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

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

//don’t start off in correction mode Boolean correctionMode = false;

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

//the number of superPixels (things start to slow down quickly if this is increased int xPixels = 100; int yPixels = 76;

float xSize, ySize, x, y; color pixelFill; color backColor = #9933FF; boolean gravity = false;

float force; float baseForce = 5;

superPixel[][] pixelArray = new superPixel[xPixels][yPixels];

void setup() { //a little small, so we don’t get artifacts size(1000, 760); noStroke();

//if we’re not in debug mode, initialize the Kinect if (!debugMode){ kinect = new Kinect(this); }

tracker = new KinectTracker();

//initialize each superPixel, with a nice blueish color for (int i = 0; i < xPixels; i++) { for (int j = 0; j < yPixels; j++) {

xSize = (float)width/xPixels;  
 ySize = (float)height/yPixels;  
 x = (float)xSize \* (i);  
 y = (float)ySize \* (j);  
  
 color pixelFill = color(50);  
  
 pixelArray[i][j] = new superPixel(x, y, pixelFill, xSize, ySize);  
}

} }

void draw() {

background(backColor);

//if we're in correction mode

if (correctionMode){ //show us what the depth camera is collecting //tracker.display();

fill(25);  
text(tracker.getModeName() + " Correction", 10, 20);  
text("Offset: " + tracker.getOffset(), 10, 35);

}

if(!debugMode){

tracker.track();  
  
//looks fucking awesome, but CPU intensive  
//tracker.display();  
  
if (tracker.tracking){  
 float force = tracker.getForce();  
   
 if (gravity){  
 force = force \* 5;  
 }  
   
 PVector position = tracker.getPos();  
   
 for (int i = 0; i < xPixels; i++) {  
 for (int j = 0; j < yPixels; j++) {  
 pixelArray[i][j].explode(force, position);  
 }  
 }  
}

}

if (debugMode){ PVector mouse = new PVector(mouseX, mouseY);

force = baseForce;  
  
if (gravity){  
 force = baseForce \* 5;  
}  
  
if (mousePressed && (mouseButton == LEFT)) {  
 for (int i = 0; i < xPixels; i++) {  
 for (int j = 0; j < yPixels; j++) {  
 pixelArray[i][j].explode(force, mouse);  
 }  
 }   
}

}

for (int i = 0; i < xPixels; i++) { for (int j = 0; j < yPixels; j++) { pixelArray[i][j].run(); } }

}

//if we hit a key void keyPressed() { //if we hit c, toggle between correction mode if (key == ‘c’) { int n = tracker.getCurrentMode(); n += 1; tracker.setCurrentMode(n); if (n <= 3){ correctionMode = true; } if (n > 3) { tracker.setCurrentMode(-1); correctionMode = false; }

}

if (correctionMode){ if (key == CODED) { if (keyCode == UP || keyCode == RIGHT) { tracker.setOffset(1); } else if (keyCode == DOWN || keyCode == LEFT) { tracker.setOffset(-1); } } }

//make it easy to adjust our threshold if (!debugMode &&! correctionMode){ int t = tracker.getThreshold(); if (key == CODED) { if (keyCode == UP) { t+=1; println(“Threshold:”+t); tracker.setThreshold(t); } else if (keyCode == DOWN) { t-=1; println(“Threshold:”+ t); tracker.setThreshold(t); } } }

//if we hit space, change the gravity! if (key == ‘’) { gravity = !gravity; if (!gravity){ force = tracker.getForce(); } println(“gravity:”+gravity); }

//make it easy to adjust our force while debugging if (debugMode &&! correctionMode){ if (key == CODED) { if (keyCode == UP) { force += 50; println(“force:”+force); } else if (keyCode == DOWN) { force -= 50; println(“force:”+force); } } } }

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