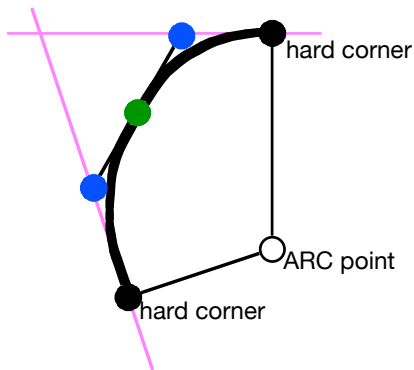


The ARC and ARC_AUTO Points

The **ARC point** is provided as an easy way to approximate a semi-circular (or semi-oval) arc without doing annoying manual calculations to derive the needed spline points. The ARC point specifies the *center* of the circle (or oval), and the rendering software calculates the appropriate spline points.

An ARC point must always come between two hard (i.e. ordinary) polygon corners of one polygon loop. Due to the way the code is written, an ARC point must not be the first point of its polygon loop, but may be the last point.

The point-in-polygon code uses an ARC point to interpolate two spline corners, each of which is .4142 of the way from a hard corner, to the intersection of the tangent lines as defined by the ARC point:



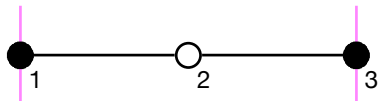
The polygon data provides the two hard corners (black) and the ARC point (white).

Two spline points (blue) are interpolated by moving .4142 down the tangents (magenta) from the hard corners toward the intersection of the tangents.

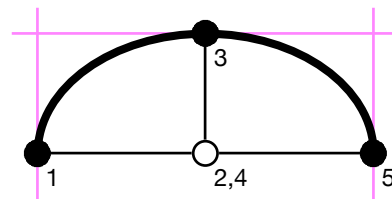
One hard corner (green) is interpolated by averaging the two (blue) spline points.

The distances from the ARC point to the two hard corners do not have to be equal (or even approximately equal), and everything will still line up perfectly.

Restriction: Don't use ARC to do angle sweeps substantially greater than 90°. (The above illustration is a good, ballpark representation of the largest angle that should be attempted.) If you need bigger sweeps, break them up into smaller ones. The reason for this restriction is that as the angle increases from 90° to 180°, the tangent-intersection point shoots off to infinity, causing massive distortion of the shape, and total failure at exactly 180°:



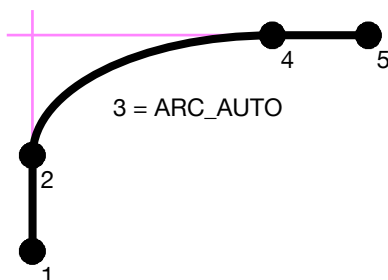
WRONG — The tangents do not intersect, or do so at an immense distance.



CORRECT — Creates a very oval-like curve of the exact, desired dimensions.

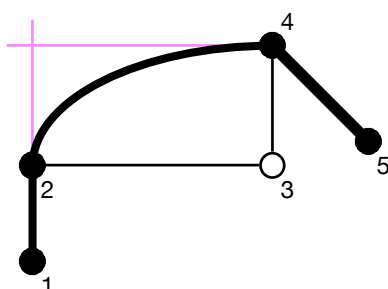
Note: The value .4142 (actually the square root of two, minus one) is used because it delivers nearly perfect circular (or oval) arcs for 90° angles, and good results for other angles too.

The **ARC_AUTO point** works exactly like the ARC point, except that it specifies no arc-center coordinates. Instead of deriving the tangent lines from a specified center-point, the tangent lines are simply assumed to be extensions of the hard-corner-specified line segments that come before and after the ARC_AUTO constant. For this reason, there must be *two* consecutive hard corners before, and two after, ARC_AUTO. Example:



As with ARC, it is not recommended to go substantially over 90° with ARC_AUTO. And due to the way the code is written, ARC_AUTO must not be in the place of the first or second point of its polygon loop, but may be in the place of the last or second-to-the-last point.

Be aware that just because an arc has two hard corners on both sides doesn't necessarily mean that ARC_AUTO should be used. For example, in this case, ARC must be used:



Important: Keep in mind that every font character in this TSG project is assumed to have its left edge lined up with the leftmost X position (-16, -1, or 0, depending on the font), and that both the left-edge alignment and the calculated width of the character are based on *hard corners only*, not spline corners, arc centers, or the horizontal reach of curves.