SCAN CONVERTING SPIRALS

Francesca Taponecco
Marc Alexa

Interactive Graphics Systems Group
Department of Computer Science
Technische Universität Darmstadt
Rundeturmstr. 6, 64283 Darmstadt, Germany
{ftapone, alexa}@gris.informatik.tu-darmstadt.de

ABSTRACT

Scan-conversion of Archimedes' spiral (a straight line in polar coordinates) is investigated.
It is shown that an exact algorithm requires transcendental functions and, thus, cannot have a fast and exact integer implementation. Piecewise polynomial approximations are discussed and a simple algorithm based on piecewise circular approximation is derived. Variations of the algorithms allow to scan convert other types of spirals.

Keywords: scan conversion, polar coordinates, linear forms, piecewise approximation.

1. INTRODUCTION

Due to the success of raster displays, scan conversion algorithms are fundamental in computer graphics. Especially the scan conversion of lines has been drawing attention since the sixties [Brese65] and seems to be still an interesting topic [Steph00, Steph01].
The underlying principle of Bresenham's line drawing algorithm -- the midpoint technique -- extends to conics [Pitte67, VanAk84] and higher order polynomials using partial forward differencing (e.g. [Foley90]).

More complicated curve primitives, however, have been less considered as objects for direct scan conversion. Most of these curves could only be scan converted using general purpose graphing techniques. Graphing is still an active research field [Tuppe01] and, naturally, generality comes at the price of speed.

This work concentrates on scan converting Archimedes' spiral. In polar coordinates this spiral is a line:

$$r(\varphi) = m \cdot \varphi + b \qquad (1)$$

This spiral has proved to be an effective primitive in information visualization [Weber01]. It effectively uses the available screen real estate and allows data comparison among subsequent data elements as well as among elements with the same phase. The latter comparison highlights cyclic patterns in the data. A smooth animation through different cycle lengths facilitates the human visual system for the detection of such patterns. Yet, to display the animation, spirals have to be drawn quickly and smoothly.

In the following, some straightforward approaches for drawing spirals on raster displays are discussed and their suitability with respect to the characteristics accuracy, drift, and speed is evaluated. We conclude that among these approaches only piecewise elliptical and circular approximations deserve attention.
However, piecewise elliptical approximation is shown to have systematic derivative error and varying approximation quality depending on the spiral's slope m.

We, then, present a simple and fast approximation algorithm based on circular arcs. This algorithm is discussed in detail and its relative performance is analyzed.

The presented algorithm has several notable features.

- The arcs are quarter-circles corresponding with the coordinate system's quadrants, which avoid initialization problems for the circle scan conversion.
- It allows a pure integer implementation (for sufficient spiral parameters) so that no drift occurs.
- It has bounded error, where the error bound decreases with increasing winding number.
- It generates smooth G^1 curves and the approximation converges against the exact spiral.

The algorithm might be altered to generate spirals other than Archimedes'.

2. DIRECT SCAN CONVERSION

A general approach for scan converting curves is the midpoint algorithm [Pitte67, VanAk84]. The idea is to represent the curve in an implicit form as

$$F(x,y)=0 \qquad (2)$$

so that F could be used to decide whether a point is above or below the line. F is applied to midpoints between two candidate pixels and depending on the below/above information one of the candidate pixels is chosen.

However, the curve has to be split into regions, which meet at points where the partial derivatives of F are equal. In these regions the x or y component is increased by one, while the other component is increased (or not increased) depending on the below/above information (*decision variable*) e.g. $F(x+1, y+0.5)=0$.
During the scanning, function evaluation is avoided by using finite partial differences. These finite partial differences are typically also used to find the end respectively start points of the regions. This approach is now applied to the spiral.

Inserting (1) in the distance equation $x^2+y^2=r^2$ and considering that φ is expressed as $\varphi=\arctan(y/x)$, the substitution leads to the following expression in Cartesian coordinates for a linear spiral:

$$F(x,y)=x^2+y^2-(m\cdot\arctan(y/x)+b)^2 \qquad (3)$$

The necessary partial differences are

$$\left(\frac{\partial F(x,y)}{\partial x},\frac{\partial F(x,y)}{\partial y}\right)=\left(2x+\frac{y}{x^2}\cdot\right.$$
$$\cdot\left(\frac{2m\cdot b}{1+(y/x)^2}+\frac{2m^2\cdot\arctan(y/x)}{1+(y/x)^2}\right),$$
$$\left.2y-\frac{1}{x}\cdot\left(\frac{2m\cdot b}{1+(y/x)^2}+\frac{2m^2\cdot\arctan(y/x)}{1+(y/x)^2}\right)\right)$$

(4)

Note that the partial differences (whether finite or not) contain the *arctan* function. This has two implications:

- The computation is not significantly simplified by a forward differencing approach as the evaluation of the *arctan* still dominates the computation.
- Error cannot be avoided, as the *arctan* function is transcendental for some arguments. Thus, the algorithm is almost prone to drift.

While a certain drift is unavoidable, the trigonometric function could be replaced with polynomial series expansion. This polynomial would allow an effective implementation of the forward differencing approach.

The inverse tangent function can be expanded to the following power series:

$$\arctan(s)\cong s-\frac{1}{3}s^3+\frac{1}{5}s^5-\frac{1}{7}s^7+\ldots \qquad (|s|<1)$$

$$\arctan(s)\cong\frac{\pi}{2}-\left[\frac{1}{s}-\frac{1}{3s^3}+\frac{1}{5s}-\frac{1}{7s^7}+\ldots\right] \qquad (s>1)$$

This expansion needs to be inserted in (3) and (4) to yield polynomial expressions for *F*. By varying the number of terms, this expansion allows to trade accuracy for complexity. We have analyzed this approach experimentally.

First, the implementation of forward differencing for high degree polynomials is cumbersome and error-prone. More importantly, even when using many terms, the error accumulates quickly so that the overall scan conversion fails. Incremental algorithms seem to strongly depend on near-exact arithmetic.

Because of these reasons we believe there is no suitable incremental algorithm for scan converting spirals.

3. PIECEWISE INTERPOLATION

A piecewise approximation is usually built by exactly computing several points of the curve and then interpolating these points with primitives that are easy to scan-convert. The simplest primitive for interpolation is a line segment; more complex alternatives are circular arcs, then elliptical arcs, general conics, and so on.

This method requires to compute some points on the spiral. The coordinates of these points are given by the following form

$$(x,y) = ((m\varphi + b)\cdot\cos(\varphi), (m\varphi + b)\cdot\sin(\varphi)) \qquad (5)$$

Some of these points are easier to compute than others. Specifically, the points on the coordinate axes

$$(x,y) = ((2km\pi + b), 0)\,,$$
$$(x,y) = (0, ((2k+1)m\pi + b)) \qquad (6)$$

are the only points which are guaranteed to be integer given that $m\pi + b$ is integer. Furthermore, they allow to avoid the evaluation of trigonometric functions. This evaluation might be avoided for many angles by using tables. However, this necessarily limits the number of points to a fixed set of angles modulo 4π.
We have found that piecewise linear as well as piecewise circular interpolation is difficult using such points.

- The approximation quality of linear pieces depends on the length of the line segments. Using a fixed set of angles, the length of a line segment (and, thus, the error) is unbounded.

- Circular arcs could not have the origin as their center and it is not clear what other point could be used as center. Note, however, that using other centers and interpolating other points on the spiral leads to the algorithm proposed in this work.

While linear or circular pieces (used in a naive way) are inadequate, elliptical arcs are an interesting alternative. The following algorithm provides such an approximation:
Define a quarter-ellipse having the value of $d = m\cdot\pi/2$ as difference between the major and minor semi-axis and draw it from $(A_0 + b, 0)$ to $(0, A_0 + d + b)$. Subsequent elliptical quarters increase every semi-axis' value by d and the center remains constant in the origin. Thus, all elliptical arcs are joined yielding a continuous increment of the radius in the resulting spiral (see Figure 1).

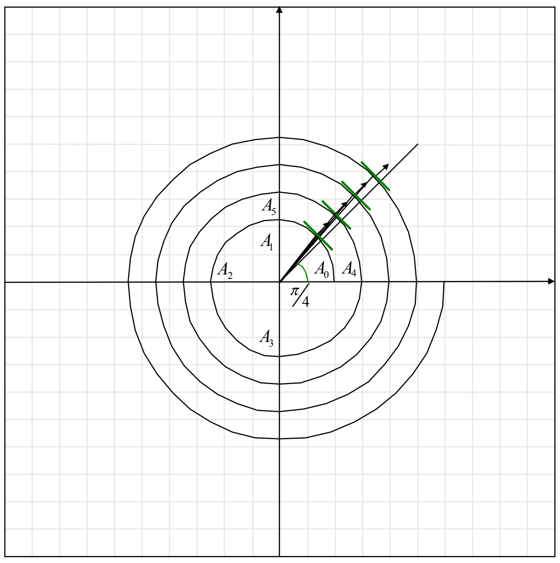

Figure 1: Quarter ellipse spiral, iterative construction.

This leads to two important consequences:

- In each of the four quadrants quarter-ellipses have a constant distance of $4d$ between each period, as in Archimedes' spiral.

- A general major semi-axis has the value of $A_n = A_{n-1} + d$; therefore, elliptical arcs meet with the same semi-axis length and the resulting curve is C^1.

The implementation of the algorithm is quite simple given a method to scan convert axis aligned quarter-ellipses.

Nevertheless this approximation has two problems. First, approximation quality strongly depends on the spirals eccentricity m and, second, the approximation does not converge against Archimedes' spiral.
Especially the first quarter of the approximation shows most notably the difference between the two semi-axes. If m is large, the difference between a spiral quarter and an ellipse is significant.

As the approximation is tangent continuous, the points of the spiral that cross the axes have tangents that result to be perpendicular to the axes. This is not the case for a linear spiral. Note that these horizontal and vertical tangents are an invariant for the elliptical approximation, thus it cannot converge

against the desired values. This problem cannot be avoided, as all elliptical approximations with the origin as their center would have wrong tangents.
On the other hand it is interesting to note that, varying the d parameter in this algorithm, it could be possible to obtain C^1 approximations for spirals of the Archimedean family.

4. QUARTER CIRCLE ALGORITHM

Here we present a fairly simple algorithm with good accuracy and bounded error. The algorithm is inspired by ancient techniques to draw approximations to spirals [Stew95]. These techniques due to Fibonacci and Padovan assemble a set of squares or triangles and use circular arcs (see Figure 2). Yet, these algorithms provide approximations to spirals with growing eccentricity (i.e. exponential spirals).

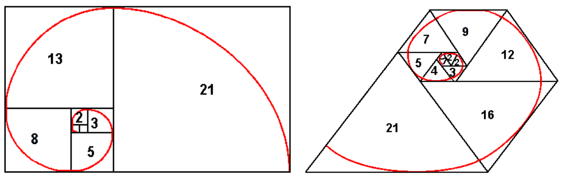

Figure 2: Fibonacci and Padovan spirals.

Let's take a closer look at Fibonacci's algorithm. Obviously, the growth the base lengths the squares controls the growth of the spiral's radius. If the base length of the squares would grow linearly, an approximation to Archimedes' spiral is found. The following simple algorithm exploits this idea.

Consider again the expression of an Archimedean spiral

$$r(\varphi) = m \cdot \varphi + b$$

and let $d = m \cdot \pi/4$ define four points $c_i = (\pm d, \pm d)$ in the four quadrants of the coordinate system (see Figure 3).
Ignoring for now the first segment of the spiral, a circular arc with center $c_1 = (-d, d)$ and radius $A_1 = 2d + b$ is drawn from $(d + b, d)$ to $(-d, 3d + b)$.

Figure 3 explains how the center points c_i are chosen in turn, and quarter circles are drawn starting from the last point of the previous circular arc.
Note that radius increases by $2d$ in each step, consequently the expression for the quarter circles radii is:

$$A_{n+1} = A_n + 2d = A_1 + 2d \cdot n \qquad (7)$$

with $A_1 = 2d + b$ and $n \in N^+$.

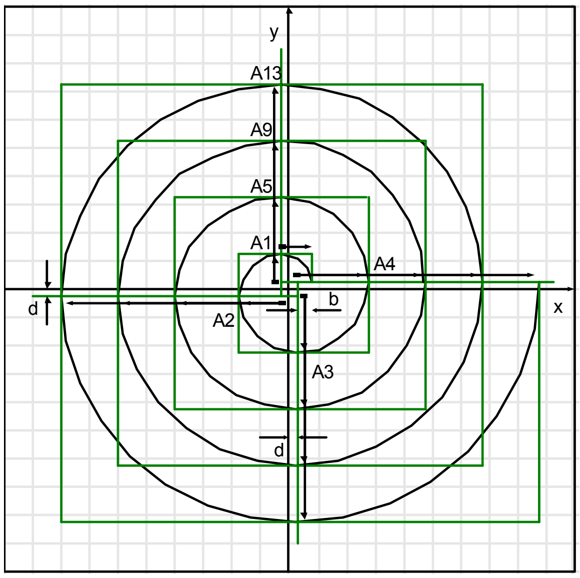

Figure 3: Illustration of the piecewise circular approximation to a spiral.

Considering the circles quarters in a quadrant the radii are

$$A_{q+4i} = A_q + 8i \cdot d \qquad (8)$$

where $i \in N$ is the periods number and $q \in [1,2,3,4]$ gives the first circle radius value for each of the four quadrants. In the first quadrant we have $q = 1$, so that the sequence of the concentric circles′ quarters radii results to be $A_1, A_5, A_9, A_{13}, \ldots$

If d and b are chosen to be integer, the algorithm can be implemented using only integer addition. The circular arcs require (after set up) only two additions, and the step from each circular arc to the next requires another two additions.

In order to better exemplify the succession of the quarter radii and the position of the related centers points, we propose the following tables, where the steps are analyzed for each of the four quadrants and iteratively for every period, considering the parameter i.

Step	Quarter circles radii	Quarters center points
(1+4i)°	2*d+b +8i*d	(-d,d)
(2+4i)°	4*d+b +8i*d	(-d,-d)
(3+4i)°	6*d+b +8i*d	(d,-d)
(4+4i)°	8*d+b +8i*d	(d,d)

Step	Quarter starting point	Quarter end point
(1+4i)°	(d+b+8i*d,d)	(-d,3*d+b+8i*d)
(2+4i)°	(-d,3*d+b+8i*d)	(-5*d-b-8i*d,-d)
(3+4i)°	(-5*d-b-8i*d,-d)	(d,-7*d-b-8i*d)
(4+4i)°	(d,-7*d-b-8i*d)	(9*d+b+8i*d,d)

Figure 4: Initializing table for the algorithm's step.

It is easy to see, see Fig. 4, how (according to the belonging quadrant) the quarter starting (end) points always have a fixed y (x) coordinate and an x (y) coordinate given by the radii values, opportunely translated by the quantity –d (+d).
Moreover, each quarter starting point is automatically given by the quarter end point of the previous step.

We previously reserved to describe the behavior of the first spiral segment. It is known, that the center of a spiral represents a particular case of singularity: i.e. it is a point about which the curve twists infinitely and at which the function is not defined. Also in our algorithm, this point leads to an exception, as it is the only arc whose center does not belong to the defined c_i, but it results to be placed in the point (b,d) (see Fig. 5).
We also investigated ways to achieve better performances by varying the length and the center position of this first circular arc, anyway the explained case (arc length $A_0 = d + b$, center position in $c_0 = (b,d)$) results to produce an easier implementation, and consequently we calculate the error in this case as an upper bound.
The argument for this algorithm with respect to approximation quality is this: The distance of two circular arcs with the same center is $8d = 2\pi \cdot m$ as required. However, this distance is not measured with respect to the origin, as it should be, but with respect to the circles´ centers in every quadrant.

Yet, the absolute approximation error depends on the ratio of d and r, which means it diminishes for large r. Furthermore, circular segments meet with horizontal or vertical tangent. Thus, the resulting approximation is G^1. Note that it is not C^1 as the meeting circular arcs have different radii.

As explained, depending on the translation between the two origins, the relative distances of the origins to the function's points are different.
Referring to the first quadrant (see Figure 5), a general radius starting from the system coordinates origin is given by:

$$R_n = R_{q+4i} = \sqrt{(A_{q+4(i-1)} + 8d - d)^2 + d^2} \qquad (9)$$

and a generic radius starting from the circles´ center is given by (8):

$$A_n = A_{q+4i} = A_{q+4(i-1)} + 8d$$

Let's consider again Figure 5: using Pitagora´s theorem it is possible to see how, increasing the rotating angle θ, the difference between the radii R_n and A_n progressively decreases, as the angles α given by the defined radii also decrease.
As a result, the approximation error is maximum close to the origin and progressively decreases when we increase the angle θ in the following quadrants and thus with increasing periods number.

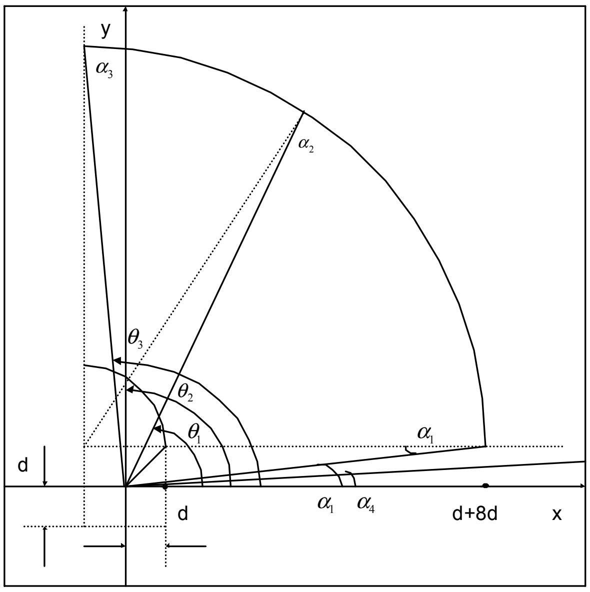

Figure 5: Error bound's calculation.

To give an error bound on the approximation depending on the radius we investigate the behavior at points on the positive x-axis. The desired radius is $r_i = m \cdot 2\pi \cdot i = 8i \cdot d$, while the approximation leads to

$$r'_i = \sqrt{(8i \cdot d)^2 - d^2} \qquad (11)$$

This leads to a relative error of

$$e(i) = 1 - \sqrt{1 - \frac{1}{64i^2}} \qquad (12)$$

which quickly vanishes with increasing cycle number i.

5. VARIATIONS & FUTURE WORK

There is an intricate relationship between the spirals' eccentricity and the center points c_i. If the distance between these points is constant the growth of the radius is linear. However, altering the distance of the center points should allow to change the growth of radius.

It seems that, instead of keeping the arcs´ centers fixed for every quadrant with a relative distance of $2d$, if the distance between these points is constantly increased this would result an exponential spiral. If it is decreased it results in a logarithmic spiral. However, this has to be investigated more thoroughly and we regard it as future work.

In general, the variation of the center points allows the scan converting of a variety of different spiral curves. All of these approximations are simple to implement and generate G^1 approximations.

6. CONCLUSIONS

We have investigated several methods to render Archimedes' spiral. Two algorithms have been derived and explained, the approximation error has been analyzed and the results are quite satisfactory.

The quarter ellipse algorithm produces a C^1 curve, but the resulting approximation has systematic error.

The quarter circle algorithm is simple in implementation and execution; the approximation error depends on a translation offset and, thus, is only relative and diminishes with increasing winding number. The resulting curve is G^1.

Both algorithms allow the use of integer arithmetic; consequently they offer great simplicity and fast implementation. They take advantage of symmetry characteristics and avoid calculations´ redundancy and trigonometric complexity.

REFERENCES

[BELS76] K. Belser. Comment on "An improved algorithm for the generation of nonparametric curves". IEEE Trans. on Comput. Vol. C-25, 103, 1976.

[BRESE65] J. E. Bresenham. Algorithm for computer control of a digital plotter. IBM Systems Journal, 4(1): 25-30, 1965.

[FOLEY90] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. Computer graphics, principles and practice, second edition. 1990. Reading, Massachusetts.

[JORD73] B. W. Jordon, W. J. Lennon, B. D. Holm. "An improved algorithm for the generation of nonparametric curves". IEEE Trans. On Comput. Vol. C-22, 1052-1060, 1973.

[PITTE67] M. L. V. Pitteway. Algorith for drawing ellipses or hyperbolae with a digital plotter. The Computer Journal, 10(3): 282-289, November 1967.

[RAMO76] J. Ramot. "Non parametric curves". IEEE Trans. On Comput. Vol. C-25, 103-104, 1976.

[STEPH00] P. Stephenson and B. Litow. Why step when you can run? iterative line digitization algorithms based on hierarchies of runs. IEEE Computer Graphics & Applications, 20(6): 76-84, November / December 2000. ISSN 0272-1716.

[STEPH01] Peter Stephenson and Bruce Litow. Running the line: Line drawing using runs and runs of runs. Computers & Graphics, 25(4): 681-690, August 2001. ISSN 0097-8493.

[STEW95] JI. Stewart, "Nature's numbers", Basic Books, 1995.

[TUPPE01] JeFF Tupper. Reliable two-dimensional graphing methods for mathematical formulae with two free variables. Proceedings of SIGGRAPH 2001, pages 77-86, August 2001. ISBN 1-58113-292-1.

[VANAK84] J. R. Van Aken. An efficient ellipse-drawing algorithm. IEEE Computer Graphics & Applications, 4(9): 24-35, September 1984.

[VANAK85] J. R. Van Aken, M. Novak. "Curve drawingalgorithms for raster displays", ACM Trans. On Graphics, 4, 147-169, 1985.

[WEBER01] Marc Weber, Marc Alexa and Wolfgang Müller. Visualizing time-series on spirals. Accepted for publication in Proceedings of InfoVis 2001, 2001.