

```
1 let points = [];
2 let pressed = false;
3 let centre;
4 let drawTangentes = false;
5 let drawPixels = false;
6 let learning = true;
7 let learningRate = 0.1;
8 let momentum = 0.95;
9 let nbGradients;
10 let gradients_x = [];
11 let gradients_y = [];
12
13 function setup()
14 {
15   createCanvas(1920, 1080);
16   centre = {x: width / 2, y: height / 2};
17 }
18
19 function gradientDescent()
20 {
21   let tangentes = calculateTangentes();
22
23   let gradient_x = 0;
24   let gradient_y = 0;
25
26   for (let i = 0; i < points.length; i++)
27   {
28     let t = tangentes[i];
29     let cte = 2 * (t.a * centre.x + t.b * centre.y + t.c) / (points.length * (t.a * t.a + t.b * t.b));
30
31     gradient_x += t.a * cte;
32     gradient_y += t.b * cte;
33   }
34
35   for (let i = 0; i < gradients_x.length; i++)
36     gradients_x[i] *= momentum;
37
38   for (let i = 0; i < gradients_y.length; i++)
39     gradients_y[i] *= momentum;
40
41   gradients_x.push(gradient_x);
42   gradients_y.push(gradient_y);
43
44   if (gradients_x.length > nbGradients)
45     gradients_x.splice(0, 1);
46
47   if (gradients_y.length > nbGradients)
48     gradients_y.splice(0, 1);
49
50   let mean_x = gradients_x.reduce((a, x) => a + x);
51   let mean_y = gradients_y.reduce((a, x) => a + x);
52
53   centre.x -= mean_x * learningRate;
54   centre.y -= mean_y * learningRate;
55   nbGradients = int(log(0.02) / log(momentum));
56 }
57
58 function calculateAngle(x, y)
59 {
60   let a;
61
62   if (y >= 0)
63     a = acos(x / sqrt(x * x + y * y));
64   else
65     a = 2 * PI - acos(x / sqrt(x * x + y * y));
66
67   return a;
68 }
69
70 function mousePressed()
71 {
72   let p = {x: mouseX, y: mouseY, radius: 0};
73   points.push(p);
74
75   if (points.length > 3)
76     points.shift();
77
78   pressed = true;
79 }
80
81 function mouseReleased()
82 {
83   pressed = false;
84 }
85
86 function keyPressed()
87 {
88   if (keyCode == 115)
89   {
```

```
90     if (drawTangentes)
91     {
92         drawTangentes = false;
93         drawPixels = true;
94     }
95     else if (drawPixels)
96     {
97         drawPixels = false;
98     }
99     else
100    {
101        drawTangentes = true;
102    }
103 }
104
105 if (keyCode == 32 || keyCode == 13)
106     learning = !learning;
107 }
108
109 function calculateTangentes(x, y)
110 {
111     let pt;
112
113     if (x != undefined && y != undefined)
114         pt = {x: x, y: y};
115     else
116         pt = centre;
117
118     let tangentes = [];
119
120     for (let p of points)
121     {
122         let inter = {x: p.x, y: p.y};
123         inter.x += p.radius * (pt.x - p.x) / sqrt(pow(pt.x - p.x, 2) + pow(pt.y - p.y, 2));
124         inter.y += p.radius * (pt.y - p.y) / sqrt(pow(pt.x - p.x, 2) + pow(pt.y - p.y, 2));
125
126         let a = pt.x - inter.x;
127         let b = pt.y - inter.y;
128         let c = -a * inter.x - b * inter.y;
129
130         if (abs(a) > abs(b) && abs(a) > abs(c))
131         {
132             b /= a;
133             c /= a;
134             a = 1;
135         }
136         else if (abs(b) > abs(c))
137         {
138             a /= b;
139             c /= b;
140             b = 1;
141         }
142         else
143         {
144             a /= c;
145             b /= c;
146             c = 1;
147         }
148
149         tangentes.push({a: a, b: b, c: c});
150     }
151
152     return tangentes;
153 }
154
155 let histo = [];
156
157 function pixelisationDraw()
158 {
159     let pas = width / 50;
160
161     stroke(255);
162     strokeWeight(2);
163
164     for (let x = 0; x < width; x += pas)
165         for (let y = 0; y < height; y += pas)
166         {
167             let tangentes = calculateTangentes(x, y);
168             let erreur = 0;
169
170             for (let t of tangentes)
171                 erreur += Math.abs(t.a * x + t.b * y + t.c, 2) / (t.a * t.a + t.b * t.b);
172
173             let coeff = erreur / 1000000;
174
175             let rouge = {r: 255, g: 0, b: 0};
176             let bleu = {r: 0, g: 0, b: 255};
177
178             fill((1 - coeff) * rouge.r + coeff * bleu.r,
179                 (1 - coeff) * rouge.g + coeff * bleu.g,
```

```
180         (1 - coeff) * rouge.b + coeff * bleu.b);
181
182         rect(x, y, x + pas - 1, y + pas - 1);
183     }
184 }
185
186
187
188 function draw()
189 {
190     background(255);
191
192     if (pressed)
193     {
194         let p = points[points.length - 1];
195         p.radius = sqrt(pow(mouseX - p.x, 2) + pow(mouseY - p.y, 2));
196     }
197
198     if (centre !== undefined)
199     {
200         histo.push({x: centre.x, y: centre.y});
201
202         while (histo.length > 200)
203             histo.shift();
204     }
205
206     if (learning)
207         gradientDescent();
208
209     if (drawPixels)
210     {
211         pixelisationDraw();
212         return;
213     }
214
215     stroke(255, 64, 64);
216     strokeWeight(8);
217
218     for (let p of points)
219         point(p.x, p.y);
220
221     strokeWeight(4);
222     noFill();
223
224     for (let p of points)
225         ellipse(p.x, p.y, 2 * p.radius);
226
227     if (drawTangentes)
228     {
229         stroke(128, 128, 255);
230
231         for (let t of calculateTangentes())
232         {
233             let x0, y0, x1, y1;
234
235             if (t.b == 0)
236             {
237                 x0 = x1 = -t.c / t.a;
238                 y0 = 0;
239                 y1 = height;
240             }
241             else
242             {
243                 x0 = 0;
244                 x1 = width;
245                 y0 = -t.c / t.b;
246                 y1 = -width * t.a / t.b - t.c / t.b;
247             }
248
249             line(x0, y0, x1, y1);
250         }
251     }
252
253     stroke(128, 192, 0);
254     strokeWeight(16);
255     point(centre.x, centre.y);
256
257     strokeWeight(5);
258
259     let x = undefined;
260     let y = undefined;
261
262     for (let pt of histo)
263     {
264         if (x && y)
265             line(x, y, pt.x, pt.y);
266
267         x = pt.x;
268         y = pt.y;
269     }
```

270 }