July 6, 2003

An Incremental Method for Drawing Circular Arcs Using Properties of Oriented Lines

by C. Bond, ©2002

1 Background

Any graphics algorithm developer who has implemented lines, circles and ellipses will at some point consider drawing arc segments. Generally, the most efficient algorithms for scan conversion are incremental methods similar to those developed by Bresenham or based on 'mid-point' decision processes. It is therefore natural to investigate the application of such methods to the scan conversion of arcs.

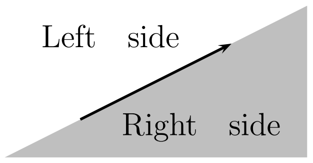
Unfortunately, incremental methods for drawing arcs are less well-known and the literature on this subject is limited.

In this paper, a new method for drawing circular or elliptical arc segments is presented. The method supports incremental scan conversion and can be extended for drawing filled circular sectors. To simplify the presentation, we assume that the centers of all circles are translated to the origin.

2 Oriented Lines

A line divides the 2-D drawing plane into two halves. If the line is oriented, *i.e.,* has a preferred direction, the two halves can be unambiguously identified as 'left side' and 'right side'. Any point in the plane will then be on the line, to the left of the line, or to the right of it.

Clearly a line through the center of a circle will divide the circle into halves and, if the line is oriented, will allow us to classify the halves as 'left' or 'right', as shown.



Extending the notion of line orientation to the case at hand, consider the lines from the center of a circle to either end of an arc segment. These lines are oriented and can be used to uniquely specify the portion of the circle between them. To show how this is done, we need to review some of the mathemetics for lines in a plane.

There are several ways to express the equation for a line from the origin to the point. Here are some of the standard forms for the equation of a line.

Point slope form:

$$y = mx + b \tag{1}$$

General line in a plane:

$$ax + by + c = 0 \tag{2}$$

The slope in (1), $m = dy/dx$, is constant and is a directed quantity indicating the tangent of the angle the line makes with the x axis. Note that either of these forms can be transformed into the other. For the case of lines through the origin, the constant terms, b in (1) and c in (2) are 0. Hence, we can write,

$$y * dx - x * dy = 0, \tag{3}$$

to determine all (x, y) pairs which are on the line.

Let x_p and y_p be the coordinates of a point in a plane. For the line associated with this point (3) becomes

$$y * x_p - x * y_p = 0. \tag{4}$$

As before, any (x, y) pair which represents a point on the line will satisfy this equation. For any (x, y) pair *not* on the line, the value of the left side of (4) will be less than or greater than zero, depending on which side of the line the point lies. If the point lies on the right side of the line, the value will be less than zero. If it lies on the left, it will be greater than zero. A better formulation for our purposes is

$$S = y * x_p - x * y_p \tag{5}$$

Although the above equation involves two multiplications, an incremental version can avoid multiplication for updates. For example, at any point, n, (5) can be written,

$$S_n = y_n * x_p - x_n * y_p. \tag{6}$$

Then, if we move to the next point by incrementing y,

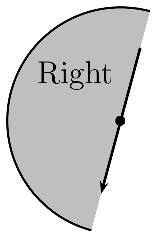
$$y_{n+1} = y_n + 1 \tag{7}$$
$$x_{n+1} = x_n \tag{8}$$
$$S_{n+1} = (y_n + 1) * x_p - x_n * y_p \tag{9}$$
$$S_{n+1} = S_n + x_p \tag{10}$$

Similar equations can be derived for any combination of positive or negative increments to y and/or x.

Eq. (6) can be updated incrementally, once initialized, to follow any trajectory in the plane. Then the sign of S_n at any point indicates which side of the line the point is on.

3 Incremental Methods for Semicircles

This is the simplest case for drawing a portion of a circle, as it involves only dividing the circle into two halves. A single line through the center is required. Using the previous terminology, we consider drawing only the *left* or *right* half.



Suppose we are generating the circle by incrementally updating a scan conversion algorithm. At each step, either the x coordinate or the y coordinate or both will change by ± 1. It is clear that we can determine when the circle intersects or crosses over (4) by also incrementally updating S_n as we move along the circle.

For example, let an origin-centered circle have a radius R. Then if we initialize the circle scan conversion algorithm to start at the uppermost point of the circle, $(0, R)$, we should initialize the value of S_n to $R * x_p$. When the circle algorithm causes a step in the x direction by ± 1, $S_{n+1} = S_n \mp y_p$. When a step in the y directions occurs, $S_{n+1} = S_n \pm x_p$. Note that both steps may occur at some points.

The plotting algorithm simply tests the sign of S_n to determine whether to plot the current point. The strategy for plotting semicircles is then,

1. Initialize circle drawing algorithm,
2. Initialize S_n,
3. Update circle algorithm,
4. Update S_n,
5. Test S_n for plot decision,
6. Loop to 3 until done.

Since the only change from a full-circle drawing algorithm is to update and test S_n, which involves one or two integer adds and a compare per pixel, the algorithm can be implemented very efficiently.

3.1 Calculating x_p and y_p

If the semicircle is specified by the angle of its diameter with respect to the horizontal axis, it will generally be given as a floating point value in radians or degrees. We need to determine a suitable value for x_p and y_p for use in integer calculations. Optimal values are not necessary, as we are only interested in a 'nearest pixel' calculations for the plot.

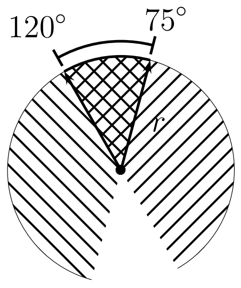
A suitable strategy is to multiply the cosine of the angle by 1000 for x_p and the sine by 1000 for y_p. This will provide sufficient resolution for today's video displays.

4 General Arcs

Having solved the problem for plotting semicircles, we now turn to the problem of plotting arcs smaller than a semicircle. It is easily seen that this problem involves two angles: one for the start of the arc, and one for its end.

The most direct solution is to simply plot points on the circle which correspond to the logical AND of two implied semicircles – one to the left of the line associated with the starting angle, and one to the right of the line associated with the ending angle.

This strategy is illustrated in the following figure, which draws an arc of radius r from 75° to 120°.



This strategy will correctly draw any arc associated with an angle less than 180°. For angles greater than 180°, simply plot the logical OR of the semicircular regions associated with the left of the starting line and the right of the ending line.

5 Extensions and Variations

There are many variations of incremental circle drawing algorithms, and each of these may be suitable for drawing arcs. A little preparatory logic in the algorithm can adapt 'octant' circle methods, for example. Such methods effectively calculate points in one octant and use symmetry to plot eight corresponding points around the circle. To plot arcs, simply create and initialize the eight auxiliary variables corresponding to each octant for the starting and ending lines, and update them in the plotting subroutine. Slightly more logic can be used to include or exclude certain octants depending on the range of the arc.

Elliptical arcs can also be drawn with the above method.

Extending the method to draw sectors or filled sectors is done by drawing the radii associated with the start and end of the arc, and invoking a scan fill routine, respectively.