A Winding Number and Point-in-Polygon Algorithm

David G. Alciatore*
Department of Mechanical Engineering
Colorado State University
Fort Collins, CO 80523

(303) 491-6589
e-mail: alciator@mardigras.lance.colostate.edu

Rick Miranda
Department of Mathematics
Colorado State University

This paper presents an efficient axis-crossing algorithm for determining the winding number of a closed planar polygon about a given point. The winding number mathematically measures the number of times a polygon encloses a point. The polygon is defined by a set of ordered vertices and need not be a simple polygon (i.e., the sides may intersect). Knowledge of the winding number immediately determines whether or not a point lies within the polygon and also determines the sense of the polygon (clockwise or counterclockwise). The point-in-polygon problem is such a fundamental geometric problem and has such wide-ranging applications that an efficient numerical algorithm is an invaluable tool. Although point-in-polygon algorithms abound in the literature, none provide the complete information that the winding number offers.

INTRODUCTION

A fundamental problem encountered in two-dimensional computation geometry is determining whether a given point lies within given closed polygon. Some applications of this include: ray tracing (determining if a ray pierces a surface) [1]; painting simulation (determining points on surface that a paint fan pattern covers) [2]; robotics (determining if a point is within the reachable workspace of a robot) [3]; acoustics (determining points at which sounds wave are considered to be reflected by walls) [4]; geosciences (contouring a data set) [5]; computer graphics surface triangulation from serial section contours (determining if a contour in one serial section overlaps a contour in the previous section) [6]; etc.

A large amount of literature addresses this simple geometric problem [2–13, 15]. Most of the algorithms presented in the literature deal only with special cases of the problem. A common assumption is that the polygon is convex, and algorithms for this case exist which have logarithmic efficiency [10, 11]. Most of the point-in-polygon algorithms presented in the literature assume that the polygon is simple (i.e., has non-intersecting sides). Although this is an important and common case, there is little discussion of generalizing these algorithms to a non-simple case.

This paper proposes the use of the polygon's winding number as the basis of a point-in-polygon algorithm. The winding number of a polygon (contour) C about a point x, w, measures not only whether C encloses x, but also how many times and in which orientation C "winds around" x. In particular,

$$w = \begin{cases} 0 & \text{if } \mathbf{x} \text{ is not inside } \mathbf{C} \\ n > 0 & \text{if } \mathbf{C} \text{ winds around } \mathbf{x} \text{ n times counterclockwise} \\ n < 0 & \text{if } \mathbf{C} \text{ winds around } \mathbf{x} \text{ (-n) times clockwise} \end{cases} \tag{1}$$

The winding number can be rigorously defined as a contour integral in the complex plane [14]:

$$w = \frac{1}{2\pi i}\int_C \frac{1}{z}dz \quad \text{where} \quad z = x + iy \tag{2}$$

Note that the winding number is not defined when the point **x** is on the polygon **C**. In this paper, we present an axis-crossing algorithm for computing the winding number of **C** about **x** which avoids numerical approximation of the above integral.

Axis crossing methods have been presented in the literature [4, 7, 12] for the point-in-polygon algorithm; and in [15], the winding number is defined using axis crossings. However, in [15], no algorithm or implementation details are presented for computing the winding number. Moreover, the data structure suggested in [15] for dealing with closed polygons is quite sophisticated and complex (a signed multiset of states encoding position and attitude at each point of a motion traversing the polygon); this level of detail is unnecessary for our restricted but important and common application. Finally, in [15], a point-in-polygon test is described, but it utilizes a "sweep number" concept instead of directly applying the winding number results.

We present a concise, complete, and efficient winding number algorithm using a common and simple data structure for a polygon (ordered set of vertices). This algorithm can be used directly as a point-in-polygon test: if the winding number is nonzero, the point is in the polygon. As noted above, the winding number gives information even for non-simple polygons and can also be used to determine the orientation of the polygon.

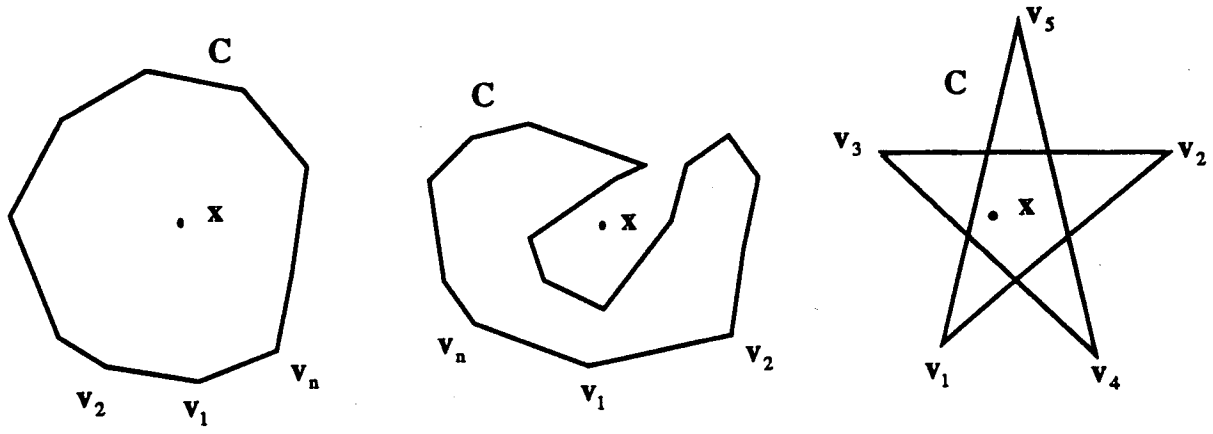
THE AXIS-CROSSING METHOD

The axis-crossing method can be used to efficiently determine the winding number of a closed polygon **C** about a point **x**. The notation to be used follows:

$\mathbf{x} = (x_0, y_0) \in R^2$	the point in question.
$\mathbf{v}_1, \cdots, \mathbf{v}_n; \quad \mathbf{v}_i = (x_i, y_i)$	the ordered vertices of C.
$\mathbf{v}_{n+1} = \mathbf{v}_1, \quad \mathbf{C} = \bigcup_{i=1}^{n} \overline{\mathbf{v}_i \mathbf{v}_{i+1}}$	the closed polygon to be tested.

Since the point-in-polygon test is invariant under horizontal and vertical translation, the geometry may be translated so that the point **x** is at the origin. This simply amounts to replacing the vertices $\mathbf{v}_i$ with $(\mathbf{v}_i - \mathbf{x})$, for each i, at the beginning of the algorithm. Therefore the algorithm can be presented in this special case with **x** as the origin with no loss in generality.

Figure 1 illustrates three types of point-in-polygon test situations which this algorithm is designed to handle. For example, for the polygons in Figure 1, w = –1 for (a), w = 0 for (b), and w = 2 for (c). We assume that x is not on C: this special case can be detected and handled either during or before the algorithm.



(a) convex C (b) non-convex C (c) convoluted C

Figure 1 Point-in-Polygon Situations

The axis-crossing method for determining the winding number consists of traversing the polygon keeping track of direction and frequency of crossing with the positive x-axis. Refer to Figure 2 for an illustration of the terminology. Briefly speaking, each time the polygon winds around the origin, the positive x-axis must be crossed; a crossing from below represents a counterclockwise winding, and from above a clockwise winding.

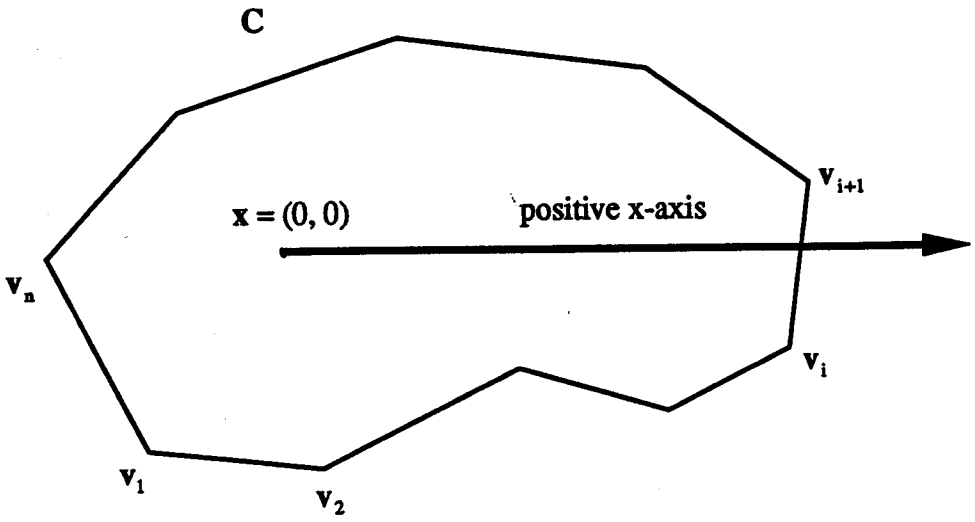


Figure 2 Axis-Crossing Method Terminology

The first step in the axis-crossing method is to initialize the winding number w to zero. Then, for each segment of the polygon, determine whether that segment crosses the positive x-axis, and in which direction. If the crossing is from below, increment w by one;

if the crossing is from above, decrement w by one. After proceeding through all of the segments of the polygon, w will equal the winding number of the polygon about the origin. Thus for each vertex v_i in the polygon the winding number is updated according to the direction and intersection of $\overline{v_i v_{i+1}}$ and the positive x-axis. This is illustrated in the first two cases in Figure 3.

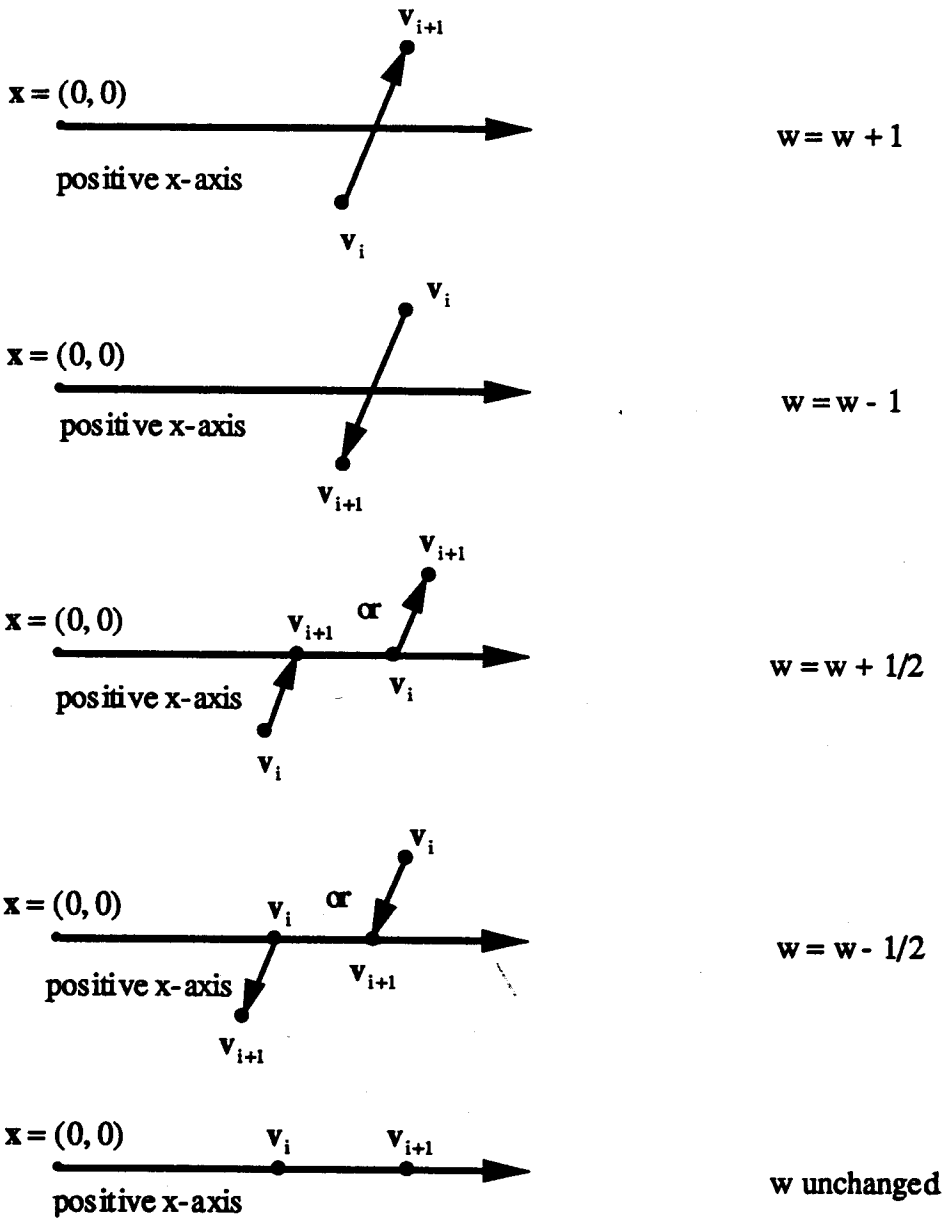


Figure 3 Winding Number Update Criterion

For digitized data of limited resolution and sometimes in cases of computer round-off, it is quite possible that one of the coordinates of a vertex in the polygon is identical to a coordinate of the test point x. Therefore it is conceivable that after translating the data so that x is the origin, one or more of the vertices may exactly lie on the positive x-axis. Many

algorithms presented in the literature ignore this case. The last three cases in Figure 3 illustrate this phenomenon. Essentially, if one of the ends of a polygon segment lies on the positive x-axis, the winding number should be incremented or decremented by one-half, depending on the direction.

Pseudocode which implements the axis-crossing method follows:

Axis-Crossing Method Winding Number Algorithm

Input: vertices v_i (i = 1 to n) of the polygon C.
Output: winding number w of polygon C about point x.

0.) Replace each v_i by $(v_i - x)$.

1.) Initialize w = 0.

2.) For each vertex v_i (i = 1 to n) in the polygon C:

if $(y_i y_{i+1} < 0)$ then [$\overline{v_i v_{i+1}}$ crosses the x-axis]

Set $r = x_i + \dfrac{y_i(x_{i+1} - x_i)}{(y_i - y_{i+1})}$

[r is the x-coordinate of the intersection of $\overline{v_i v_{i+1}}$ and the x-axis]

if (r > 0) then [$\overline{v_i v_{i+1}}$ crosses positive x-axis]

if $(y_i < 0)$ then $w = w + 1$ else $w = w - 1$

else if ($(y_i = 0)$ and $(x_i > 0)$) then [v_i is on the positive x-axis]

if $(y_{i+1} > 0)$ then $w = w + 1/2$ else $w = w - 1/2$

else if ($(y_{i+1} = 0)$ and $(x_{i+1} > 0)$) then [v_{i+1} is on the positive x-axis]

if $(y_i < 0)$ then $w = w + 1/2$ else $w = w - 1/2$

3.) Return w = winding number of C about x.

CONCLUSIONS

Presented here is a detailed axis-crossing algorithm for determining the winding number of a polygon about a point. As was shown, this number provides an efficient solution to the point-in-polygon problem with O(n) complexity (where n is the number of polygon vertices). It deals with the general case of non-simple and/or non-convex polygons and it can also be used to determine the orientation (clockwise or counterclockwise) of the polygon.

REFERENCES

[1] Arvo, J., editor, Graphics Gems II, Academic Press, 1991, pp. 247-263

[2] Freund, M., A. Traver, and D. Alciatore, "Simulation of Multiple Nozzle Surface Finishing Operations", ASME Technical Paper No. 91-PET-33, 1991.

[3] Ng, D., "Determining Workspace Boundaries of Articulated Robot Arms," Masters Thesis, Colorado State University, to appear December, 1993.

[4] Kulowski, A., "Optimization of a Point-in-Polygon Algorithm for Computer Models of Sound Field in Rooms," Applied Acoustics, V. 35, 1992, pp. 63-74.

[5] Salomon, K., "An Efficient Point-in-Polygon Algorithm," Computers and Geosciences, V. 4, 1978, pp. 173-178.

[6] Miranda, R., Fedde, C., McCracken, T., "Triangulating Between Parallel Splitting Polygons Using a Simplicial Algorithm", Proceedings of the Biostereometrics Conference, Boston, Mass, 1990.

[7] Anderson, K., "Simple Algorithm for Positioning a Point Close to a Boundary," Mathematical Geology, V. 8, N. 1, 1976, pp.105-106.

[8] Davis, M. and David, M., "An Algorithm for Finding the Position of Point Relative to a Fixed Boundary," Journal of Mathematical Geology, V. 12, N. 1, 1980, pp.61-68.

[9] Hall, J., "PTLOC – A FORTRAN Subroutine For Determining the Position of Point Relative to a Closed Boundary," Journal of Mathematical Geology, V. 7, N. 1, 1975, pp.75-79.

[10] Larkin, B., "An ANSI C Routine to Determine if a Point is Within a Specified Convex Polygon in Logarithmic Time," Computers and Geosciences, V. 17, N. 6, 1991, pp. 841-847.

[11] Lee, D. and Preparata, F., "Computational Geometry – A Survey," IEEE Transactions on Computers, V. C-33, N. 12, 1984, pp. 1072-1101.

[12] Shimrat, M., "Position of Point Relative to Polygon," Communications of the ACM, V. 5, N. 8, Algorithm 112, 1962, pg. 434.

[13] Yamaguchi, F., Niizeki, M., Fukunaga, H., "Two Robust Point-in-Polygon Tests Based on the 4x4 Determinant Method," Advances in Design Automation, V. 1, Computer-aided and Computational Design, 1990, pp.89-95.

[14] Churchill, R., Complex Variables and Applications, 2nd ed., McGraw-Hill, New York, 1960.

[15] Guibas, L., Ramshaw, L., and Stolfi, J., "A Kinetic Framework for Computational Geometry," Proceedings of the 24th Annual Symposium on Foundations of Computer Science, November, 1983, pp. 100-111.