package util.opencv;

import java.util.ArrayList;

import java.util.List;

public class Polygon {

private final BoundingBox \_boundingBox;

private final List<Line> \_sides;

private Polygon(List<Line> sides, BoundingBox boundingBox) {

\_sides = sides;

\_boundingBox = boundingBox;

}

/\*\*

\* Get the builder of the polygon

\*

\* @return The builder

\*/

public static Builder Builder() {

return new Builder();

}

/\*\*

\* Builder of the polygon

\*

\* @author Roman Kushnarenko (sromku@gmail.com)

\*/

public static class Builder {

private List<Point> \_vertexes = new ArrayList<Point>();

private List<Line> \_sides = new ArrayList<Line>();

private BoundingBox \_boundingBox = null;

private boolean \_firstPoint = true;

private boolean \_isClosed = false;

/\*\*

\* Add vertex points of the polygon.<br>

\* It is very important to add the vertexes by order, like you were drawing them one by one.

\*

\* @param point

\* The vertex point

\* @return The builder

\*/

public Builder addVertex(Point point) {

if (\_isClosed) {

// each hole we start with the new array of vertex points

\_vertexes = new ArrayList<Point>();

\_isClosed = false;

}

updateBoundingBox(point);

\_vertexes.add(point);

// add line (edge) to the polygon

if (\_vertexes.size() > 1) {

Line Line = new Line(\_vertexes.get(\_vertexes.size() - 2), point);

\_sides.add(Line);

}

return this;

}

public Builder addVertex(java.awt.Point point) {

Point p = new Point(point.x, point.y);

if (\_isClosed) {

// each hole we start with the new array of vertex points

\_vertexes = new ArrayList<Point>();

\_isClosed = false;

}

updateBoundingBox(p);

\_vertexes.add(p);

// add line (edge) to the polygon

if (\_vertexes.size() > 1) {

Line Line = new Line(\_vertexes.get(\_vertexes.size() - 2), p);

\_sides.add(Line);

}

return this;

}

/\*\*

\* Close the polygon shape. This will create a new side (edge) from the <b>last</b> vertex point to the <b>first</b> vertex point.

\*

\* @return The builder

\*/

public Builder close() {

validate();

// add last Line

\_sides.add(new Line(\_vertexes.get(\_vertexes.size() - 1), \_vertexes.get(0)));

\_isClosed = true;

return this;

}

/\*\*

\* Build the instance of the polygon shape.

\*

\* @return The polygon

\*/

public Polygon build() {

validate();

// in case you forgot to close

if (!\_isClosed) {

// add last Line

\_sides.add(new Line(\_vertexes.get(\_vertexes.size() - 1), \_vertexes.get(0)));

}

Polygon polygon = new Polygon(\_sides, \_boundingBox);

return polygon;

}

/\*\*

\* Update bounding box with a new point.<br>

\*

\* @param point

\* New point

\*/

private void updateBoundingBox(Point point) {

if (\_firstPoint) {

\_boundingBox = new BoundingBox();

\_boundingBox.xMax = point.x;

\_boundingBox.xMin = point.x;

\_boundingBox.yMax = point.y;

\_boundingBox.yMin = point.y;

\_firstPoint = false;

} else {

// set bounding box

if (point.x > \_boundingBox.xMax) {

\_boundingBox.xMax = point.x;

} else if (point.x < \_boundingBox.xMin) {

\_boundingBox.xMin = point.x;

}

if (point.y > \_boundingBox.yMax) {

\_boundingBox.yMax = point.y;

} else if (point.y < \_boundingBox.yMin) {

\_boundingBox.yMin = point.y;

}

}

}

private void validate() {

if (\_vertexes.size() < 3) {

throw new RuntimeException("Polygon must have at least 3 points");

}

}

}

/\*\*

\* Check if the the given point is inside of the polygon.<br>

\*

\* @param point

\* The point to check

\* @return <code>True</code> if the point is inside the polygon, otherwise return <code>False</code>

\*/

public boolean contains(Point point) {

if (inBoundingBox(point)) {

Line ray = createRay(point);

int intersection = 0;

for (Line side : \_sides) {

if (intersect(ray, side)) {

// System.out.println("intersection++");

intersection++;

}

}

/\*

\* If the number of intersections is odd, then the point is inside the polygon

\*/

if (intersection % 2 == 1) {

return true;

}

}

return false;

}

public List<Line> getSides() {

return \_sides;

}

/\*\*

\* By given ray and one side of the polygon, check if both lines intersect.

\*

\* @param ray

\* @param side

\* @return <code>True</code> if both lines intersect, otherwise return <code>False</code>

\*/

private boolean intersect(Line ray, Line side) {

Point intersectPoint = null;

// if both vectors aren't from the kind of x=1 lines then go into

if (!ray.isVertical() && !side.isVertical()) {

// check if both vectors are parallel. If they are parallel then no intersection point will exist

if (ray.getA() - side.getA() == 0) {

return false;

}

float x = ((side.getB() - ray.getB()) / (ray.getA() - side.getA())); // x = (b2-b1)/(a1-a2)

float y = side.getA() \* x + side.getB(); // y = a2\*x+b2

intersectPoint = new Point(x, y);

} else if (ray.isVertical() && !side.isVertical()) {

float x = ray.getStart().x;

float y = side.getA() \* x + side.getB();

intersectPoint = new Point(x, y);

} else if (!ray.isVertical() && side.isVertical()) {

float x = side.getStart().x;

float y = ray.getA() \* x + ray.getB();

intersectPoint = new Point(x, y);

} else {

return false;

}

// System.out.println("Ray: " + ray.toString() + " ,Side: " + side);

// System.out.println("Intersect point: " + intersectPoint.toString());

if (side.isInside(intersectPoint) && ray.isInside(intersectPoint)) {

return true;

}

return false;

}

/\*\*

\* Create a ray. The ray will be created by given point and on point outside of the polygon.<br>

\* The outside point is calculated automatically.

\*

\* @param point

\* @return

\*/

private Line createRay(Point point) {

// create outside point

float epsilon = (\_boundingBox.xMax - \_boundingBox.xMin) / 100f;

Point outsidePoint = new Point(\_boundingBox.xMin - epsilon, \_boundingBox.yMin);

Line vector = new Line(outsidePoint, point);

return vector;

}

/\*\*

\* Check if the given point is in bounding box

\*

\* @param point

\* @return <code>True</code> if the point in bounding box, otherwise return <code>False</code>

\*/

private boolean inBoundingBox(Point point) {

if (point.x < \_boundingBox.xMin || point.x > \_boundingBox.xMax || point.y < \_boundingBox.yMin || point.y > \_boundingBox.yMax) {

return false;

}

return true;

}

private static class BoundingBox {

public float xMax = Float.NEGATIVE\_INFINITY;

public float xMin = Float.NEGATIVE\_INFINITY;

public float yMax = Float.NEGATIVE\_INFINITY;

public float yMin = Float.NEGATIVE\_INFINITY;

}

}