Procedural Generation

Kaarel Tõnisson

2018-04-20

Contents

- Procedural generation overview
  - Development-time generation
  - Execution-time generation
- Noise-based techniques
  - Perlin noise
  - Simplex noise
- Synthesis-based techniques
  - Tiling
  - Image quilting
  - Deep learning
- Procedural content generation
  - Early history
- Case study: Minecraft

What is procedural generation?

- Method of creating something algorithmically
  - As opposed to creating something manually
- Few inputs can generate many different outputs
  - One seed number can generate a unique world

Where can procedural generation be used?

In theory

- Every field of creative development
  - Textures
  - Models (characters, trees, equipment)
  - Worlds (terrain geometry, object placement)
  - Item parameters
  - Stories and history
  - Sound effects and music

Where is it actually used?

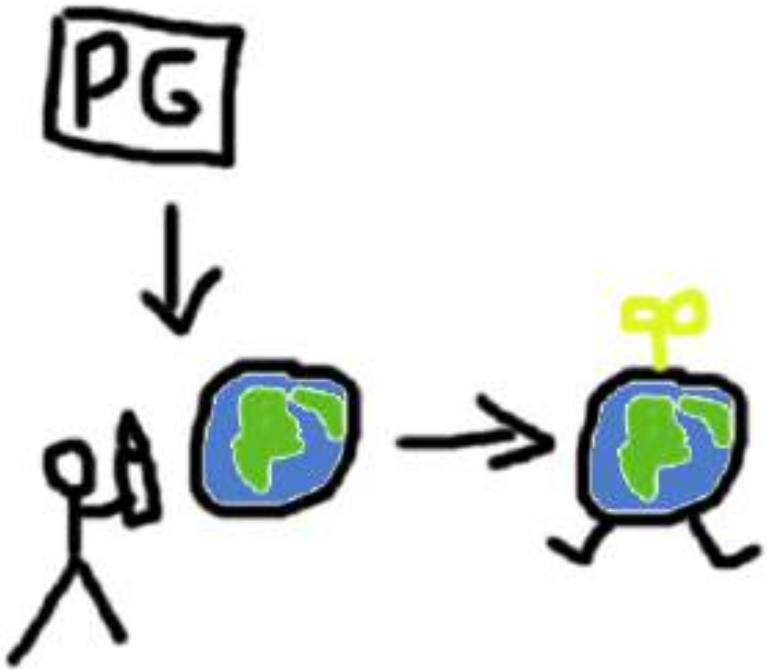
In practice

- *Often:* Extent is limited
  - Most assets are made by hand
  - Procedural parts are edited by hand
- *Sometimes*: Heavily reliant on procedural generation
  - Certain games
  - Size-limit challenges

When is generation performed? (Option 1)

During development

- Asset is generated, then enhanced by hand
- Examples:
  - Algorithm generates terrain, developer adds objects and detail
  - Algorithm generates basic texture, developer adds detail
- Used in games, movies, images



What is development-time procedural generation good for?

- Hand-crafting assets requires extensive work
- Allows developers to focus on important areas and details
  - Not every tree has to be made by hand
- Reduced number of manual mistakes

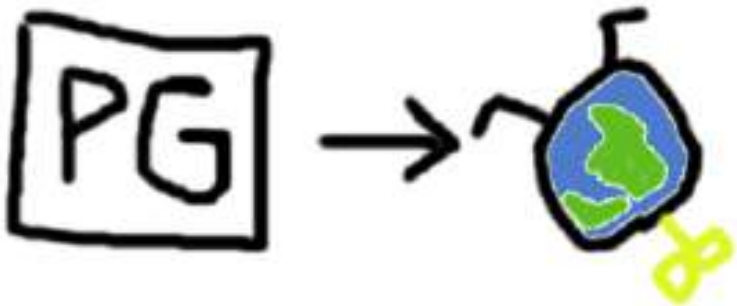
What are the drawbacks of development-time generation?

- Algorithm development can be difficult
  - May be easier to create assets by hand
- Using the algorithm may require skill
  - The user may require additional training
- Algorithm has to produce *desirable* results
  - Hand-made assets are *nicer*

When does generation take place? (Option 2)

During execution

- Generation happens when the program is executed by the end-user
- Results are not edited by hand
- Examples:
  - Generate a procedural landscape when the game is first started
  - Generate procedural objects when a treasure chest is opened
- Used in video games
- Mostly **procedural content generation**



What is execution-time generation good for?

- Smaller *initial* installation
- Added variance and replayability
- Enables emergent events
  - Something exciting happens as a coincidence

What are the drawbacks of execution-time generation?

- Less attractive than human-made assets
  - Generated textures can look *ugly*
  - Generated worlds can feel *boring*
- Heavy focus on algorithm development
- Asset generation requires storage space and computation time

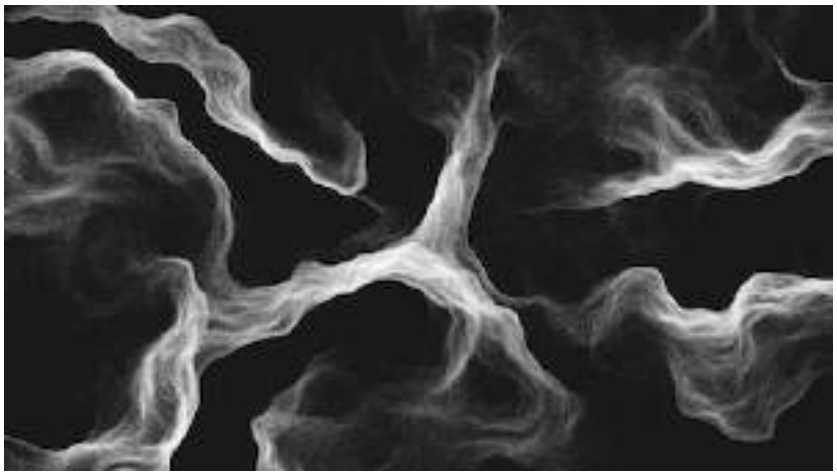
Procedural generation approaches

- Noise-based techniques
  - Perlin noise
  - Simplex noise
- Synthesis-based techniques
  - Tiling
  - Image quilting
  - Deep learning

(This list of techniques is not exhaustive)

Noise-based techniques

- *Idea*: Generate assets from randomness
- *Challenge*: Pure randomness is not appealing
- *Several uses*: terrain, fog, clouds, skies, certain textures

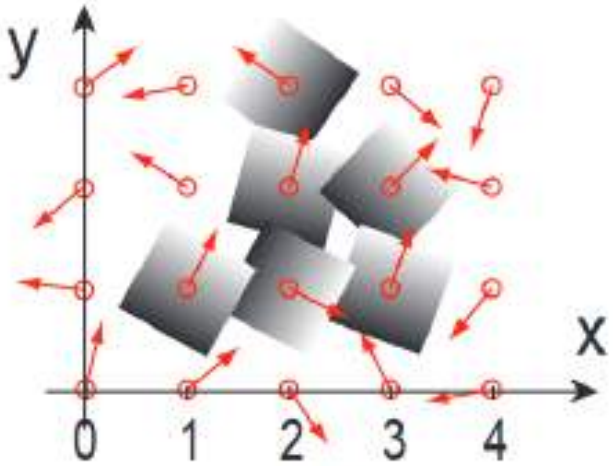


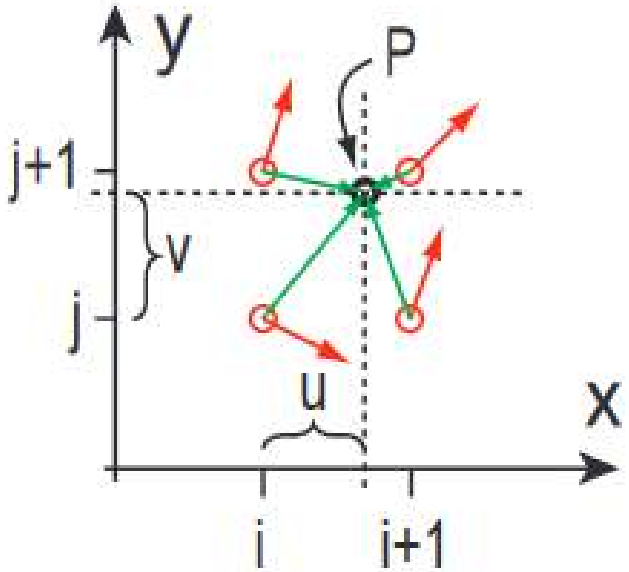
Perlin noise

- *Idea*: Interpolate a smooth function from randomly generated gradients in a grid
- Developed by Ken Perlin in 1983
- 2^n time complexity for n dimensions

Mechanism

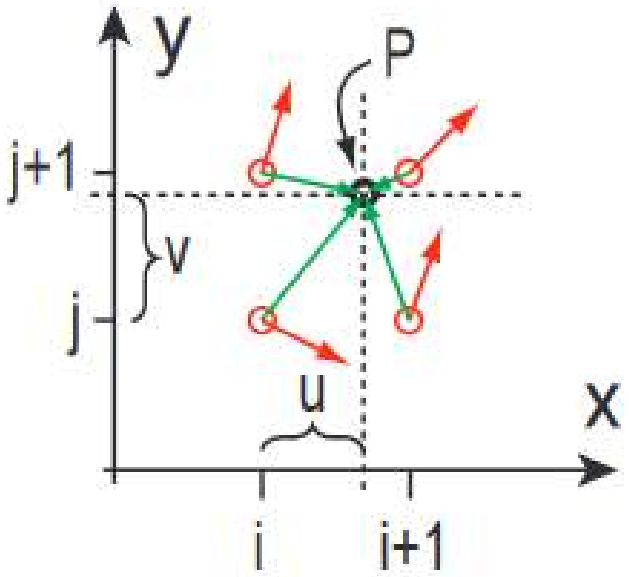
- Generate grid with a random gradient (vector) at each node
- To calculate value at point P, find vectors to nearest nodes

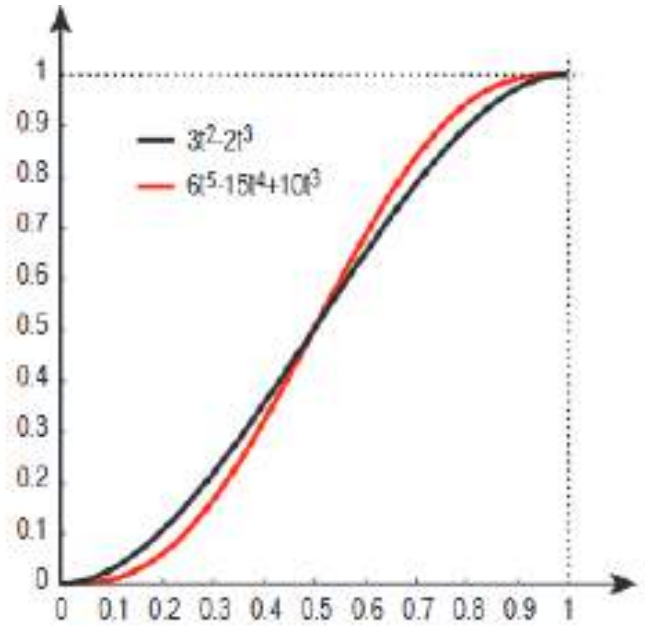




Perlin noise cont.

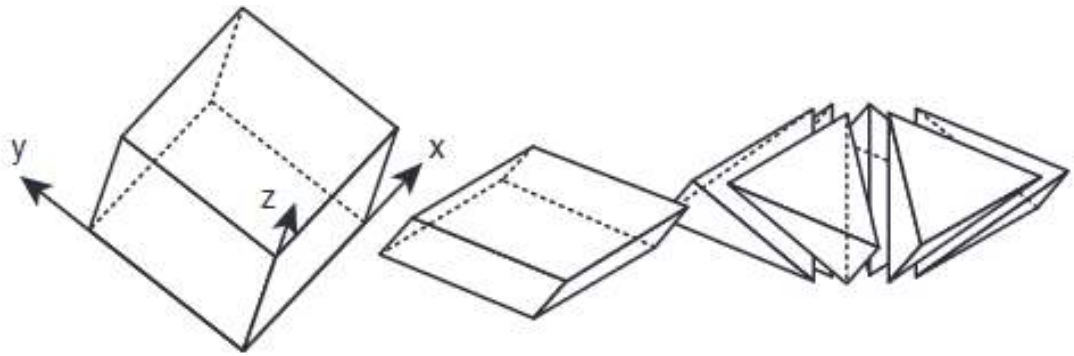
- For each nearest node, calculate dot product between distance vector and gradient vector
- Blend the noise contribution for each node using the curve: $f(t) = 6t^5 - 15t^4 + 10t^3$
- $n_{x0} = n_{00}(1 - f(x - i)) + n_{10}f(x - i)$
- $n_{x1} = n_{01}(1 - f(x - i)) + n_{11}f(x - i)$
- $n_{xy} = n_{x0}(1 - f(y - j)) + n_{x1}f(y - j)$





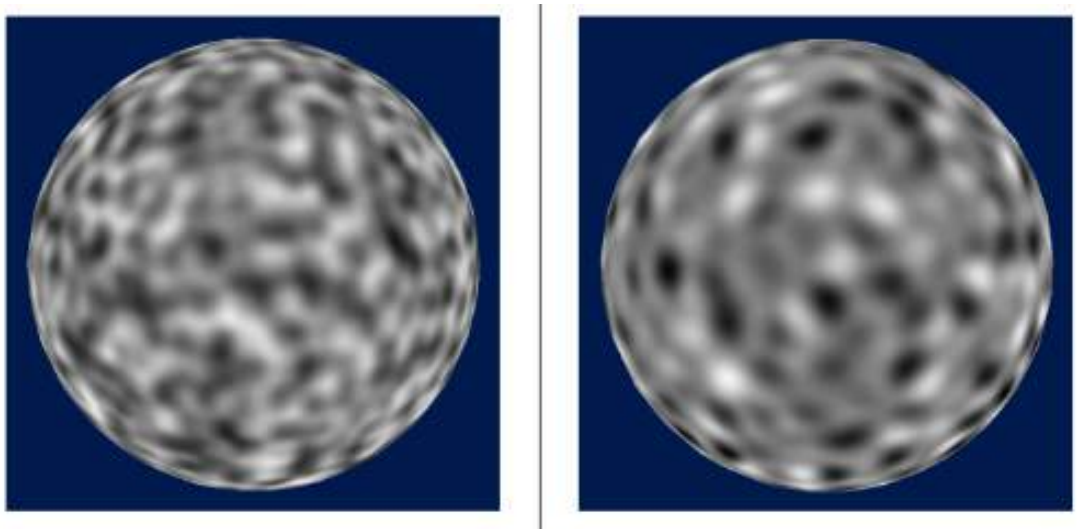
Simplex noise

- *Idea*: Use simplex shapes instead of points
- Also by Ken Perlin
- n^2 time complexity for n dimensions
- Better higher-dimension scaling than Perlin noise
- Fewer artifacts than Perlin noise
- Patented
  - OpenSimplex noise is a free alternative



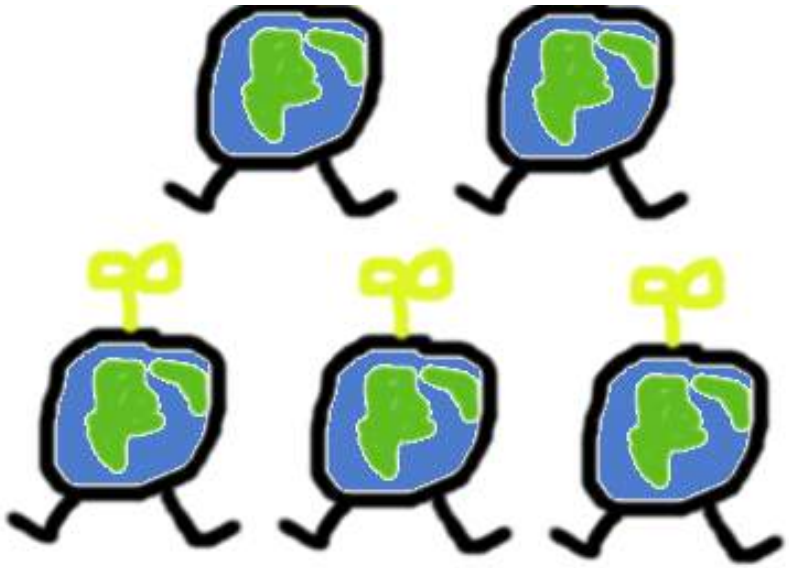
Simplex noise cont.

- An *n*-dimensional simplex has *n+1* corners
- For a point inside a simplex, each corner contributes according to distance function
- 2D example of simplex noise usage:
  `https://codepen.io/jwagner/pen/BNmpdm?editors=001`
- 3D example of simplex noise usage:
  `https://29a.ch/sandbox/2012/voxelworld/`



Synthesis-based techniques

- A texture is an array of colored pixels
  - The number of pixels is finite
  - Can't enlarge without losing detail
- *Idea*: Manipulate small sample textures to create large textures
- *Also possible*: Manipulate other assets to create more assets

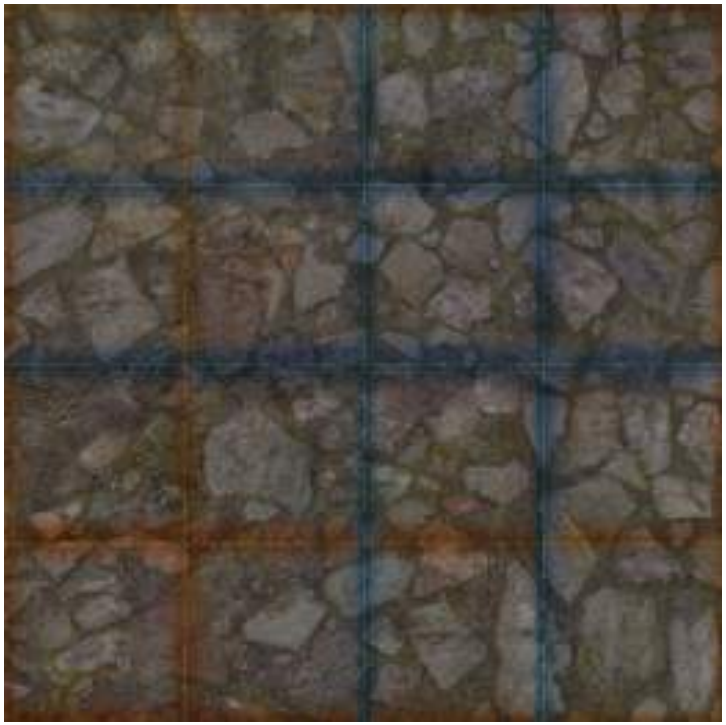


Tiling

- *Idea*: Replicate one texture sample until the area is filled
- Can have noticeable tile edges and repetition
- Cheap to compute
- Simple to implement

Better tiling

- *Improved idea*: Tile different samples with randomized locations
- *Requirement*: Each edge must match edge of another sample
- Example: blue edges match blues, orange edges match oranges
- Requires more samples than simple tiling



`http://www.pathofexile.com/forum/view-thread/55091`

Image quilting

- *Idea*: Take small blocks from input texture, paste to output *with overlap*
- Required parameters: block size, overlap size
- More natural outcome than tiling
- Requires more configuring than tiling

Image quilting cont.

Mechanism

- Go through target image in steps of one block (minus overlap)
- For each target block, search input for blocks which satisfy overlap constraints, pick one at random
- Find overlap location with minimum overlap error
- Paste new block at the location

Deep learning

- *Idea*: Teach machine learning algorithms how to create assets
- Potentially excellent results
- Difficult to control

BETHGE LAB `http://bethgelab.org/deeptextures/`

Procedural content generation

- Includes procedural assets which directly influence gameplay
  - Can include terrain, objects, characters, events

Early procedural content generation

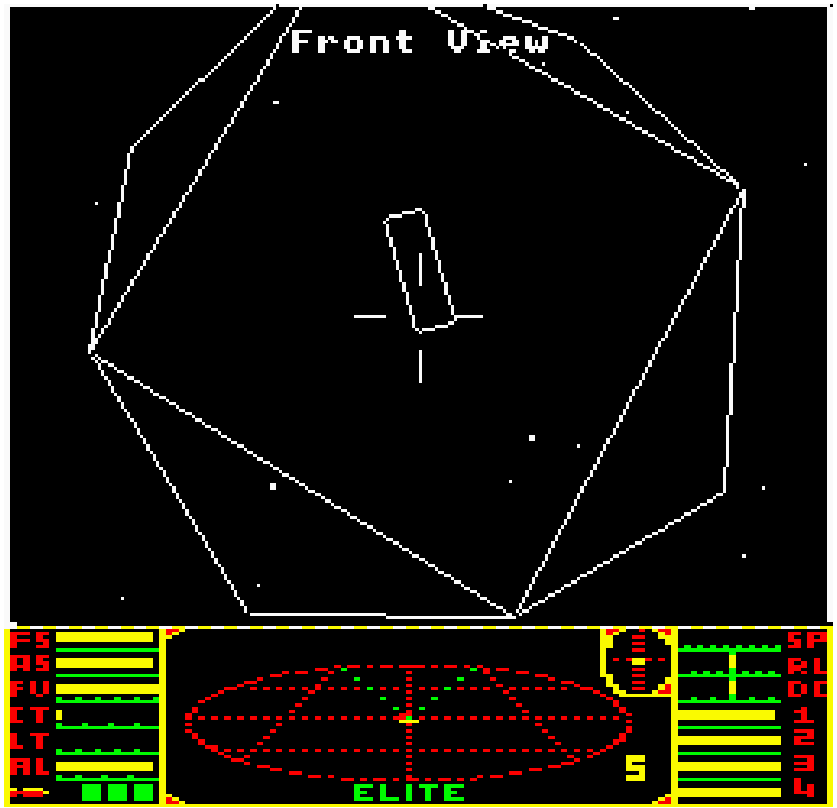
Early games: Beneath Apple Manor(1978), Rogue(1980)

- Procedural world because of memory limits
- Not enough space to include the world with the game

Early procedural content generation cont.

Elite(1984): seed-based world generation

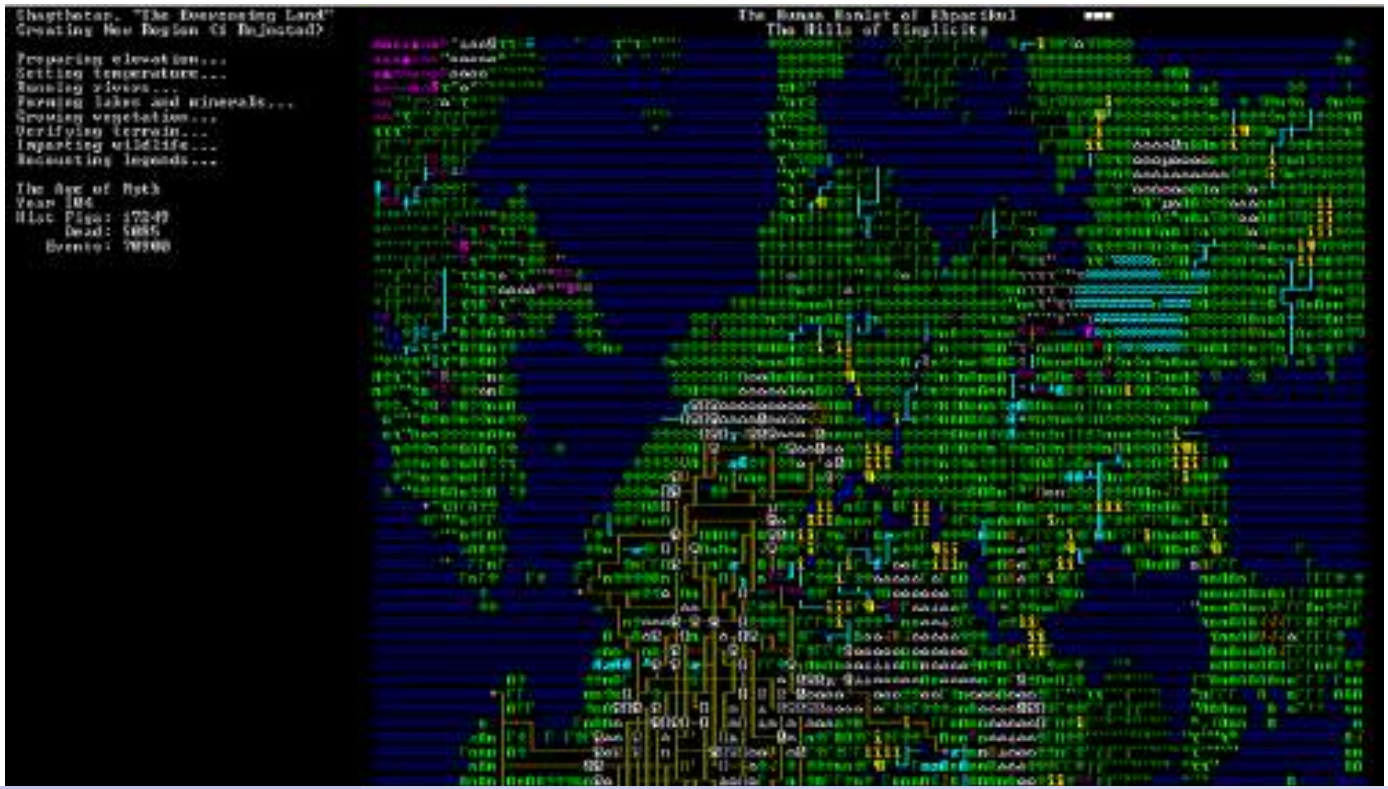
- Fixed seeds
- 8 galaxies with 256 planets each
- Planet properties (trade, inhabitants) are all generated from the seed
- Intentionally small number of galaxies to hide artificiality



Contemporary procedural generation

Dwarf fortress(2006-): procedurally generated world with history

- World generation includes several hundred years of generated history
- World generation takes several (tens of) minutes
- Many world generation options
- The generator may reject the world in the progress and start over
- Generates continents, nations, terrain, individuals and all history



Case study: Minecraft(2009-)

- Adventure/sandbox game
- Heavily procedural world generation
- World is composed of blocks
  - Blocks exist in a discrete 3D grid
  - Players and entities can move in continuous space

Minecraft worlds

- A world is potentially infinite
  - World height is limited (no blocks exist below or above certain heights)
- 30+ main biome types
  - Each biome has certain properties
    - Pre-determined block types (ground cover, plants)
    - Pre-determined rainfall and temperature values
    - Category (snow-covered, cold, medium, dry/warm, neutral)
- Transition biomes (jungle edge, extreme hills edge, river) to improve appearance of biome edges

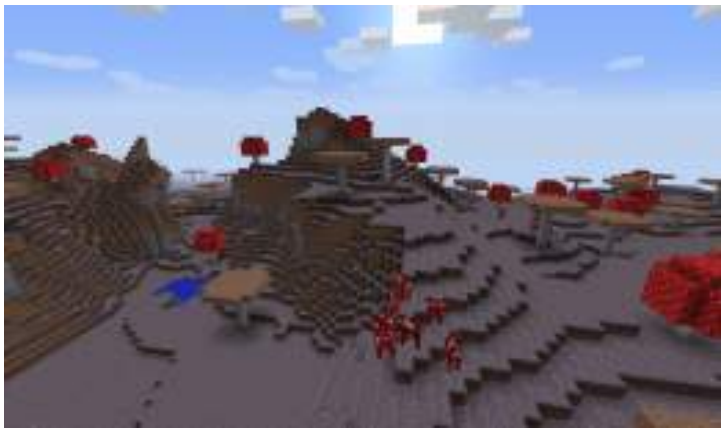
Minecraft colors

- Grass, foliage, water, and sky color are determined by rainfall and temperature
- Rainfall and temperature are adjusted by height
- Grass and foliage sampled from a gradient colormap
- Some biomes use other coloration
  - Swamplands: Perlin noise is used for small temperature variation
  - Roofed forest: grass color sample is averaged with a dark green constant

Minecraft world generation algorithm

World generation algorithm is not exactly known

- *Early versions*: 2D Perlin noise
  - Boring outcomes
  - No overhangs
- *Later*: 3D Perlin noise
  - Value less than 0 is air, greater than 0 is ground
  - Higher computational cost
    - *Solution*: Generate some points, interpolate others
- *Now*: possibly another algorithm



World generation rules

- World is loaded in chunks
  - Only a small number of chunks around the player are kept loaded
  - If a chunk does not exist, it is generated immediately
  - Generated chunks are saved to disk
- Biomes from distant categories are not generated next to each other
- Secondary features are generated after main terrain
  - *Caves and ravines*: fully procedural
  - *Villages, strongholds, mineshafts*: prefabricated components with randomized layouts

Chunk-based caveats

Chunk-based generation strengths

- Allows world to persist
- Initial world takes little space
- New worlds can be created quickly

Chunk-based generation weaknesses

- World size can increase to several gigabytes
- Chunk generation can be slower than player movement
- Chunk generation can hinder server performance

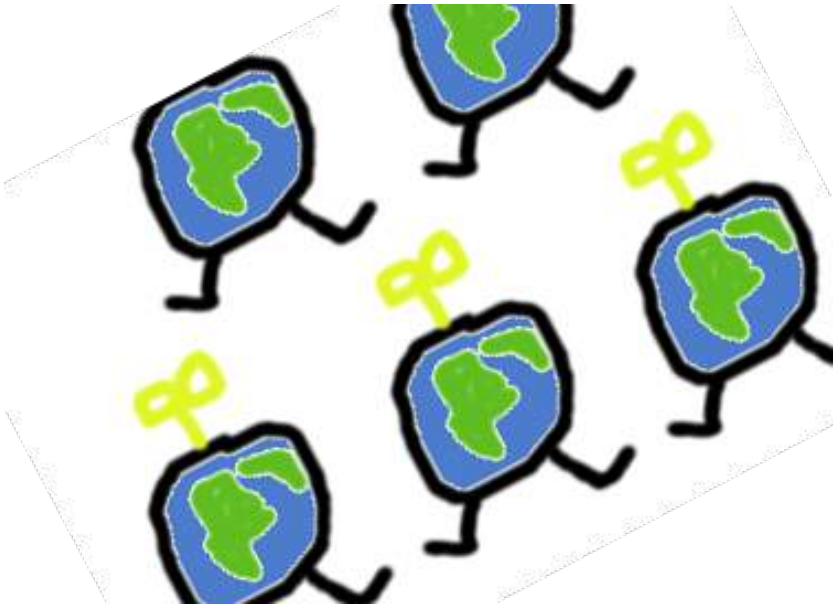
Generation caveats

- Secondary feature generation may create anomalies
  - houses without floors
  - doors too high or underground
- Discovering necessary resources can be bothersome

To sum up

- There are numerous methods of procedural generation
- Noise-based methods create from randomness
- Synthesis-based methods create from existing assets
- Asset generation can reduce human workloads
- Algorithm design can increase human workloads
- Computers do not have a sense of beauty (yet)

Thank you for listening!



References

Minecraft Wiki: Biomes https://minecraft.gamepedia.com/Biome
http://people.wku.edu/qi.li/teaching/446/cg13_texturing.pdf
J. Freiknecht, W. Effelsberg: A Survey on the Procedural Generation of Virtual Worldswww.mdpi.com/2414-4088/1/4/27/pdf
Path of Exile Dev Diary: Tile-Based Texture Maps for Games http://www.pathofexile.com/forum/view-thread/55091
H-Immersion (64K intro competition) http://www.ctrl-alt-test.fr/2018/a-dive-into-the-making-of-immersion/
Kotaku: No Man's Sky creature generation
https://kotaku.com/a-look-at-how-no-mans-skys-procedural-generation-works-1787928446
Graphcut algorithm https://www.cc.gatech.edu/cpl/projects/graphcuttextures/
B. Kachscovski, Interactive Methods for Procedural Texture Generation with Noise
http://www.diva-portal.org/smash/get/diva2:839579/FULLTEXT01.pdf
S. Gustavson: Simplex Noise Demystified http://staffwww.itn.liu.se/~stegu/simplexnoise/simplexnoise.pdf
necessarydisorder blog: GIF loop tutorialshttps://necessarydisorder.wordpress.com/
A.A. Efros, W.T. Freeman: Image Quilting for Texture Synthesis and Transfer
https://people.eecs.berkeley.edu/~efros/research/quilting/quilting.pdf
The Guardian: Masters of their universe https://www.theguardian.com/books/2003/oct/18/features.weekend
The Word of Notch blog: Minecraft Terrain generation
https://notch.tumblr.com/post/3746989361/terrain-generation-part-1

Image Sources

https://minecraft.gamepedia.com/File:Swampland_M.png http://www.bay12games.com/dwarves/screens.html
https://upload.wikimedia.org/wikipedia/commons/b/bc/Imagequilting.gif
https://blendercgtips.files.wordpress.com/2014/12/fog2.png?w=672&h=372&crop=1
http://genekogan.com/code/p5js-perlin-noise/
https://upload.wikimedia.org/wikipedia/commons/1/17/Rogue_Screen_Shot_CAR.PNG
http://mcpedl.com/50975-3/
https://www.rollapp.com/app/dwarffortress