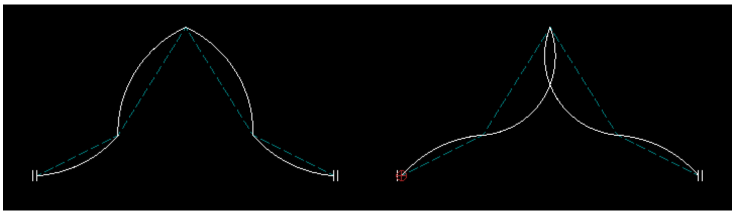
Andrew,

because you did not answer if it is ok to continue to comment on a closed bug report (FS#1756) please read my findings below.

After finding my way trough the QCAD code, what was uniquely facilitated by the name "Robert S." in this case, I found where polyline area code fails.
https://github.com/qcad/qcad/commit/7234c460dc645d236a91d8b3925a7ac8bccf95b0
Snapshot included because of the dynamic nature of the line numbering.

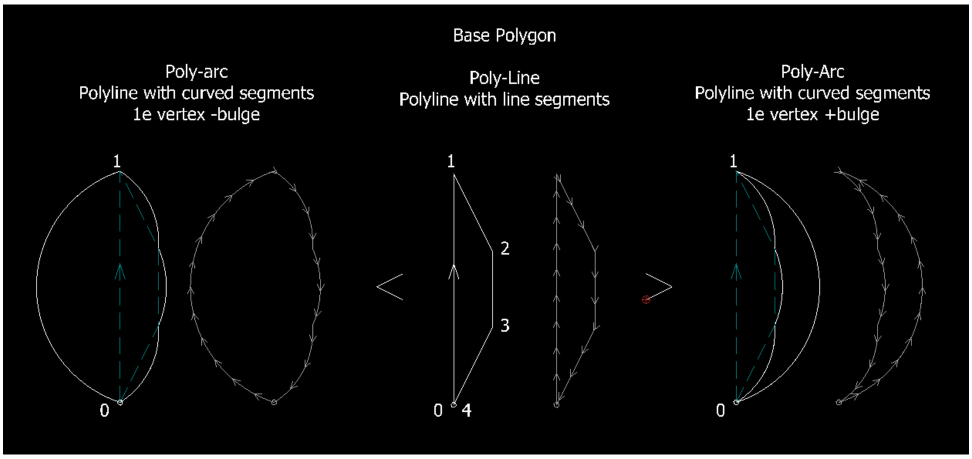
In core/math/RPolyline.cpp lines 1304 - 1311 is the implemention of the Green's Theorem.

> **Remark 1:** This is for the polygonal Area (The surface area circumscribed by a closed polyline) as defined by https://en.wikipedia.org/wiki/Simple_polygon including weakly simple polygons. Excluded are polygons that self-intersect by segments and this is also found true for self-intersection at 2 nodes on the same spot.
> For QCAD 2D usage of collinear line segments don't make a difference.
>
> **Remark 2:** Not simple polygons should return NaN. There is no easy solution for those, eg. pentagram; figure 8; loopholes;.... and solutions are or can be disambiguous. In good practice such polygons are split into sub polygons and sub areas are summed by preferences.
> Checking the ‘not simple’ nature is not straightforward.
> For polygons with curved edges (there is no unique name for them) this gets even harder.



Green's Theorem confide to anticlockwise paths to return a positive answer because an area should always be positive like a length or a distance.

> **Remark 3**: It is found that this is not necessary, and can be even adverse, for computational purpose as long as we return any final value for an area as an absolute. This will eliminate the disambiguous nature of polygons with curved segments.


Base Polygon
Poly-arc
Polyline with curved segments
1e vertex -bulge
Poly-Line
Polyline with line segments
Poly-Arc
Polyline with curved segments
1e vertex +bulge
1
0
<
1
2
3
0
4
>
1
0

In the coding:

The closedCopy should be of the nature that the last node(x_{n+1},y_{n+1}) is the same as the first node.

> node(x_0,y_0) – vertex(0) -- node(x_1,y_1) -- vertex(1)......vertex(n) -- node(x_{n+1},y_{n+1})<

Explicitly avoiding the terms logically open/closed, physically open/closed.

I would like to see a comprehensive definition of those because there is still something fuzzy going on while swapping only the closed property in the property editor.

The method below does not rely on any oriëntation.

Lines 1296 – 1310 calculates the area of the base polygon making this absolute (line 1311) is like setting the base polygon to CCW while it could be CW and this is where it fails.
The half of the sum (lines 1304 – 1309) should be called areaP.
Replace area with areaP.
This is the area of the polygon with straight edges with a sign.

Strip line 1311

lines 1313 – 1334 calculates the sum of the surfaces of the individual circle-segments for curved segments if any.
This should be summed to areaC:

- With chordArea always positive by definition but with the sign of the bulge.
  areaC = $\sum_0^n[\mathrm{ChordArea} \bullet \mathrm{sign}(\mathrm{bulge})]$ with chordArea(0/+) and bulge(-/0/+)
- Stepping the closedCopy the same way lines 1296 – 1310 do.
  Plain forward from 0 to countVertices.
  In fact this can be done in the same count loop but then it checks isArcSegmentAt(i) all the time or one can use a flag depending on closedCopy.hasArcSegments().
  All the getOrientation; isReversed; plReversed is not nececary for this.

Finally add areaC to areaP and make this term positive (as in line 1336):

area = | areaP + areaC |

This method is thoroughly tested for many 'Simple polygon' including 'Weakly simple polygon' in any quadrant, crossing axises, curled, with curved segments not crossing other segments or other nodes (eg. mooncrest) and with both oriëntations of the base polygon.

Correct results remains at user’s discretion.

I hope you can wade your way through my post.
I would be happy to precheck your adapted coding, just send the part to me in plain text.

Implementation in a QCAD release would be of great benefit to me for the study of a better method to asymmetrically scale polys with arcs. Or you could sent me a pre-path like once before.

Kind Regards
CVH