Procedural shapes

CMSC425.01 Spring 2019

Administrivia

- Exam graded, solutions posted
- Project 2b due Tuesday, April 16^{th}

Today’s question

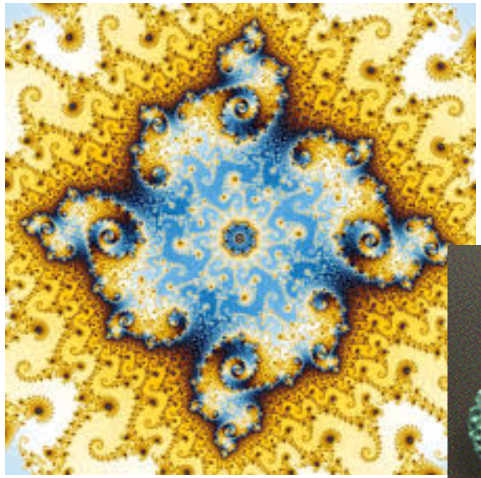
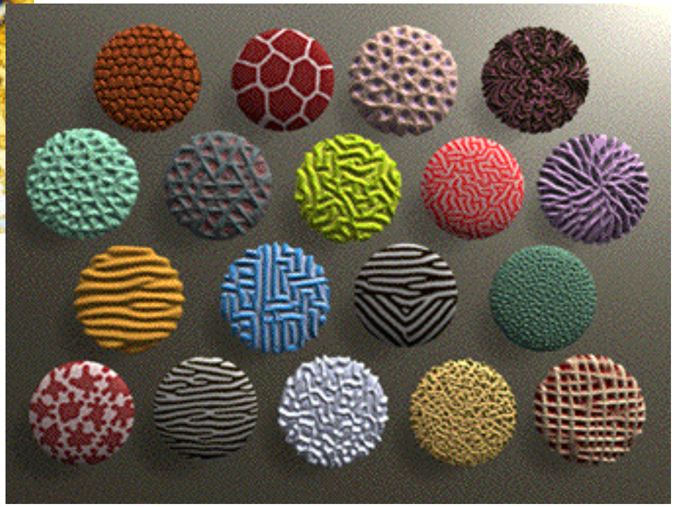
How to create interesting shape assets

Autogenerating terrain, objects, animations

- Why
  - Efficiency – faster than by hand
  - Variety – generate variations on a template
  - LOD – Generate level of detail as needed (*billboarding* in Unity)
- How:
  - Office procedural generation: create, store mesh, add
  - Online – during game: shape represented by subroutine/object, not a fixed
- What:
  - 3D shapes, 2D textures
  - Fake real (eg, trees), non-real things

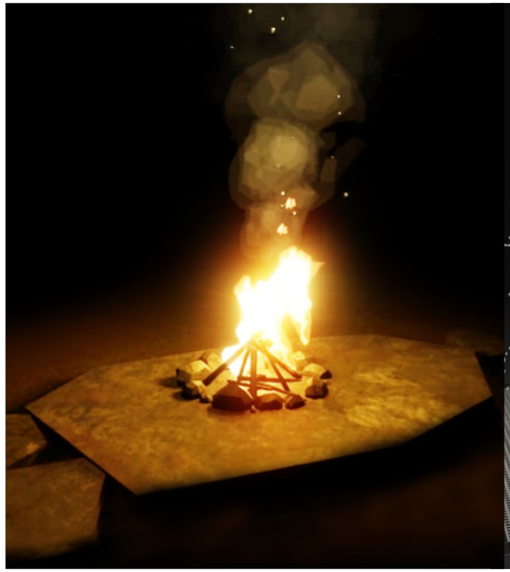
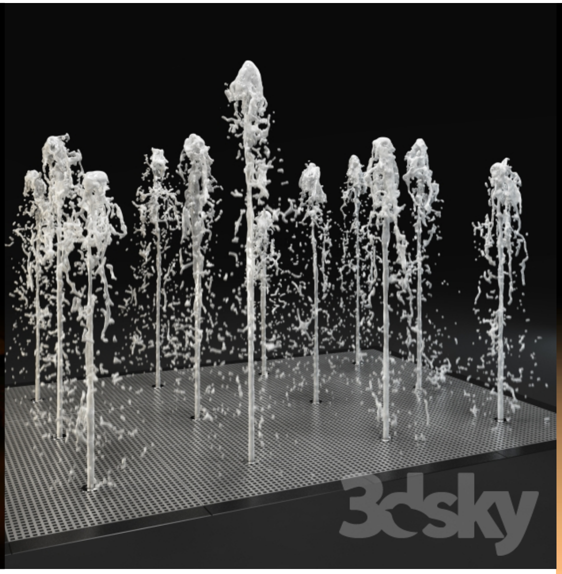
“Fractal” examples

- ***Midpoint displacement terrain***
- Particle systems
- **Lindermeyer Systems (L-systems)**
- Perlin noise
- Attractor sets
- *Iterated function systems (IFS)*
- *Reaction-diffusion textures*
- *Recursive fractals*

Particle systems (no Mount lecture)

- Reeves, *Particle Systems A Technique for Modeling a Class of Fuzzy Objects, 1983*
  - https://www.lri.fr/~mbl/ENS/IG2/devoir2/files/docs/fuzzyParticles.pdf
- Used for fire, explosions, smoke, water, hair, grass, and more …

Particle system basics

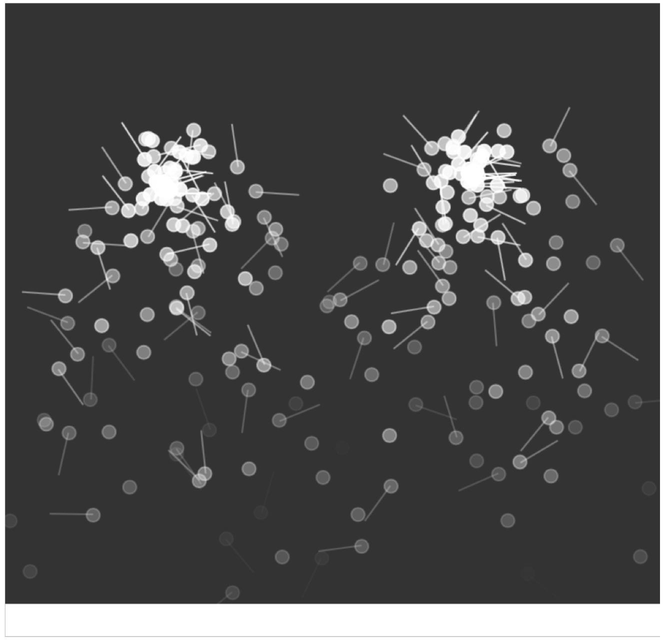
- Shiffman, Nature of Code link:
- https://natureofcode.com/book/chapter-4-particle-systems/
- Khan Academy
- https://www.khanacademy.org/computing/computer-programming/programming-natural-simulations/programming-particle-systems/a/particle-types

- https://processing.org/examples/multipleparticlesystems.html

Particle properties

```
class Particle {
        PVector position;
        PVector velocity;
        PVector acceleration;
        float lifespan;
```

- Motion
  - Position
  - Forces (gravity, wind, explosion/impetus, friction)
- Lifetime
- Rendering
  - Color/size as function of position/motion
  - Rendering style and persistence



Basic algorithm

- Step 1: Generate new particles
  - Apply constraints on initial velocity
- Step 2: Retire particles past their lifetime
- Step 3: Simulate motion for all existing particles
  - Apply forces to get acceleration
  - Apply acceleration to get motion
  - Move
  - Handle collisions

Dynamics of particle interaction

- Karl Sims 1989. Modeling
- https://vimeo.com/114622025

- SIGGRAPH 2014
- https://vimeo.com/94622661

Forces

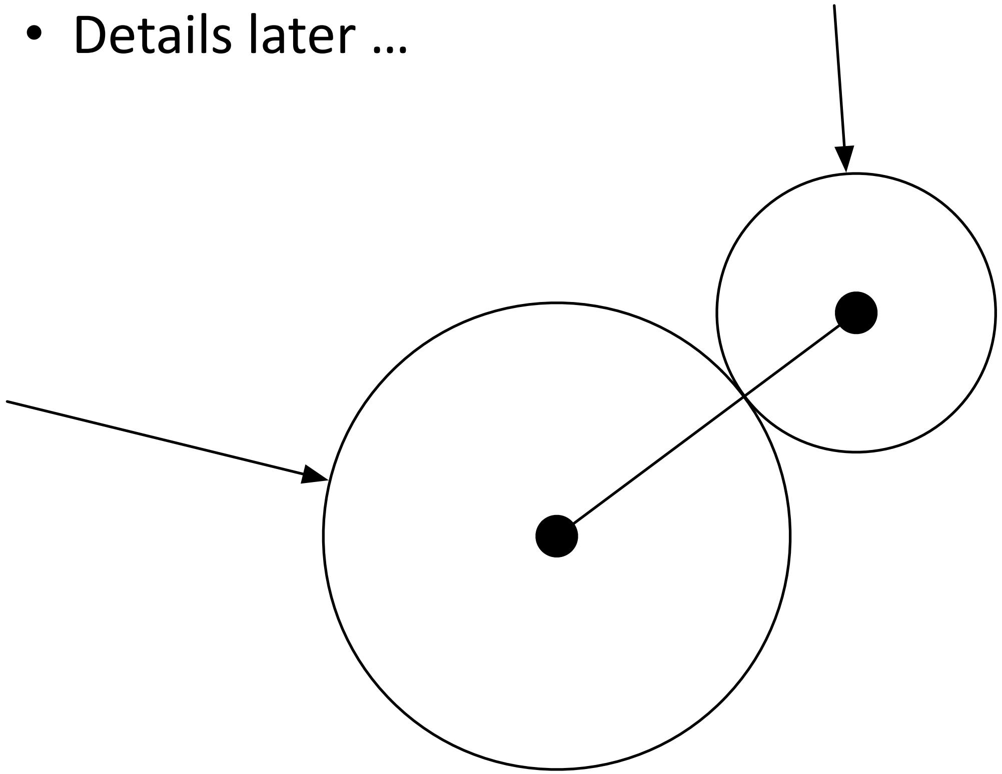
- From environment: gravity, viscosity (drag)
- From objects in scene: repulsion, attraction forces; collisions
- From fellow particles: spring, flocking

Bouncy Bubbles

- https://processing.org/examples/bouncybubbles.html

- Elastic collision
- Details later …

```
void move() {
  vy += gravity;
  x += vx; y += vy;
  if (x + diameter/2 > width)
     {  // Hit right wall
     x = width - diameter/2;
     vx *= friction;
     }
```

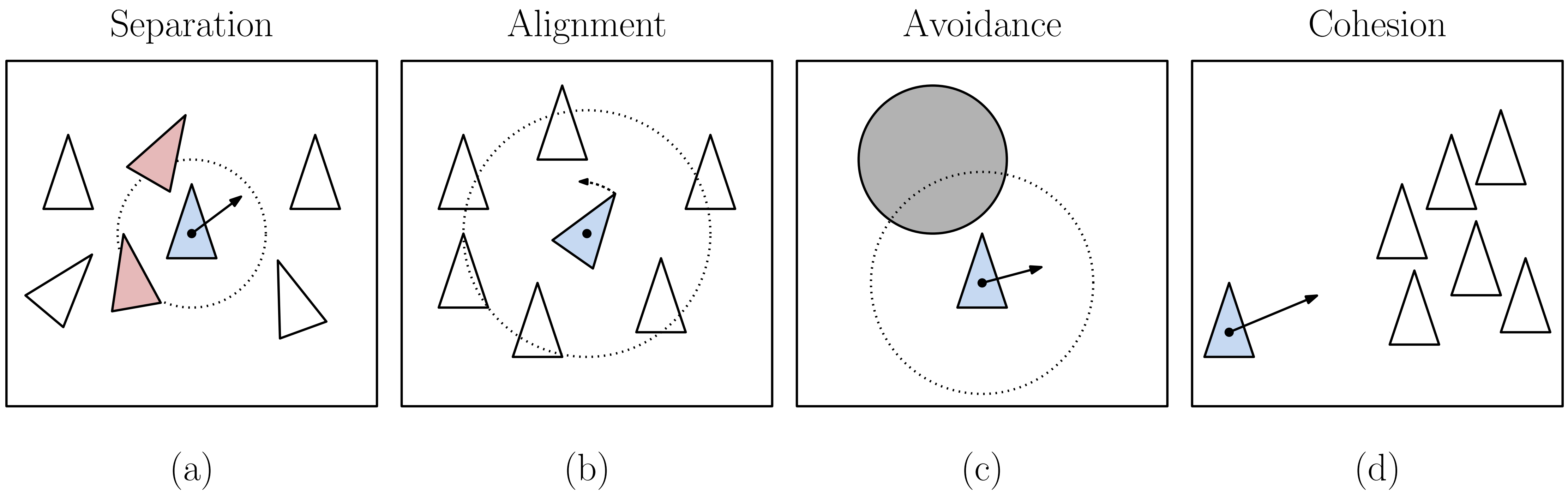


Flocking force (Mount

- For textiles, bird flocks, other common motion
- https://www.youtube.com/watch?v=gUF7ObEat0U

- Boids
- Craig Reynolds 1986

Flocking forces



Forces on boids

- Compute forces Fi at time t
- Compute acceleration vector ai
  - Scale forces by mass i
- Compute velocity from acceleration
- Compute position from velocity

$$\vec{a}_i(t) \;\leftarrow\; \frac{\vec{F}_i(t)}{m_i}.$$

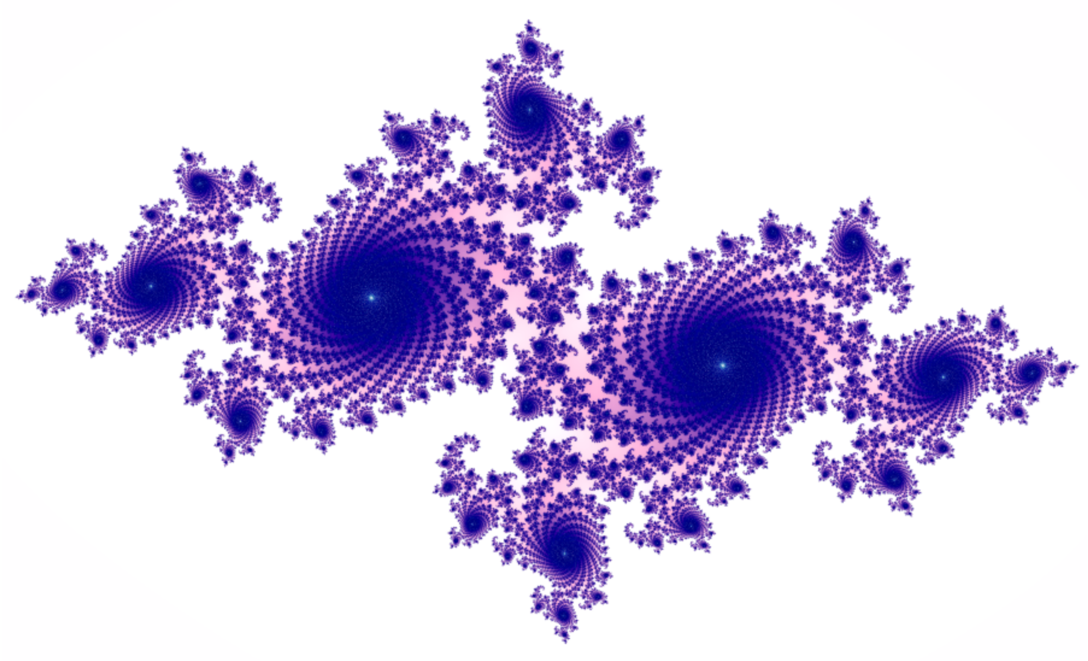
$$\vec{v}_i(t+\Delta) \;\leftarrow\; \vec{v}_i(t) + \Delta \cdot \vec{a}_i(t).$$

$$p_i(t+\Delta) \;\leftarrow\; p_i(t) + \Delta \cdot \vec{v}_i(t+\Delta)$$

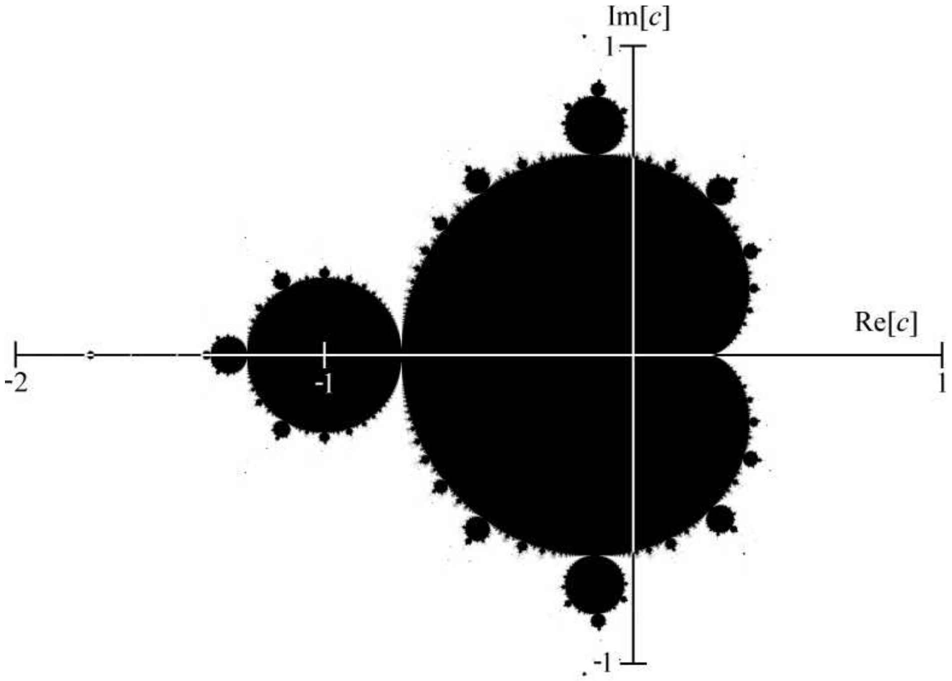
Iterated Functions and Attractor sets

Julia set

Mandelbrot set



(a)



(b)

Iterate complex function

- Complex numbers review

Modulus $\sqrt{a^2 + b^2}$.

- Definition

$$a + bi, \text{ where } i^2 = -1.$$

- Addition

$$(a + bi) + (c + di) = (a + c) + (b + d)i,$$

- Multiplication

$$(a + bi)(c + di) = ac + adi + bci + bdi^2 = (ac - bd) + (ad + bc)i.$$

Iterate complex function

- Iterate simple function of z

$$z_i \leftarrow z_{i-1}^2 + c \qquad \text{for } i = 1, 2, 3, \ldots$$