import org.openkinect.*; import org.openkinect.processing.*;

//turns off the Kinect sensing, uses the mouse as input Boolean debugMode = true;

// Showing how we can farm all the kinect stuff out to a separate class KinectTracker tracker; // Kinect Library object Kinect kinect;

Balloon[] balloons = new Balloon[150]; color fillColor = color(165,33,26);

int direction = -1;

void setup() { size(1024,768);

if (!debugMode){ kinect = new Kinect(this); tracker = new KinectTracker(); }

for (int i = 0; i < balloons.length; i++) {

float mass = random(200.0, 400.0);  
balloons[i] = new Balloon(  
 mass,  
 random(50, width - 50),  
 random(50, height - 50),   
 fillColor);

} }

void draw() { background(255);

if (!debugMode){

// Run the tracking analysis  
tracker.track();  
  
// Let's draw the raw location  
PVector v1 = tracker.getPos();  
float force = tracker.getForce();  
if (tracker.tracking) {  
  
 fill(100);  
 noStroke();  
 ellipse(v1.x, v1.y, 20, 20);  
}  
  
//for every balloon   
for (int i = 0; i < balloons.length; i++) {  
 if (tracker.tracking){  
 balloons[i].repel(v1, force);   
 }  
 balloons[i].run();  
}

}

if (debugMode){ PVector mouse = new PVector(mouseX, mouseY); fill(100); noStroke(); ellipse(mouse.x, mouse.y, 20, 20);

for (int i = 0; i < balloons.length; i++) {  
 balloons[i].repel(mouse, 100);   
 balloons[i].run();  
}

}

} void keyPressed() {

if (!debugMode){

//make it easy to adjust our threshold  
int t = tracker.getThreshold();  
if (key == CODED) {  
 if (keyCode == UP) {  
 t+=5;  
 tracker.setThreshold(t);  
 }   
 else if (keyCode == DOWN) {  
 t-=5;  
 tracker.setThreshold(t);  
 }  
}

}

//if we hit space, change the color and direction! if (key == ‘’) { direction \*= -1; for (int i = 0; i < balloons.length; i++) { balloons[i].changeColor(); } } }

void stop() { tracker.quit(); super.stop(); }