Clipping Polygons

Paul Kainberger Jakob Moosbauer Philipp Nuspl

June 26, 2019

Johannes-Kepler-University Linz

Polygon Clipping

A clipping algorithm cuts a

- candidate polygon such that it is contained in a
- clipping polygon,

i.e. the intersection of two such polygons is computed.

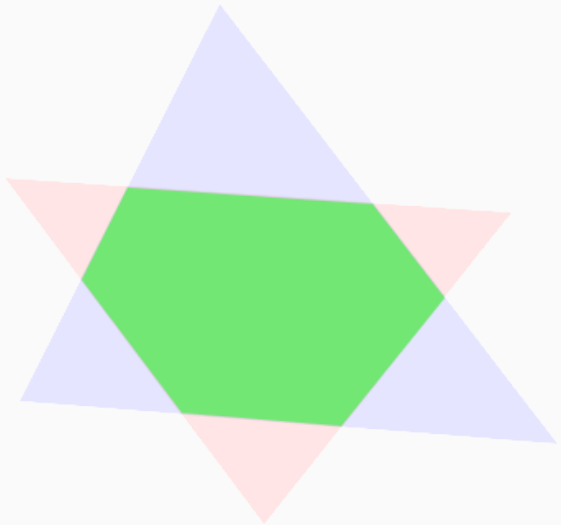

Figure 1: Clipping of two triangles

Polygons

A polygon is an

- ordered list of two-dimensional points, called vertices,
- which are connected with a line, called edge.

A polygon is called convex if all the inner angles are strictly less than 180 degrees (otherwise called concave).

(a) Convex polygon

(b) Concave polygon

Orientation

A polygon can be oriented clockwise or anti-clockwise. Method to determine this: Signed area of polygon given by vertices (x_i, y_i) for $i = 1, \ldots, n$ is given by (called Shoelace formula)

$$A = \frac{1}{2}\left(x_1 y_2 - x_2 y_1 + x_2 y_3 - x_3 y_2 + \cdots + x_n y_1 - x_1 y_n\right)$$

If area positive, then clockwise. If negative, then anticlockwise.

Self-intersecting

A polygon
is called non-self-intersecting
if two arbitrary edges do not
intersect (except in vertices).

What is interior? Use even-odd
(winding number) rule.

Figure 3: Self-intersecting polygon

> *'This rule determines*
> *the "insideness" of*
> *a point on the canvas*
> *by drawing a ray from that point to infinity in any*
> *direction and counting the number of path segments*
> *from the given shape that the ray crosses. If this number*
> *is odd, the point is inside; if even, the point is*
> *outside.'*
>
> SVG Definition

Overview Clipping Algorithms

We implemented three different clipping algorithms: Can handle different clipping polygons.

	concave	self-intersecting
Sutherland-Hodgman	✗	✗
Weiler-Atherton	✓	✗
Greiner-Hormann	✓	✓

Sutherland-Hodgman

Algorithm works as follows:

- Traverse clipping polygon edge by edge in anti-clockwise order.
- Extend edge to a line and compute intersection points with candidate polygon.
- Add those intersections to candidate polygon.
- Only take the vertices of the candidate polygon which lie on the left side of the line.

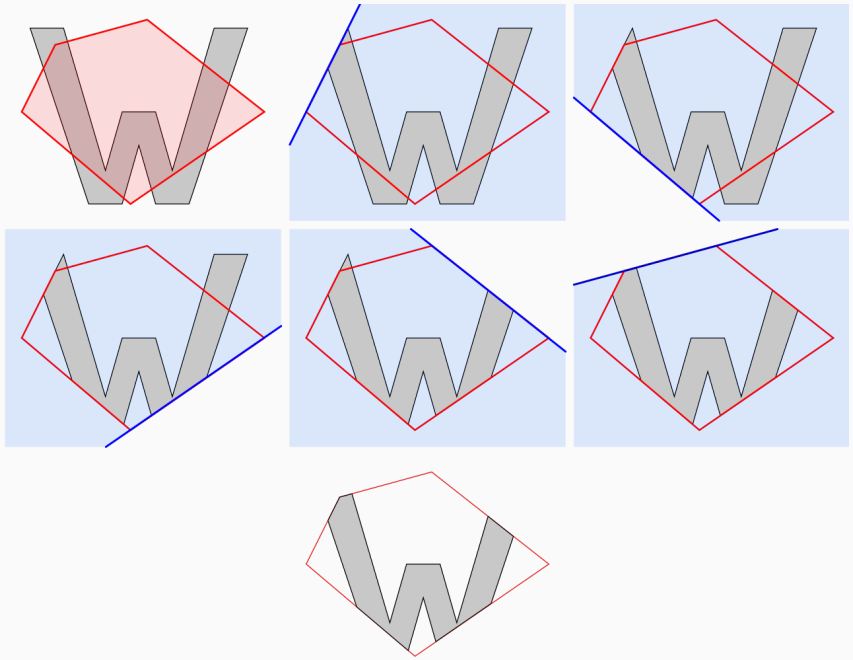

Figure 4: Sutherland-Hodgman in action, Source: Wikipedia

Sutherland-Hodgman

Problem: Algorithm might introduce double edges.

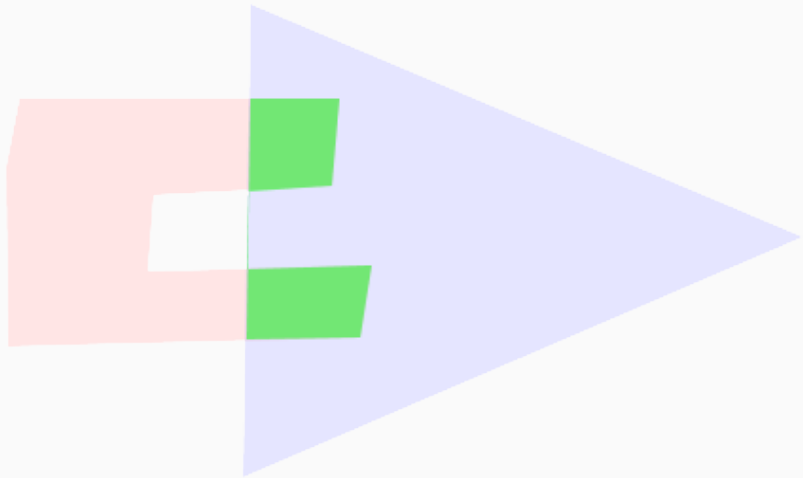

Figure 5: Double edge

Weiler-Atherton

- Applicable for convex clipping polygons
- Solves Sutherland-Hodgman's "Double-edge-problem"

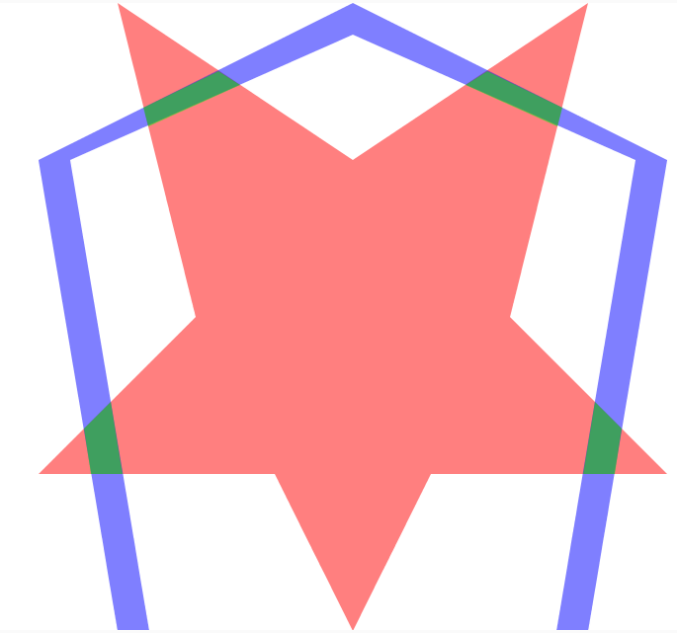

Figure 6: Convex polygons

Weiler-Atherton

Algorithm works as follows:

- If one polygon contains the other, return the inner polygon.
- Otherwise compute intersection points.
- Add intersection points to both polygons at the right position (and mark them).
- Start clipping from one intersection point on the clipping polygon.
- If we reach the next intersection point, we jump on the candidate polygon.
- Continue clipping from that point on the candidate polygon until we reach the next intersection point.
- Continue this process until the clipped polygon is closed.
- Repeat the process using the remaining intersection points.

Greiner-Horman

Algorithm works as follows:

- Create all intersection points.
- Insert the intersection points into the polygons at the right position.
- Mark each intersection point as entry or exit.
- Traverse through the polygons, switching to the neighbor vertex at each intersection.

Greiner-Horman

Datastructure:

- vertex
  - next: vertex
  - previous: vertex
  - intersect: boolean
  - entry_exit: boolean
  - neighbor: vertex
  - alpha: double

Greiner-Horman

Can handle self-intersecting polygons:

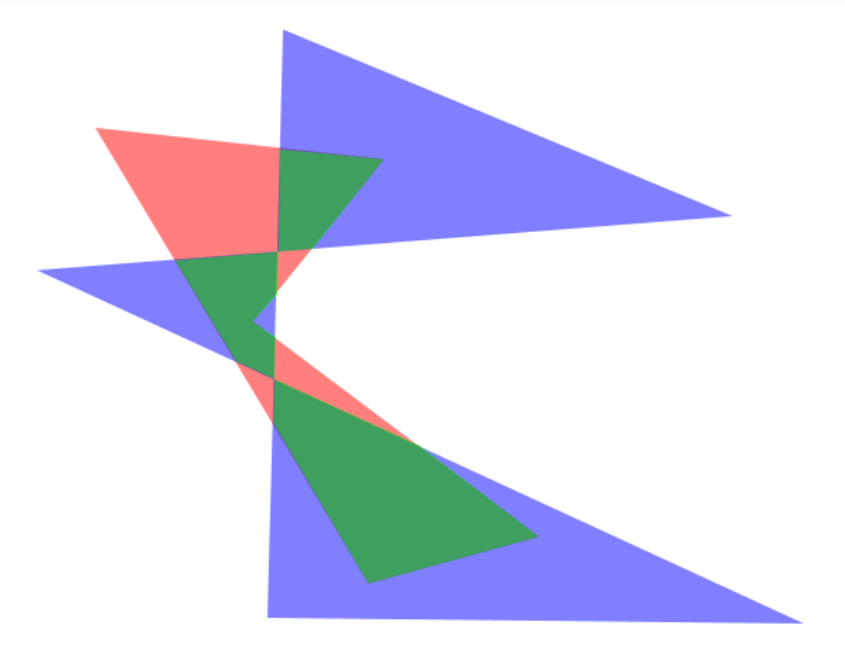

Figure 7: self-intersection

Degenerate cases

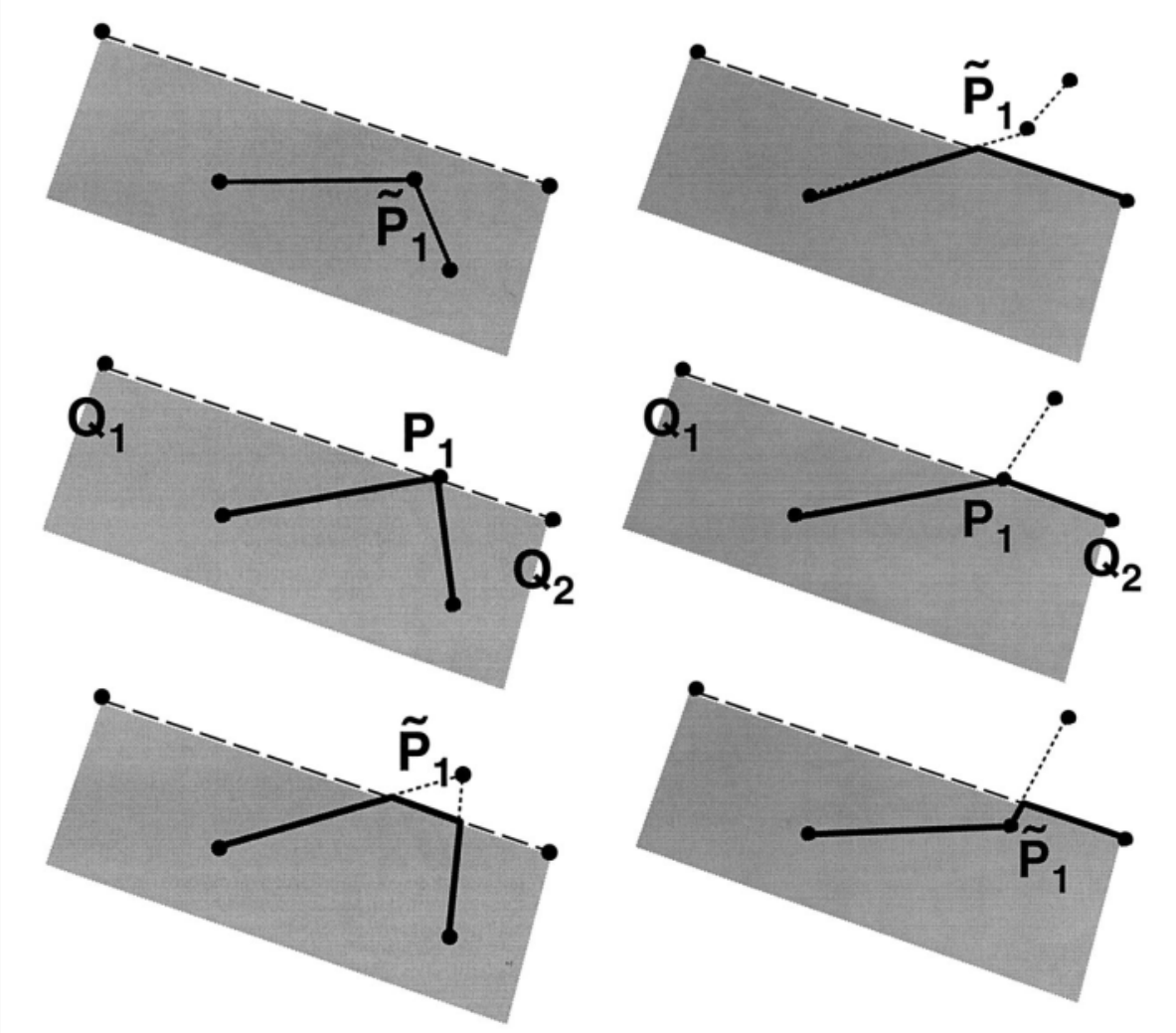

Figure 8: Degenerate cases, Source: G. Greiner and K.Hormann, Efficient Clipping of Arbitrtary Polygons

Program Structure

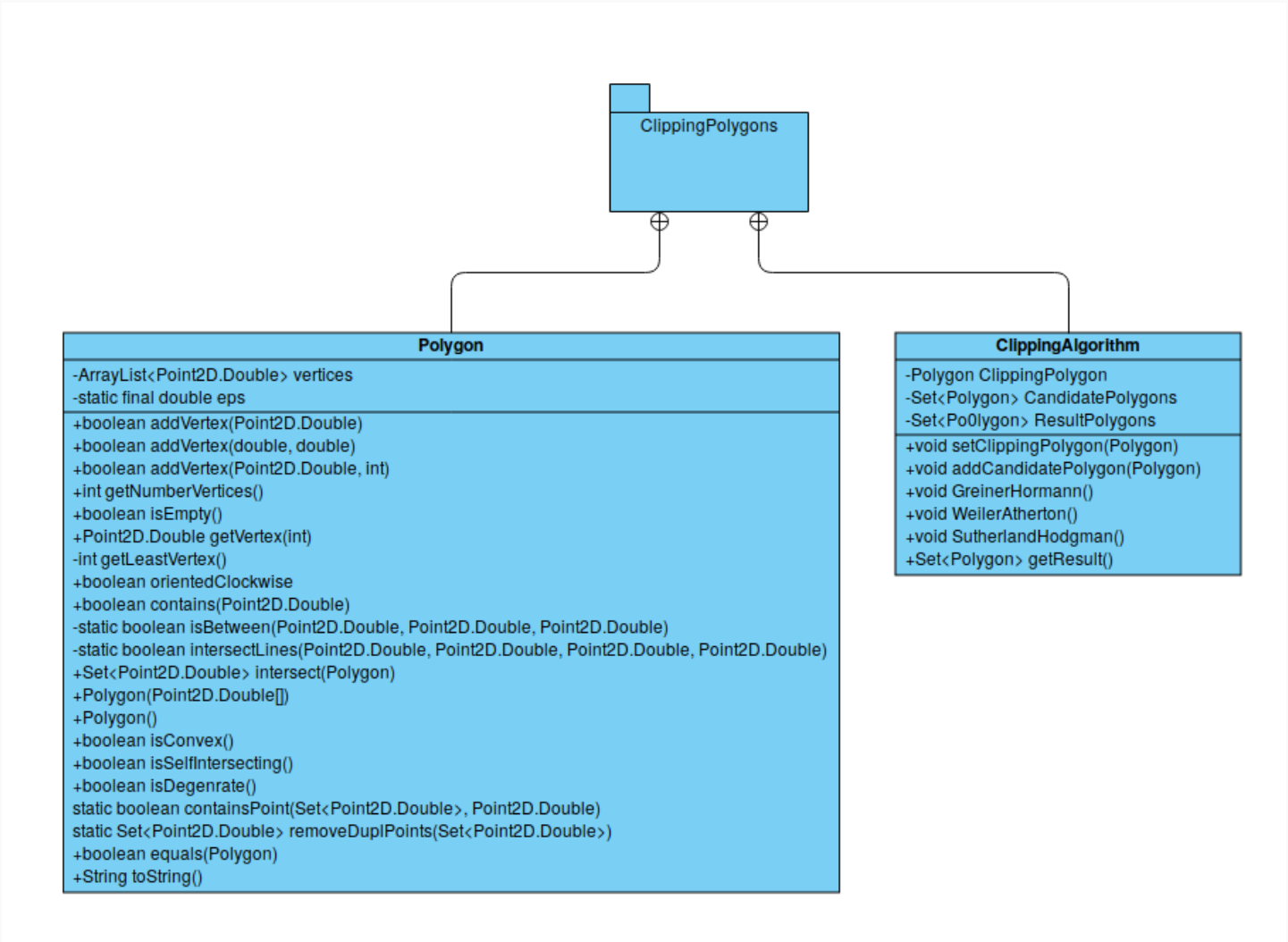

Figure 9: Class diagram

Implementation Polygon

We implemented our own Polygon class based on `List<Point2D.Double>`. Features include:

- Adding/getting vertex at specific position.
- Check if point is in polygon.
- Intersect polygons to get points of intersections.
- Check if self-intersecting.
- Check if convex.
- Check orientation: Clockwise or anti-clockwise.

Polygon files

Our program can open polygons from files and store them into files.
We chose the following structure for the files:

- Each vertex is in one line.
- The coordinates are separated by a comma.
- Different polygons are separated by blank lines.

Graphical User Interface

- Display
- Menu
- Draw polygons
- Manage polygons
- Clip polygons

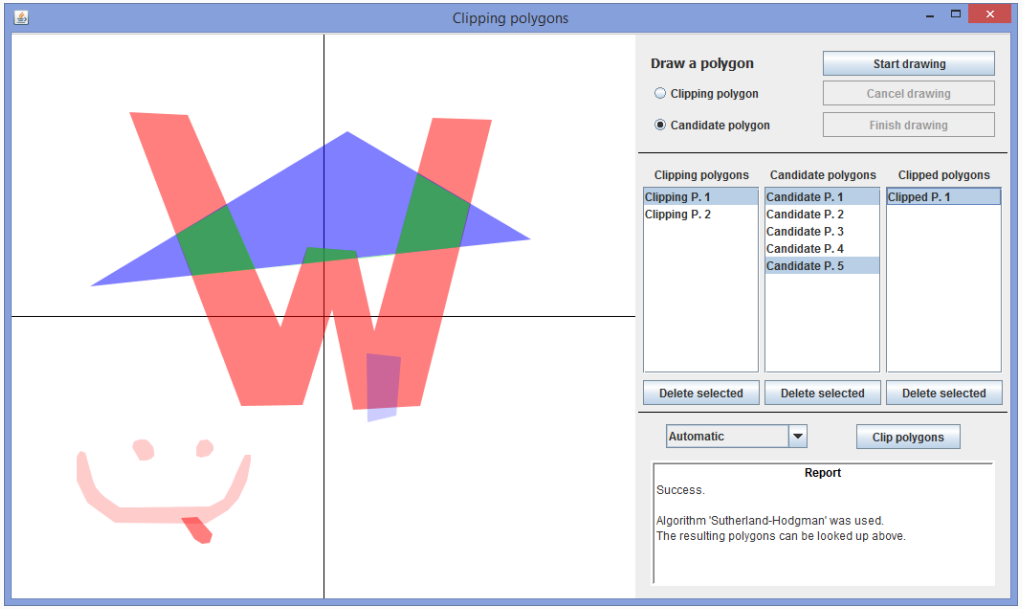

Figure 10: GUI