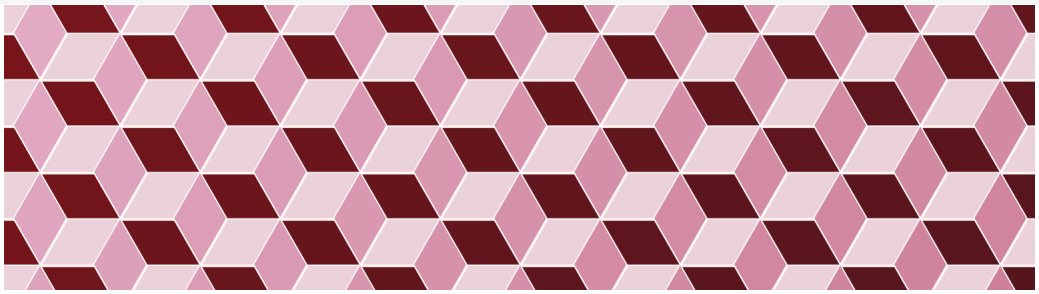


15-104 Introduction to Computing for Creative Practice

Fall 2021

22 Mutual Interaction (Physics of Motion)

Instructor: Tom Cortina, tcortina@cs.cmu.edu, GHC 4117, 412-268-3514

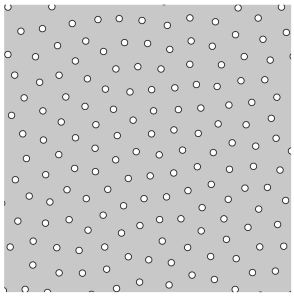
1

Mutual Repulsion

- In mutual interaction, the idea is that particles try to move away from each other.
- To implement this idea, we represent "try to move away" as a force.
  - The closer particles are together, the greater the force that pushes them apart.
  - We use the inverse square law and make the force proportional to 1/(distance*distance).
  - As distance increases, this force approaches zero, but at small distances, e.g. distance = 1, it is significant.
  - Caution: the force is infinite at zero distance, so to avoid huge forces and the resulting high velocities, we do not apply the force when distance is very small.

2

Example



```
// make a new particle (constructor)
function makeParticle(x, y, dx, dy) {
    var p = {px: x, py: y, vx: dx, vy: dy,
             mass: 1.0, damping: 0.9,
             bFixed: false,
             bLimitVelocities: false,
             bPeriodicBoundaries: false,
             bHardBoundaries: false,
             addForce: particleAddForce,
             update: particleUpdate,
             limitVelocities: particleLimitVelocities,
             handleBoundaries: particleHandleBoundaries,
             draw: particleDraw
            }
    return p;
}
```

Functions in red not shown in slides.

3

Repel: setup

```
var myParticles = [];
var nParticles = 400;
function setup() {
    createCanvas(400, 400);
    for (var i = 0; i < nParticles; i++) {
        var rx = random(width);
        var ry = random(height);
        myParticles[i] = makeParticle(rx, ry, 0, 0);
    }
}
function keyPressed() {
    for (var i = 0; i < myParticles.length; i++) {
        myParticles[i].px = random(width);
        myParticles[i].py = random(height);
    }
}
```

4

How many pairs of particles?

- We need to compute the force between each pair of particles.
- If we have *n* particles, how many unique pairs are there?

Number of particles	Unique pairs
2	1
3	3
4	6
5	10
6	15
n	1 + 2 + … + (n-1)

Six particles:
A B C D E F

```
AB AC AD AE AF
BC BD BE BF
CD CE CF
DE DF
EF
```

Or

```
BA
CA CB
DA DB DC
EA EB EC ED
FA FB FC FD FE
```

5

Repel: draw

```
function draw() {
    background(200);
    var gravityForcex = 0;
    var gravityForcey = 0.05;
    var mutualRepulsionAmount = 2.5;
    for (var i = 0; i < myParticles.length; i++) {
        var ithParticle = myParticles[i];
        var px = ithParticle.px;
        var py = ithParticle.py;
        for (var j = 0; j < i; j++) {
            var jthParticle = myParticles[j];
            var qx = jthParticle.px;
            var qy = jthParticle.py;
            ...  // add forces to each particles i and j
        }
    } ...
```

6

The nested loops (for 5 particles)

```
for (var i = 0; i < 5; i++) {

    for (var j = 0; j < i; j++) {


    }

}
```

Could we have had the inside loop run while `j < 5` ?
If no, why not?
If yes, why didn't we do this?

j

i	0	1	2	3	4
0					
1	X				
2	X	X			
3	X	X	X		
4	X	X	X	X	

Particle pairs that have forces computed.

7

Another way…

```
for (var i = 0; i < 5; i++) {

    for (var j = i+1; j < 5; j++) {


    }

}
```

j

i	0	1	2	3	4
0		X	X	X	X
1			X	X	X
2				X	X
3					X
4					

Particle pairs that have forces computed.

8

Repel: draw

```
function draw() {
    background(200);
    var gravityForcex = 0;
    var gravityForcey = 0.05;
    var mutualRepulsionAmount = 2.5;
    for (var i = 0; i < myParticles.length; i++) {
        var ithParticle = myParticles[i];
        var px = ithParticle.px;
        var py = ithParticle.py;

        for (var j = 0; j < i; j++) {
            var jthParticle = myParticles[j];
            var qx = jthParticle.px;
            var qy = jthParticle.py;
            // cont'd...
```

9

Repel: draw

```
            var dx = px - qx;
            var dy = py - qy;
            var dh = sqrt(dx * dx + dy * dy);
            if (dh > 1.0) {
                var componentInX = dx / dh;
                var componentInY = dy / dh;
                var proportionToDistanceSquared = 1.0 / (dh * dh);
                var repulsionForcex = mutualRepulsionAmount *
                    componentInX * proportionToDistanceSquared;
                var repulsionForcey = mutualRepulsionAmount *
                    componentInY * proportionToDistanceSquared;
                // add in forces
                ithParticle.addForce( repulsionForcex,  repulsionForcey);
                jthParticle.addForce(-repulsionForcex, -repulsionForcey);
            }
        }  // cont'd...
```

How would you compute dh using the `dist` function?

```
dh = dist(px,py,qx,qy);
```

Why are we checking `dh` ?

Why negative values here?

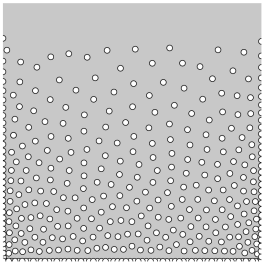
10

Repel: draw

```
for (var i = 0; i < myParticles.length; i++) {
       myParticles[i].bPeriodicBoundaries = false;
       myParticles[i].bElasticBoundaries = true;
       myParticles[i].update(); // update all particles
   }
for (var i = 0; i < myParticles.length; i++) {
       myParticles[i].draw(); // draw all particles
   }
}
```

11

Repel with Gravity



```
...
for (var i = 0; i < myParticles.length; i++) {
       var ithParticle = myParticles[i];
       var px = ithParticle.px;
       var py = ithParticle.py;
       if (mouseIsPressed) {
           ithParticle.addForce(gravityForcex, gravityForcey);
       }
       for (var j = 0; j < i; j++) {
           var jthParticle = myParticles[j];
           var qx = jthParticle.px;
           var qy = jthParticle.py;
           ...
```

12

Efficiency

- If we have n particles, how much work is done?
  (What is the algorithmic complexity of this algorithm?)
- The predominant amount of work is done in the nested loop.
- In the worst case, the number of times we compute the inside of the inner loop is:
  $S = 1 + 2 + \ldots + (n - 2) + (n - 1)$ (Why?)
- What is S as a function of n in closed form (no ellipses)?

$$
\begin{array}{llllll}
 & S = (n-1) & + (n-2) & + \ldots & + 2 & + 1 \\
+ & S = 1 & + 2 & + \ldots & + (n-2) & + (n-1) \\
\hline
 & 2S = n & + n & + \ldots & + n & + n \quad = (n-1)\,n
\end{array}
$$

$S = (n - 1)\, n / 2 = n^2/2 + n/2 \rightarrow O(n^2)$

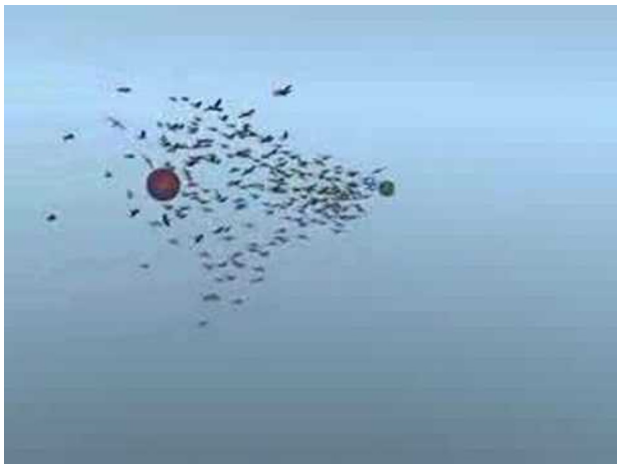
13

Flocking

- Flocking is inspired by flocks of birds that achieve mass coordination of movement even though each bird is making independent decisions.
- How does the flocking behavior emerge from local behavior?
- In the flocking model, there are 3 rules:
  - Separation — avoid collisions with neighbors
  - Alignment — steer in the same direction as your neighbors
  - Cohesion — steer to the center of your neighbors to stay together

14

Flocking



https://www.youtube.com/watch?v=GUkjC-69vaw

15

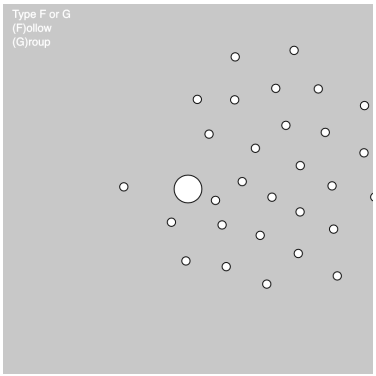
Flocking example

This example is based on code by Conrad Parker.

There is an obstacle in the middle that the particles will try to avoid.

The following keys modified the simulation:

- "S" key randomizes particle locations
- "G" key toggles grouping
- "F" key toggles following



16

Flock – setup

```
var myParticles = [];
var nParticles = 30;
var group = true;
var follow = true;
function setup() {
    createCanvas(400, 400);
    for (var i = 0; i < nParticles; i++) {
        var rx = random(width);
        var ry = random(height);
        var p = makeParticle(rx, ry, 0, 0);
        p.bHardBoundaries = true;
        p.damping = 0.99;
        myParticles[i] = p;
    }
    frameRate(10);
});
```

Booleans to store whether grouping and following are turned on or off.

17

Flock – keyPressed

```
function keyPressed() {
    if (key === "S") {
        for (var i = 0; i < myParticles.length; i++) {
            myParticles[i].px = random(width);
            myParticles[i].py = random(height);
        }
    } else if (key === "G") {
        group = !group;
    } else if (key === "F") {
        follow = !follow;
    }
}
```

What are these two Boolean statements essentially doing?

18

Flock – draw (excerpt)

```
for (var i = 0; i < myParticles.length; i++) {
     var ithParticle = myParticles[i];
     ithParticle.fx = 0;
     ithParticle.fy = 0;
     separate(ithParticle);
     alignment(ithParticle);
     cohesion(ithParticle);
     avoid(ithParticle, height / 2, width / 2, 40);
     if (follow) seek(ithParticle);
 }
 for (var i = 0; i < myParticles.length; i++)
 {
     var p = myParticles[i]
     p.update(); // update all locations
     p.draw(); // draw all particles
 }
```

Applies force in opposite direction if particle is closer than a set separation value. (not shown)

Applies force to align particles over time to move in the same direction.

Applies force to steer particles toward the center of the flock.

Applies force to avoid the obstacle in the center.

Applies force to follow the mouse if feature is turned on.

19