Bounding Box and Hit Detection Algorithms for 2D Graphic Objects

Keith Andrews

Institute for Information Processing and Computer Based New Media (IICM), Graz University of Technology

Abstract

This paper presents a selection of bounding box and hit detection algorithms for the 2D graphic objects defined in the Computer Graphics Interface (CGI), which is the ISO standard for low-level computer graphics. The algorithms are based on well-known geometric principles, except for the ellipse algorithms whose underlying methods are new. Such algorithms are used (among other things) to efficiently implement the picking of graphic objects.

1 Introduction

Object-oriented graphics editors are becoming essential in increasingly diverse fields of application, from computer aided design to desk top publishing. A common feature of such editors is the selection of graphic objects by *picking* (eg. moving a pointing device such as a mouse and clicking on the desired graphic object). In order to implement efficient picking of graphic objects, *bounding box* and *hit detection* algorithms are required for each of the graphic objects supported. Some of the graphic objects frequently available include polylines, circular arcs, polygons, ellipses and cell arrays.

In general, two kinds of picking are possible:

- *Single Picking*: picking a single graphic object by specifying a point.
- *Fence Picking*: picking a group of graphic objects by specifying a rectangular area.

A graphic object's bounding box is defined as the smallest rectangle which totally encloses the graphic object. It is used in single picking as a first check to see whether a graphic object is a possible candidate for selection: if the user-specified pick position lies outside the bounding box (slightly stretched to allow for a margin of error), then the object is certainly not a candidate. If the pick position lies inside the bounding box, then a more refined algorithm (the hit detection algorithm) is used. It determines whether the user-specified pick position lies on or near the graphic object proper. Fence picking is implemented by comparing the user-specified rectangle with the bounding boxes of all the graphic objects currently displayed.

Another use for bounding boxes is to reduce the amount of work involved when an area of the display has to be redrawn (eg. after being obscured by a menu). The clipping rectangle is set to the area concerned and only those graphic objects whose bounding boxes intersect with this area need to be redrawn.

The algorithms presented here were developed for the EDEN graphics editing environment [1] and can be found in the author's Master's thesis [2]. The ellipse algorithms (sections 4.4 and 4.5) were developed from scratch; the other algorithms make use of well-known geometric methods.

2 Bounding Box Algorithms

There follows a presentation of the bounding box calculation algorithms for some of the various graphic objects defined in CGI [3].

2.1 Polyline

The bounding box of the definition points is determined and then stretched by the line width.

2.2 Polygon

The bounding box of the definition points is determined. If the edge visibility flag is true, the bounding box is stretched by the edge width.

2.3 Circle

The bounding box of the four extreme points is determined. If the edge is visible, the bounding box is stretched by the edge width.

2.4 Circular Arc 3 Point

First the circular arc is checked for degeneracy. For a Circular Arc 3 Point, the three definition points (start point, intermediate point and end point) are checked to see if they are collinear or coincident. If the arc is degenerate, then the bounding box is calculated according to the degenerate representation, as defined in [3]. Otherwise, the centre, radius and start, intermediate and end vectors are determined and the four extreme points (leftmost, rightmost, topmost, bottommost) on the full circle are calculated (see section 4.2).

The bounding box of the mathematical (infinitely thin) arc is calculated as the bounding box of the start point, the end point and any extreme points included in the angle from start vector through intermediate vector to end vector. The bounding box is then stretched by the line width of the arc to give the actual bounding box of the object.

2.5 Circular Arc 3 Point Close

Analogous to Circular Arc 3 Point, except that in the case of pie closure (a sector of a circle), the centre is also included in the set of points making up the bounding box. If the edge is visible, the bounding box is stretched by the edge width.

2.6 Ellipse

An ellipse is defined by its centre and two conjugate radius (diameter) endpoints. If the centre and conjugate radius endpoints are collinear or coincident, the ellipse is degenerate and the bounding box is calculated according to the degenerate representation, as defined in [3]. If the ellipse is non-degenerate, the bounding box is calculated as follows. The four extreme points of the ellipse are calculated (see section 4.4) and their bounding box determined. If the edge is visible, the bounding box is stretched by the edge width.

2.7 Elliptical Arc

An elliptical arc is defined by its centre, two conjugate radius endpoints (x_1, y_1) and (x_2, y_2), a start vector and an end vector. If the centre and conjugate radius endpoints are collinear or coincident, then the elliptical arc is degenerate and the bounding box is calculated according to the degenerate representation, as defined in [3].

If the arc is non-degenerate, the bounding box is calculated as follows. The four extreme points on the full ellipse, the intersection of the start vector with the ellipse (start point) and the intersection of the end vector with the ellipse (end point) are calculated. The bounding box of the mathematical (infinitely thin) elliptical arc is now calculated as the bounding box of the start point, the end point and any extreme points included in the angle from start vector to end vector. This bounding box is then stretched by the line width of the arc to give the actual bounding box of the object.

Whether one proceeds in a clockwise or anti-clockwise direction from start vector to end vector is determined as follows: "The elliptical arc is drawn from the start point to the end point in the direction from the first conjugate radius endpoint to the second conjugate radius endpoint through the smaller of the two ellipse segments enclosed by the conjugate radii." [3]. In practice this means that the arc is drawn clockwise if $x_1y_2 < x_2y_1$ and anti-clockwise if $x_1y_2 > x_2y_1$.

2.8 Elliptical Arc Close

Analogous to Elliptical Arc, except that in the case of pie closure (a sector of an ellipse), the centre is also included in the set of points making up the bounding box. If the edge is visible, the bounding box is stretched by the edge width.

3 Hit Detection Algorithms

Hit detection refers to the process of determining whether a test point is on or near (within a certain tolerance, δ) a graphic object. Hit detection is required for the single-picking of graphic objects, as explained in the introduction.

This section describes some of the hit detection algorithms used for particular graphic objects. A first check is always whether the test point is inside or outside the graphic object's bounding box (stretched by δ). If it is outside, then the object is certainly not hit. If it is inside, then the object may or may not be hit, according to its type and geometry: further inspection is required. The following notes present some of the hit detection algorithms used for graphic objects.

3.1 Polyline

The polyline is hit, if the distance of the test point from any of the individual line segments is less than the tolerance, δ. The "distance from point to line" algorithm (see section 4.1) is used, with the line width of the object also taken into account:

$$\text{hit detected} \quad \Leftrightarrow \quad \text{distance} < \tfrac{1}{2}\ (\text{line width}) + \delta$$

3.2 Polygon

The polygon is hit, if the test point is within δ of any edge. A filled polygon is also hit if the test point lies inside the polygon. The edge check uses the "distance from point to line" algorithm (see section 4.1), with the edge width taken into account if the edge is visible. For a filled Polygon, the "point inside polygon" algorithm (see section 4.3) is used for the interior point test.

3.3 Circle

The circle is hit, if the test point is within δ of the perimeter. If the edge is visible, the edge width is also taken into account. For a filled circle, the circle is also hit if the test point lies inside the circle.

3.4 Circular Arc 3 Point

The circular arc is hit, if the test point is within δ of the full circle and is also included within the angle defining the arc. The line width of the arc must also be taken into account.

3.5 Circular Arc 3 Point Close

The closed circular arc is hit, if the test point is within δ of the perimeter and is included within the angle defining the arc. For pie closure, a hit is also registered if the test point is within δ of either the line between centre and start point or the line between centre and end point. For chord closure, a hit is registered if the test point is within δ of the line between start point and end point. If the edge is visible, the edge width is also taken into account.

For a filled closed arc, a hit is also registered if the test point lies inside the closed arc. For pie closure this means the test point must be inside the full circle and included in the angle defining the arc. For chord closure there are two cases, as shown in Figure 1:

1. The arc's defining angle is less than or equal to π, in which case the test point must be inside the pie, but outside the triangle formed by centre, start and end points.

2. The arc's defining angle is greater than π, in which case the test point must either be inside the pie or inside the triangle formed by centre, start and end points.

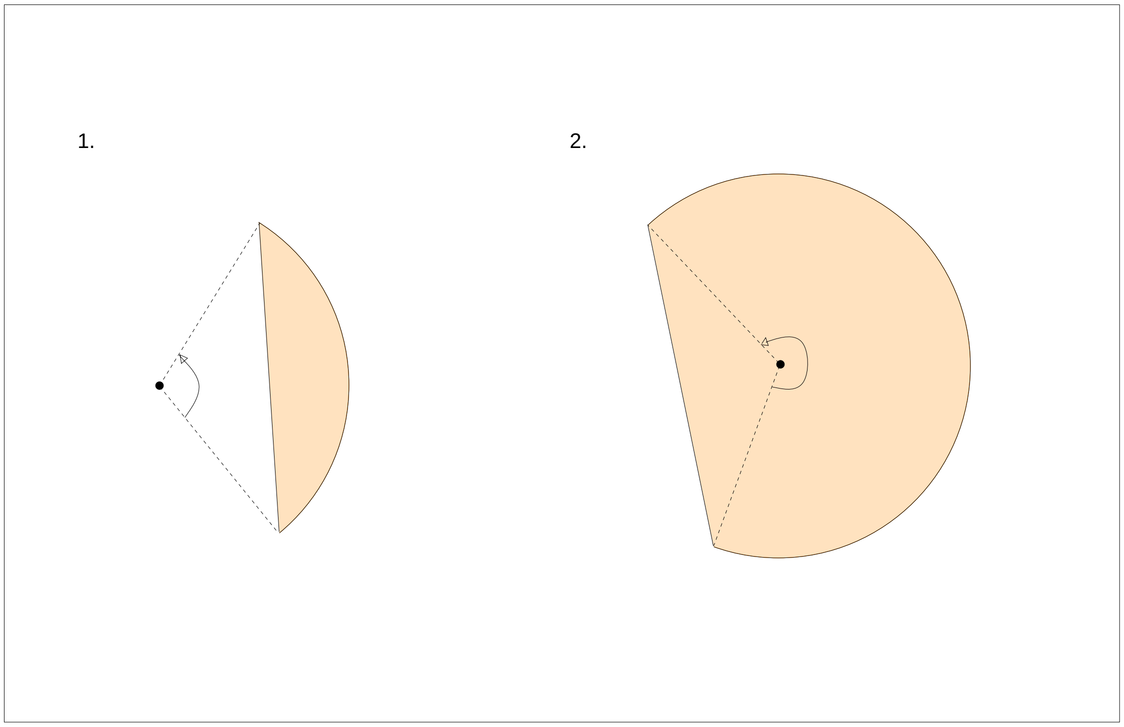

Figure 1: Interior of a Circular Arc 3 Point Close (Chord)

3.6 Ellipse

The ellipse is hit, if the test point is within δ of the perimeter. If the edge is visible, the edge width is also taken into account. For a filled ellipse, the ellipse is also hit if the test point lies inside the ellipse. The algorithm "point near ellipse" (see section 4.5) and the simple interior point test are used.

3.7 Elliptical Arc

The elliptical arc is hit, if the test point is within δ of the full ellipse and is also included within the angle defining the arc. The "point near ellipse" algorithm (see section 4.5) is used, with the line width of the arc also taken into account.

3.8 Elliptical Arc Close

Analogous to Circular Arc 3 Point Close.

4 Geometric Algorithms

4.1 Distance from Point to Line

For hit detection during picking, a common problem is to find the distance of a given point from a given line. Assume that the point is $P = (x_p, y_p)$ and the line is the line between $A = (x_a, y_a)$ and $B = (x_b, y_b)$, as shown in Figure 2.

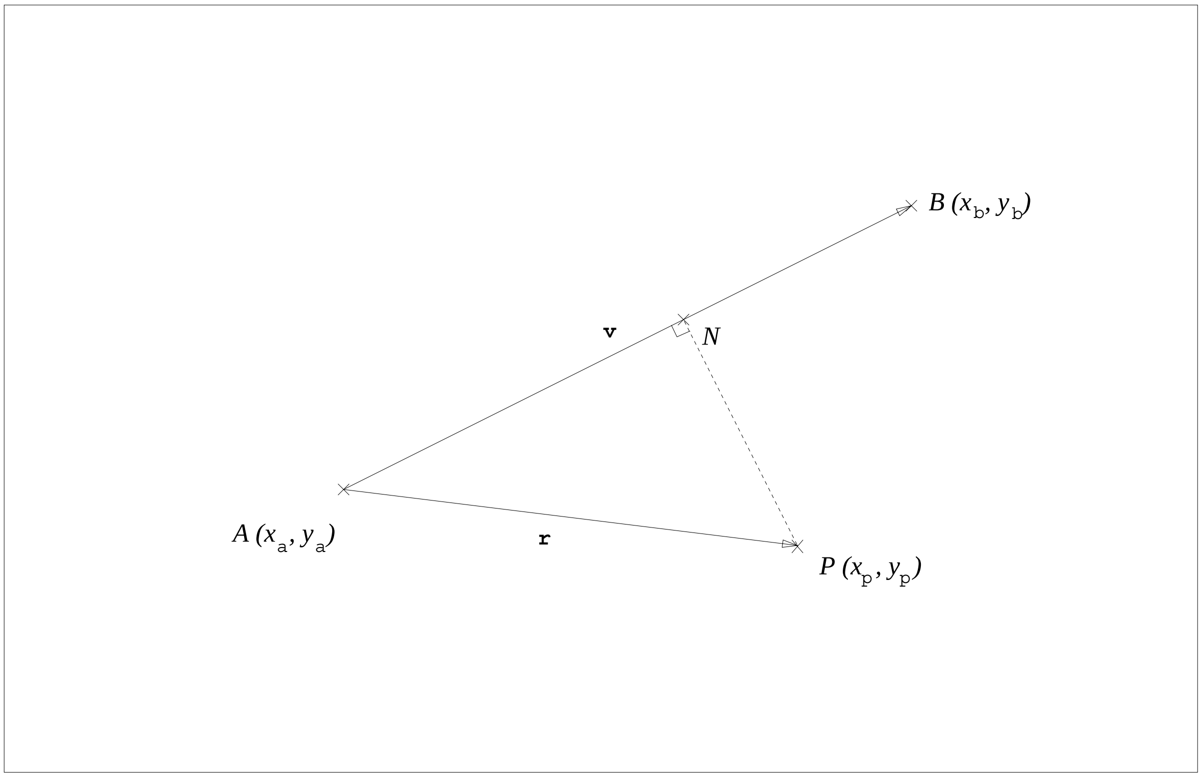

Figure 2: Distance from Point to Line

Defining **v** as the vector AB, **r** as the vector AP and using the vector form of the line equation, the point (N) on the line nearest P can be determined as follows:

$$\begin{aligned} v &= (x_v, y_v) &:= B - A &= (x_b - x_a, y_b - y_a) \\ r &= (x_r, y_r) &:= P - A &= (x_p - x_a, y_p - y_a) \end{aligned}$$

$$N = A + tv \ ; \quad t = \frac{r \cdot v}{v \cdot v} = \frac{x_r x_v + y_r y_v}{x_v^2 + y_v^2}$$

The distance, d, from P to N is given by:

$$\begin{aligned}
d &= \|r - tv\| \\
d^2 &= (x_r - tx_v)^2 + (y_r - ty_v)^2 \\
&= x_r^2 + y_r^2 + t^2(x_v^2 + y_v^2) - 2t(x_r x_v + y_r y_v) \\
&= x_r^2 + y_r^2 + \frac{(x_r x_v + y_r y_v)^2}{(x_v^2 + y_v^2)} - 2\frac{(x_r x_v + y_r y_v)^2}{(x_v^2 + y_v^2)} \\
&= x_r^2 + y_r^2 - \frac{(x_r x_v + y_r y_v)^2}{(x_v^2 + y_v^2)} \\
d^2 &= \frac{(x_r y_v - y_r x_v)^2}{(x_v^2 + y_v^2)} \\
d &= \frac{x_r y_v - y_r x_v}{\sqrt{x_v^2 + y_v^2}}
\end{aligned}$$

In order to check if the distance is smaller than a given ϵ, it is sufficient to use d^2 and ϵ^2, since $d < \epsilon \Leftrightarrow d^2 < \epsilon^2$. This avoids the need to take a square root.

4.2 Centre and Radius of Circle Passing Through 3 Given Points

For certain graphic objects (Circular Arc 3 Point, Circular Arc 3 Point Close) it is necessary to determine the centre and radius of a circle passing through three given points.

Assuming that the three points are $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$ and $P_3 = (x_3, y_3)$ and using straightforward geometry, the centre of the circle can be determined as the intersection of the bisectors of P_1P_2, P_2P_3 and P_3P_1. This is shown in Figure 3.

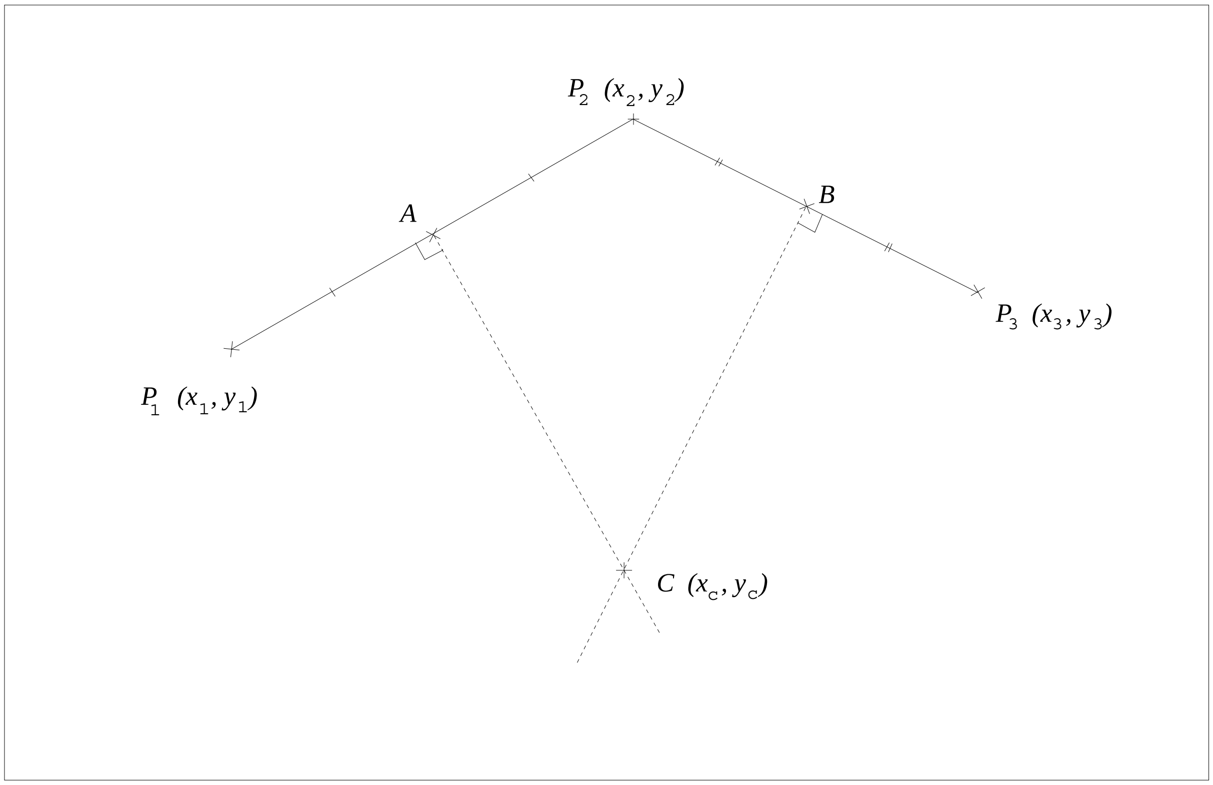

Figure 3: Centre and Radius of Circle Through 3 Points

The points A and B are given by:

$$A = (\frac{x_1 + x_2}{2}, \frac{y_1 + y_2}{2}), \quad B = (\frac{x_2 + x_3}{2}, \frac{y_2 + y_3}{2})$$

The equation of AC is:

$$y = \frac{2(x_1 - x_2)x + (y_2 + y_1)(y_2 - y_1) + (x_2 + x_1)(x_2 - x_1)}{2(y_2 - y_1)}$$

The equation of BC is:

$$y = \frac{2(x_2 - x_3)x + (y_3 + y_2)(y_3 - y_2) + (x_3 + x_2)(x_3 - x_2)}{2(y_3 - y_2)}$$

The centre of the circle $C = (x_c, y_c)$ is found by setting the two equations equal:

$$\begin{aligned} x_c \;=\; & [\ (y_2 - y_1)(y_3 + y_2)(y_3 - y_2) + (y_2 - y_1)(x_3 + x_2)(x_3 - x_2) \\ & - (y_3 - y_2)(y_2 + y_1)(y_2 - y_1) - (y_3 - y_1)(x_2 + x_1)(x_2 - x_1)\ ]\ / \\ & 2[\ (y_3 - y_2)(x_1 - x_2) - (y_2 - y_1)(x_2 - x_3)\ ] \end{aligned}$$

If $(y_3 - y_2)(x_1 - x_2) - (y_2 - y_1)(x_2 - x_3) = 0$, the three points are collinear or coincident and no solution is possible.

$$y_c = \frac{2(x_1 - x_2)x_c + (y_2 + y_1)(y_2 - y_1) + (x_2 + x_1)(x_2 - x_1)}{2(y_2 - y_1)} \qquad y_2 \neq y_1$$

$$y_c = \frac{2(x_2 - x_3)x_c + (y_3 + y_2)(y_3 - y_2) + (x_3 + x_2)(x_3 - x_2)}{2(y_3 - y_2)} \qquad y_2 = y_1$$

The radius, r, of the circle can now be found as the distance between C and any of P_1, P_2 or P_3:

$$r = \sqrt{(x_c - x_1)^2 + (y_c - y_1)^2}$$

4.3 Point Inside Polygon

For hit detection while picking a filled polygon, it is necessary to determine whether a given point (the current pick position) lies inside or outside the polygon.

To solve this problem the odd/even parity test is used. A test ray from the given point extends infinitely in any direction (for example in the positive y direction). If the ray intersects an odd number of the polygon's edges, the point lies inside the polygon. If the ray intersects an even number of edges, the point lies outside the polygon. Should the ray pass through a (polygon) vertex tangentially, no intersection is counted. How the test works is illustrated in Figure 4.

Kappe [4] presented an efficient implementation of the odd/even parity test, with a test ray extending to $+\infty$ in the y direction. It is based on the observation that the test point divides the plane into four quadrants. If both endpoints of an edge are in the same quadrant, then the test ray certainly does *not* intersect the edge. If the two edge endpoints are in different quadrants, there are six cases, as shown in Figure 5. A specifies the left endpoint of an edge, B the right endpoint. The ray only intersects the edge in cases 1, 5a and 6a.

The special case of the test ray passing through a vertex point is handled by designating the vertical dividing line as belonging to the lefthand quadrants (but not to the righthand quadrants). If the test ray passes through the vertex tangentially, then either zero (ray to left of polygon) or two (ray to right of polygon) intersections are counted; in either case an even number.

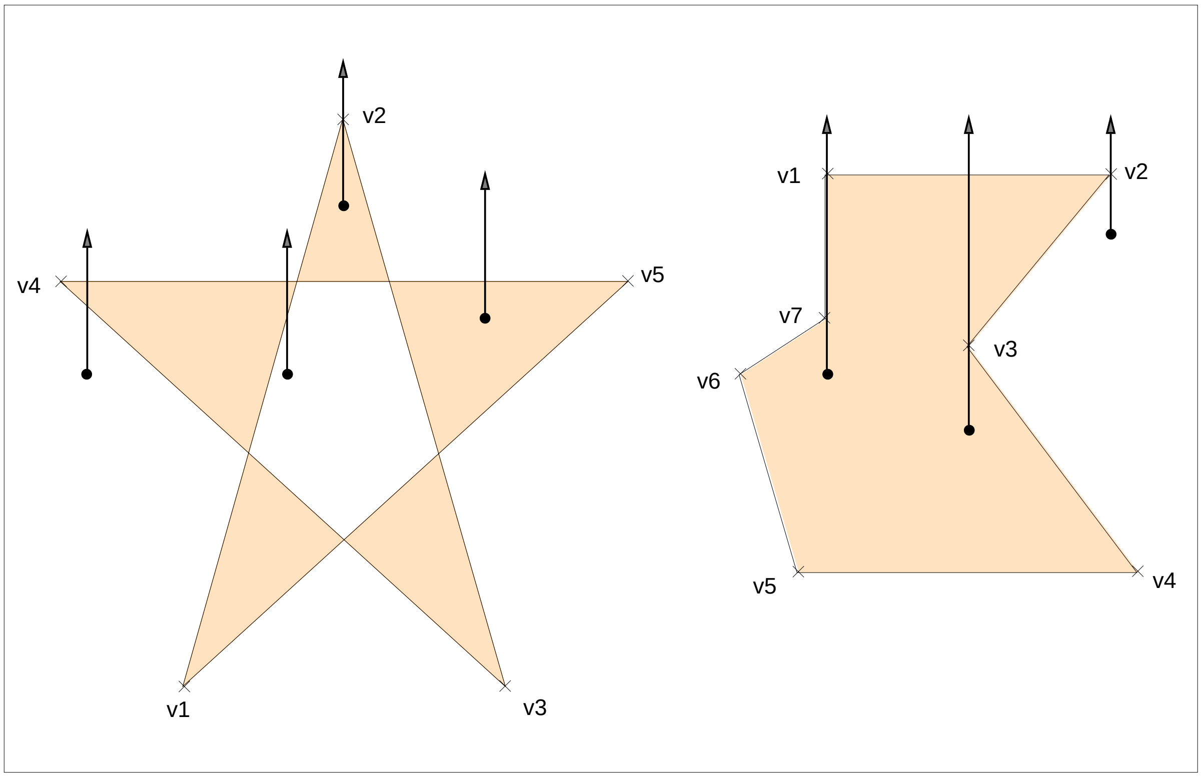

Figure 4: Odd/Even Parity Test

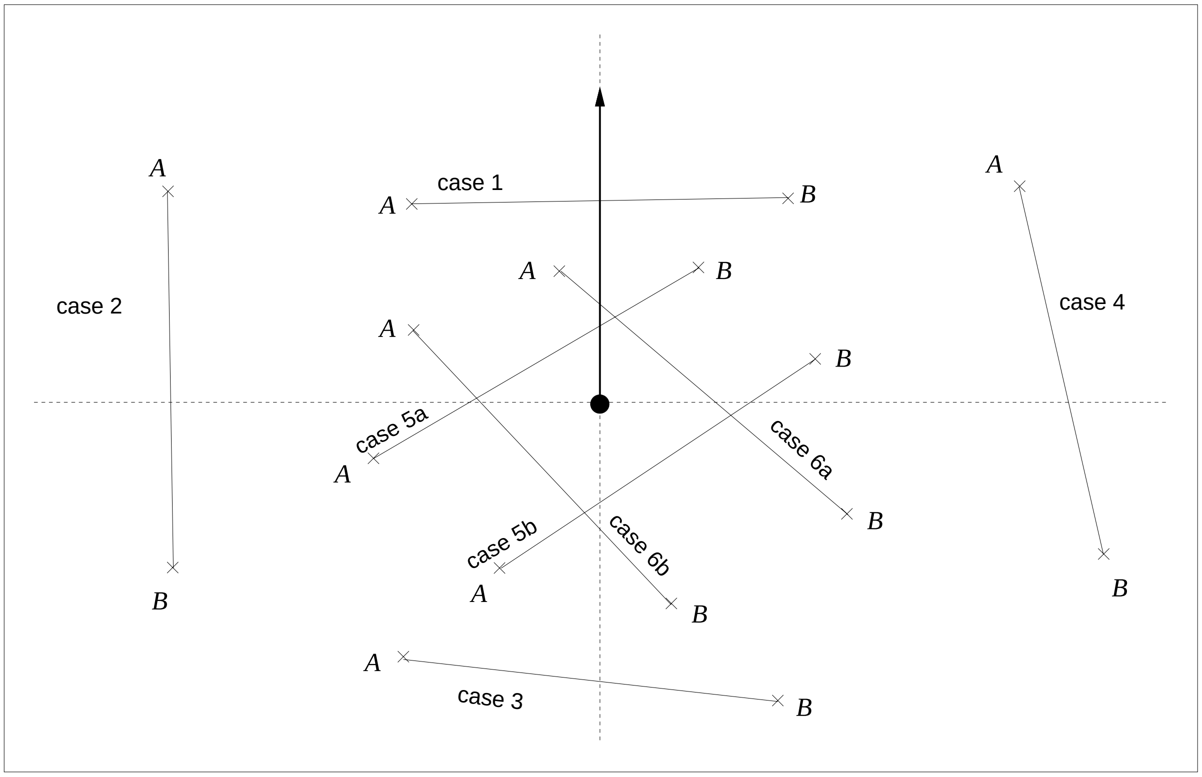

Figure 5: Polygon Edge Intersection by Quadrants

4.4 Extreme Points of Ellipse

The equation of an ellipse centered at the origin has the form:

$$S(x,y) = Ax^2 + Bxy + Cy^2 + F = 0$$

where the coefficients A, B, C and F are determined by the coordinates of the two conjugate diameter endpoints. Any ellipse can be assumed, without loss of generality, to be centered at the origin, since if this is not the case, then a simple translation can be applied.

Assuming that the first conjugate diameter endpoint, cd1 $= (x_1, y_1)$, and the second, cd2 $= (x_2, y_2)$, then the coefficients are given by:

$$\begin{aligned} A &= y_1^2 + y_2^2 \\ B &= -2(x_1y_1 + x_2y_2) \\ C &= x_1^2 + x_2^2 \\ F &= -(x_1y_2 + x_2y_1)^2 \end{aligned}$$

A point $P = (x_p, y_p)$ is inside the ellipse if:

$$S(x_p, y_p) = Ax_p^2 + Bx_py_p + Cy_p^2 + F < 0$$

The four extreme points of an ellipse (leftmost, rightmost, bottommost and topmost) are used in calculating the bounding box of an ellipse or elliptical arc, as shown in Figure 6.

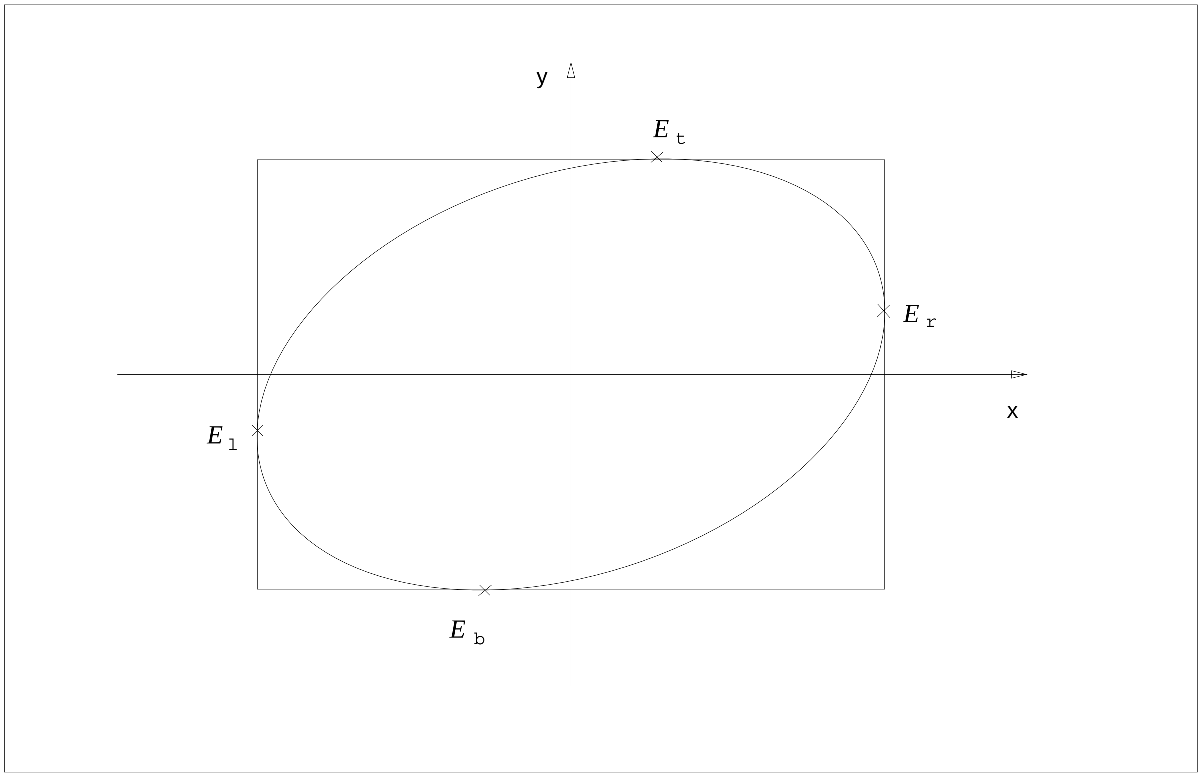

Figure 6: Extreme Points of Ellipse

The parametric form of the ellipse equation:

$$\begin{aligned} x &= c\cos(t) \\ y &= d\cos(t + \delta) \end{aligned}$$

where $c = \sqrt{x_1^2 + x_2^2} = \sqrt{C}$ and $d = \sqrt{y_1^2 + y_2^2} = \sqrt{A}$, makes it easy to see that the four extreme points E_l, E_r, E_b and E_t on the ellipse are where $x = -\sqrt{C}$, $x = +\sqrt{C}$, $y = -\sqrt{A}$ and $y = +\sqrt{A}$ respectively.

Hence the four extreme points are given by:

$E_l = \left(-\sqrt{C}, \frac{B}{2\sqrt{C}}\right)$	leftmost
$E_r = \left(\sqrt{C}, \frac{-B}{2\sqrt{C}}\right)$	rightmost
$E_b = \left(\frac{B}{2\sqrt{A}}, -\sqrt{A}\right)$	bottommost
$E_t = \left(\frac{-B}{2\sqrt{A}}, \sqrt{A}\right)$	topmost

4.5 Point Near Ellipse

For hit detection while picking an ellipse, it is necessary to determine whether the pick position $P = (x_p, y_p)$ is within a certain distance δ of the ellipse.

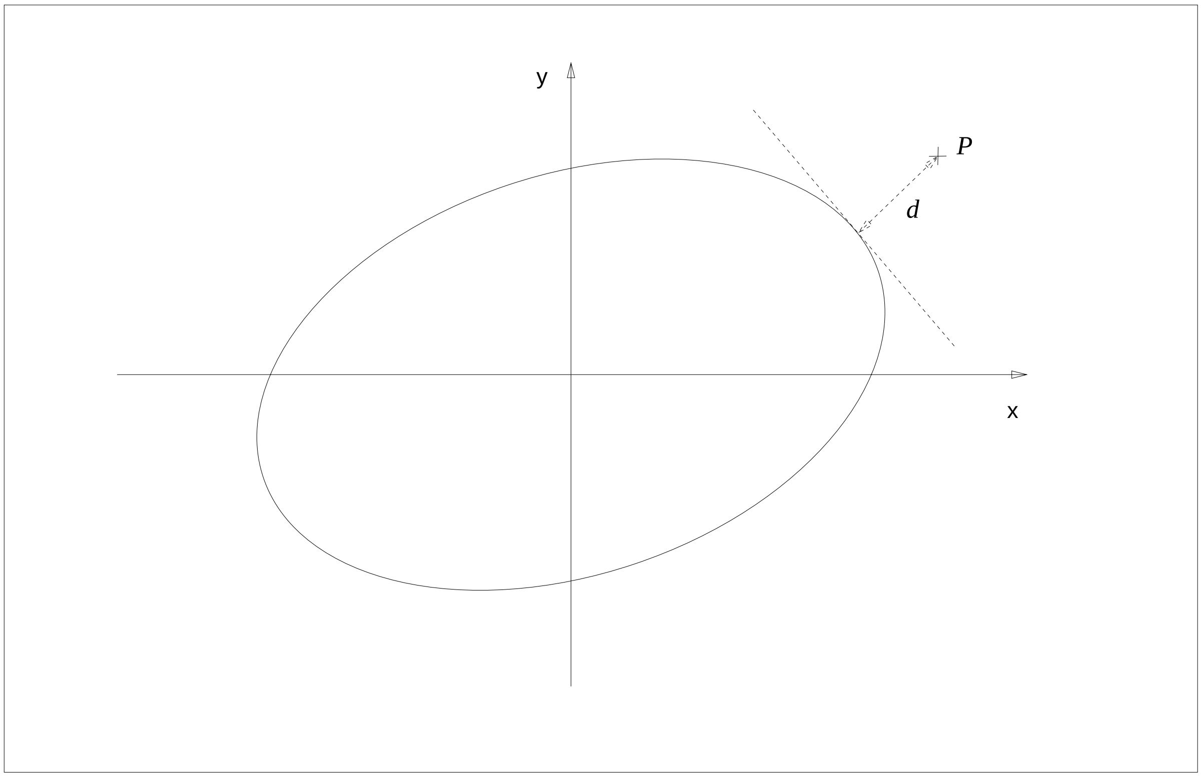

Figure 7: Distance of Point P from Ellipse

Since there is no easy formula for the distance d of a point from an ellipse (Figure 7), the following mechanism was derived. An 8-way star of radius δ is constructed around the point P, as shown in Figure 8. If at least one of the nine points is within the ellipse and at least one is outside the ellipse, then the star straddles the ellipse and P is within δ of it. The simple test for an interior point:

$$S(x, y) = Ax^2 + Bxy + Cy^2 + F < 0$$

is used on each of the star's nine points.

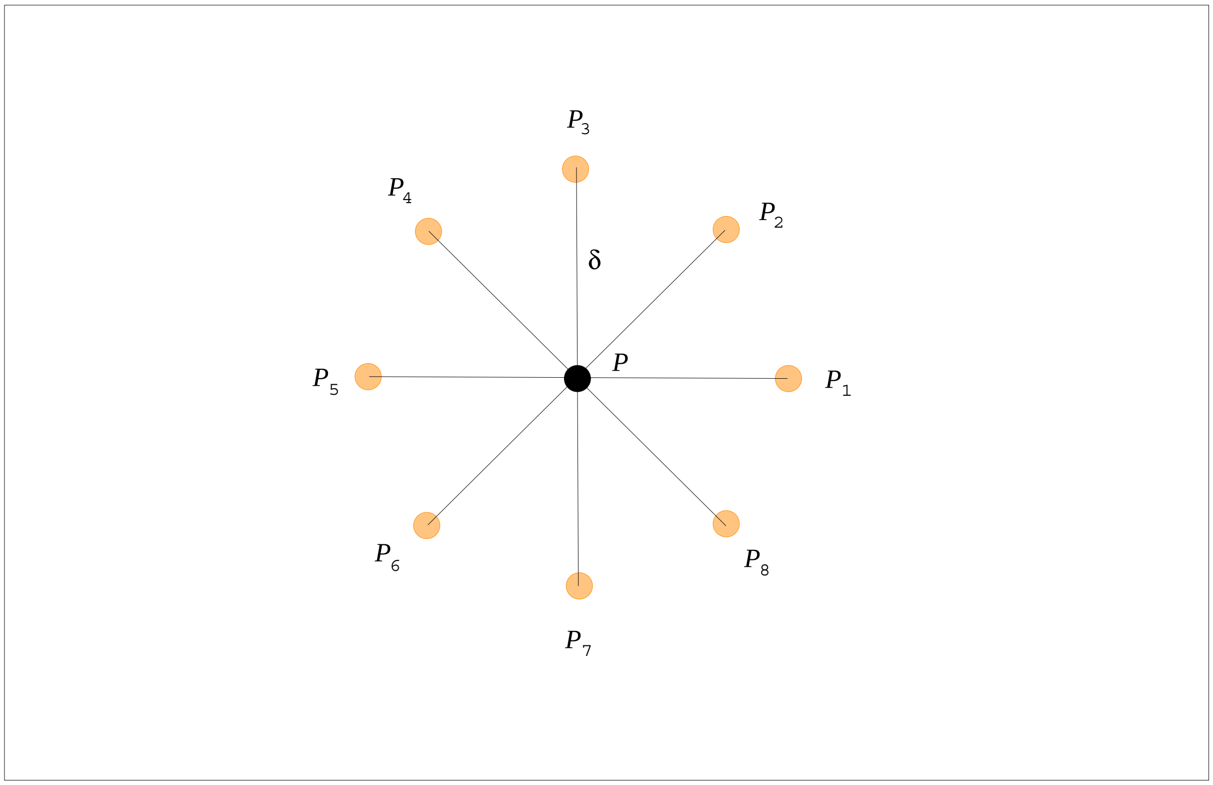

Figure 8: 8-Way Star Around Point P

It is possible that certain very thin ellipses may fall between the spokes of the star and not be hit, even though they are within δ of P. However, such ellipses are rare in practice and can still be picked by moving P slightly inside the ellipse.

References

[1] Fellner W. D., Kappe F.: *EDEN – An Editor Environment for Object-Oriented Graphics Editing*; Eurographics '90, North-Holland, (Sept. 90), pp. 425-437.

[2] Andrews K.: *Distributed EDEN – Object-Oriented Design of the EDEN Kernel as a Distributed System*; Master's Thesis, IICM, Graz University of Technology, Austria (May 1991).

[3] ISO: *Information Processing Systems – Computer Graphics – Interfacing techniques for dialogues with graphical devices (CGI), Part 1-6*; ISO 2nd DP 9636 (Nov. 1988).

[4] Kappe F.: *Design und Implementierung eines EDEN-basierenden Bildschirmtext-Editors (Design and Implementation of an EDEN-based Videotex Editing System)*; Master's Thesis, IIG, Graz University of Technology, Austria (Feb. 1988).