甲天下

Present-day Earth topography [m]
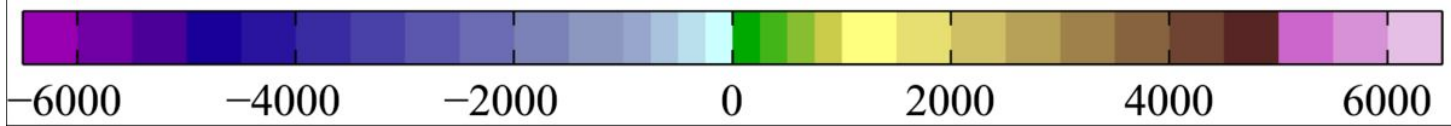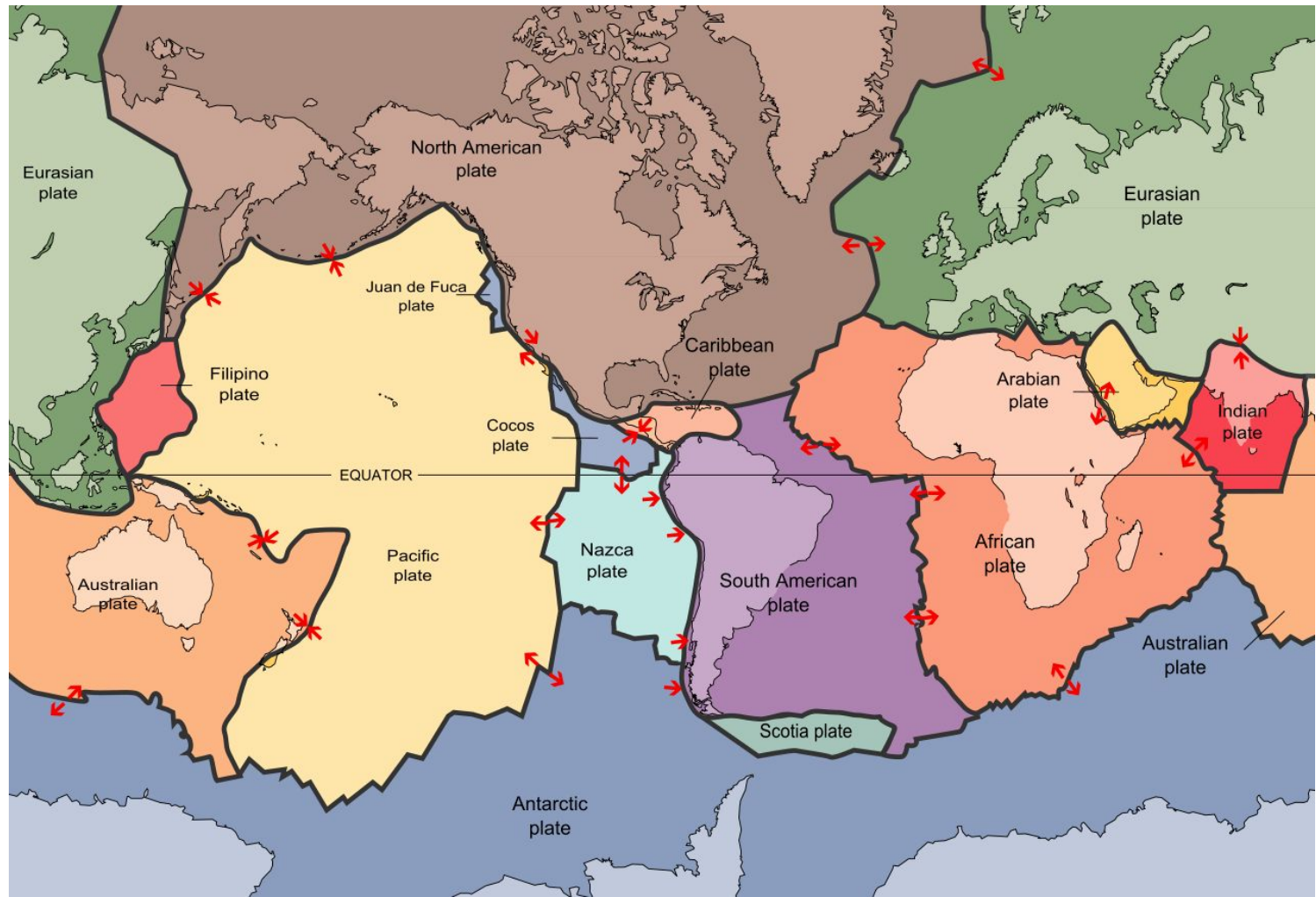
Physically-Based Terrain Generation

Continental Features ➤

Plate Tectonics

- Determines location of mountain ranges, rift valleys
- Shape of landmasses
- Distribution of karsts

Large Features ➤

- "Hot Spots"
- Tectonic Orogenics

Small Features ➤

- Streams
- Lakes
- Karst Features

PlaTec

lithosphere

- hmap : float*
- plates : plate**
- collisions : vector< vector<plateCollision> >
- subductions : vector< vector<plateCollision> >

+ reset() : void
+ update() : void
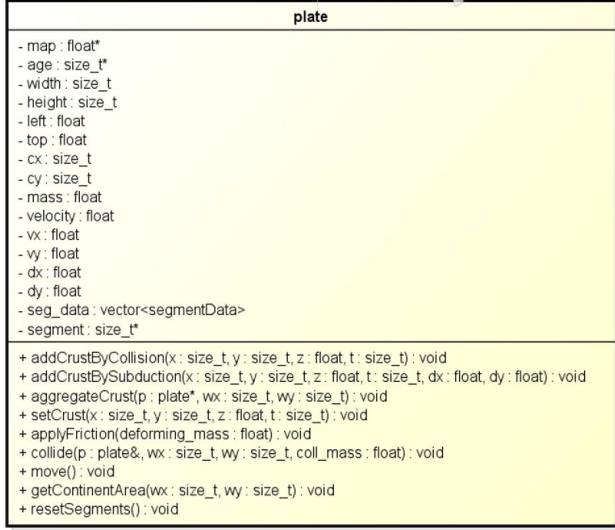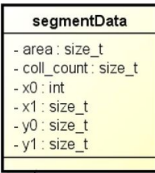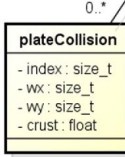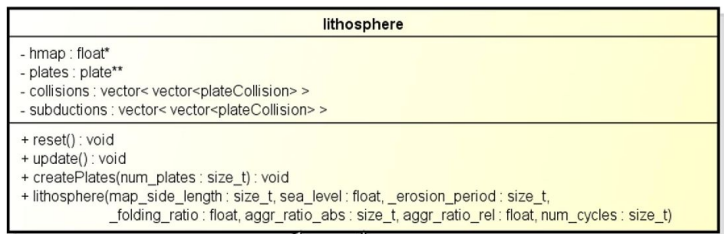+ createPlates(num_plates : size_t) : void
+ lithosphere(map_side_length : size_t, sea_level : float, _erosion_period : size_t,
          _folding_ratio : float, aggr_ratio_abs : size_t, aggr_ratio_rel : float, num_cycles : size_t)

1
0..*
1

plateCollision

- index : size_t
- wx : size_t
- wy : size_t
- crust : float

segmentData

- area : size_t
- coll_count : size_t
- x0 : int
- x1 : size_t
- y0 : size_t
- y1 : size_t

0..*
1
0..*

plate

- map : float*
- age : size_t*
- width : size_t
- height : size_t
- left : float
- top : float
- cx : size_t
- cy : size_t
- mass : float
- velocity : float
- vx : float
- vy : float
- dx : float
- dy : float
- seg_data : vector<segmentData>
- segment : size_t*

+ addCrustByCollision(x : size_t, y : size_t, z : float, t : size_t) : void
+ addCrustBySubduction(x : size_t, y : size_t, z : float, t : size_t, dx : float, dy : float) : void
+ aggregateCrust(p : plate*, wx : size_t, wy : size_t) : void
+ setCrust(x : size_t, y : size_t, z : float, t : size_t) : void
+ applyFriction(deforming_mass : float) : void
+ collide(p : plate&, wx : size_t, wy : size_t, coll_mass : float) : void
+ move() : void
+ getContinentArea(wx : size_t, wy : size_t) : void
+ resetSegments() : void

References / Further Reading

- "Procedural Content Generation in Games" by Noor Shaker et al. (2016)
- "Physically Based Terrain Generation" by Lauri Viitanen (2012)
- "Procedurally Generating Terrain" by Travis Archer (2011)
- "Procedural Generation: An Algorithmic Analysis of Video Game Design and Level Creation" by Logan Bond (2017)