**sketch09.js**

// Seeking a target (multiple Things, all identical except for

// starting position, chasing the mouse)

// This example of Craig Reynolds'steering formula in action

// (steering = desired-velocity) is from Dan Shiffman's

// Processing book, The Nature of Code and modified for p5.js by al.

// al 14 November 2016

var gPop = 200;

var gThings = [];

function setup() {

createCanvas(1000,600);

background("#fefefe")

gThings.length = gPop;

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

gThings[i] = new Thing(random(width),random(height));

}

createP("Target seeking (mouse) with Reynolds' steering and avoidance behaviours");

createP("sketch09.js");

}

function draw() {

gTarget = createVector(mouseX,mouseY);

background("#fefefe");

for (var i = 0; i < gThings.length; i++) {

gThings[i].seekTarget(gTarget);

gThings[i].separate(gThings);

gThings[i].update();

gThings[i].render();

}

}

function Thing(startX,startY) { // thing constructor

this.d = 40;

this.maxSpeed = 4;

this.maxForce = 0.04;

// euler integration physics engine

this.acc = createVector(0,0);

this.vel = createVector(0,0);

this.pos = createVector(startX,startY);

// vectors for Reynolds' steering formula

this.desired = createVector(0,0);

this.steering = createVector(0,0);

this.update = function() {

this.vel.add(this.acc);

this.vel.limit(this.maxSpeed);

this.pos.add(this.vel);

this.acc.mult(0); // reset acc after each update

} // update

this.applyForce = function(aForce) {

this.acc.add(aForce);

} // applyForce

this.seekTarget = function(target) {

this.desired = p5.Vector.sub(target,this.pos);

this.distTarget = this.desired.mag();

this.desired.normalize();

this.desired.mult(this.maxSpeed);

this.steering = p5.Vector.sub(this.desired,this.vel);

this.steering.limit(this.maxForce);

this.applyForce(this.steering);

} // seekTarget

this.separate = function(thingArray) {

this.desSep = this.d\*1.1// desired separation in body widths

closeSum = createVector();

closeCount = 0;

for (var i = 0; i < gThings.length; i++) {

locSep = p5.Vector.dist(this.pos,gThings[i].pos);

// print(sepLocal);

if (locSep>0 && locSep<this.desSep){

diffVect = p5.Vector.sub(this.pos,gThings[i].pos);

diffVect.normalize();

diffVect.div(locSep);

closeSum.add(diffVect);

closeCount++;

}

}

if (closeCount>0) {

closeSum.div(closeCount);

closeSum.normalize();

closeSum.mult(this.maxSpeed);

this.steering = p5.Vector.sub(closeSum,this.vel);

this.steering.limit(this.maxForce);

this.applyForce(this.steering);

}

}

this.render = function() {

noStroke();

fill(127,127,0,72);

ellipse(this.pos.x,this.pos.y,this.d);

} // render

} // Thing