CS354R

DR SARAH ABRAHAM

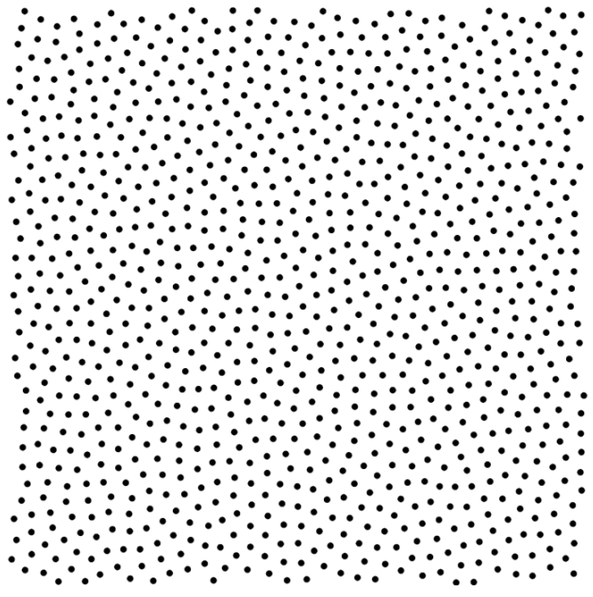
PROCEDURAL GENERATION

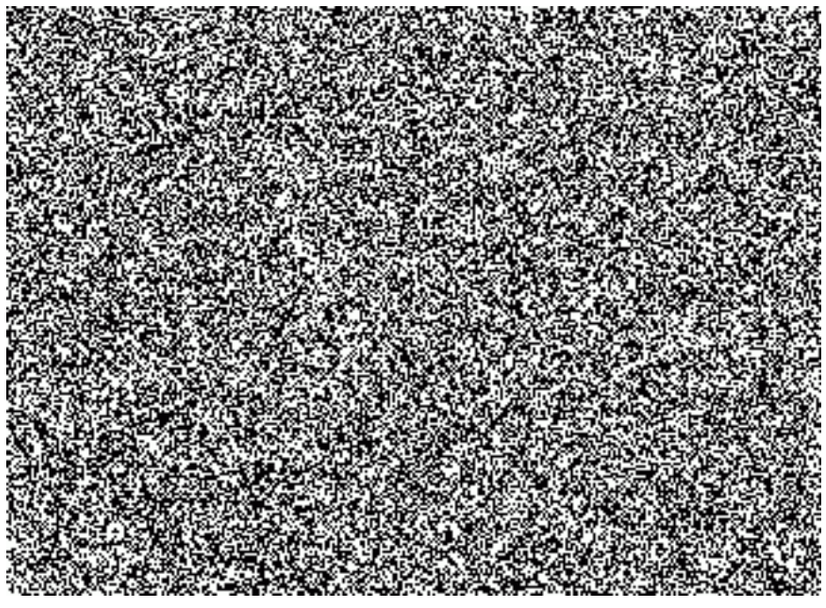
WHAT IS PROCEDURAL GENERATION?

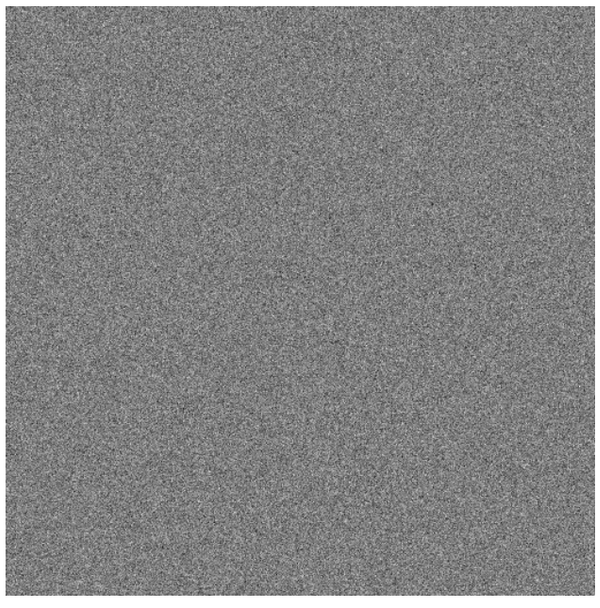
- Use of mathematical functions to create assets
  - Usually we want it to both follow a pattern and have some amount of randomness
- Broad category with many subtopics and types of applications
- Today we will be discussing:
  - Noise functions
  - Dungeon generation
  - L-Systems

NOISE

- Random (stochastic) fluctuations in an expected signals
  - Different concentrations of energy create different patterns

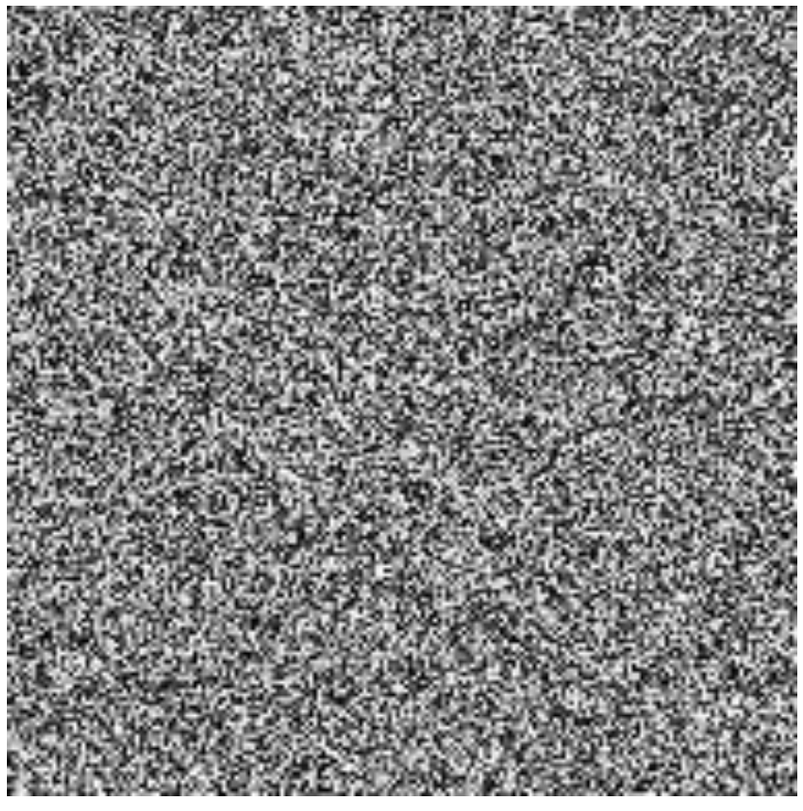






WHITE NOISE

- White noise problems:
  - Isn’t smooth
  - Isn’t correlated

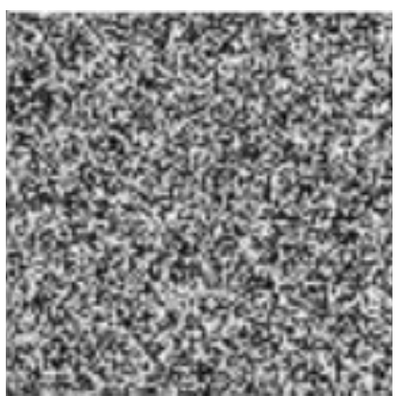
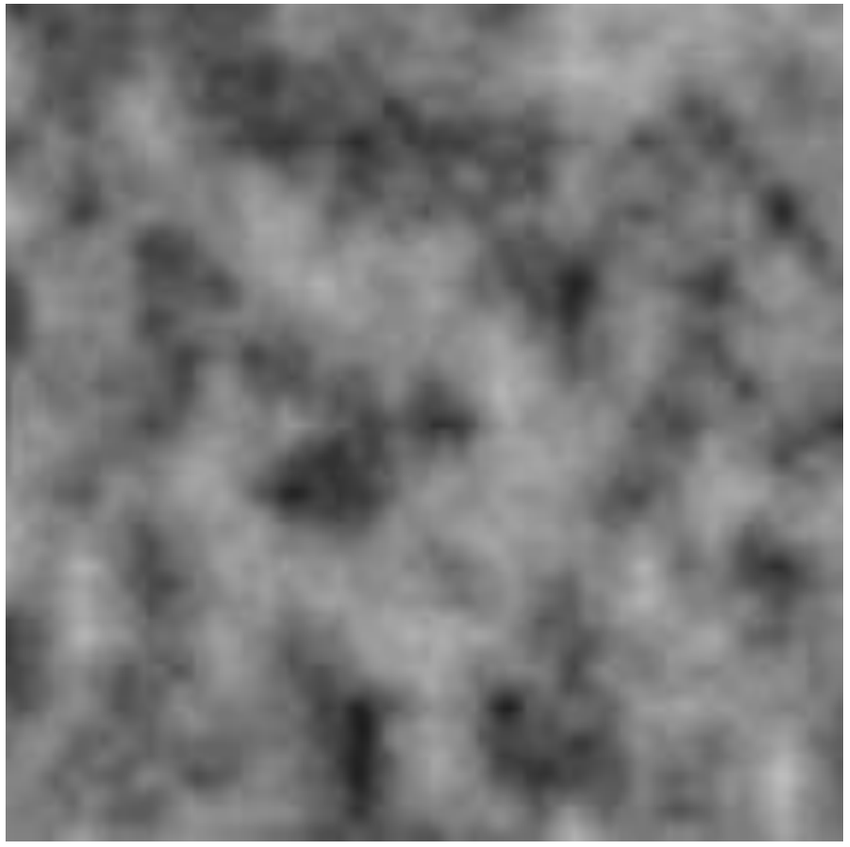


$$I(u, v) = \text{rand}()$$

PERLIN NOISE

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



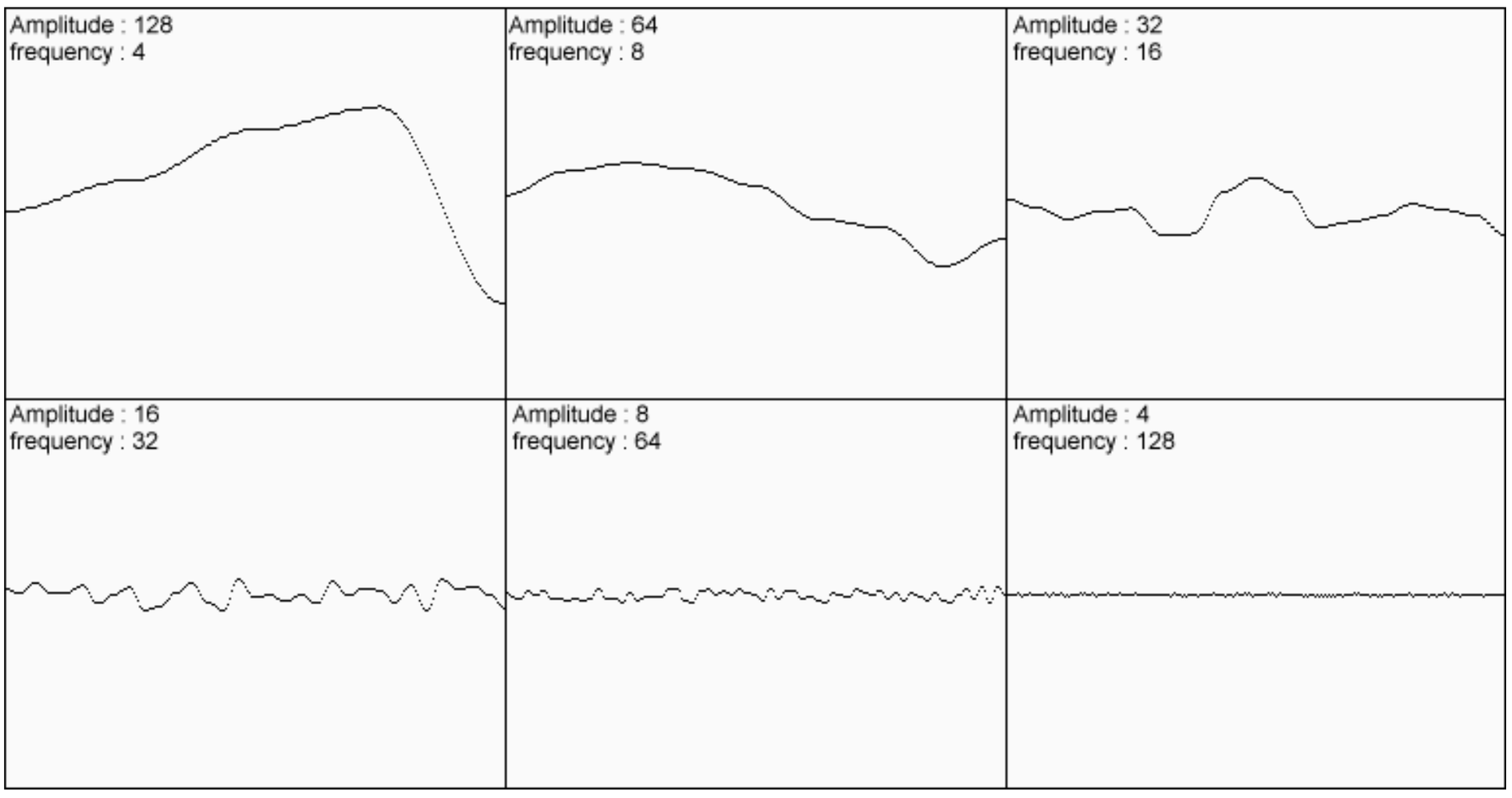




GENERATING SMOOTH NOISE

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

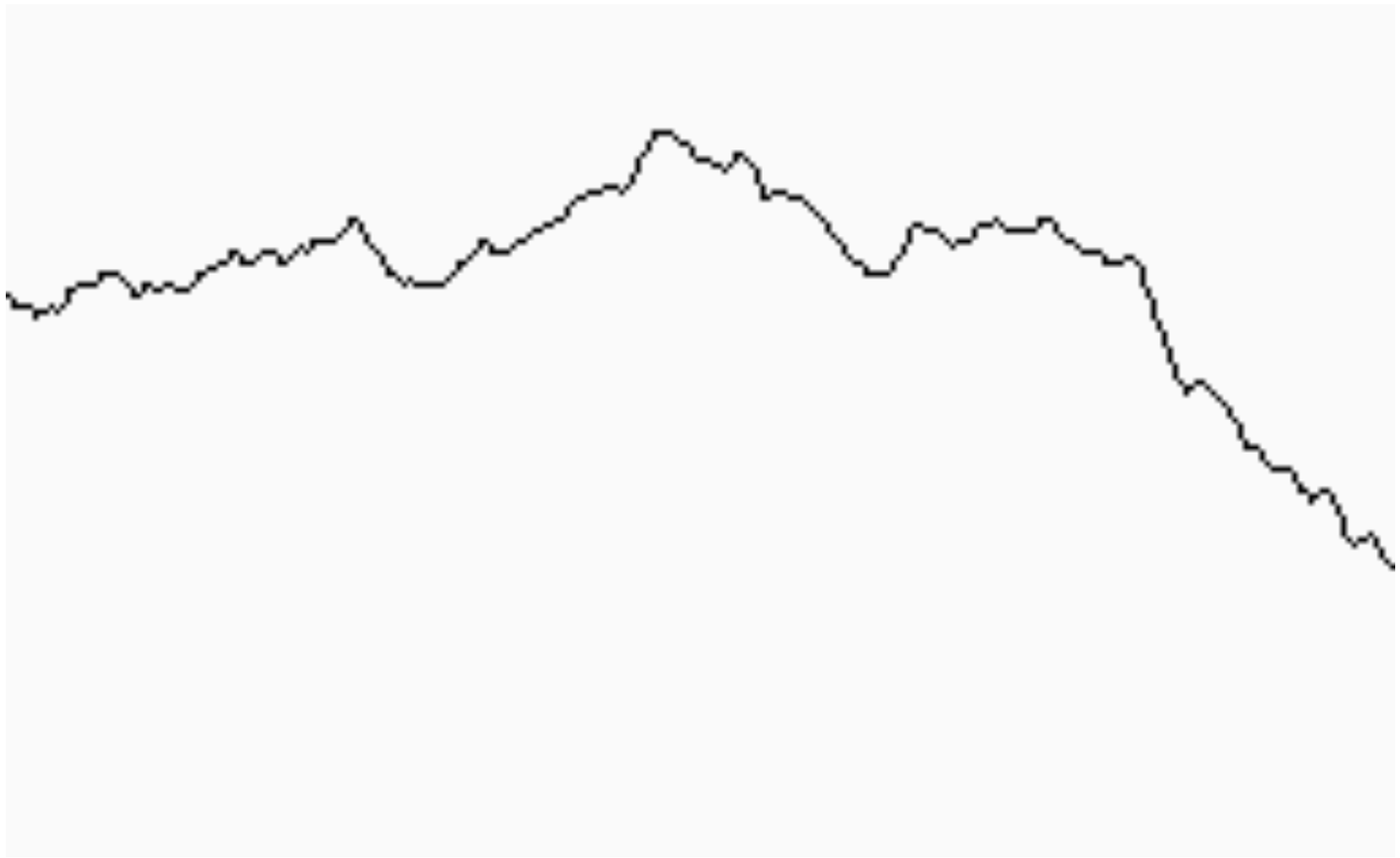
NOISE OCTAVES

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



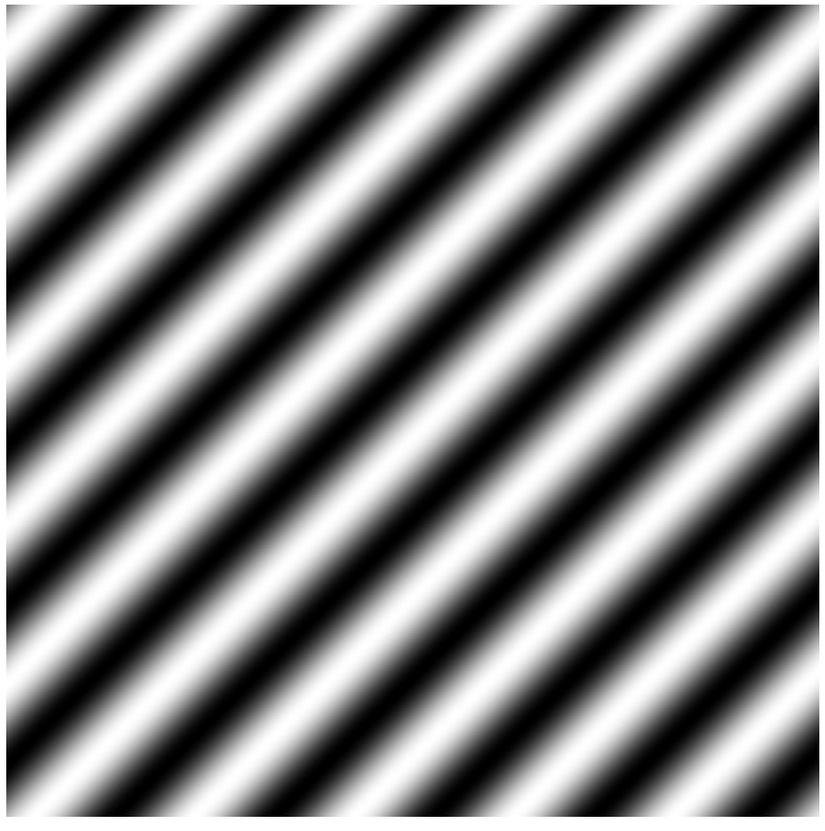
NOISE OCTAVES

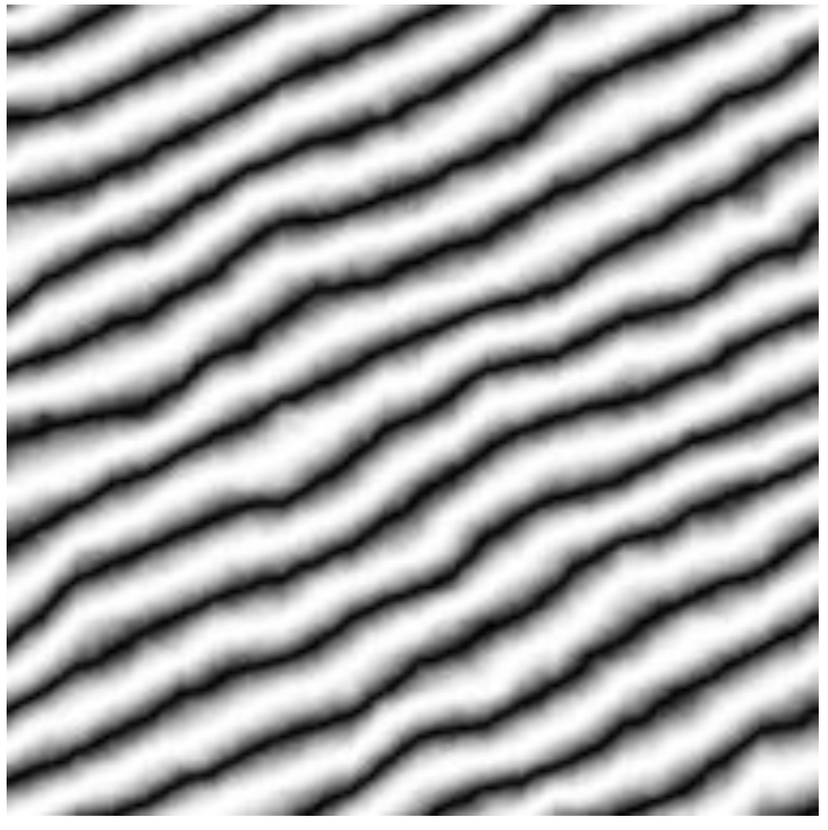
- Can combine multiple octaves to get better looking results via Perlin noise



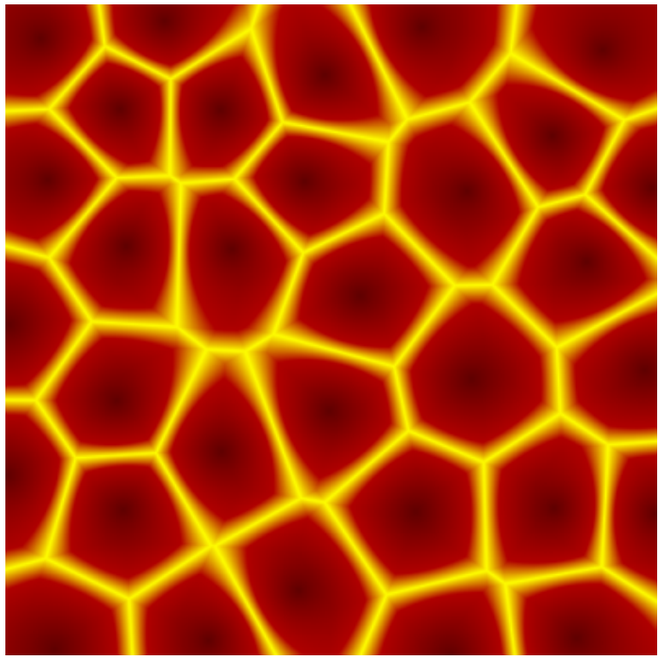
PERLIN NOISE APPLICATIONS

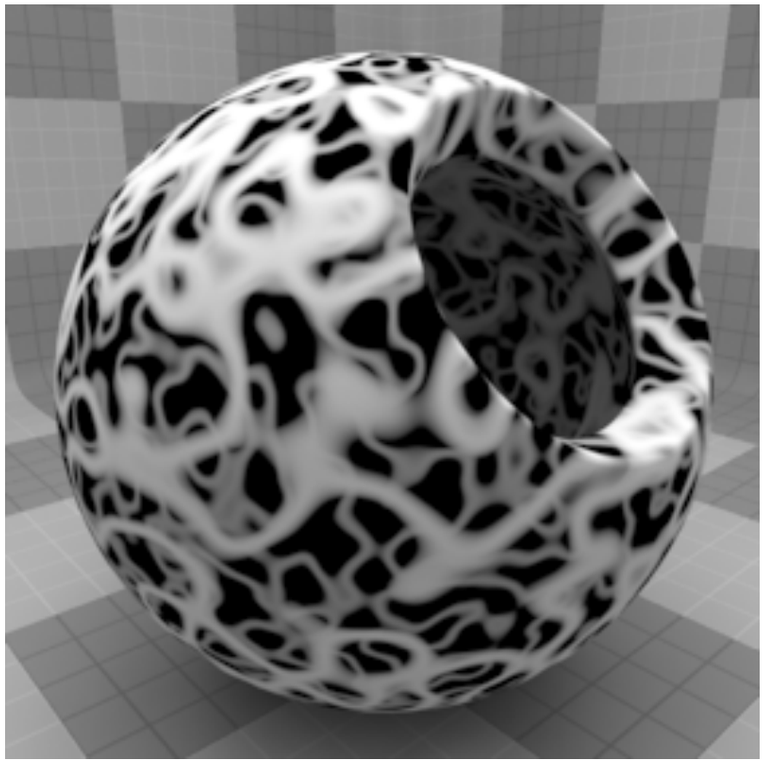
- Perlin noise applied to existing functions…

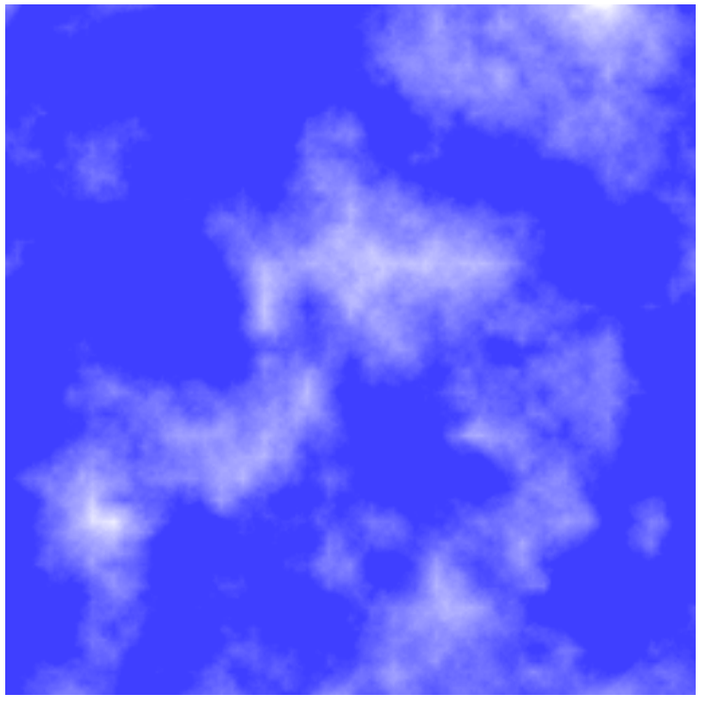


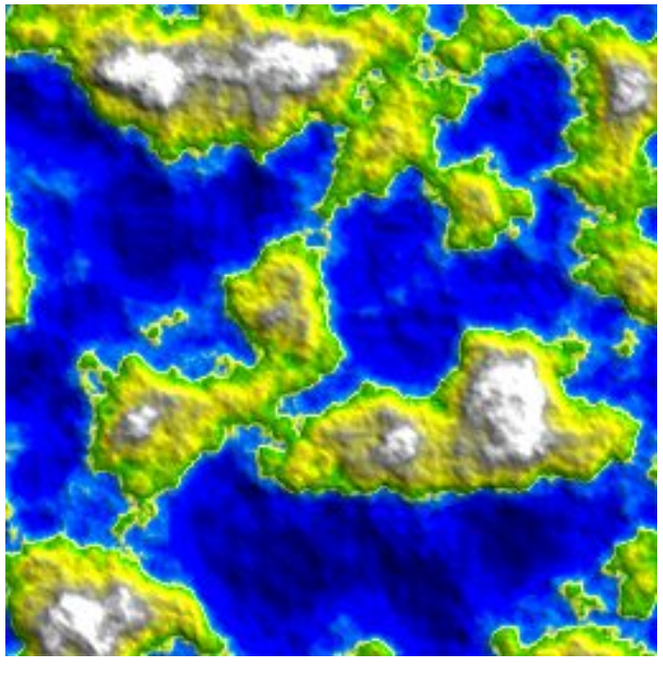


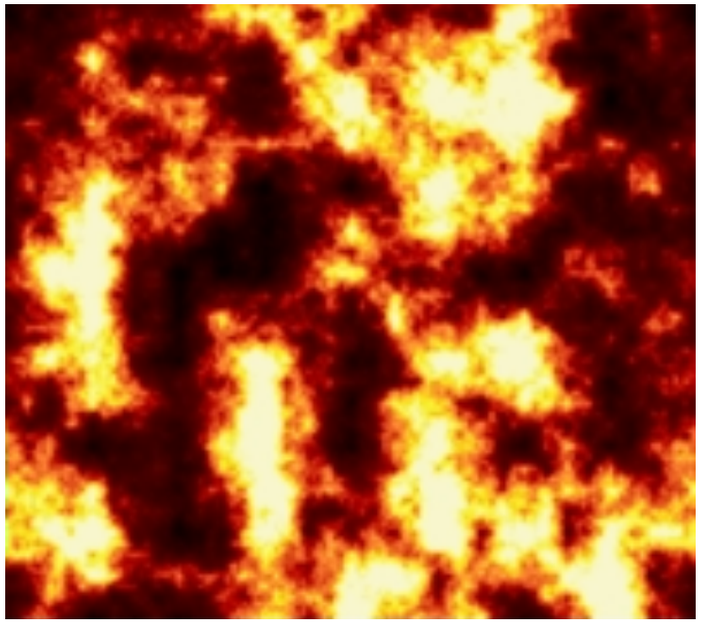
PERLIN NOISE EXAMPLES



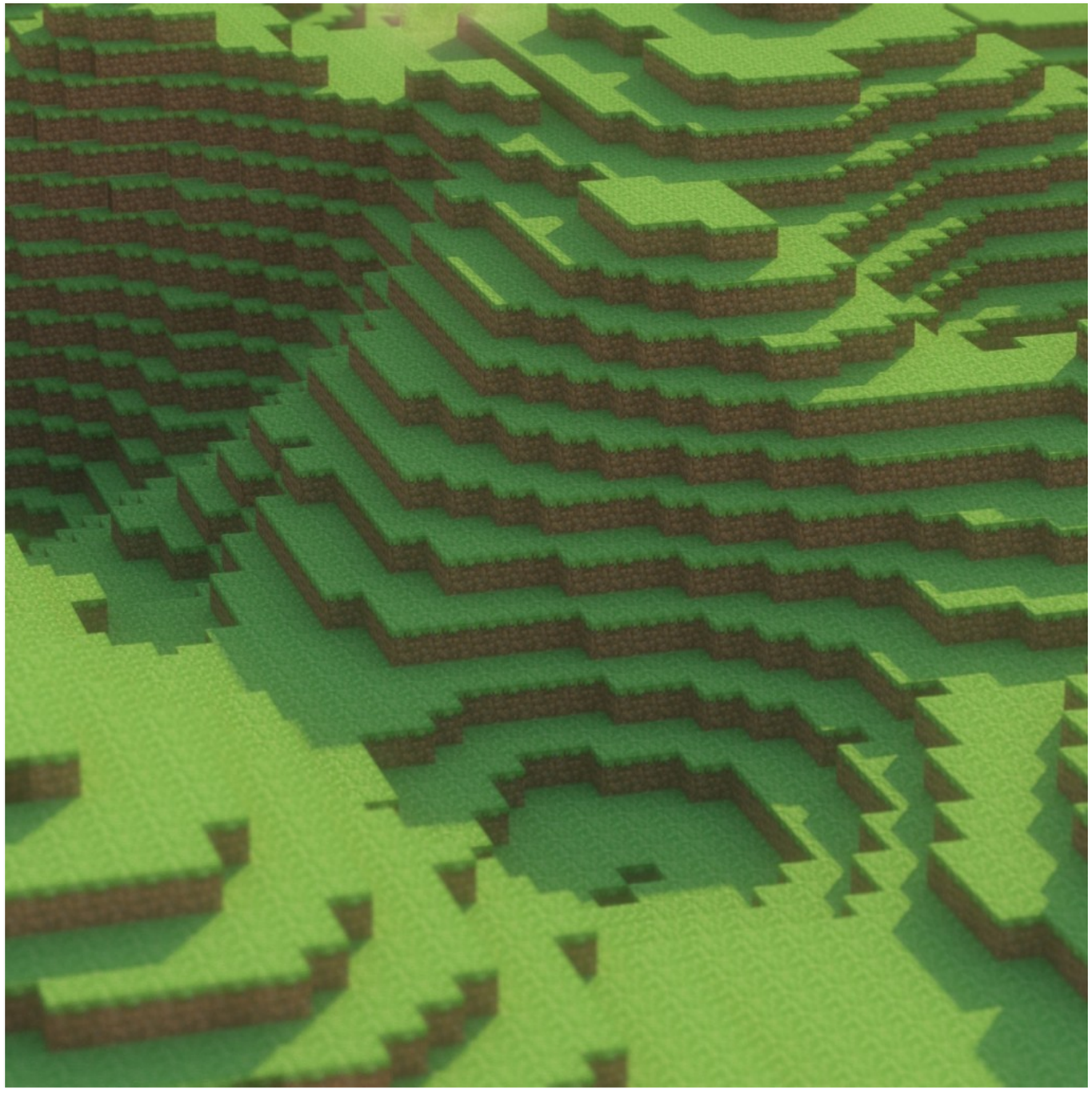






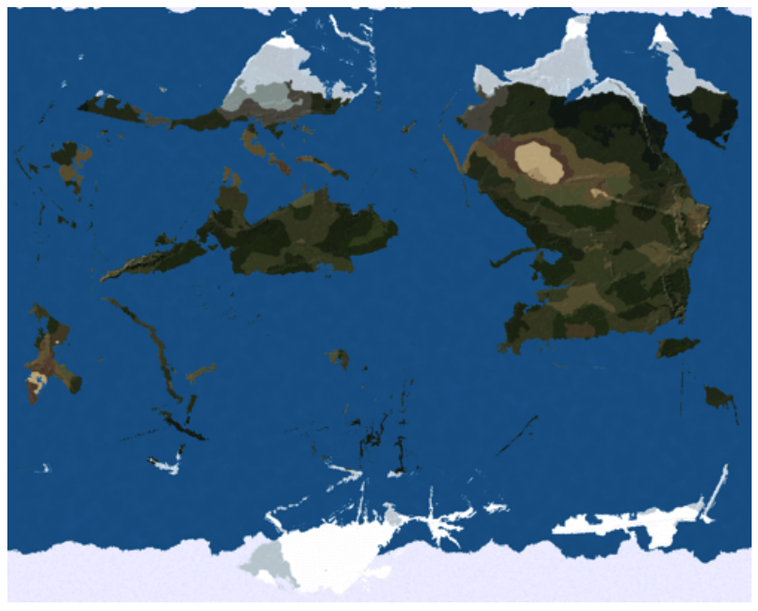


TERRAIN GENERATION EXAMPLE: MINECRAFT



PERLIN NOISE AND VORONOI

- Possible to combine Perlin noise with other algorithms
- Voronoi diagrams partition planes based on distance to provided points
- Used together for creating biomes!



DUNGEON GENERATION

- Distinct from terrain generation
  - Room-based rather than open-world
- Usually want to generate the entire dungeon (rooms, enemies, drops, obstacles, etc)

Rogue

Diablo 1

BUILDING THE LEVEL

- Dungeons are effectively mazes with rooms
  - Many, many ways to approach this but usually involves multiple passes over the level
- The high-level idea is:
  1. Build out the rooms and corridors
  2. Build out world features of the level
- Each of these “passes” can (and should) have multiple sub-passes as well

BUILDING ROOMS AND CORRIDORS

- Must determine size of level and number of rooms
  - How big should the rooms be?
  - How dense should the rooms be?
- Run a maze generation algorithm to build out a maze of corridors between these rooms
- Doors can be placed in a brute-force manner
  - Must ensure all rooms are accessible
  - Maybe want to make them “logically” placed

BUILDING WORLD FEATURES

- Once the level is built out, it is populated with expected items and features
- Tile sets based on features of room and corridor (e.g. floor tiles, wall tiles, doorway tiles, etc)
  - Applies to 3D assets as well
  - Can generate metadata to allow for greater customizability
- Place items and enemies based on features of room or corridor
  - Use of metadata to know the types and frequencies of these actors
- Place a starting and ending point for the level
  - Should have some reasonable distance between these two points

ALLOWING FOR HAND-TUNING

- Generation tools ideally should allow artists/designers to hand-tune areas that are unique places in the world
- Creates a more exciting experience
  - Allows for better world-building and points of interest
- True rogue-likes don’t have this, but they can accomplish a similar feel using extensive metadata for building a level

NOTE: PROCEDURAL GENERATION CAN'T MAKING THINGS INTERESTING...

No Man's Sky

L-SYSTEMS

- Recursive definition of an object using a string rewriting system and formal grammar
- Invented by botanist, Aristid Lindenmayer
- Designed to model plants
- Przemyslaw Prusinkiewicz brought concepts to graphics

L-SYSTEM DEFINITION

- **Axiom**: Starting string
- **Variables**: Set of symbols to be rewritten according to rules
- **Terminals**: Set of symbols that have no rewriting rules
- **Rules**: Set of substitutions possible for variables

USING L-SYSTEMS IN GRAPHICS

1. Associate actions (e.g. draw line, rotate, etc) with each variable and terminal

2. Recursively expand the axiom *n* times

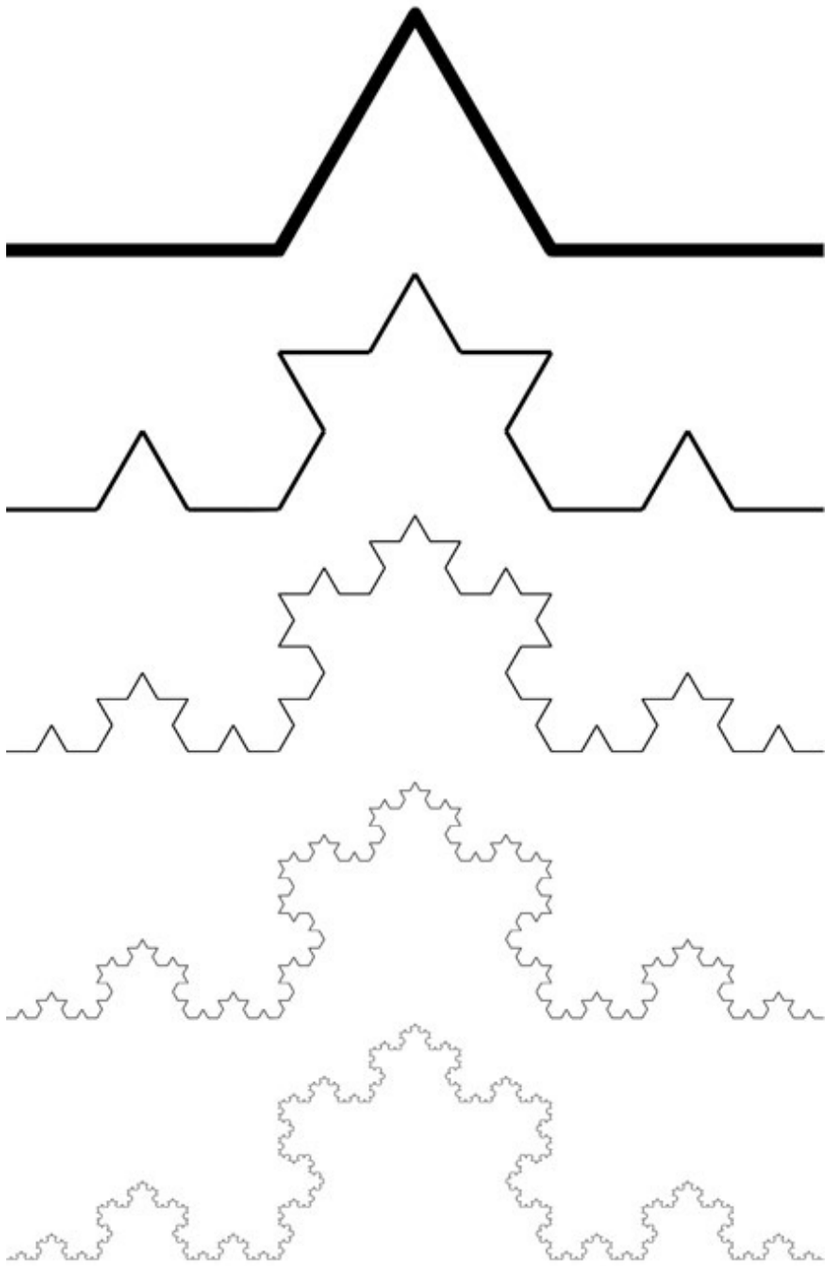
1. Execute actions of resulting string

2. Generate image from string

EXAMPLE: KOCH CURVE

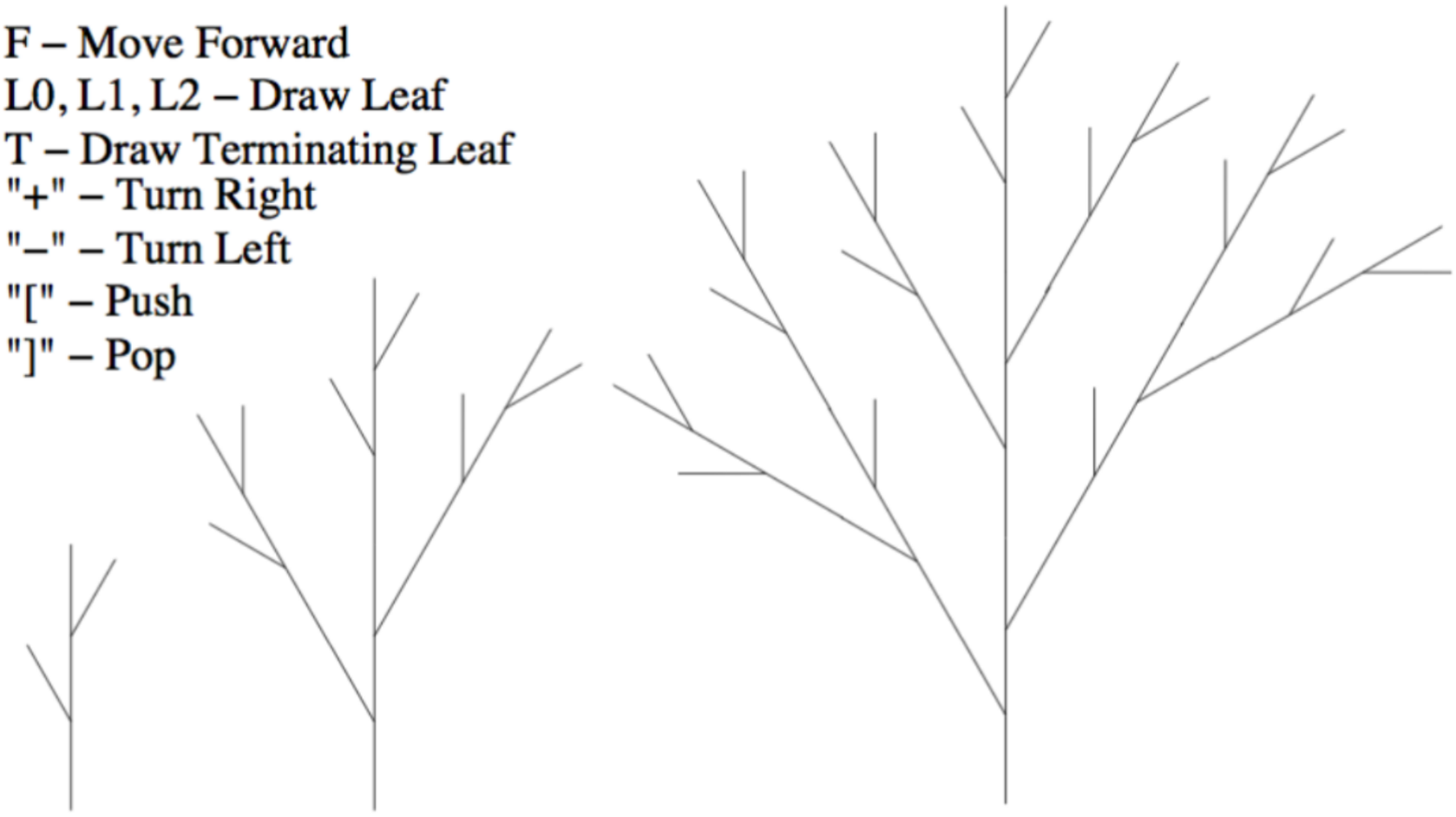
- Rule:
- F = F-F++F-F

- F: Draw line segment scaled by 1/3
- -: Turn 60° left
- +: Turn 60° right



EXAMPLE: 2D TREE

F – Move Forward
L0, L1, L2 – Draw Leaf
T – Draw Terminating Leaf
"+" – Turn Right
"–" – Turn Left
"[" – Push
"]" – Pop

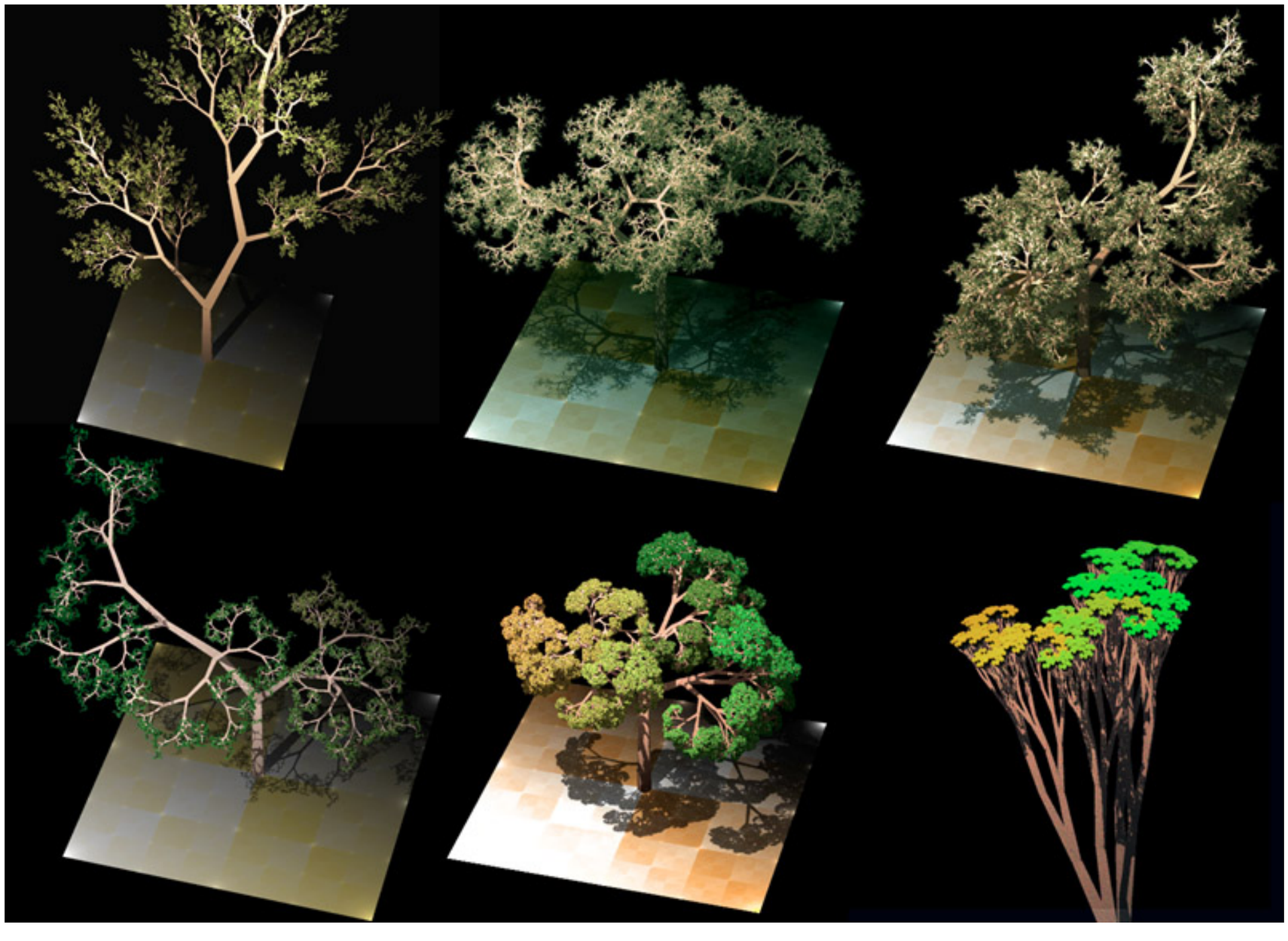


Depth 1 Depth 2 Depth 3

Axiom: L0

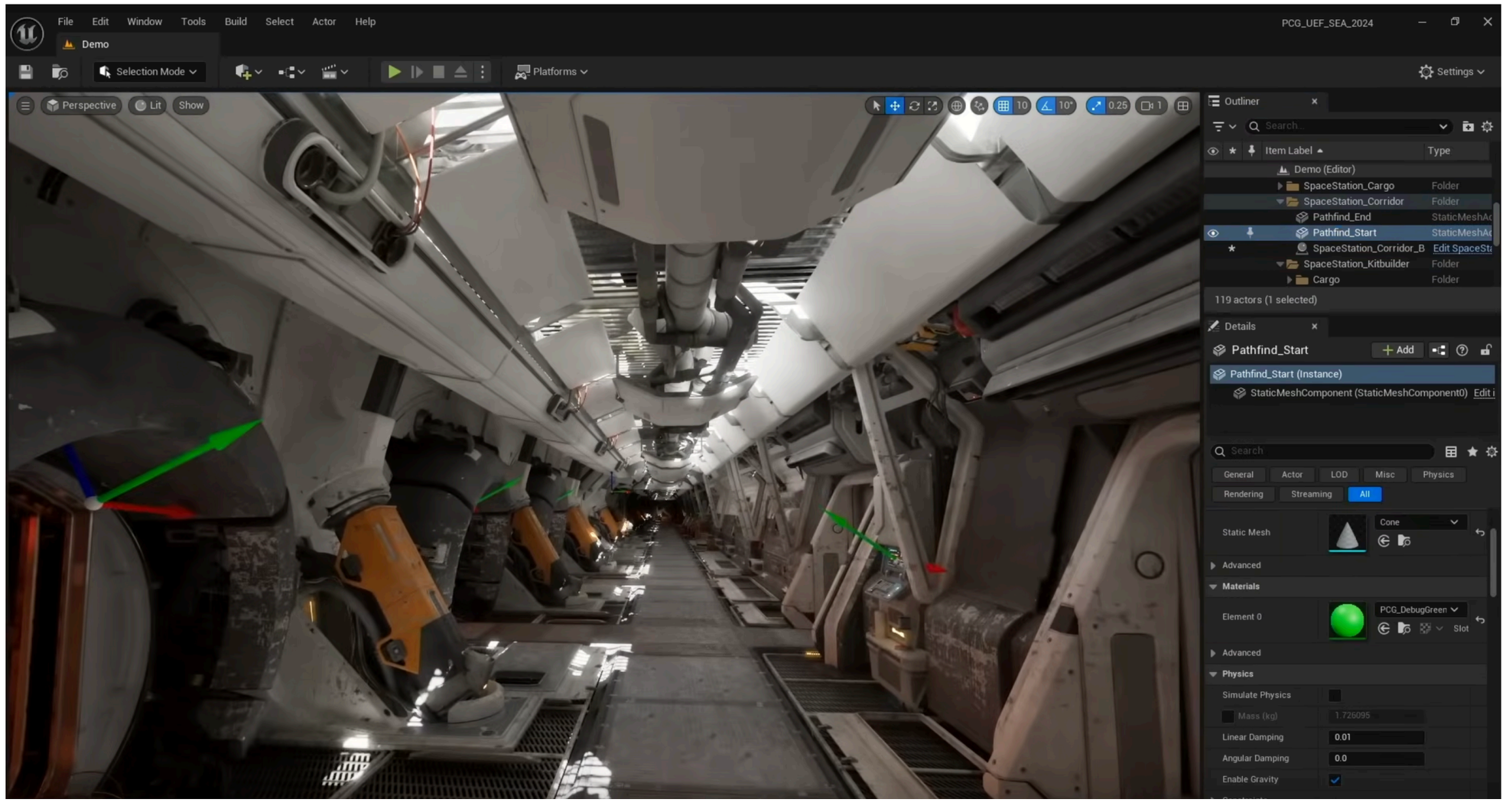
1. L0 -> F [- F L1] F [+ F L2] F L0 (center branch)
2. L1 -> F [- F L1] F [+ F T] F L1 (left half of tree)
3. L2 -> F [- F T] F [+ F L2] F L2 (right half of tree)

L-SYSTEM EXAMPLES



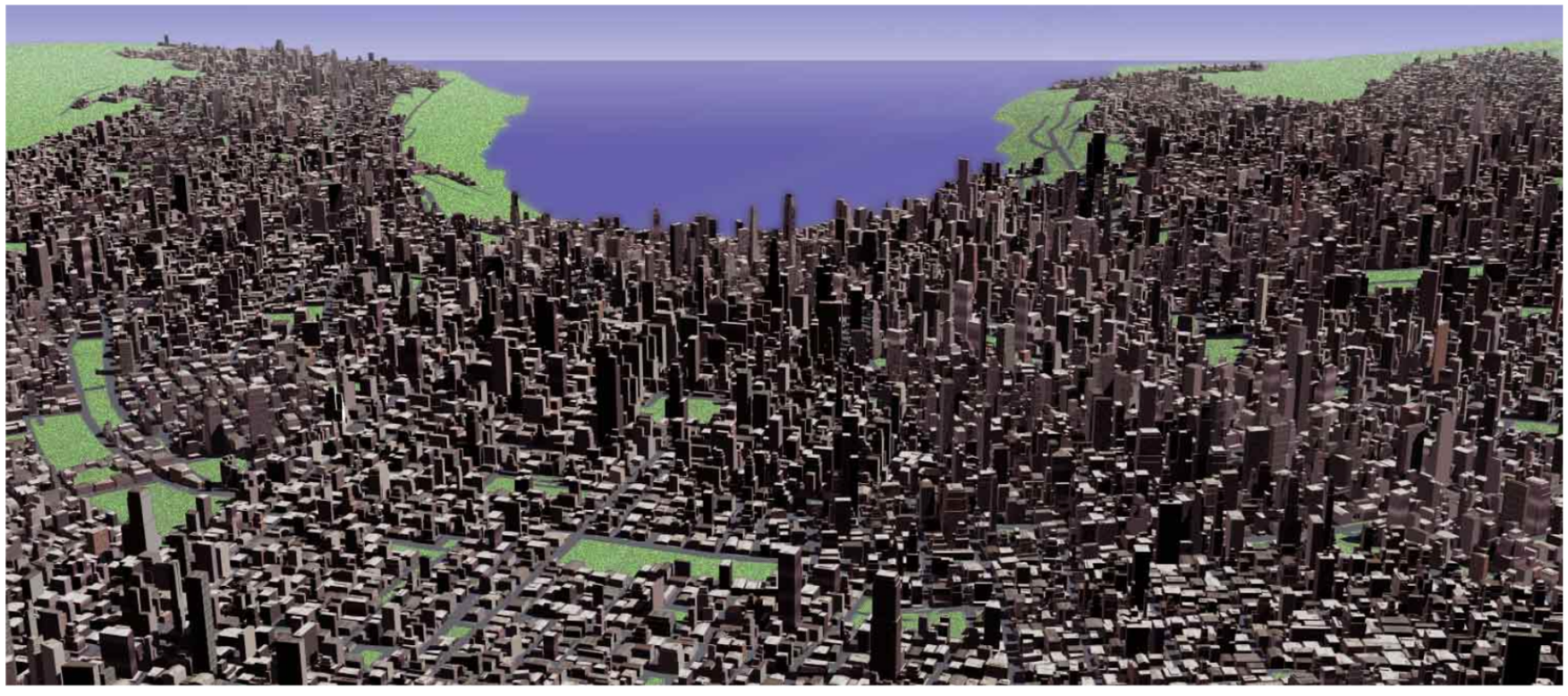
UNREAL PCG FRAMEWORK

- https://youtu.be/j3ke6MmcaeY?si=VZM8J_eJ_c-_S-qY&t=784



GENERATING CITIES

- Same idea with different symbols and rules
  - Good idea to having working understanding of the modeled system



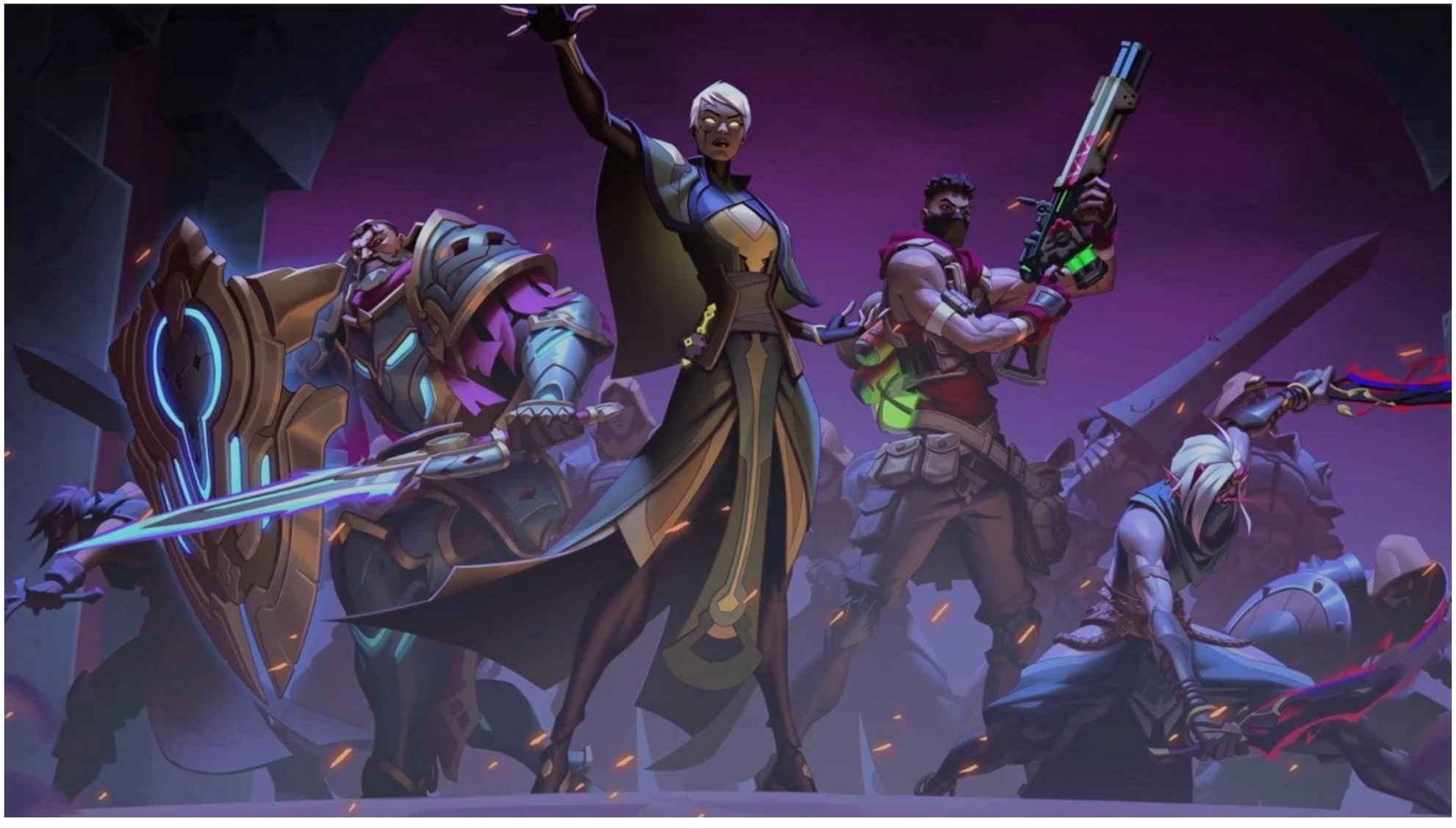
EXAMPLE: INFAMOUS

https://www.youtube.com/watch?v=Br5uKs-Fp-E

EXAMPLE: HOME FREE

https://www.youtube.com/watch?v=ahBSQrX1yOE

EXAMPLE: WAYFINDER



https://youtu.be/iaglHKM1-2Y?si=ODJ5YwjHQ3D2TmRl&t=1134

FURTHER READING

- Graphical Applications of L-Systems <http://algorithmicbotany.org/papers/graphical.gi86.pdf>
- <http://algorithmicbotany.org/papers/>
- Procedural Modeling of Cities<https://graphics.ethz.ch/Downloads/Publications/Papers/2001/p_Par01.pdf>
- TreeIt (Free tree-generation L-System) <http://www.evolved-software.com/treeit/treeit>