Circular Arc Subdivision

Ken Turkowski
Media Technologies: Computer Graphics
Advanced Technology Group
Apple Computer, Inc.

Abstract: A circular arc is converted into a series of straight line segments, by taking advantage of the property that when a circular arc is bisected, each half-arc has a chordal deviation one quarter that of the original arc.

3 October 1994

Apple Technical Report No. 97

Circular Arc Subdivision

Ken Turkowski
3 October 1994

Introduction

This gem presents an algebraic solution to the rendering problem of circular arcs. These forms commonly arise in graphic design. For instance, well-designed typefaces may apply a very slight curvature on a portion of a letterform[1] to produce a profound aesthetic effect.

A circular arc may be represented in terms of the center and radius of its parent circle, plus the starting and ending angles θ_{st} and θ_{end}. This suggests a first-principles solution which generates successive points on the arc by evaluating the sine and cosine of a series of intermediate angles.

$$\begin{aligned} x &= x_0 + r\cos\theta \\ y &= y_0 + r\sin\theta \end{aligned} \qquad \theta_{st} \le \theta \le \theta_{end}$$

Clearly, this approach is computationally intensive. Moreover, the method has numerical problems if the radius is large, the circle's center is remote or the values have limited precision (e.g., fixed-point or even single-precision floating point). All these undesirable conditions arise for any arc having a very small bend, as its radius of curvature rapidly grows to infinity.

An algebraic, vector-based approach instead subdivides the arc into two halves until the (inverse radius of) curvature goes to zero and a vector suffices. This suggests a recursive implementation which can also terminate when a sufficient number of intermediate vertices have been produced (i.e., the length of the intermediate vector is very small, independent of curvature). This approach avoids both trigonometric operations and ill-conditioned formulas. The method is derived from related work [Karow87] based upon a suggestion. Both the expressions and an error analysis are presented below.

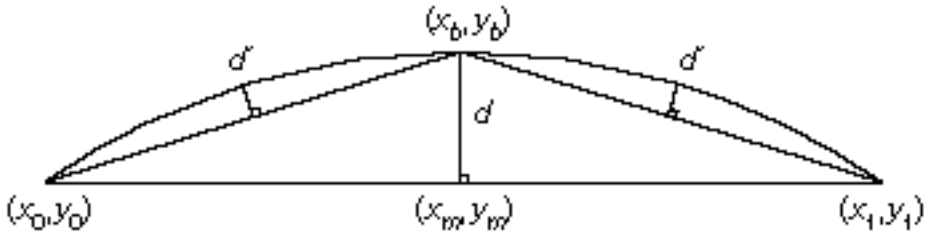
Derivation

The circular arc is represented by its endpoints and the chordal deviation:

$$(x_0, y_0, x_1, y_1, d).$$

The sign of *d* is used to distinguish between the two arcs that lie on either side of the chord that joins the endpoints (In the figure below, *d* is positive).

[1] Herman Zapf's *Optima* is such an example, resembling his well-known *Helvetica*, save for a slight curve on otherwise vertical strokes.



The bisection point of each arc is used as an endpoint for each half arc:

$$x_b = \frac{x_0 + x_1}{2} - d\frac{y_1 - y_0}{L}$$
$$y_b = \frac{y_0 + y_1}{2} + d\frac{x_1 - x_0}{L}$$

where

$$L = \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2}$$

is the length of the chord.

The chordal deviation *d´*, or *sagitta*, of the bisected arc can be approximated by:

$$d' \approx \frac{d}{4}$$

In other words, when an arc is divided into congruent “sub-arc” halves, their chordal deviation is divided by about four.

By repeatedly bisecting the arc, the chordal deviation is reduced with *quadratic* convergence. The arc is then replaced with a series of chords that approximate the sub-arcs to within a given tolerance. This approximation works well for arcs subtending less than about 75°; however, this depends upon the resolution and arc radius (estimated by *d*).

Proof

The ratio between the two chordal deviations can be expressed as:

$$\frac{d}{d'} = \frac{1 - \cos\theta}{1 - \cos\frac{\theta}{2}}$$

Where θ is the angle subtended by the smaller arc.
Substituting the double-angle formula

$$\cos\theta = 2\cos^2\frac{\theta}{2} - 1$$

yields

$$\frac{d}{d'} = \frac{1 - \left(2\cos^2\frac{\theta}{2} - 1\right)}{1 - \cos\frac{\theta}{2}}$$

$$= 2\frac{1 - \cos^2\frac{\theta}{2}}{1 - \cos\frac{\theta}{2}}$$

$$= 2\left(1 + \cos\frac{\theta}{2}\right)$$

This is an exact representation. Expanding its reciprocal in a Taylor series around zero gives

$$\frac{d'}{d} = \frac{1}{2\left(1 + \cos\frac{\theta}{2}\right)} = \frac{1}{4} + \frac{\theta^2}{64} + \frac{\theta^4}{1536} + O\left(\theta^6\right),$$

the desired result.

The error in the approximation of d' can be approximated well by retaining terms through the quadratic, $\theta^2/64$. The exact error is

$$\varepsilon = d' - \tilde{d}' = d\left(\frac{1 - \cos\frac{\theta}{2}}{1 - \cos\theta} - \frac{1}{4}\right)$$

C Implementation

The algorithm is illustrated with a floating-point implementation in C.

```
#define DMAX 0.5   /* max chordal deviation = 1/2 pixel */

#include <math.h>
#include <GraphicsGems.h>

/* Function prototype for externally defined functions */
void DrawLine(Point2 p0, Point2 p1);

void
DrawArc(Point2 p0, Point2 p1, double d)
{
    if (fabs(d) <= DMAX) {
        DrawLine(p0, p1);
    }
    else {
        Vector2 v;
        Point2  pm, pb;
```

```
            double  dSub;

            v.x = p1.x - p0.x;       /* vector from p0 to p1 */
            v.y = p1.y - p0.y;

            pm.x = p0.x + 0.5 * v.x; /* midpoint */
            pm.y = p0.y + 0.5 * v.y;

            dSub = d / 4;
            V2Scale(&v, dSub);       /* subdivided vector */

            pb.x = pm.x - v.y;       /* bisection point */
            pb.y = pm.y + v.x;

            DrawArc(p0, pb, dSub);   /* first half arc */
            DrawArc(pb, p1, dSub);   /* second half arc */
        }
    }
```

Discussion

This method quickly produces a polyline that inscribes the circular arc, and should be faster than a previous gem [Musial91], which employs a secant-based root finder and trigonometric functions to maintain arc length while splitting errors between the outside and inside of the arc.

A variant method [Paeth88] uses a non-uniform chord subdivision to locate the point where the saggita has dropped to half its height. Given a chord of interval [–1…+1] on the *x*-axis having saggita *s*, the points of half-saggita descent lie at $\pm\frac{1}{2}\sqrt{2+s}$. These converge to a constant offset for $a \approx 0$ — a parabola then approximates the circle). This formulation adds floating-point overhead to account for the non-uniform subdivision.

For IEEE-based (radix 2) floating-point representations, halving operations are almost free.

A fixed-point implementation of this uniform subdivision algorithm will be faster on many machines than a floating-point implementation. Multiplication by two and four can be accomplished by a right shift, and a fixed-point square root [Turkowski95] can be used to help rescale the vector.

Given d_{max}, the maximum chordal deviation allowed by approximating a circular arc by a series of line segments, the number of arc bisections can be computed as

$$n = \left\lceil \frac{|d_0|}{4 d_{\max}} \right\rceil$$

where d_0 is the initial chordal deviation and $\lceil \cdot \rceil$ is the *ceiling* function. This fact can be used to write a faster implementation that uses iteration instead of recursion, as advocated in another gem [Lindgren92] which also uses maximum chordal deviation as a subdivision criterion.

Bibliography

Karow87 Peter Karow, *Digital Formats for Typefaces*, 1987, URW Verlag, Hamburg, Germany, pp. 236-251.

Lindgren92 Terence Lindgren, Juan Sanchez and Jim Hall, Curve Tessellation criteria through sampling, in David Kirk, editor, *Graphics Gems III*, pages 262-265, Academic Press, Boston, 1992.

Musial91 Christopher Musial, A Good Straight-Line Approximation of a Circular Arc, In James Arvo, editor, *Graphics Gems II*, Academic Press, 1991, pp. 435-439.

Paeth88 Alan Paeth, Lemming Editor, *IRIS Software Exchange*, Summer 1988, 1(17).

Turkowski95 Ken Turkowski, Fixed Point Square Root, In Alan Paeth, editor, *Graphics Gems V*, Academic Press, 1995, pp. -.