Clipping Polygons

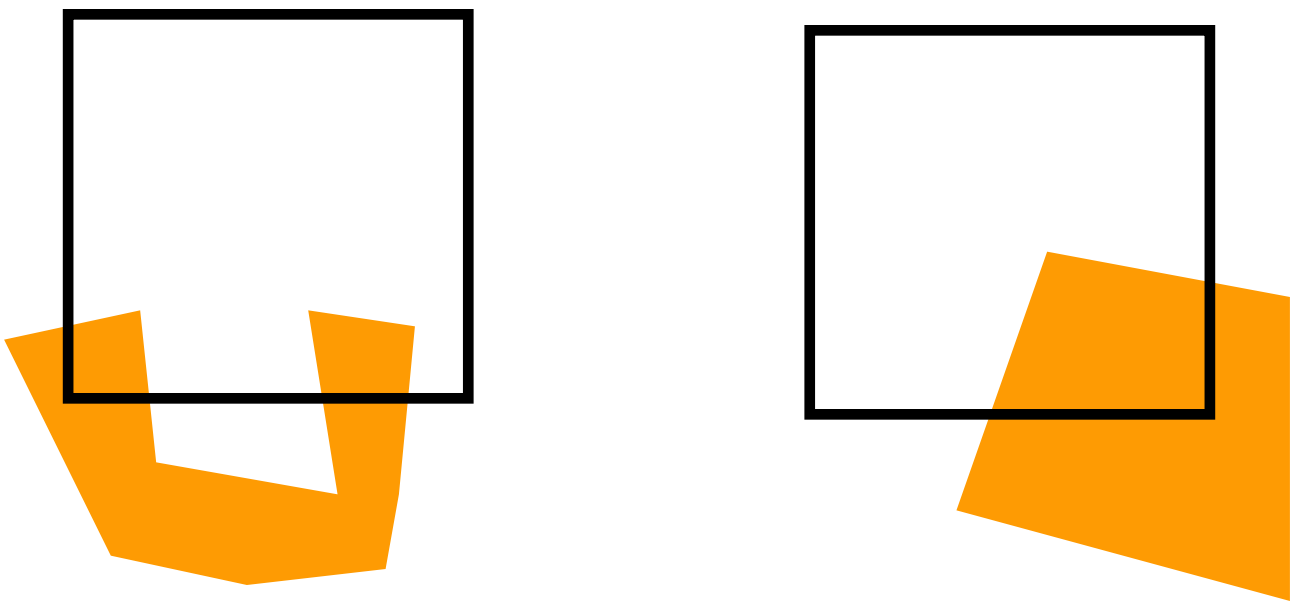
Dr Nicolas Holzschuch
University of Cape Town
e-mail: holzschu@cs.uct.ac.za

Map of the lecture

- Problems in clipping polygons
- Clipping polygons
  - against one edge of the window
  - against the whole window
- Clipping other shapes
- Accelerating the clipping

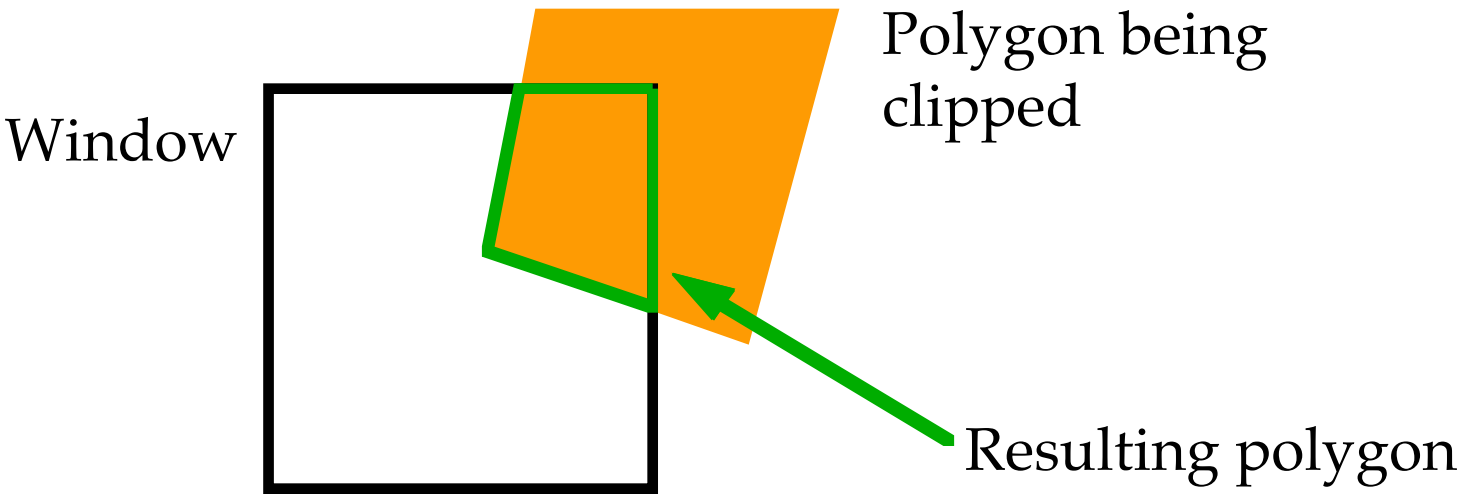
Clipping a polygon

- Different possible cases
- We have to fill the result of clipping



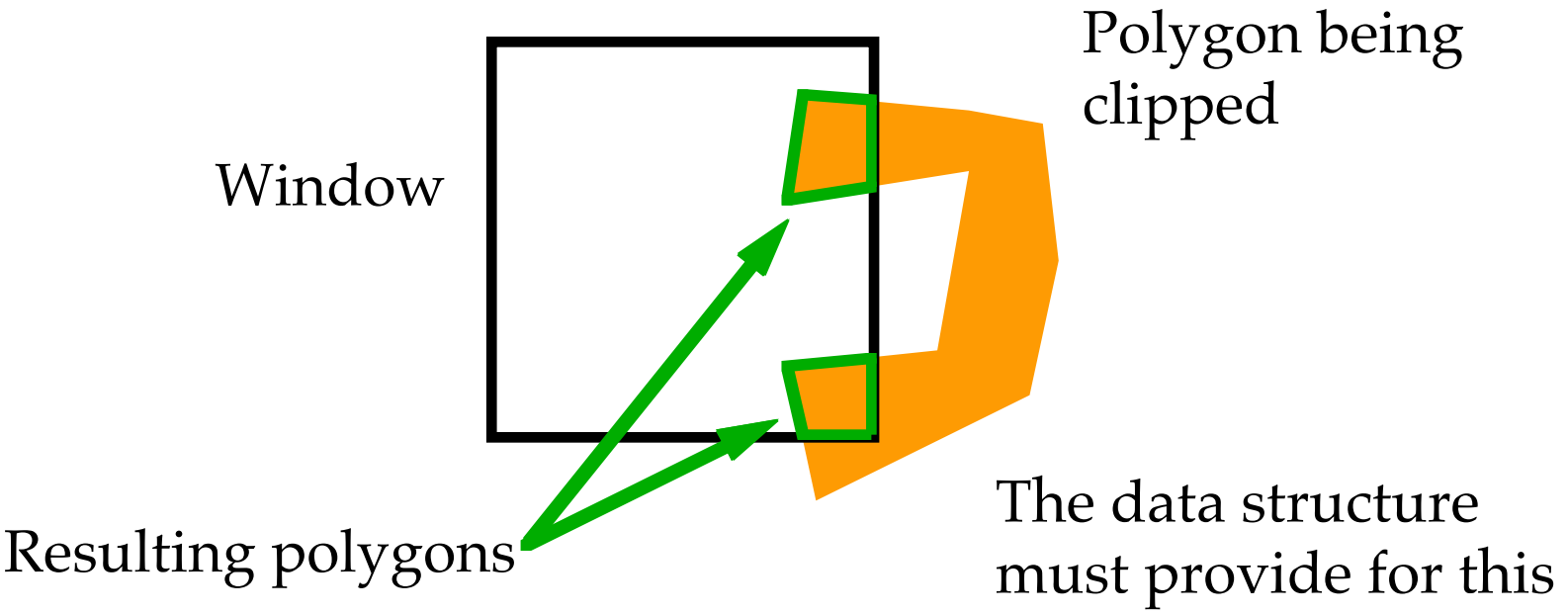
A polygon must be filled

- We have to fill the result:
  - must indentify the new edges
  - the result must be closed



Several new polygons

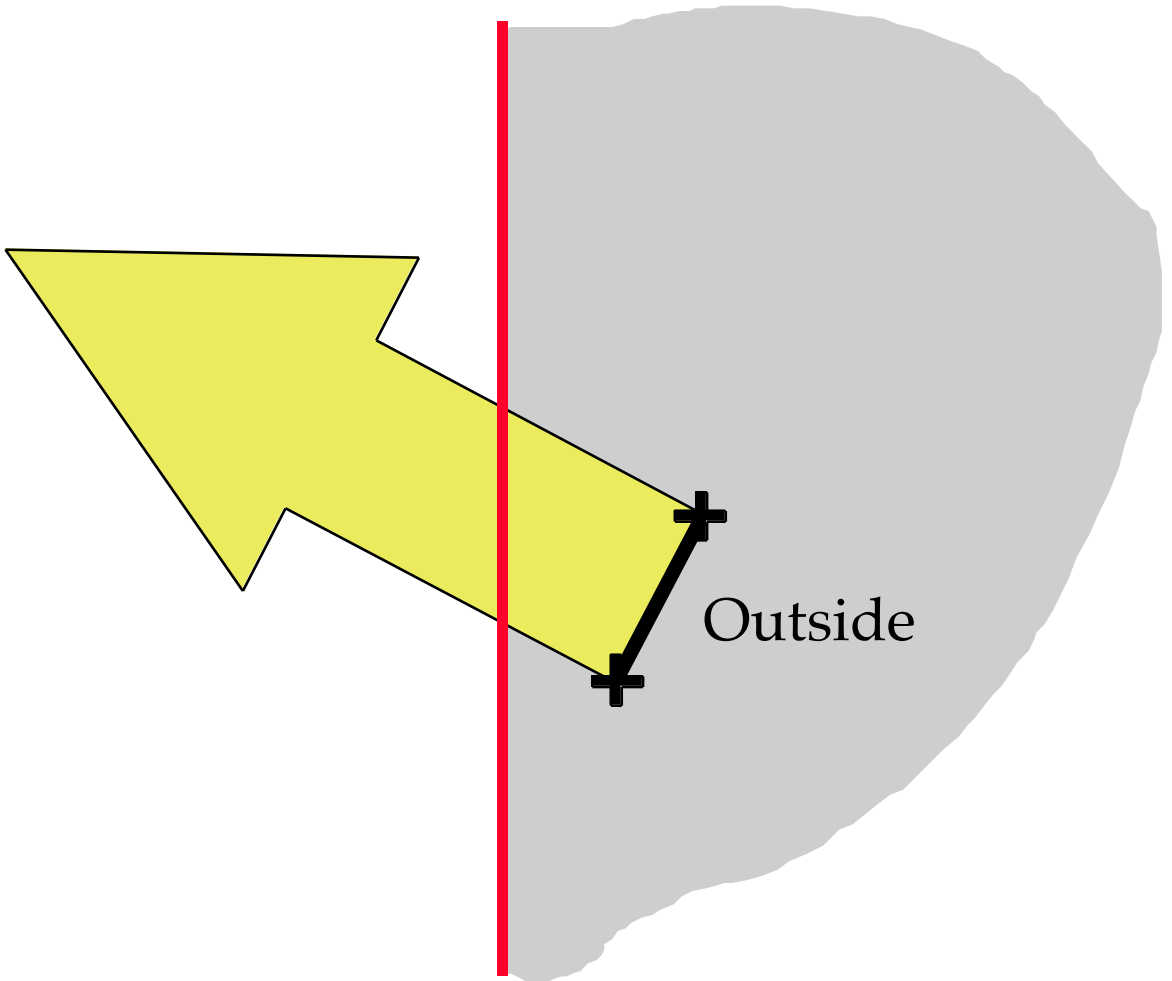
- Clipping a polygon may result in several new polygons:



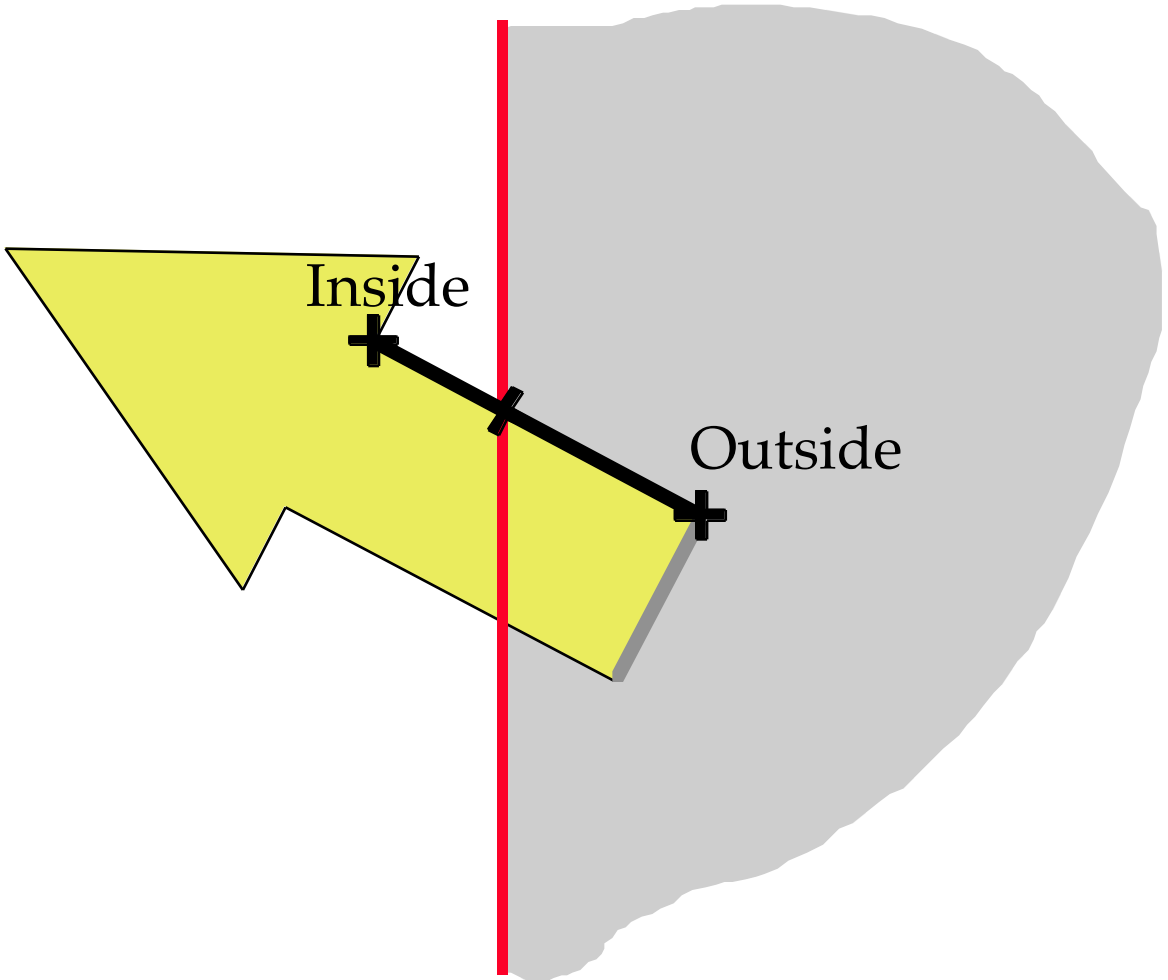
Clipping a polygon: algorithm

- Start by clipping against a window boundary
- Do each polygon edge in turn
- Clip each polygon edge against the boundary:
  - If *leaving*, connect to latest *entering*
  - If *entering*, connect to latest *leaving*

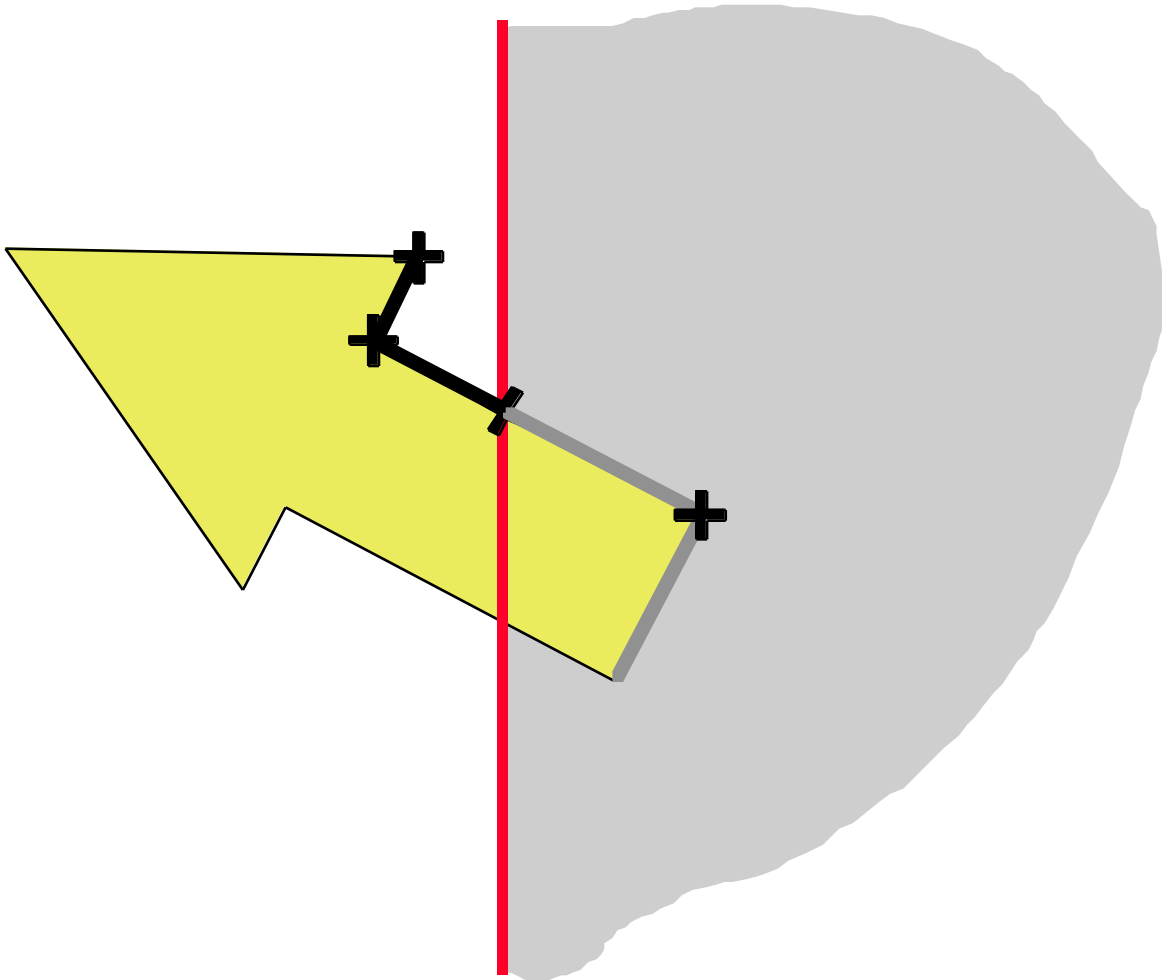
Polygon against boundary



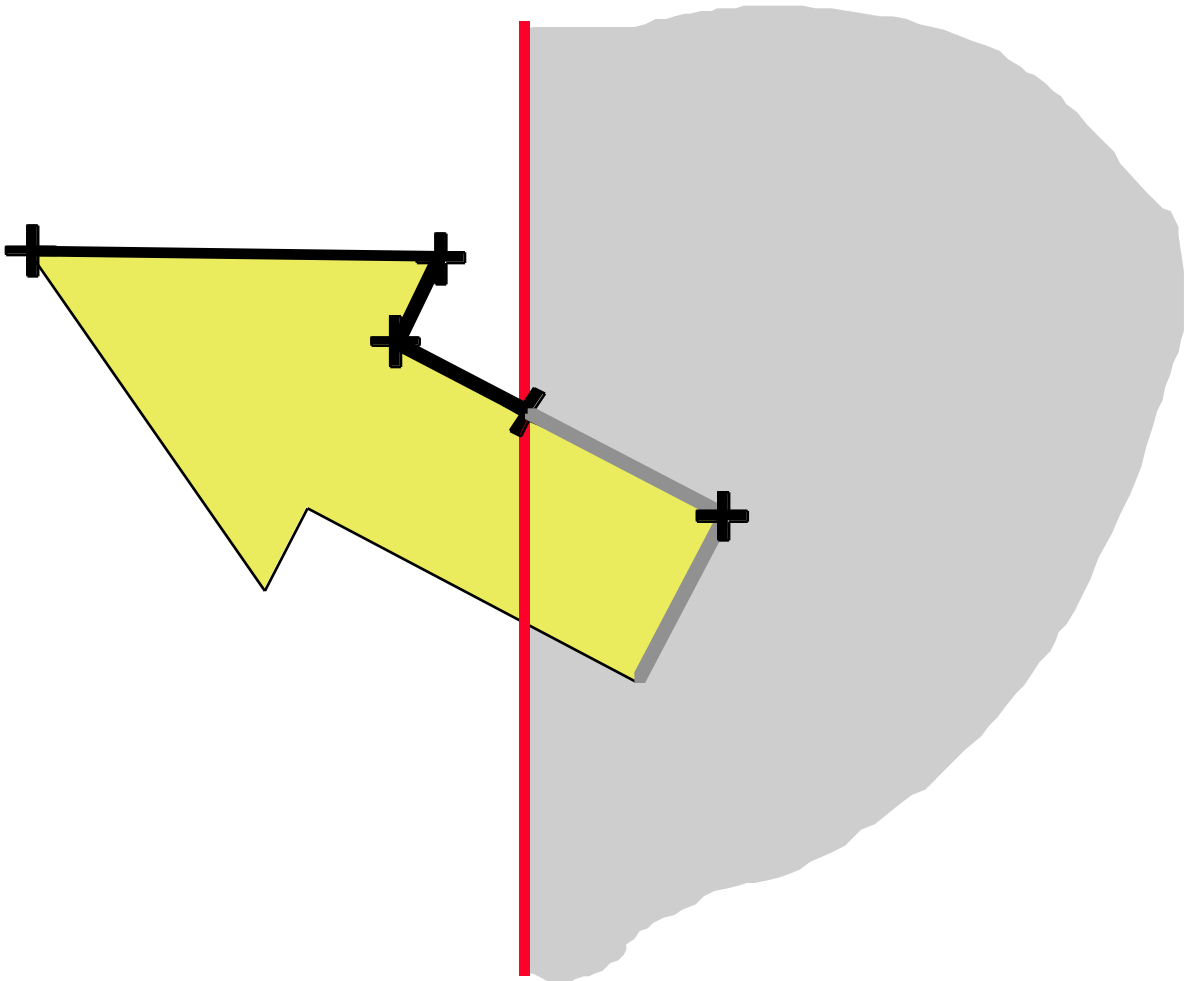
Polygon against boundary



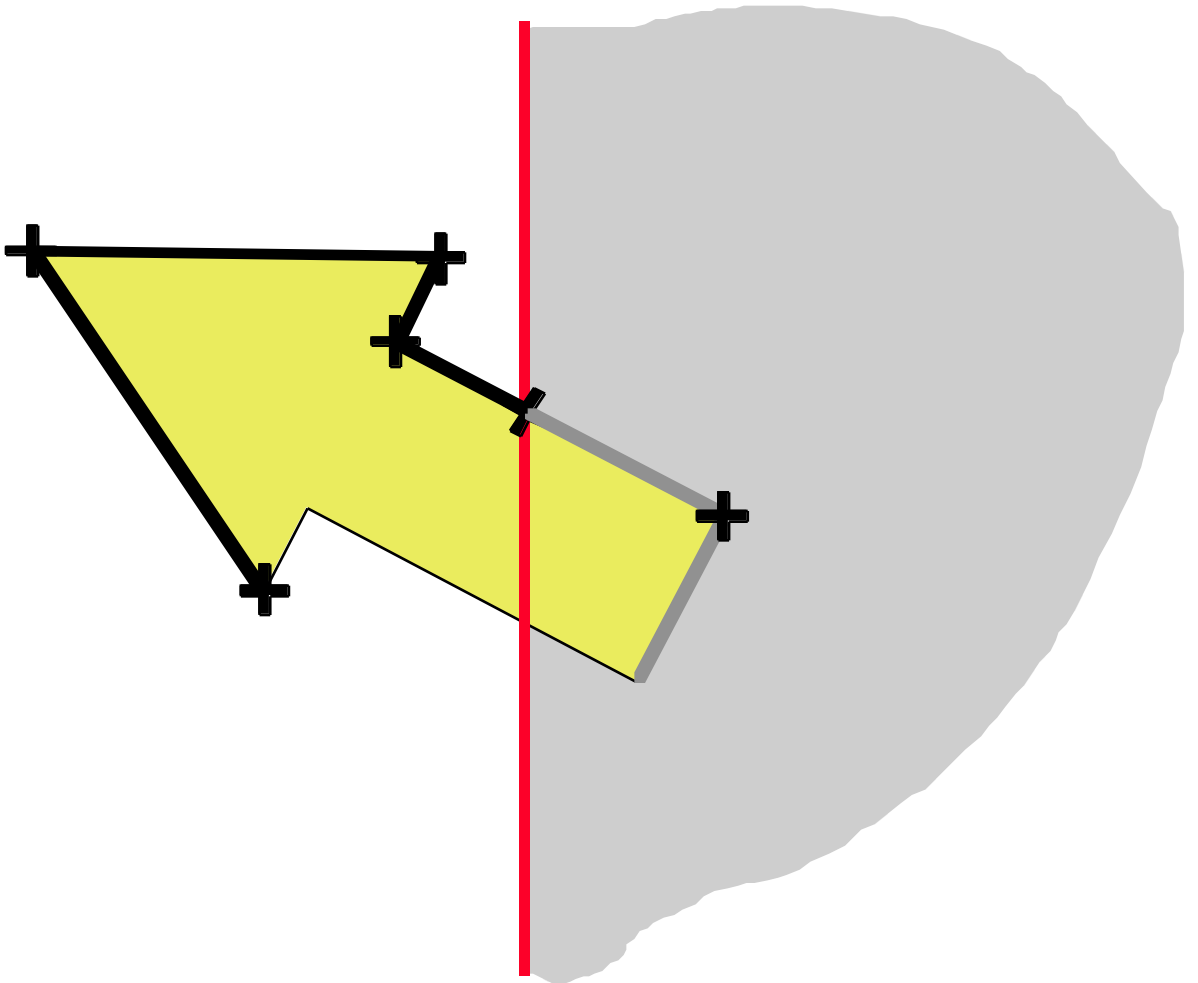
Polygon against boundary



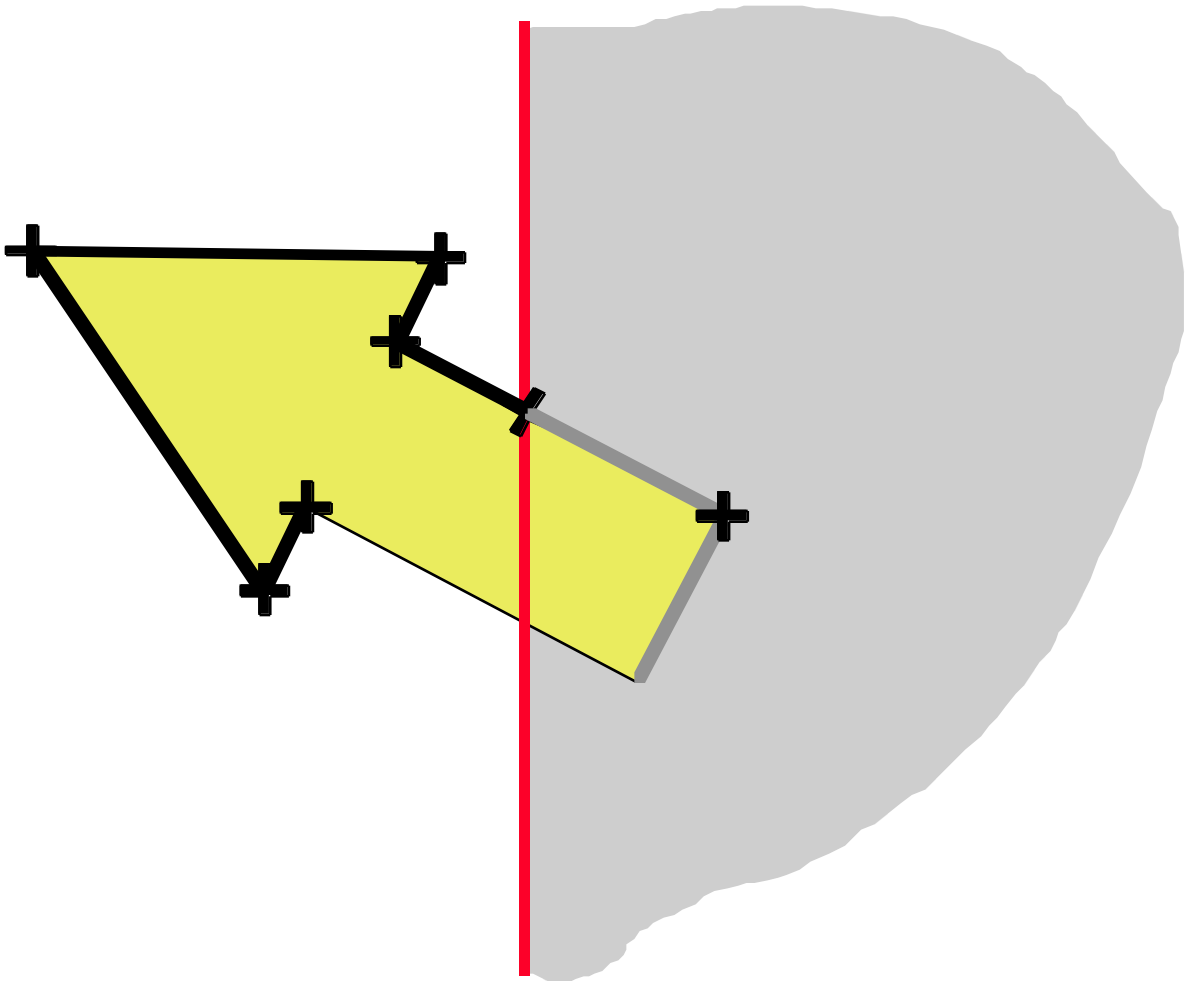
Polygon against boundary



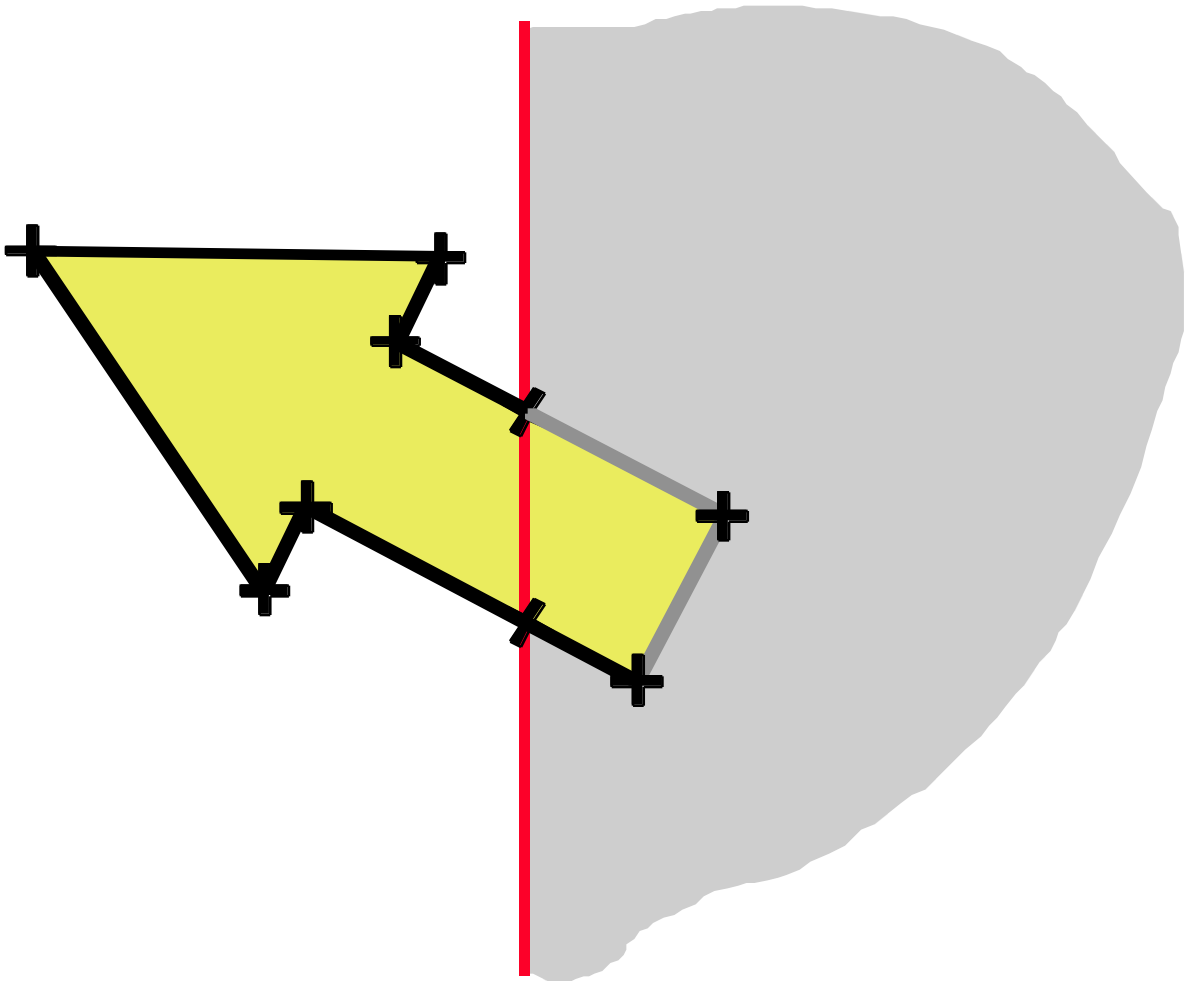
Polygon against boundary



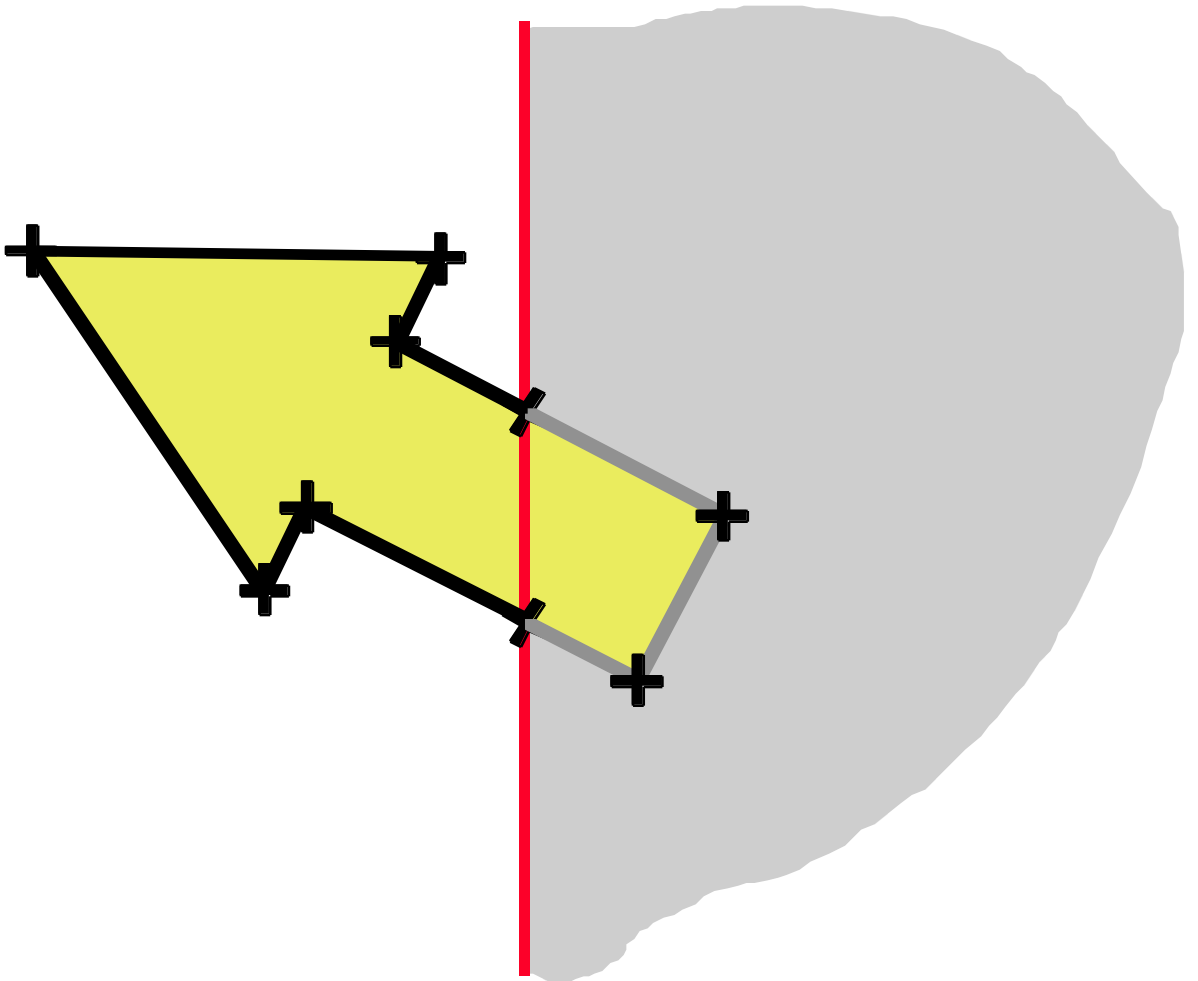
Polygon against boundary



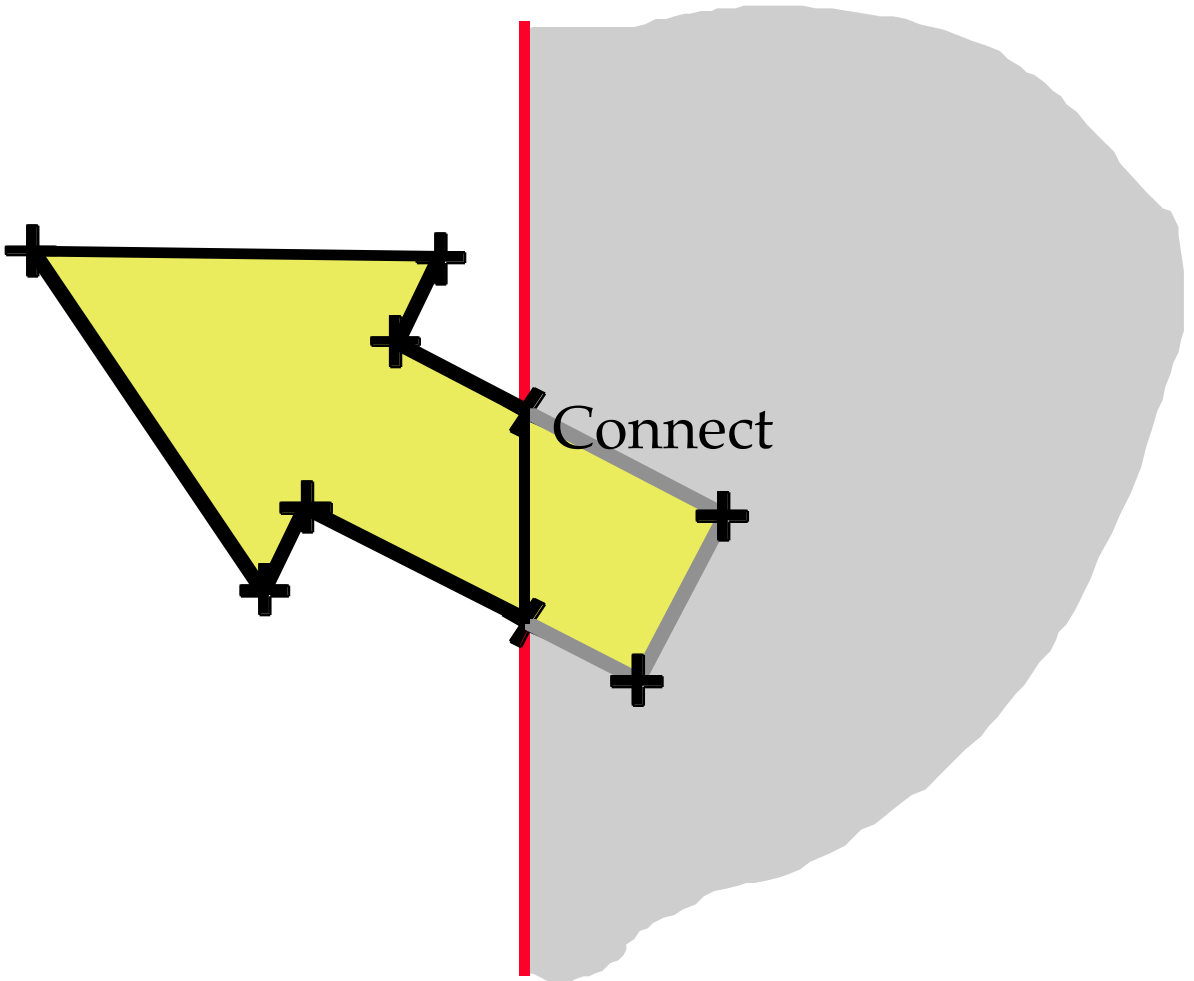
Polygon against boundary



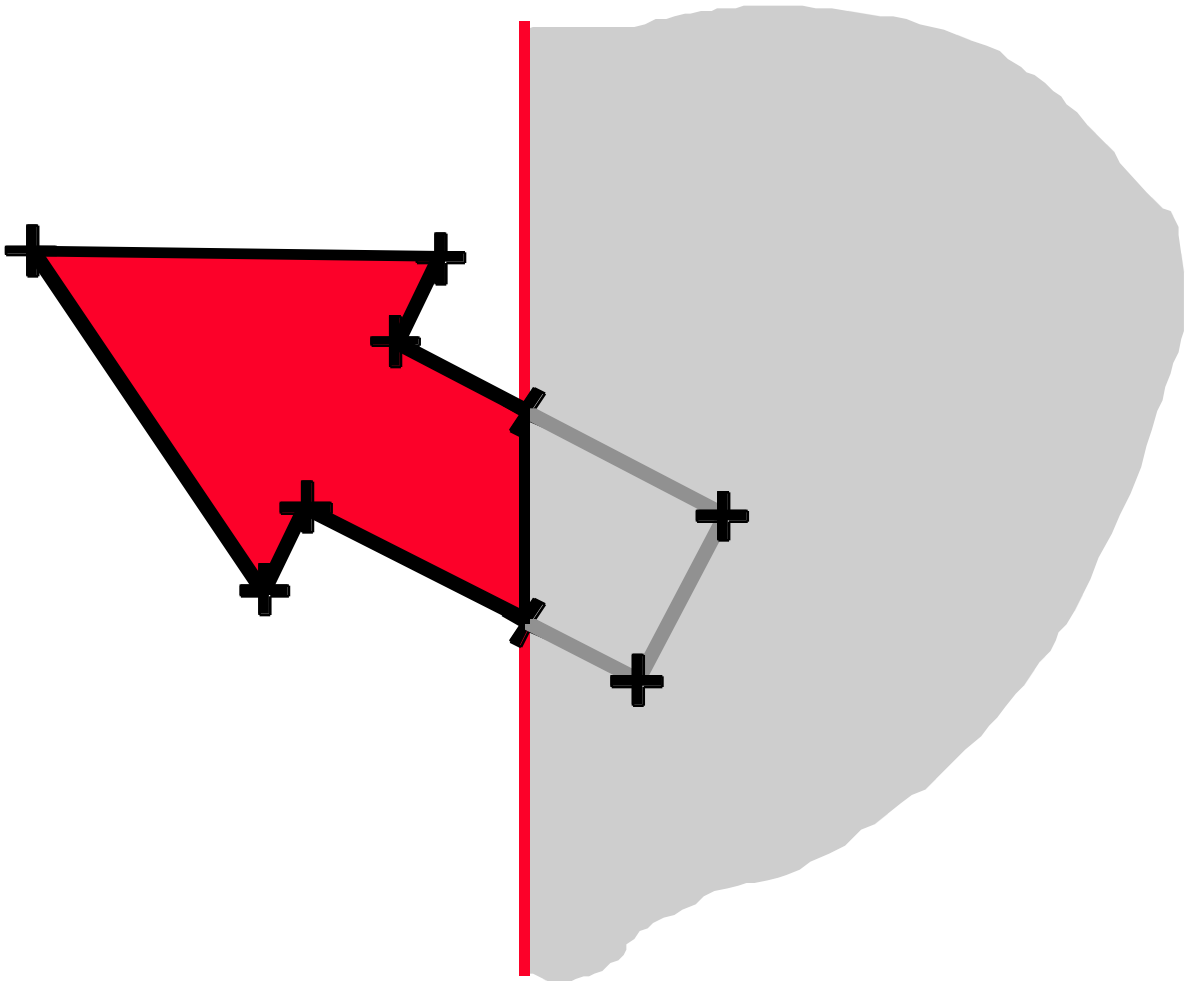
Polygon against boundary



Polygon against boundary



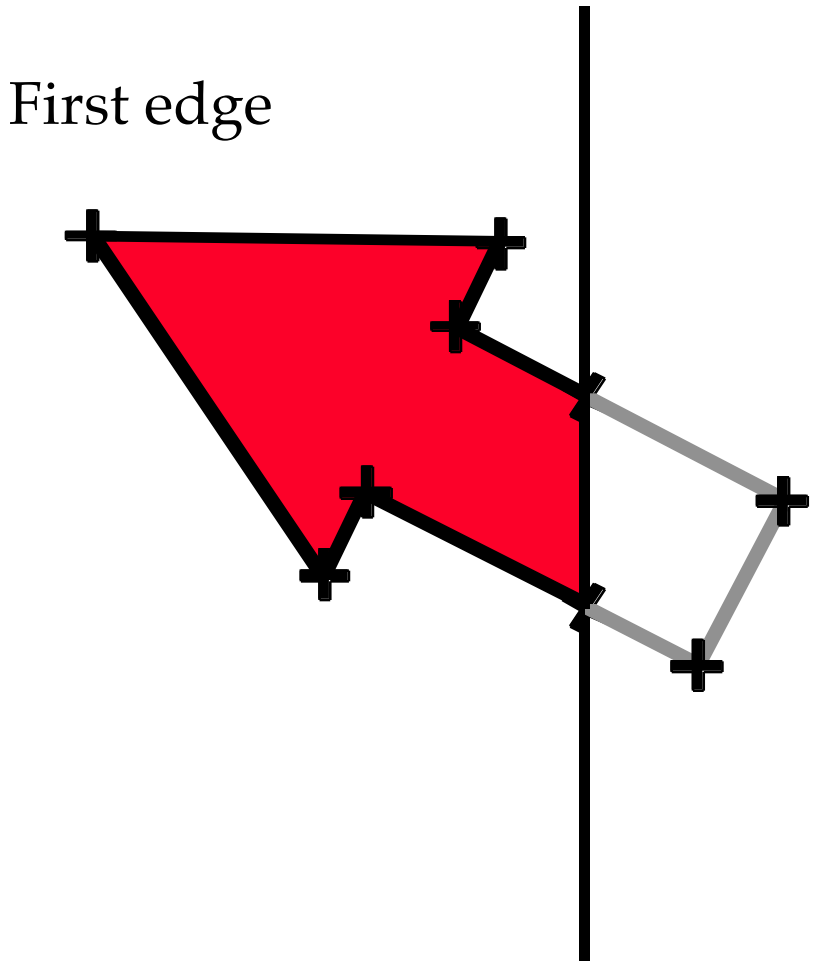
Polygon against boundary



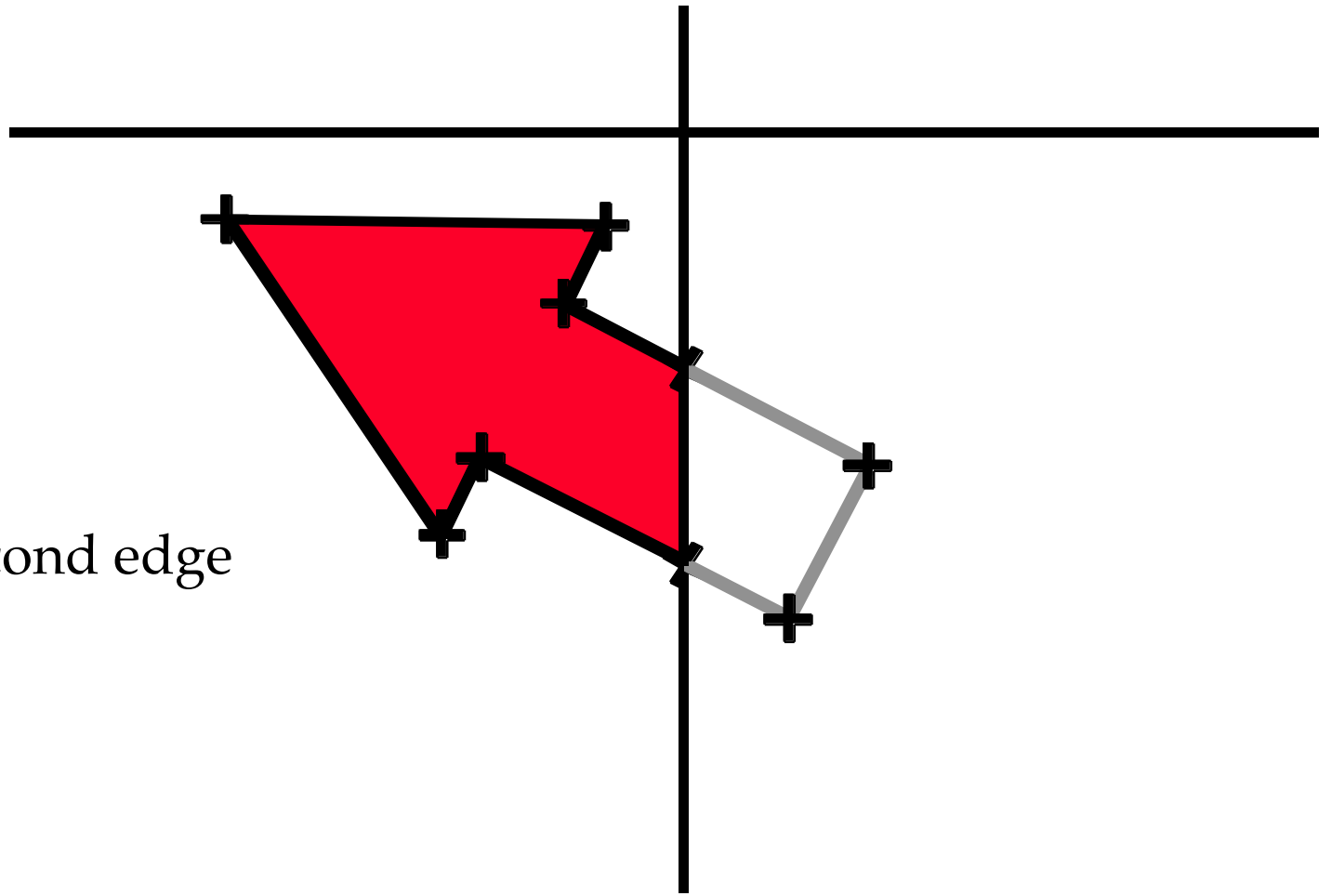
Polygon against window

- Do each window boundary in turn
- For each window boundary, clip the polygon using previous algorithm
- The result is a closed polygon or several closed polygons
- These are fed to the next window boundary

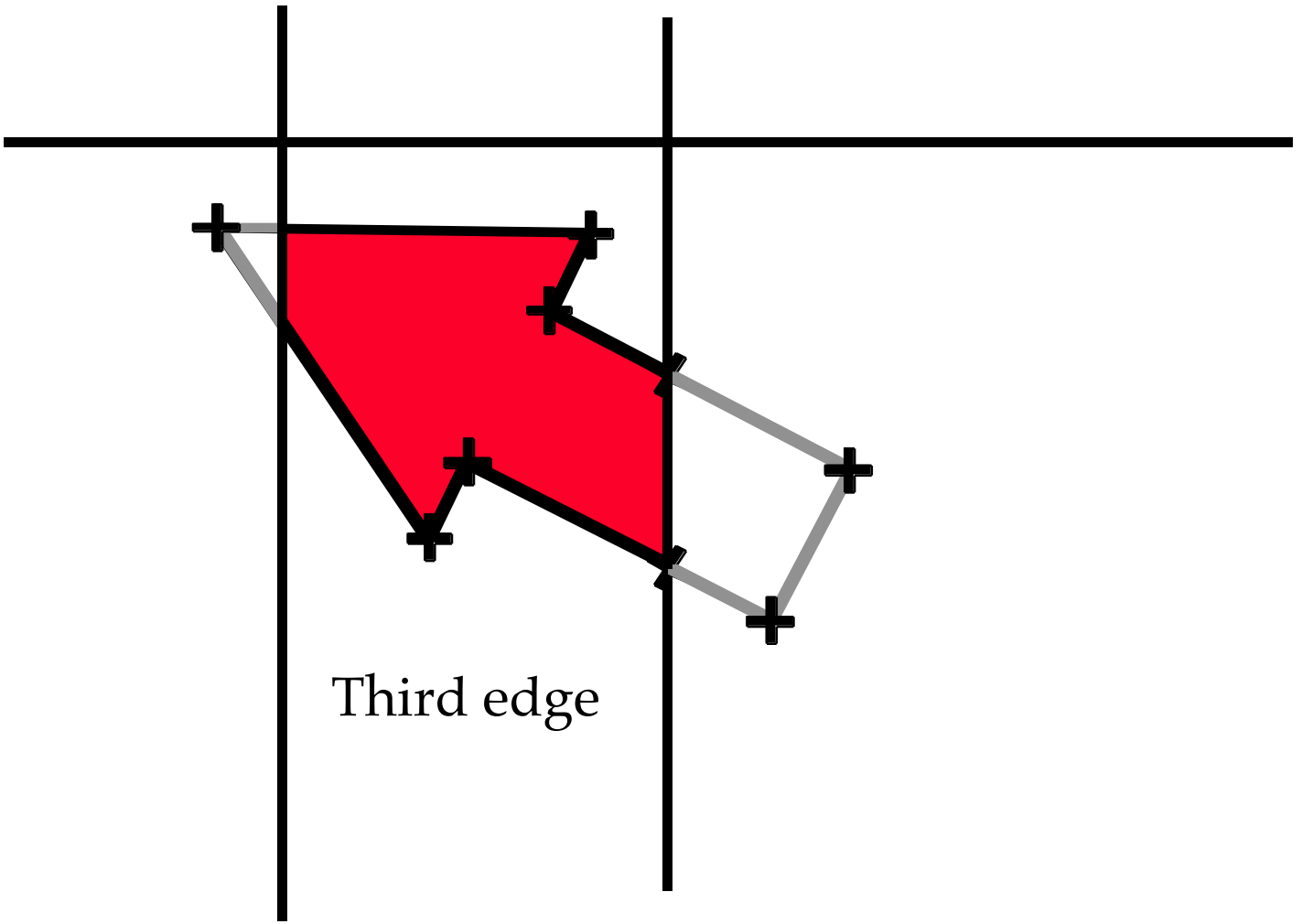
Polygon against window



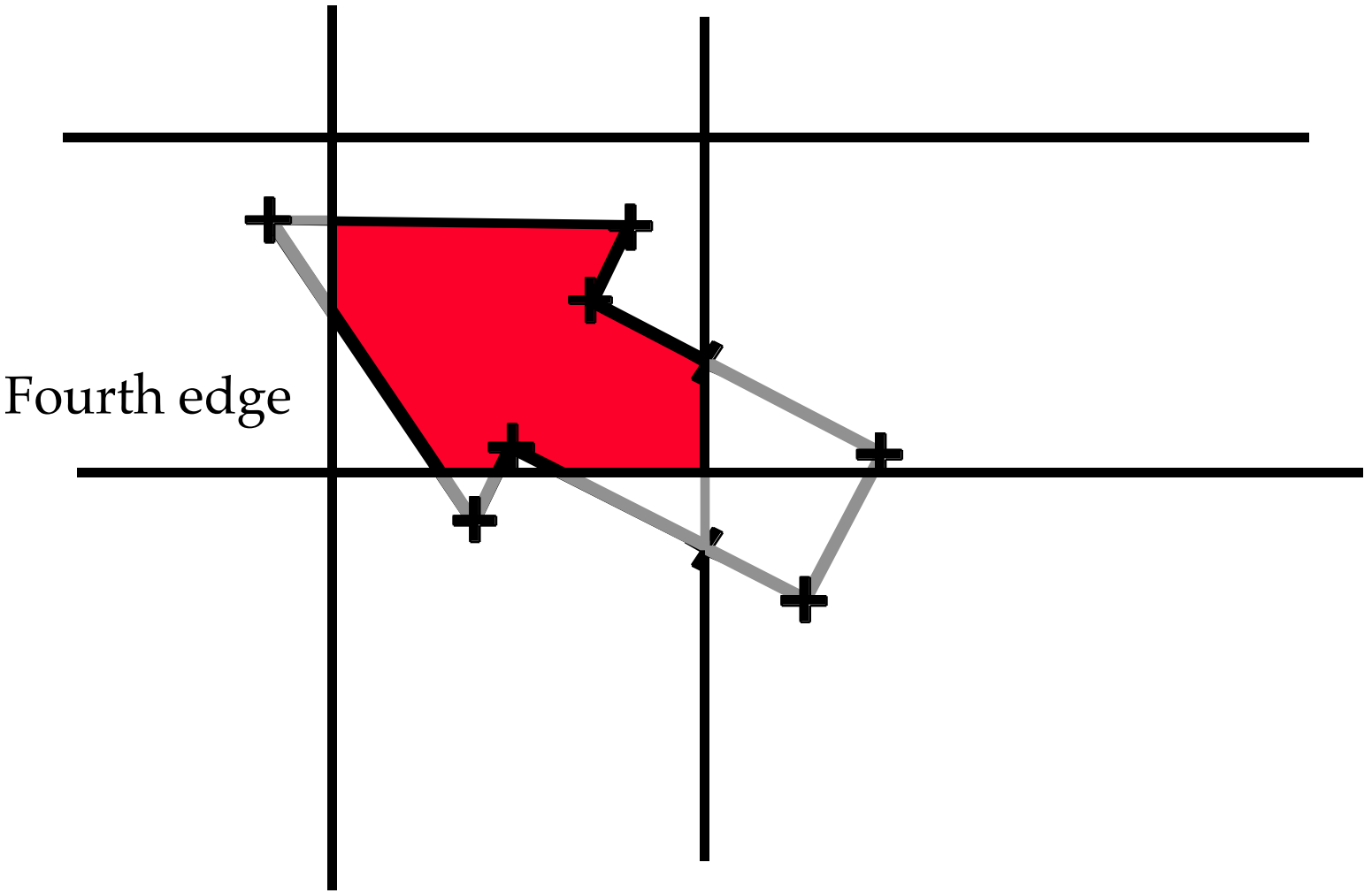
Polygon against window



Polygon against window



Polygon against window



Clipping Polygons: conclusion

- A more complex algorithm
- Algorithmic complexity: 4 times the number of edges
- Easy to implement using standard languages
  - try it in Java

Other shapes (circles)

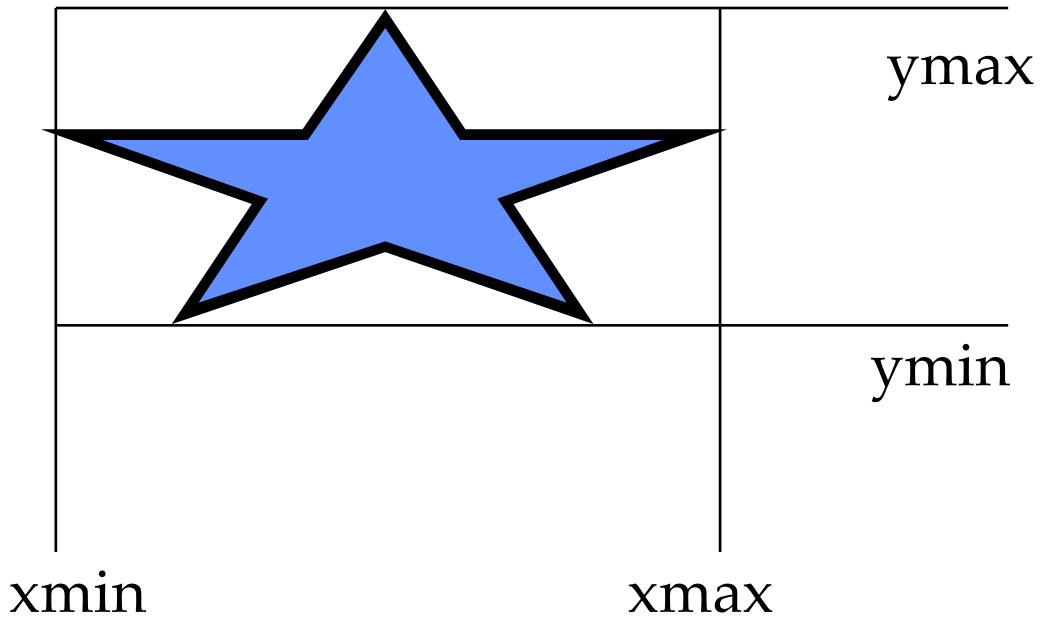
- Convert the shape to polygon, then clip the polygon
  - problem: too many edges on the polygon
- Clip the shape during rasterization
  - problem: rasterizing invisible parts
  - can be accelerated by eliminating parts that are trivially invisible

Accelerating clipping

- Clipping can be a costly step
- Optimization:
  - use extents and don't clip parts that are:
    - trivially invisible
    - trivially visible

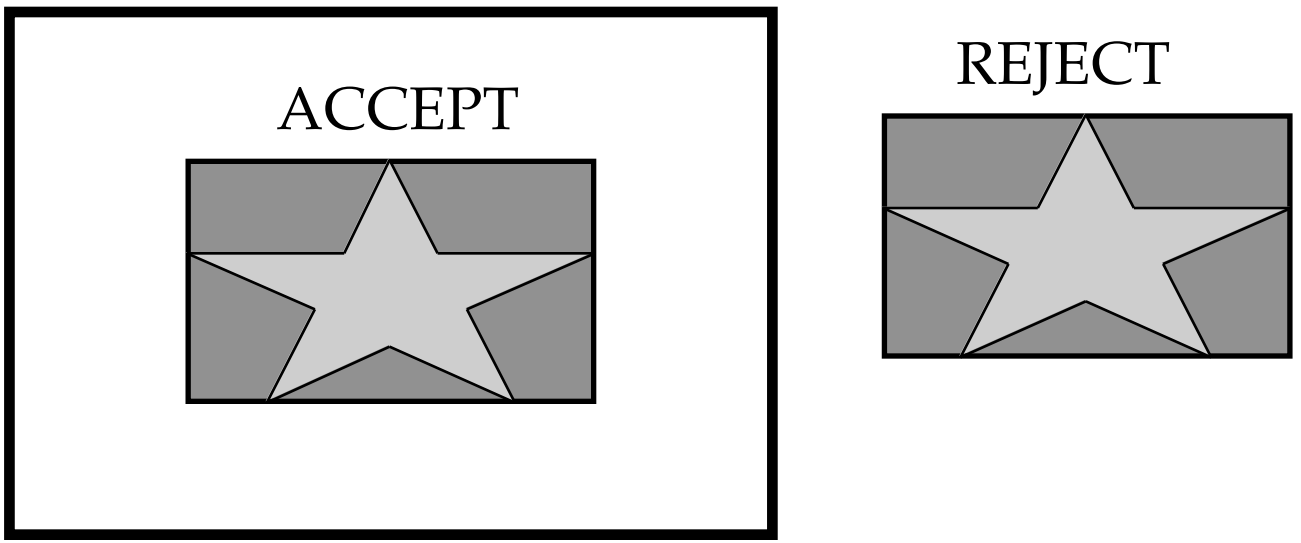
Extents

- Compute the x-extent and the y-extent of the shape:



Trivial accept/reject

- Check the extent with the window:



Trivial accept/reject

- Spend some time doing a trivial test
- Gain more time by avoiding complex computations
- A common idea to many computer graphics algorithms