

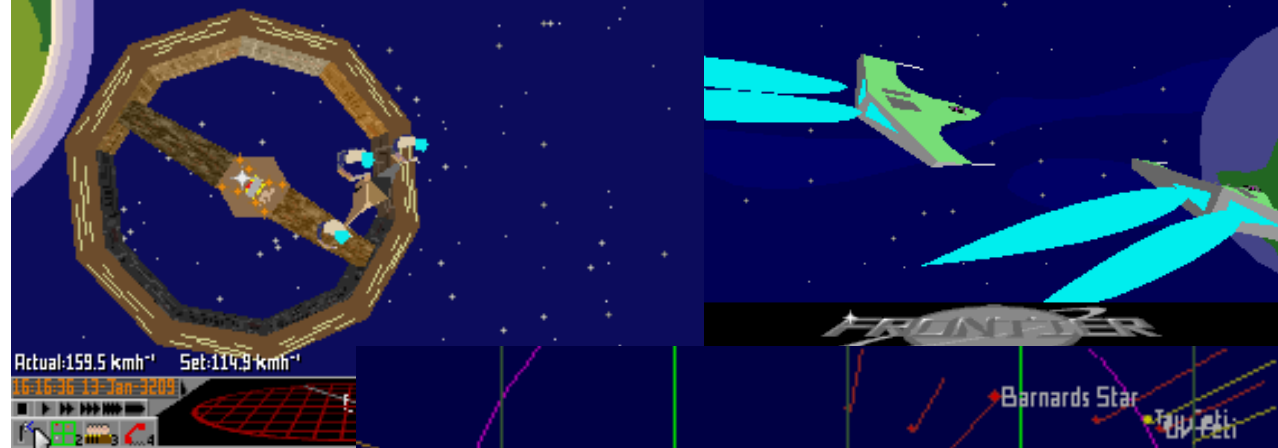
Procedural Content Generation

Paweł Marczewski

Frontier (1993)

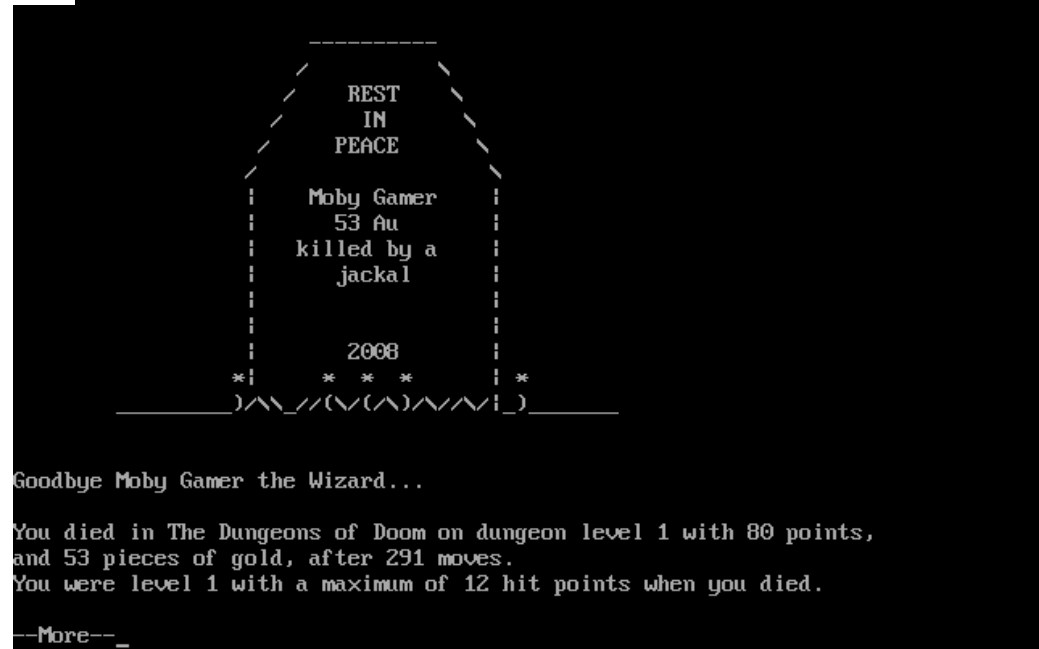
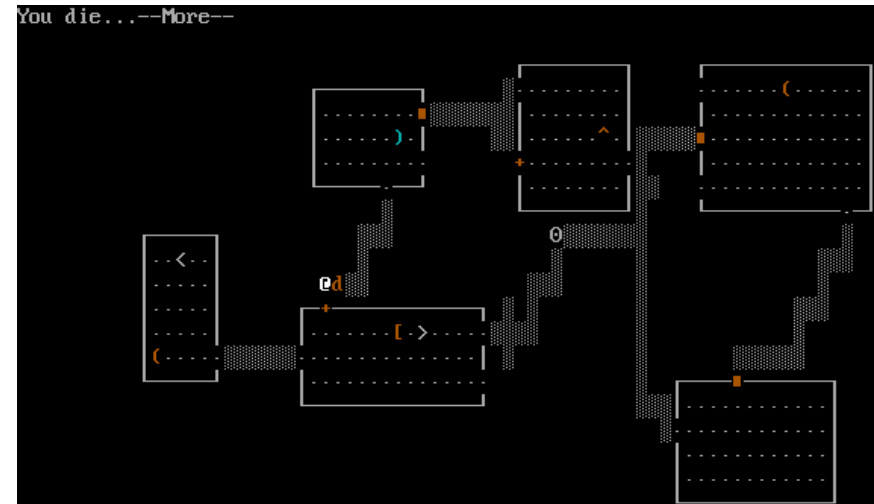
- 10 billion planets on a floppy disk!
- Procedurally generated stars, planets, moons, human bases, resources

see also [Noctis](#)



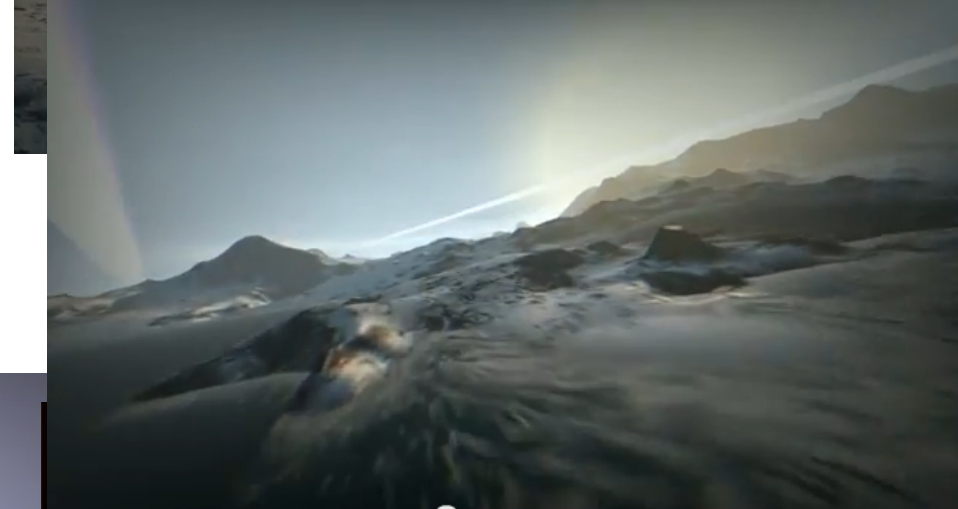
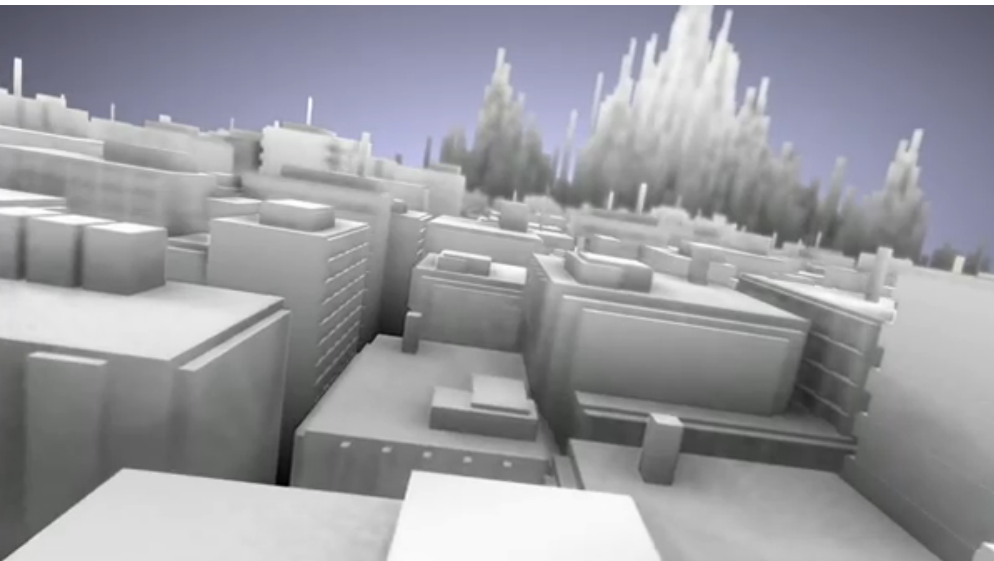
Nethack (and other roguelikes)

- You're going to die *a lot*...
- A new dungeon every time
- Easy to adjust the generator to allow for more variety (e.g. cities, mines, special rooms)



Demoscene

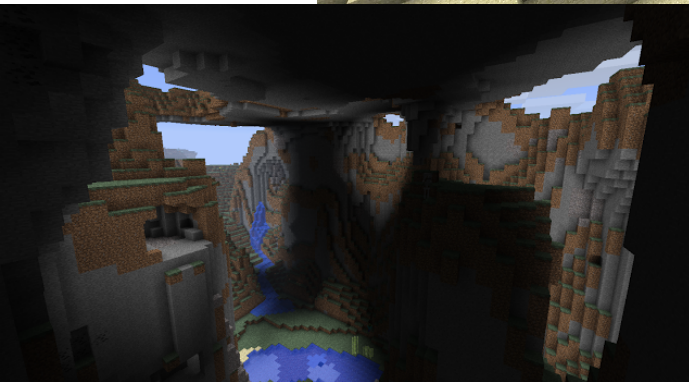
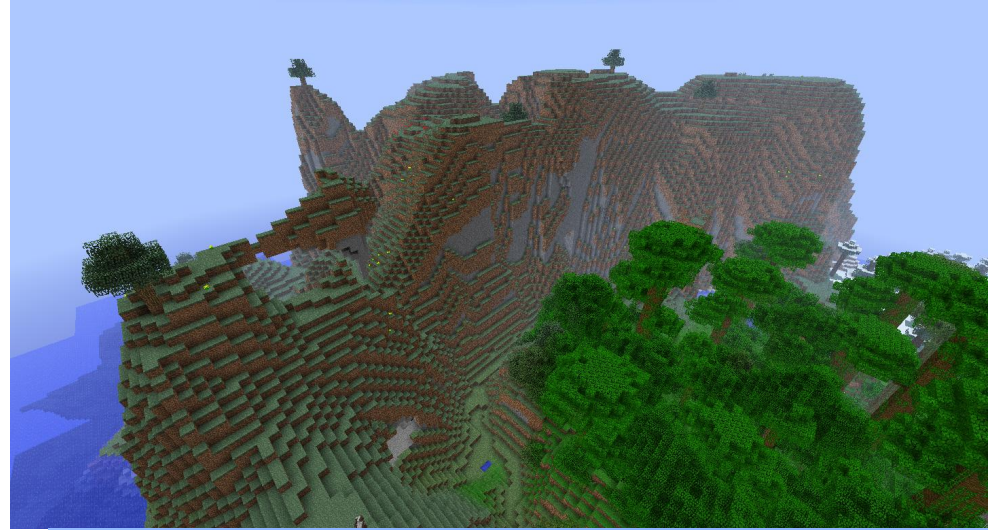
- Create a demo with amazing graphics...
- ...now do it in 4 KB
- Everything has to be generated on-the-fly



Various demoscene productions ("elevated" - 4K, ".kkrieger" - 96K, "SCOTTIE" - 64K)

Minecraft

Explore a new, *infinite* world in every game



Why generate

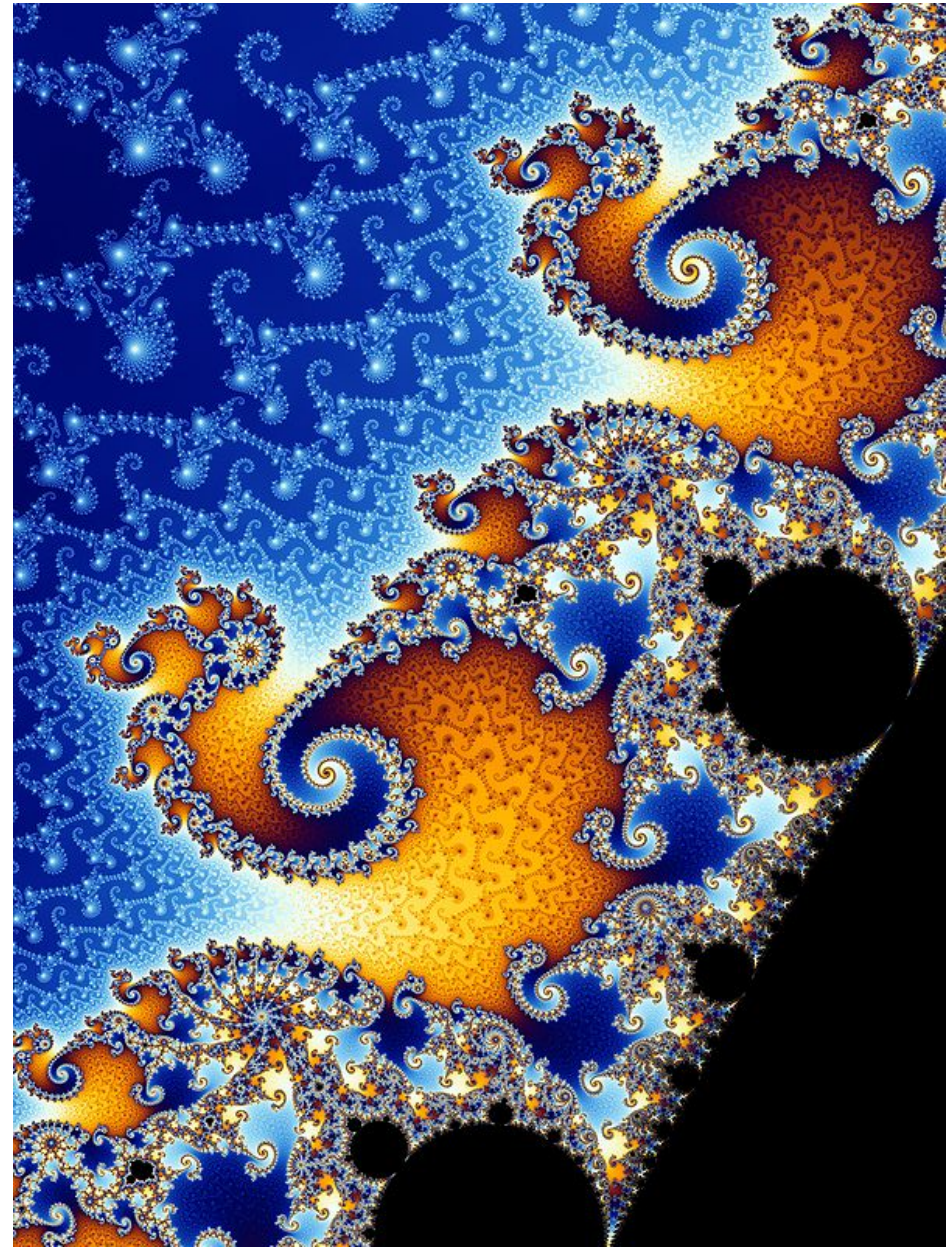
- Space constraints
 - not that relevant nowadays
- Developer/artist time constraints
- Replayability
 - new content to explore
 - especially in roguelike games, but not only
- Emergent gameplay
 - if the game is complex enough,
your content can create new *experiences*
 - a player writes their own story

$$z_{n+1} = z_n^2 + z_0$$

How to do it

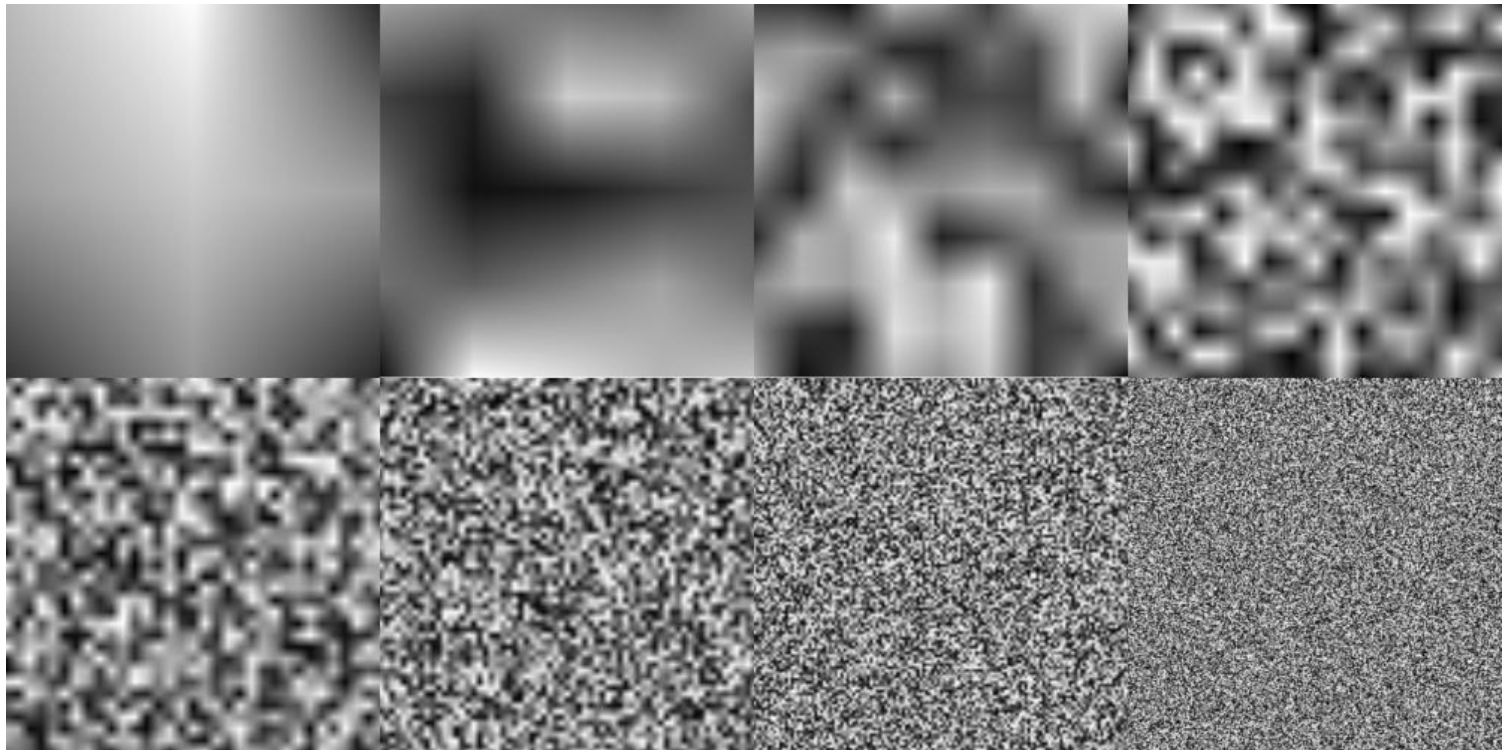
- Simple algorithms that generate complex-looking shapes
 - fractals are a good example
- Throw in some randomness
- Adjust the parameters, combine several methods

Now for some examples.



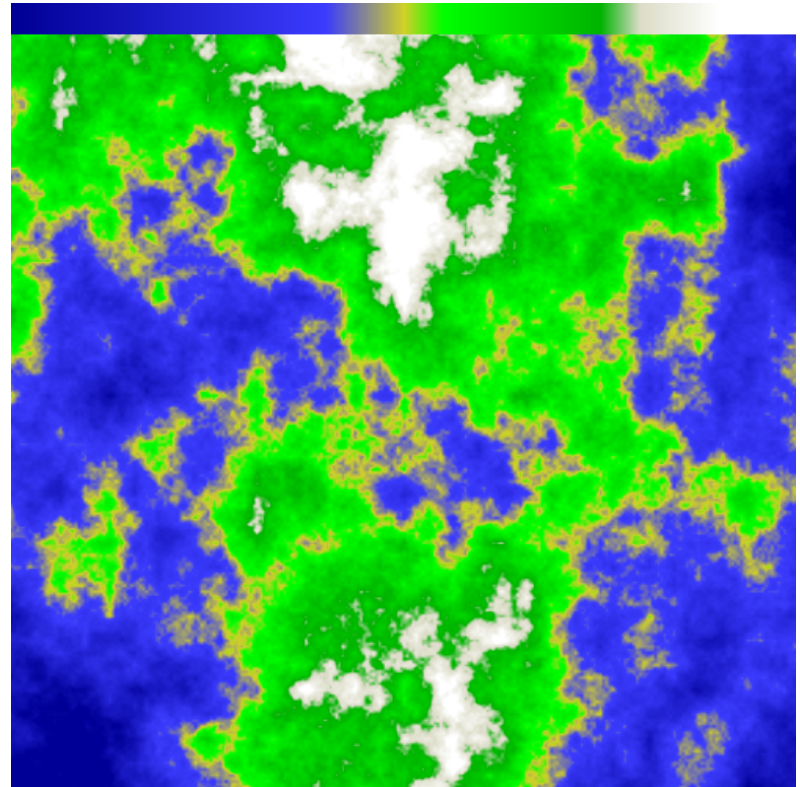
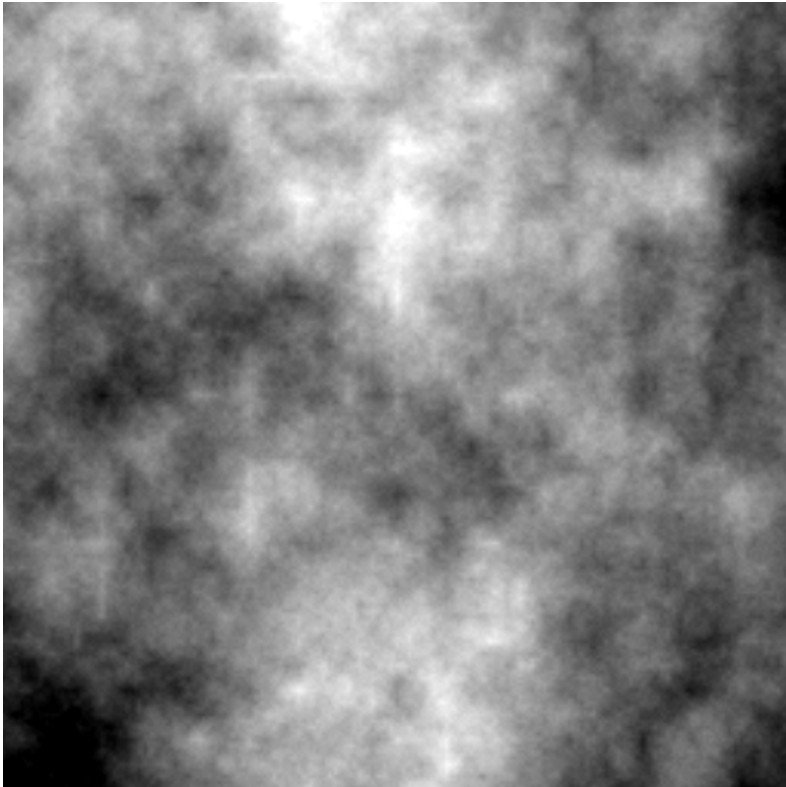
Example: Perlin noise

Generate smooth random noise in several scales ("octaves")....



Perlin noise

Now add it all together and apply a color map.



- Fractal-like (self-similar at different scales)
- Works for higher dimensions as well (volumetric noise)
- Many uses for maps, textures...

Example: Cellular automata

Start with a random distribution of cells, and a simple rule:

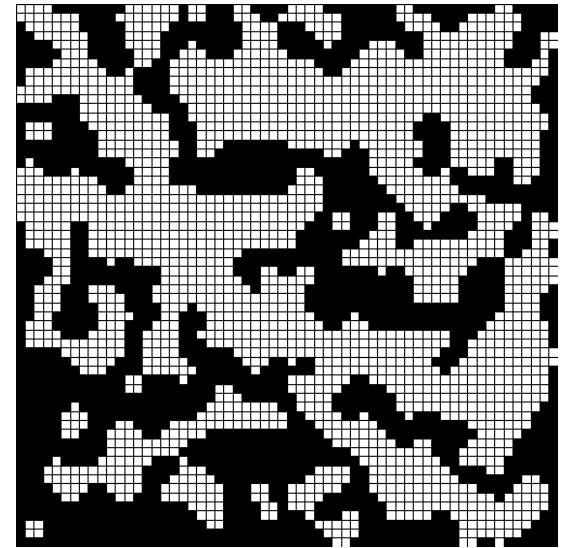
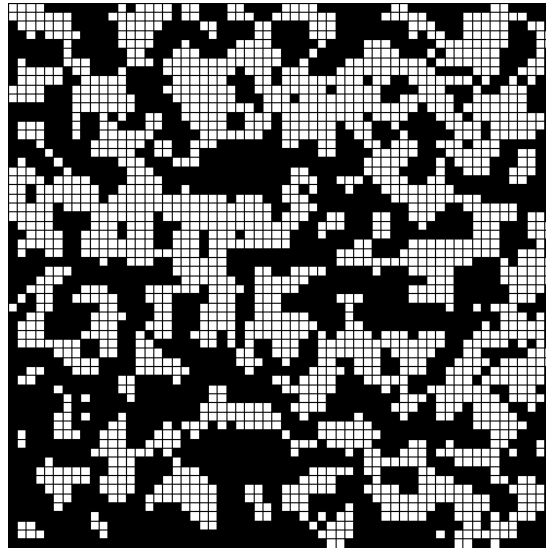
- If a cell is empty, and >2 neighbors are empty, it stays empty.
- If a cell is full, and >4 neighbors are empty, it becomes empty.
- Otherwise, it becomes full.

Kind of like Game of Life, but more stable.



Cellular automata

After several iterations, we get a nice cavern system.



We might want to test for connectedness, and join separate components.

Example: L-systems

We define a simple "turtle graphics" set of commands:

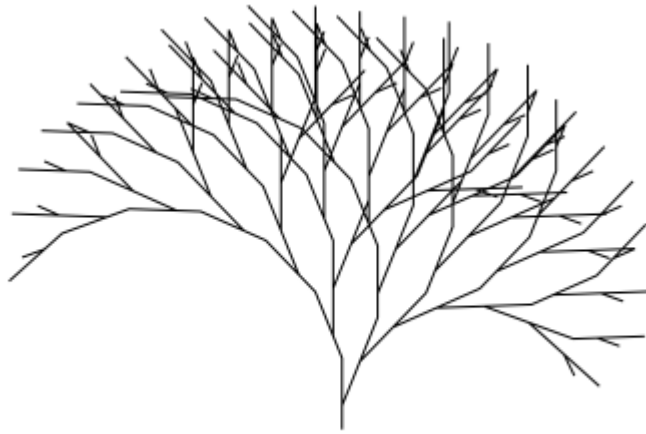
- **F** - go forward
- **-**, **+** - turn left/right
- **[** - save our position
- **]** - return to saved position

and a rewriting rule:

- start with **X**
- **X** \rightarrow **F[FF-X][+FX]**

Iterate that a few times...

L-systems



Normal



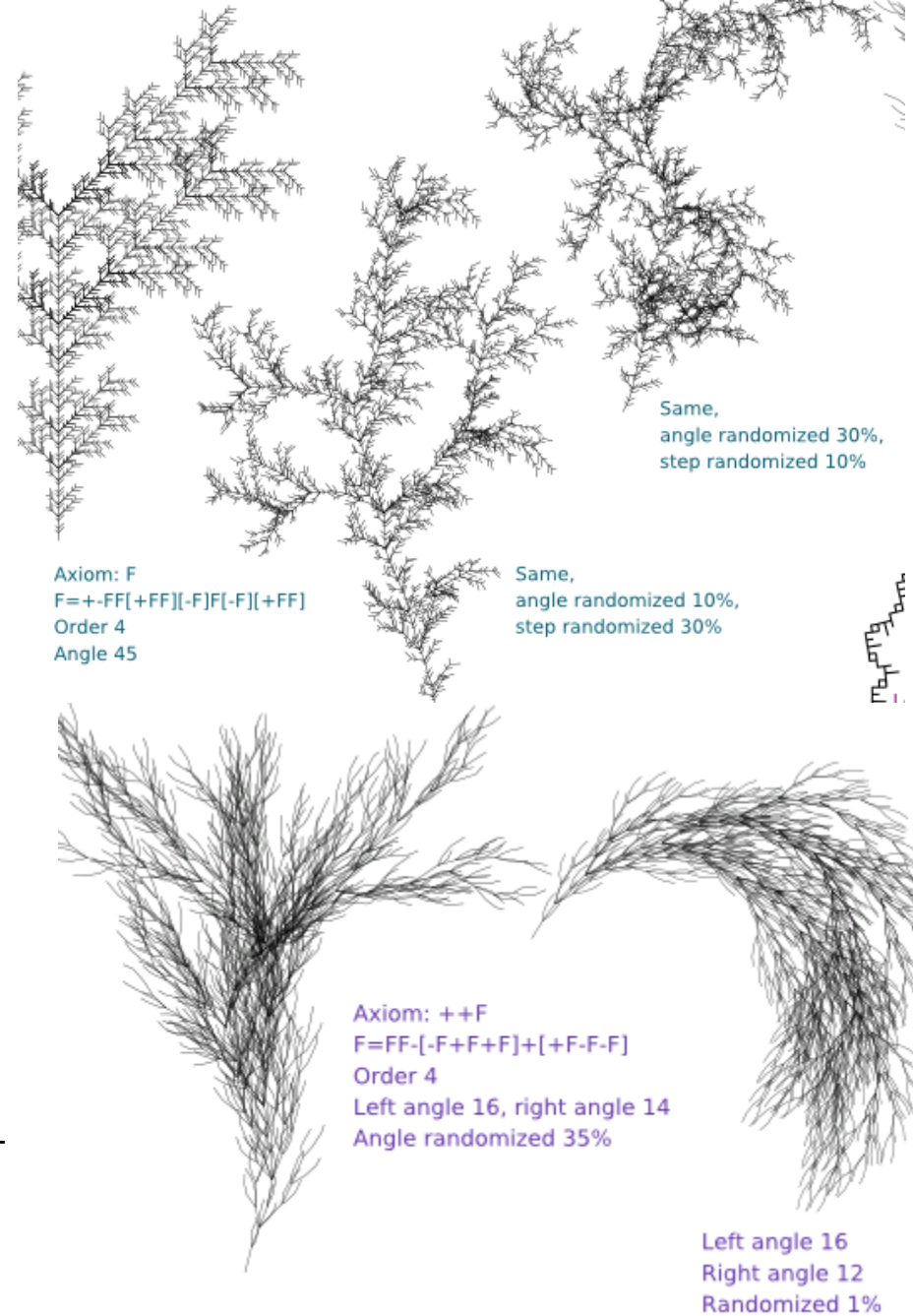
Randomized angle and length

- start with **X**
- **X** \rightarrow **F[FF-X][+FX]**

L-systems

- A (stochastic) grammar / rewriting system
- Created to describe plant growth
- Can be used to draw other fractals (e.g. Sierpiński's triangle, Koch's curve...)
 - or... Penrose tiling

If you want to play with it, [Inkscape](http://inkscape.org/) has support for L-systems.



Some other ideas

- Floor plan generation
- City generation ([paper](#))
 - start with a height map, population map
 - generate street plan, buildings
 - L-systems come in handy!
- Procedural generation of events
 - Left 4 Dead 2's "AI director"

Simulationism

- Problem: results of simple methods can be boring, too uniform
- Instead of faking everything, make a simple model
 - geologically motivated terrain generation
 - agent-based city generation
- Example of extreme simulationism?
Dwarf Fortress!
 - geological data
 - *historical and cultural data* (societies, wars, myths, heroes)
 - simulates weather for the whole world in real time
 - all necessary (and unnecessary) details about every character
- ...at this point, simulation becomes a goal in itself

Thanks for listening