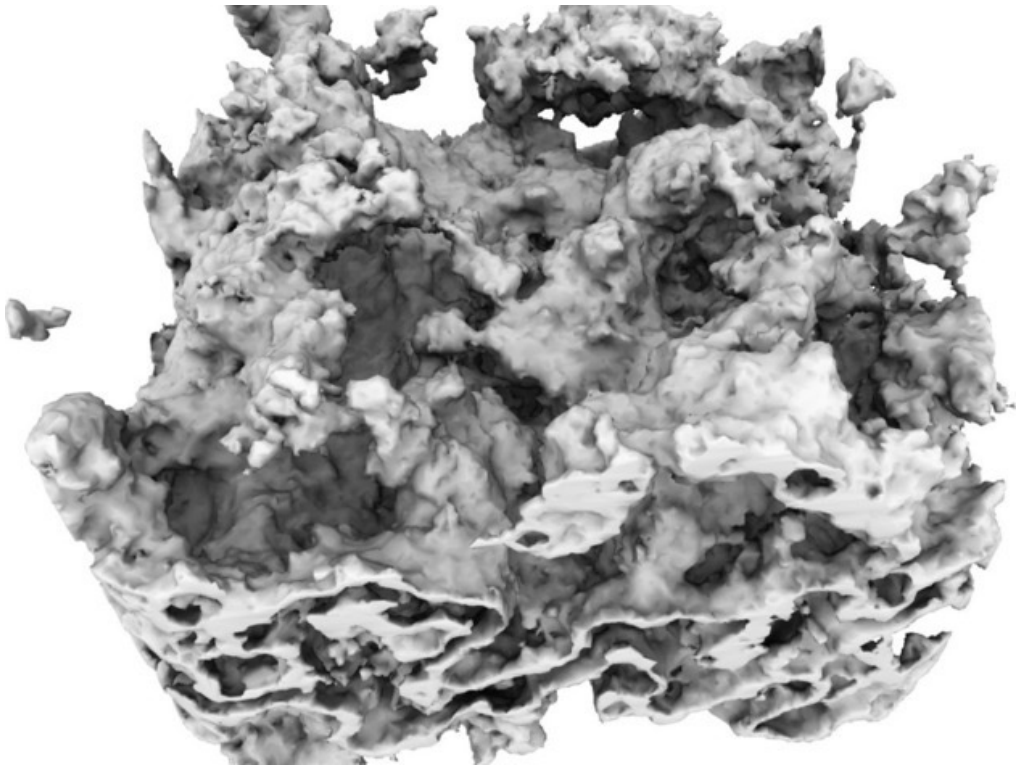
L-Systems and Particle Systems

Procedural Modeling

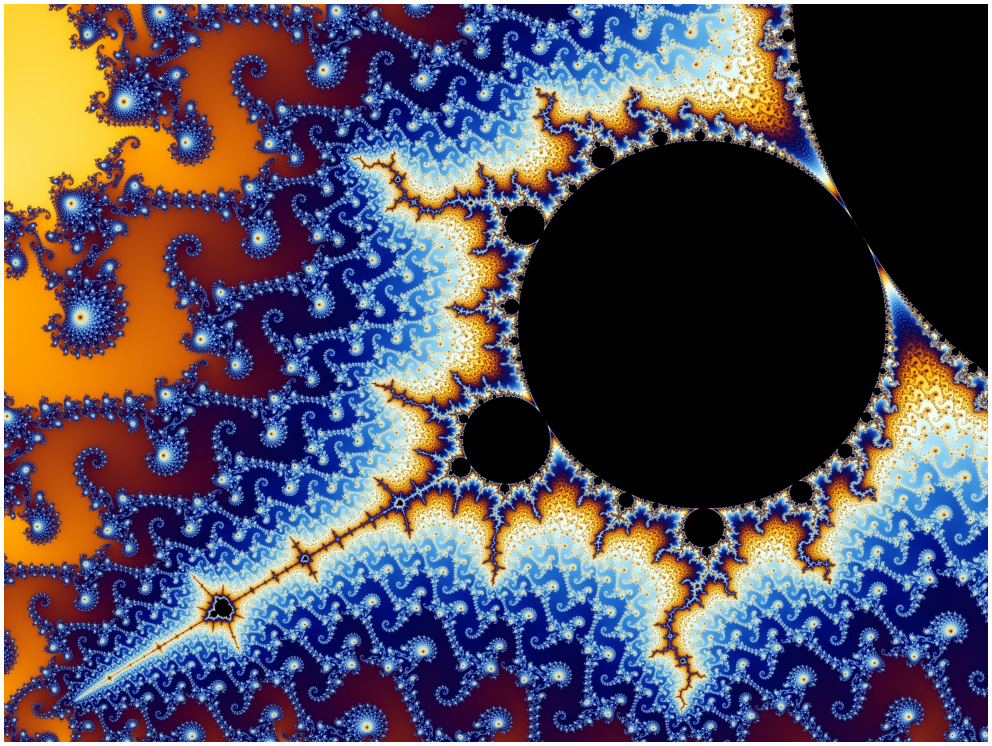
Idea: Detailed meshes are hard to build by hand, so let's create a function that builds out meshes for us

Same idea as Perlin noise but in 3D!



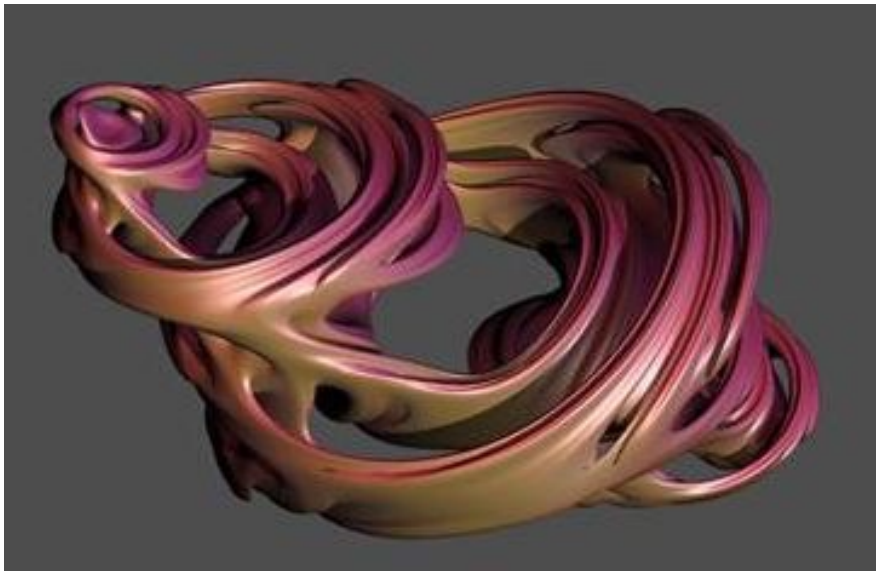
Another Example: Fractals

Iterated function system leads to infinite detail



4D Fractals

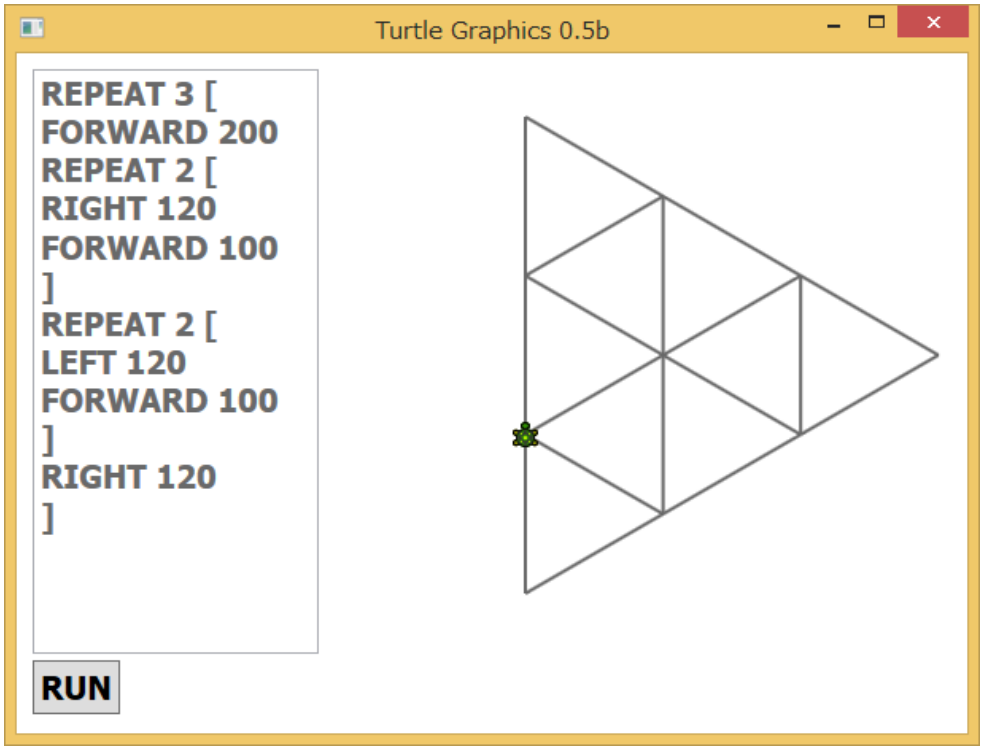
Can be created using quaternions



https://www.youtube.com/watch?v=eS7qCfttmBk

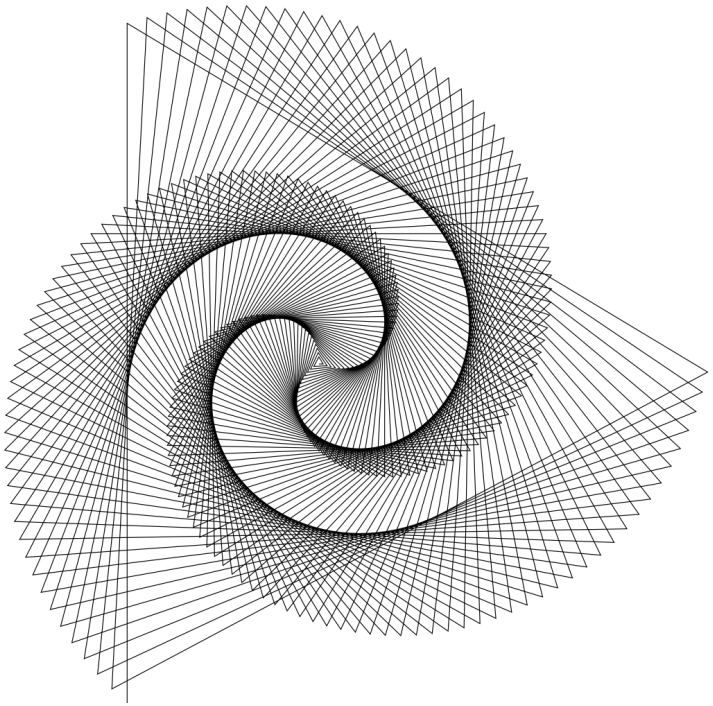
Turtle Graphics

- Graphics system implemented in LOGO (1967)
- Cursor is “turtle” with position and orientation
- Code moves turtle, creating a line trail



Turtle Graphics

Simple code generates very complex results



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

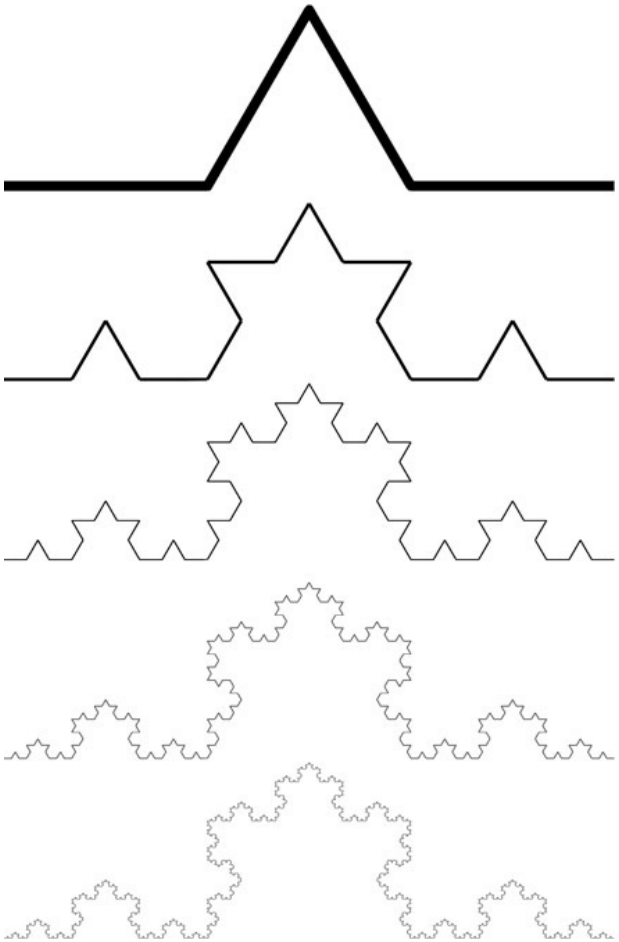
Rule:

F = F-F++F-F

F: Draw line segment
scaled by 1/3

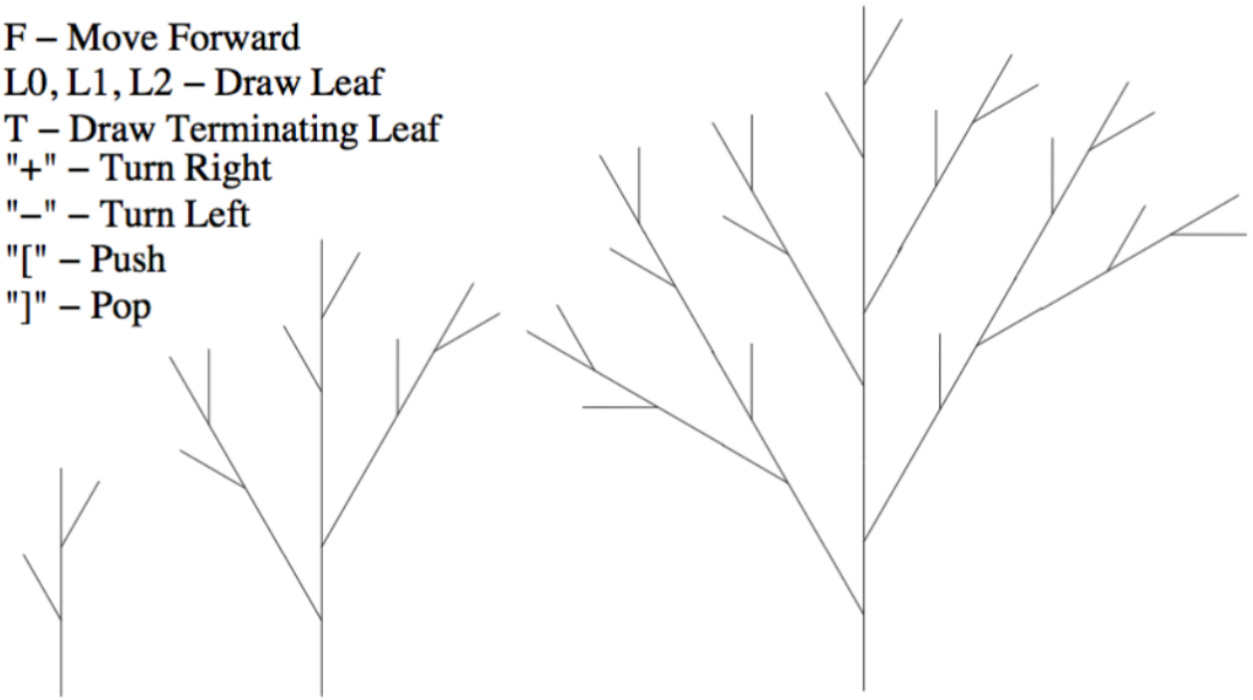
-: Turn 60° left

+: 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

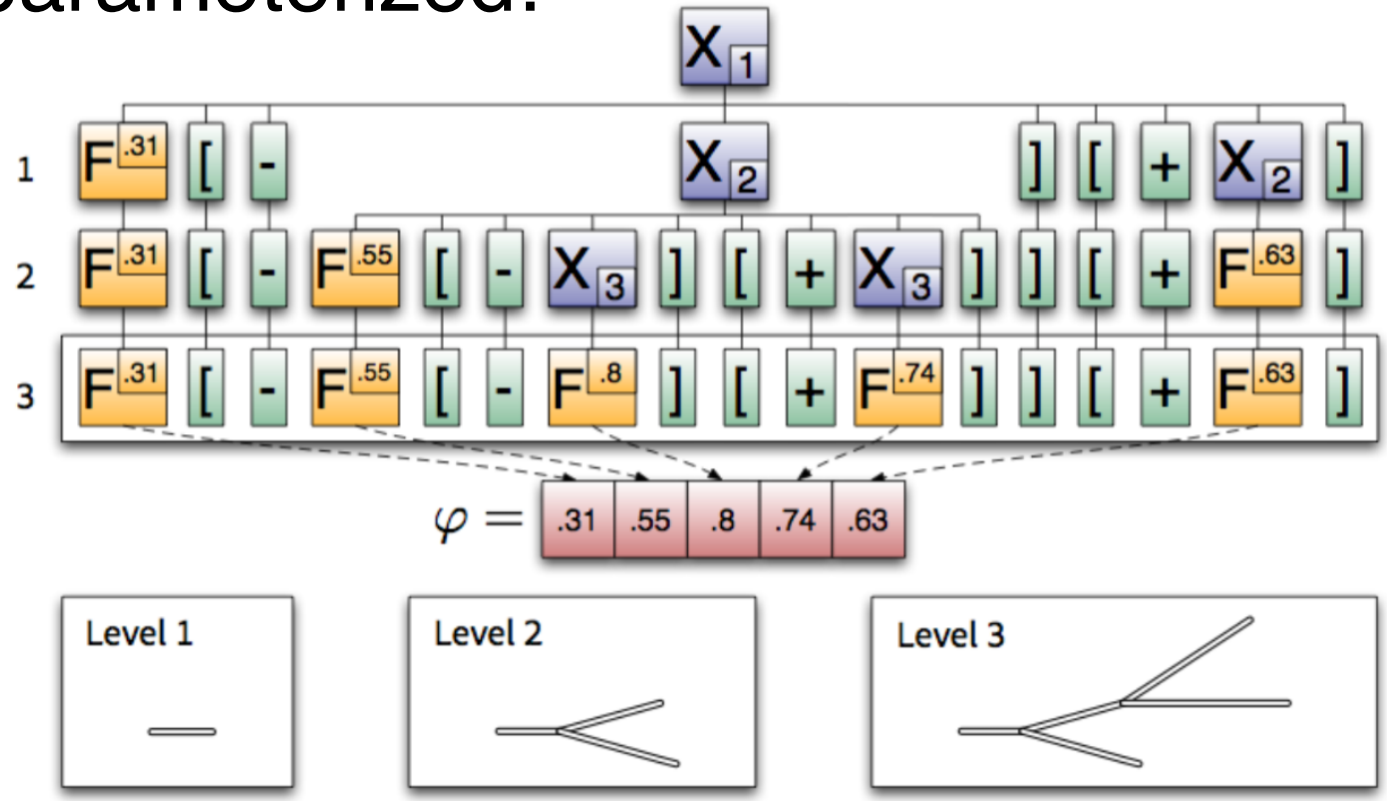
1. L0 -> F [- F L1] F [+ F L2] F L0 (center branch)

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)

Parameterized L-Systems

Action specified by symbol can be parameterized:

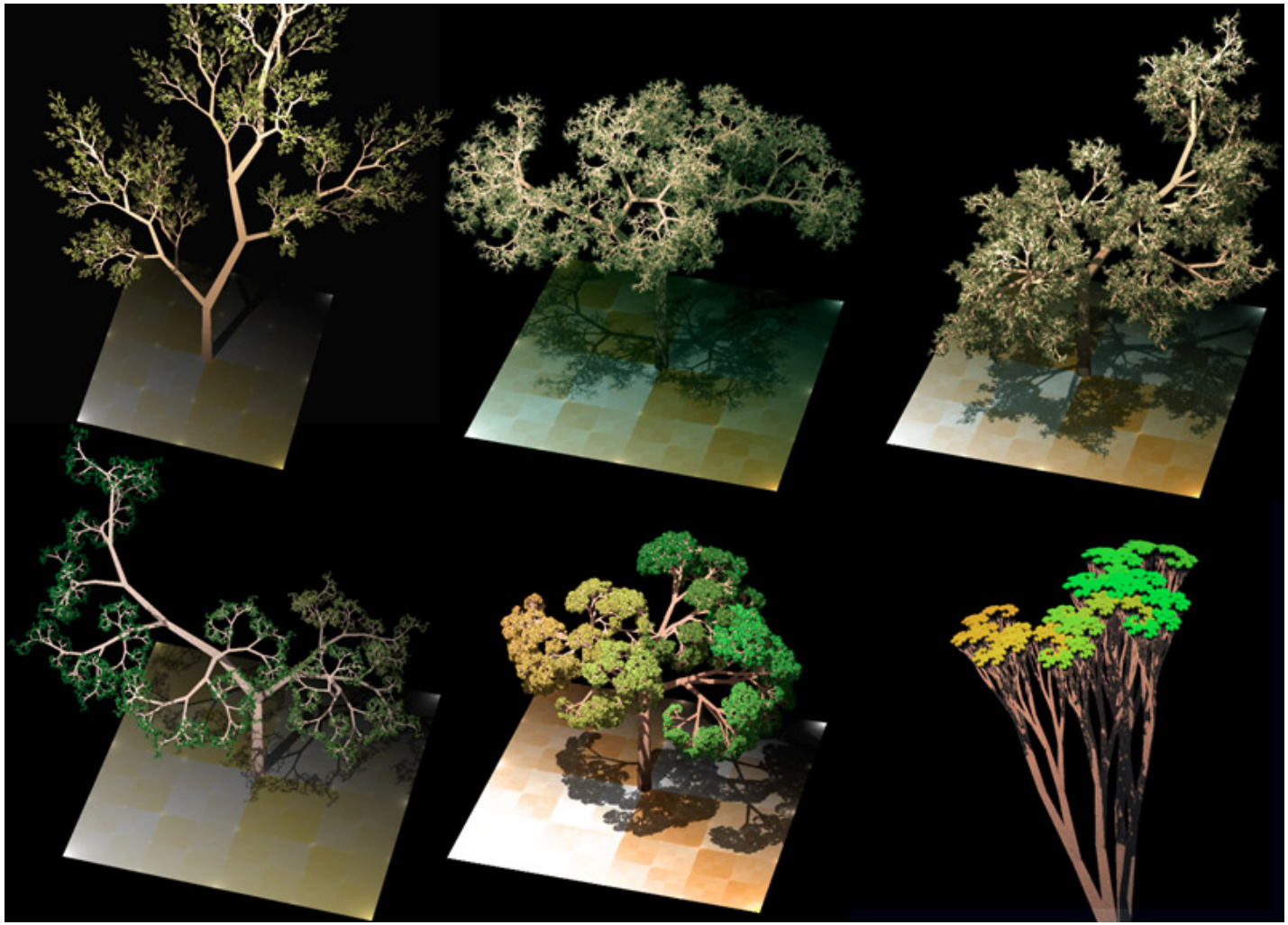


Parameterized L-Systems

Not just parameterized symbols!

- Randomized rule-selection
- Parameterization based on depth
- Changes in parameters over time

L-System Examples



SpeedTree

Leading vegetation generator:

http://www.speedtree.com/

https://www.youtube.com/watch?v=N2wmmdKzp8E

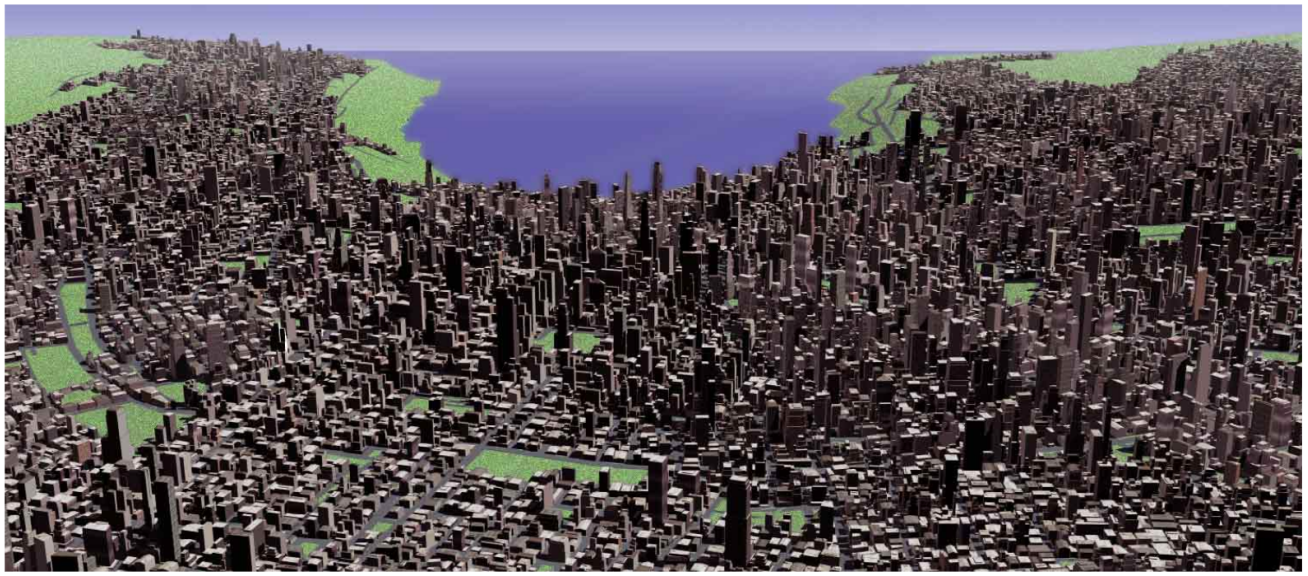
TreeIt (a free L-System I use):

http://www.evolved-software.com/treeit/treeit

Generating Cities

Same idea with different symbols and rules

- Good idea to having working understanding of the modeled system



https://graphics.ethz.ch/Downloads/Publications/Papers/2001/p_Par01.pdf

City Generation

What are the “rules” for generating a city?

City Generation

What are the “rules” for generating a city?

Based on terrain, types of buildings and plots, population, architecture style, and city history (i.e. planned versus sprawling)

Example: Home Free

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

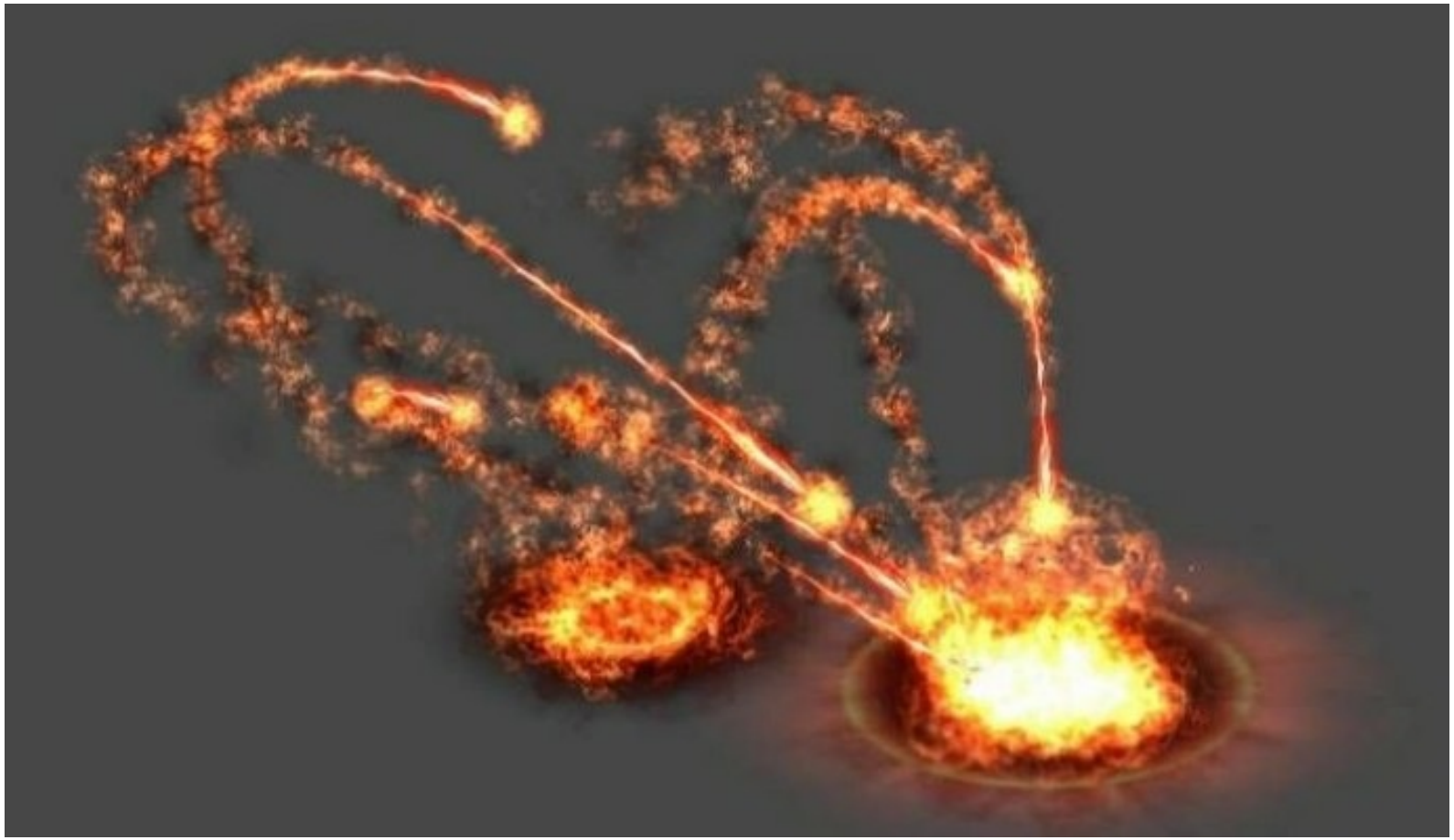
Example: Infamous

Additional Reading

http://algorithmicbotany.org/papers/graphical.gi86.pdf

http://algorithmicbotany.org/papers/

Particle Effects



General Particle Systems

- Particles treated as point masses with orientation
- Simple rules control how they move
- Controlled/rendered to simulate different “group” phenomenon
  - Fireworks
  - Waterfalls, spray, foam,
  - Explosions
  - Clouds/Atmospherics
  - Crowds/herds

Particle System Steps

1. Inject new particles into system with individual attributes
   - Generated at source(s)
2. Remove particles that exceed lifespan
   - Fixed lifespan or death upon some condition
3. Move current particles
   - Script provides rules for movement
4. Render current particles
   - Billboards, shaders, etc

Particle Generation

- Expensive to create and destroy objects
- We also want coherency in memory
- Use pools to solve both problems!

- Sources can vary based on desired effect
  - Random generation (e.g. clouds)
  - Stream generation (e.g. waterfall)
  - Scripted generation (e.g. fireworks)

Particle Movements

Rules of movement based on desired behavior:

- Emulate laws of physics
- Use environmental conditions
- Use particle neighborhood

Particle death also defined by rules

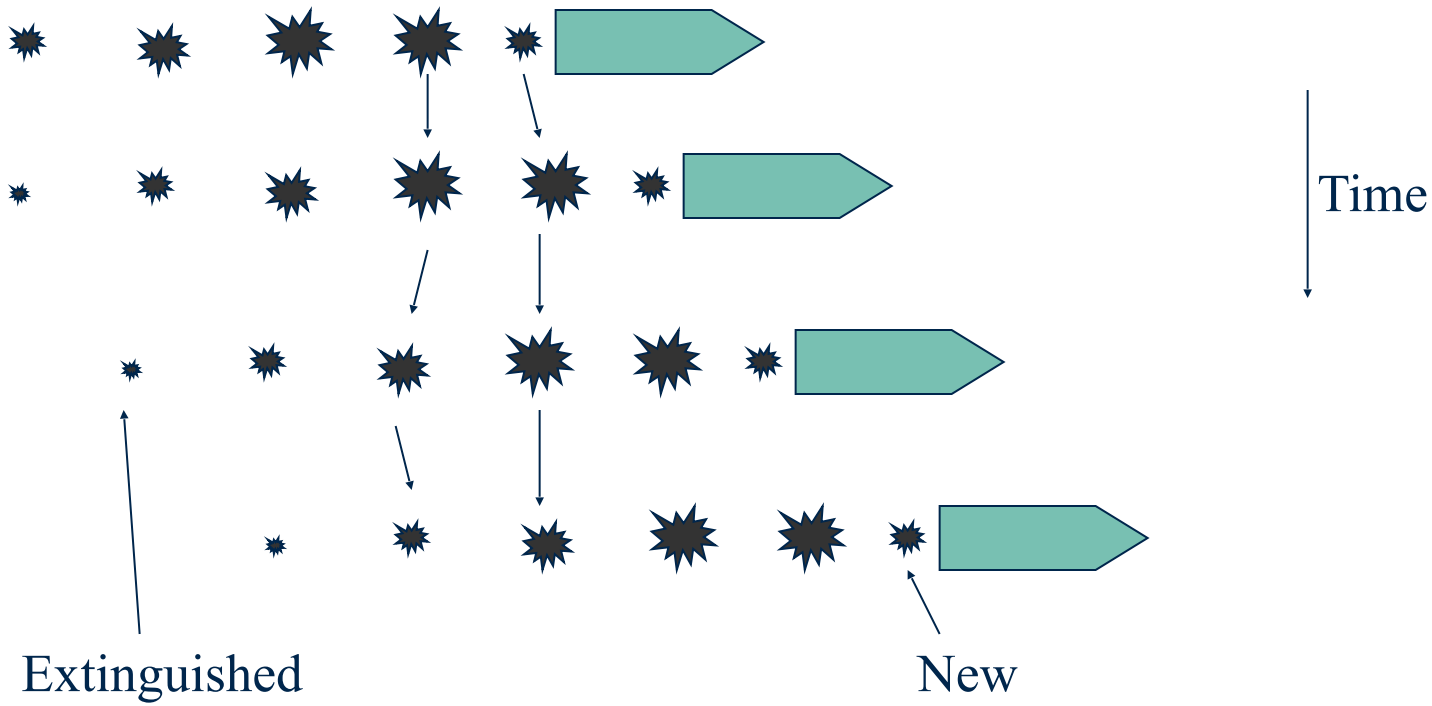
Concept: Smoke Trails

We want to create a rocket that leaves a smoke trail in its wake

What do we need to consider in terms of particle creation, movement, death, and appearance?

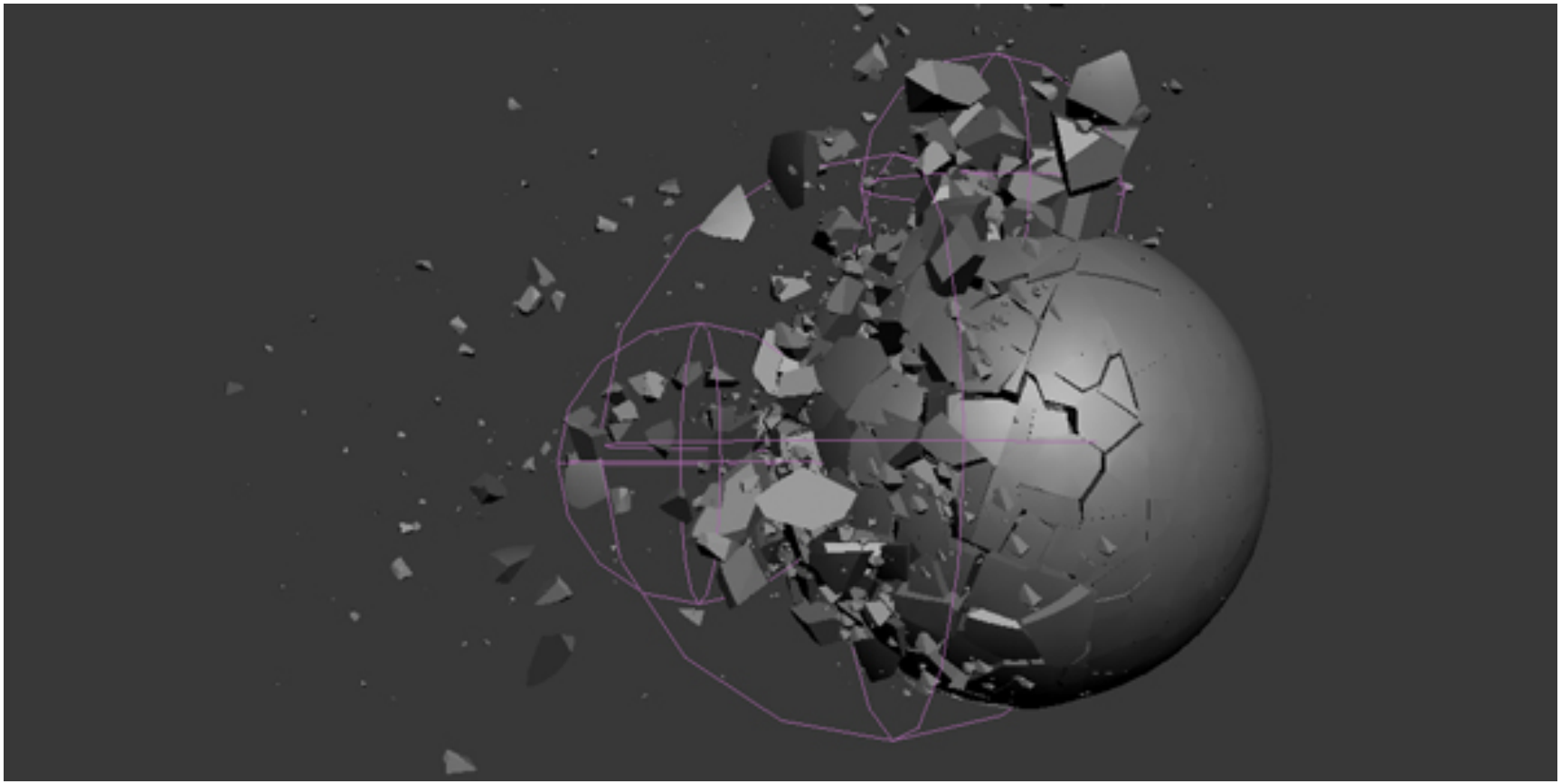
Example: Smoke Trails

1. Spawned at constant rate from end of rocket
   - Their initial velocity is 0 m/s (or perhaps some small velocity away from the rocket)
   - Given a density value that grows rapidly then falls off slowly
2. Movement in vertical direction (rise or fall as if from wind)
3. Extinguished when density is below some threshold
4. Render as a billboard facing the view or in shaders
   - Size and color of smoke puff based on density


Time
Extinguished
New

Example: Object Fracturing

- System starts when target breaks
- As target breaks into pre-determined pieces, a particle is assigned to each piece
- Each particle gets an initial velocity away from the center of the explosion
- Movement rules:
  1. Move ballistically unless collision
  2. Compute rigid body rotation or generate random rotation
  3. Resolve any collisions elastically
- Render target geometry with particle location and orientation



Laurent Renaud (http://cgcookie.com/max/2009/08/18/creating-an-exploding-planet/)

Particles in Games

They’re everywhere!

https://www.youtube.com/watch?v=6_NsaYtooQA

Particles in Movies

https://www.youtube.com/watch?v=A4QuKwfv6Wk

Particles in Movies

https://www.youtube.com/watch?v=ent02yltm60

Flocking Behavior

Particles can also model flocks, swarms, crowds, etc

(https://portraitsofwildflowers.wordpress.com/2011/12/10/grackles-revisited/)

Flocks and Herds

Idea: Flocks and herds are composed of individual, autonomous agents.

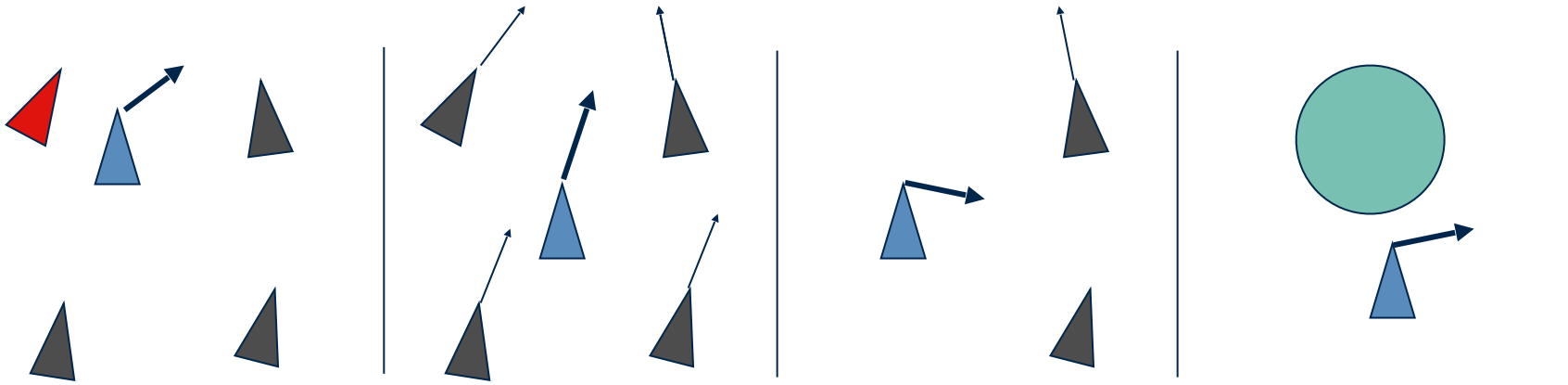
Goal: Define simple individual rules to obtain global emergent behavior

Flocking Models

- Each flock member follows same set of movement rules
- Rules contribute to the member's ultimate direction, velocity, acceleration, etc
- Different rules can create different forms of group behavior (flocks vs schools vs herds vs crowds)

www.cs.toronto.edu/~dt/siggraph97-course/cwr87/

Flocking Rules



Separation: fly away away from neighbors that are “too close”	Alignment: steer toward average velocity	Cohesion: steer toward average position	Avoidance: steer away from obstacles

How can these rules be combined?

Combining Rules

- Each rule acts as an acceleration in a direction based on its weight
  - e.g. high avoidance and low cohesion

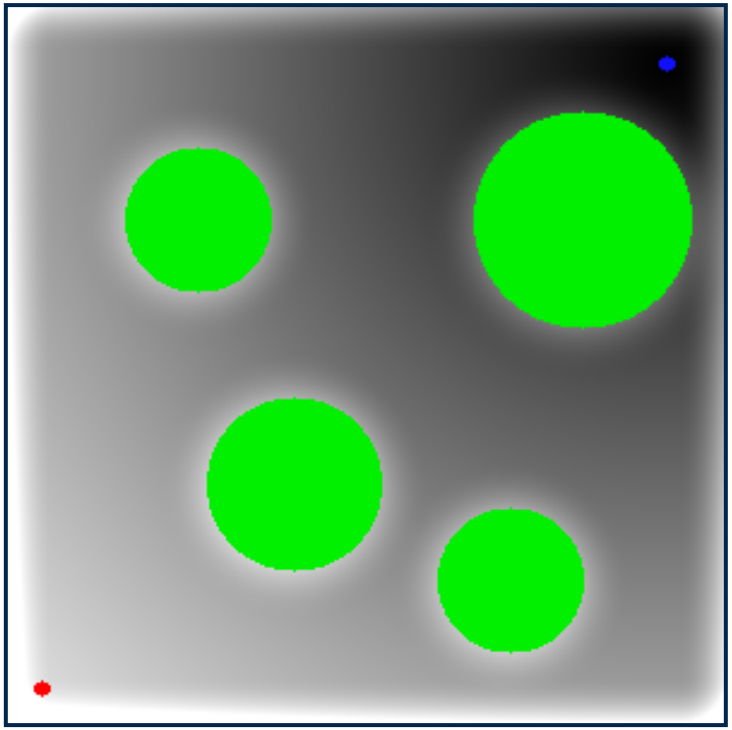
Combine using one of these strategies:

1. Apply in order of highest weight until maximum acceleration is reached
2. Take weighted sum and truncate to maximum acceleration

How do these combination strategies differ in practice?

Using Potential Fields

- Models each object as having an outward force field that pushes all other objects away from it
- Useful way to keep agents from colliding with each other or additional obstacles
- Common technique in motion planning



Flocking Demo

https://www.youtube.com/watch?v=rN8DzlgMt3M

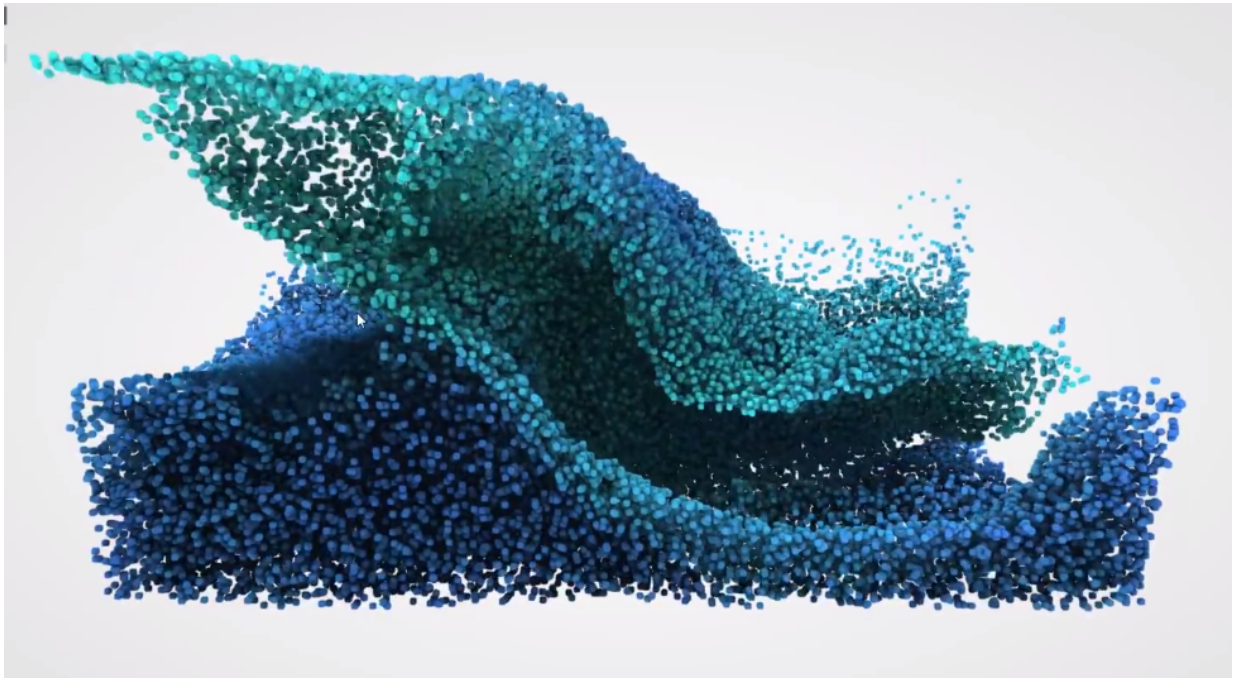
Particles as Fluid

Particles also work well for modeling fluid simulation!

Particle-based is one of the two over-arching categories of fluid simulation (the other being grid-based)

But we'll talk (a little) about this next time!

Particle-based Fluid Simulation



https://www.youtube.com/watch?v=DhNt_A3k4B4