```java
package Serveur.triangulation;

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

import Serveur.fenetre.Drawer;
import Serveur.maths.Fonction;
import Serveur.maths.Utils;
import Serveur.maths.vectors.Vector2d;
import Serveur.maths.vectors.Vector3d;

public class Map implements Closeable {
    public ArrayList<Capteur> capteurs = new ArrayList<Capteur>();
    public ArrayList<Balle> balles = new ArrayList<Balle>();

    public Map() {

    }

    public void addCapteur(Capteur capteur) {
        capteurs.add(capteur);
    }

    public void addCapteur(double x, double y) {
        capteurs.add(new Capteur(x, y));
    }

    public boolean removeCapteur(Capteur capteur) {
        boolean succes = capteurs.remove(capteur);
        return succes;
    }

    public void addBalle(double x, double y) {
        balles.add(new Balle(x, y));
    }

    public void addBalle(Balle balle) {
        balles.add(balle);
    }

    public boolean removeBalle(Balle balle) {
        System.out.println("remove balle " + balle);
        boolean succes = balles.remove(balle);
        return succes;
    }

    public ArrayList<Vector3d> getCercles() {
        ArrayList<Vector3d> cercles = new ArrayList<Vector3d>();

        for (Capteur capteur : capteurs) {
            Vector2d pos = capteur.getPosition();
            cercles.add(new Vector3d(pos.x, pos.y, capteur.getDistance()));
        }

        return cercles;
    }

    public Vector2d getSource() {
```

```
63          return getSource(1000);
64      }
65
66      public Vector2d getSource(int iterations) {
67          Vector2d sumPos = (Vector2d) doModifications(() -> {
68              ArrayList<Vector3d> cercles = getCercles();
69              Vector2d sumPos2 = new Vector2d();
70
71              for (Balle balle : balles) {
72                  Vector2d pos;
73
74                  if (iterations > 0)
75                      pos = balle.train(cercles, iterations);
76                  else
77                      pos = balle.getPosition();
78
79                  sumPos2 = sumPos2.add(pos);
80              }
81
82              cercles.clear();
83
84              return sumPos2;
85          });
86
87          return sumPos.div(balles.size());
88      }
89
90      public int removeExtremesBalles(Vector2d source) {
91          Iterator<Balle> it = balles.iterator();
92          double variance = variance(source);
93          int nb = 0;
94
95          while (it.hasNext()) {
96              if (it.next().getPosition().distanceSquared(source) > variance) {
97                  it.remove();
98                  nb++;
99              }
100         }
101
102         return nb;
103     }
104
105     public double variance(Vector2d source) {
106         double v = 0;
107
108         for (Balle balle : balles) {
109             Vector2d pos = balle.getPosition();
110             v += pos.distanceSquared(source);
111         }
112
113         return v / balles.size();
114     }
115
116     private void drawAxis(Graphics g, double xmin, double xmax, double ymin,
117             double ymax) {
118         Rectangle rect = g.getClipBounds();
119
120         int x0 = (int) Utils.map(0, xmin, xmax, rect.x, rect.x + rect.width);
121         int y0 = (int) Utils.map(0, ymin, ymax, rect.y, rect.y + rect.height);
122
123         g.drawLine(x0, rect.y, x0, rect.y + rect.height);
124         g.drawLine(rect.x, y0, rect.x + rect.width, y0);
```

```
125
126         double delta_x = (double) rect.width / (xmax - xmin);
127         double delta_y = (double) rect.height / (ymax - ymin);
128
129         for (double x = x0; x < rect.x + rect.width; x += delta_x)
130             g.drawLine((int) x, y0 + 5, (int) x, y0 - 5);
131
132         for (double x = x0; x > rect.x; x -= delta_x)
133             g.drawLine((int) x, y0 + 5, (int) x, y0 - 5);
134
135         for (double y = y0; y < rect.y + rect.height; y += delta_y)
136             g.drawLine(x0 + 5, (int) y, x0 - 5, (int) y);
137
138         for (double y = y0; y > rect.y; y -= delta_y)
139             g.drawLine(x0 + 5, (int) y, x0 - 5, (int) y);
140     }
141
142     public void paintComponent(Graphics g, double xmin, double xmax,
143             double ymin, double ymax) {
144         g.setColor(Color.BLACK);
145         drawAxis(g, xmin, xmax, ymin, ymax);
146
147         new Drawer(g).setAntiAliasing(true);
148         g.setColor(new Color(134, 95, 255));
149
150         for (Capteur capteur : capteurs)
151             capteur.paintComponent(g, xmin, xmax, ymin, ymax);
152
153         g.setColor(new Color(34, 177, 76));
154
155         for (Balle balle : balles)
156             balle.paintComponent(g, xmin, xmax, ymin, ymax);
157     }
158
159     public void close() {
160         for (Balle balle : balles)
161             balle.close();
162
163         for (Capteur capteur : capteurs)
164             capteur.close();
165
166         balles.clear();
167         capteurs.clear();
168     }
169
170     public synchronized Object doModifications(Fonction f) {
171         return f.method();
172     }
173 }
```