```java
package Serveur.triangulation;

import java.awt.Graphics;
import java.awt.Rectangle;
import java.io.Closeable;
import java.util.ArrayList;

import Serveur.maths.Utils;
import Serveur.maths.vectors.Vector2d;
import Serveur.maths.vectors.Vector3d;

public class Balle implements Closeable {
    private ArrayList<Vector2d> gradients = new ArrayList<Vector2d>();
    private Vector2d pos;

    public double learningRate = 0.1;
    public double momentum = 0.5;

    private int nbGradientsMax;
    private volatile boolean training = false;

    public Balle(double x, double y) {
        pos = new Vector2d(x, y);

        setMomentum(momentum);
    }

    public Vector2d getPosition() {
        return pos.clone();
    }

    public void setPosition(Vector2d pos) {
        this.pos = pos.clone();

        gradients.clear();
    }

    private void setMomentum(double momentum) {
        this.momentum = momentum;
        nbGradientsMax = (int) (Math.log(0.02) / Math.log(momentum));

        while (gradients.size() > nbGradientsMax)
            gradients.remove(0);
    }

    private void gradientDescent(final ArrayList<Vector3d> cercles) {
        training = true;
        Vector3d[] tangentes = calculateTangentes(cercles);

        Vector2d gradient = new Vector2d();
        int nbCercles = cercles.size();

        for (Vector3d t : tangentes) {
            double cte = 2 * (t.x * pos.x + t.y * pos.y + t.z)
                    / (nbCercles * (t.x * t.x + t.y * t.y));

            gradient.x += cte * t.x;
            gradient.y += cte * t.y;
        }

        for (Vector2d g : gradients) {
            g.x *= momentum;
```

```
63              g.y *= momentum;
64          }
65
66          gradients.add(gradient);
67
68          while (gradients.size() > nbGradientsMax)
69              gradients.remove(0);
70
71          Vector2d totalGradient = new Vector2d();
72
73          for (Vector2d g : gradients)
74              totalGradient = totalGradient.add(g);
75
76          training = false;
77
78          pos = pos.sub(totalGradient.mult(learningRate));
79      }
80
81      private Vector3d[] calculateTangentes(final ArrayList<Vector3d> cercles) {
82          ArrayList<Vector3d> tangentes = new ArrayList<Vector3d>(); // (a, b, c)
83                                                                     // tel que a
84                                                                     // * x + b *
85                                                                     // y + c = 0
86
87          for (Vector3d cercle : cercles) {
88              if (cercle.z == 0)
89                  continue;
90
91              Vector3d inter = cercle.clone();
92              Vector2d centre = new Vector2d(cercle.x, cercle.y);
93              double distance = centre.distance(pos);
94
95              if (distance == 0)
96                  continue;
97
98              double coeff = cercle.z / distance; // rayon / distance
99
100             inter.x += coeff * (pos.x - cercle.x);
101             inter.y += coeff * (pos.y - cercle.y);
102
103             double a = pos.x - inter.x;
104             double b = pos.y - inter.y;
105             double c = -a * inter.x - b * inter.y;
106
107             if (Math.abs(a) > Math.abs(b) && Math.abs(a) > Math.abs(c)) {
108                 b /= a;
109                 c /= a;
110                 a = 1;
111             } else if (Math.abs(b) > Math.abs(c)) {
112                 a /= b;
113                 c /= b;
114                 b = 1;
115             } else {
116                 a /= c;
117                 b /= c;
118                 c = 1;
119             }
120
121             Vector3d tangente = new Vector3d(a, b, c);
122
123             if (tangente.normSquared() > 0)
124                 tangentes.add(tangente);
```

```
125         }
126
127         Vector3d[] list = tangentes.toArray(new Vector3d[tangentes.size()]);
128         tangentes.clear();
129
130         return list;
131     }
132
133     public Vector2d train(final ArrayList<Vector3d> cercles, int iterations) {
134
135         for (int i = 0; i < iterations; i++)
136             gradientDescent(cercles);
137
138         return pos.clone();
139     }
140
141     public void paintComponent(Graphics g, double xmin, double xmax,
142             double ymin, double ymax) {
143         Rectangle rect = g.getClipBounds();
144
145         int x = (int) Utils.map(pos.x, xmin, xmax, rect.x, rect.x + rect.width);
146         int y = (int) Utils.map(pos.y, ymin, ymax, rect.y + rect.height,
147                 rect.y);
148         int radius = 5;
149
150         g.fillOval(x - radius, y - radius, 2 * radius, 2 * radius);
151     }
152
153     public void close() {
154         while (training);
155         gradients.clear();
156     }
157 }
```