```
let points = [];
let pressed = false;
let centre;
let drawTangentes = false;
let drawPixels = false;
let learning = true;
let learningRate = 0.1;
let momentum = 0.95;
let nbGradients;
let gradients_x = [];
let gradients_y = [];

function setup()
{
    createCanvas(1920, 1080);
    centre = {x: width / 2, y: height / 2};
}

function gradientDescent()
{
    let tangentes = calculateTangentes();

    let gradient_x = 0;
    let gradient_y = 0;

    for (let i = 0; i < points.length; i++)
    {
        let t = tangentes[i];
        let cte = 2 * (t.a * centre.x + t.b * centre.y + t.c) / (points.length * (t.a * t.a + t.b * t.b));

        gradient_x += t.a * cte;
        gradient_y += t.b * cte;
    }

    for (let i = 0; i < gradients_x.length; i++)
        gradients_x[i] *= momentum;

    for (let i = 0; i < gradients_y.length; i++)
        gradients_y[i] *= momentum;

    gradients_x.push(gradient_x);
    gradients_y.push(gradient_y);

    if (gradients_x.length > nbGradients)
        gradients_x.splice(0, 1);

    if (gradients_y.length > nbGradients)
        gradients_y.splice(0, 1);

    let mean_x = gradients_x.reduce((a, x) => a + x);
    let mean_y = gradients_y.reduce((a, x) => a + x);

    centre.x -= mean_x * learningRate;
    centre.y -= mean_y * learningRate;
    nbGradients = int(log(0.02) / log(momentum));
}

function calculateAngle(x, y)
{
    let a;

    if (y >= 0)
        a = acos(x / sqrt(x * x + y * y));
    else
        a = 2 * PI - acos(x / sqrt(x * x + y * y));

    return a;
}

function mousePressed()
{
    let p = {x: mouseX, y: mouseY, radius: 0};
    points.push(p);

    if (points.length > 3)
        points.shift();

    pressed = true;
}

function mouseReleased()
{
    pressed = false;
}

function keyPressed()
{
    if (keyCode == 115)
    {
```

```
        if (drawTangentes)
        {
            drawTangentes = false;
            drawPixels = true;
        }
        else if (drawPixels)
        {
            drawPixels = false;
        }
        else
        {
            drawTangentes = true;
        }
    }

    if (keyCode == 32 || keyCode == 13)
        learning = !learning;
}

function calculateTangentes(x, y)
{
    let pt;

    if (x != undefined && y != undefined)
        pt = {x: x, y: y};
    else
        pt = centre;

    let tangentes = [];

    for (let p of points)
    {
        let inter = {x: p.x, y: p.y};
        inter.x += p.radius * (pt.x - p.x) / sqrt(pow(pt.x - p.x, 2) + pow(pt.y - p.y, 2));
        inter.y += p.radius * (pt.y - p.y) / sqrt(pow(pt.x - p.x, 2) + pow(pt.y - p.y, 2));

        let a = pt.x - inter.x;
        let b = pt.y - inter.y;
        let c = -a * inter.x - b * inter.y;

        if (abs(a) > abs(b) && abs(a) > abs(c))
        {
            b /= a;
            c /= a;
            a = 1;
        }
        else if (abs(b) > abs(c))
        {
            a /= b;
            c /= b;
            b = 1;
        }
        else
        {
            a /= c;
            b /= c;
            c = 1;
        }

        tangentes.push({a: a, b: b, c: c});
    }

    return tangentes;
}

let histo = [];

function pixelisationDraw()
{
    let pas = width / 50;

    stroke(255);
    strokeWeight(2);

    for (let x = 0; x < width; x += pas)
        for (let y = 0; y < height; y += pas)
        {
            let tangentes = calculateTangentes(x, y);
            let erreur = 0;

            for (let t of tangentes)
                erreur += Math.abs(t.a * x + t.b * y + t.c, 2) / (t.a * t.a + t.b * t.b);

            let coeff = erreur / 1000000;

            let rouge = {r: 255, g: 0, b: 0};
            let bleu = {r: 0, g: 0, b: 255};

            fill((1 - coeff) * rouge.r + coeff * bleu.r,
                 (1 - coeff) * rouge.g + coeff * bleu.g,
```

```
                (1 - coeff) * rouge.b + coeff * bleu.b);

            rect(x, y, x + pas - 1, y + pas - 1);
        }
}



function draw()
{
    background(255);

    if (pressed)
    {
        let p = points[points.length - 1];
        p.radius = sqrt(pow(mouseX - p.x, 2) + pow(mouseY - p.y, 2));
    }

    if (centre != undefined)
    {
        histo.push({x: centre.x, y: centre.y});

        while (histo.length > 200)
            histo.shift();
    }

    if (learning)
        gradientDescent();

    if (drawPixels)
    {
        pixelisationDraw();
        return;
    }

    stroke(255, 64, 64);
    strokeWeight(8);

    for (let p of points)
        point(p.x, p.y);

    strokeWeight(4);
    noFill();

    for (let p of points)
        ellipse(p.x, p.y, 2 * p.radius);

    if (drawTangentes)
    {
        stroke(128, 128, 255);

        for (let t of calculateTangentes())
        {
            let x0, y0, x1, y1;

            if (t.b == 0)
            {
                x0 = x1 = -t.c / t.a;
                y0 = 0;
                y1 = height;
            }
            else
            {
                x0 = 0;
                x1 = width;
                y0 = -t.c / t.b;
                y1 = -width * t.a / t.b - t.c / t.b;
            }

            line(x0, y0, x1, y1);
        }
    }

    stroke(128, 192, 0);
    strokeWeight(16);
    point(centre.x, centre.y);

    strokeWeight(5);

    let x = undefined;
    let y = undefined;

    for (let pt of histo)
    {
        if (x && y)
            line(x, y, pt.x, pt.y);

        x = pt.x;
        y = pt.y;
    }
```

```
270 }
```