Welcome to the 13th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision 2005!

M L V Pitteway
Brunel University,
Uxbridge, UB8 3PH,
United Kingdom

1. INTRODUCTION

Jack Bresenham, in his introduction to this conference two years ago, presented an eloquent case for the software engineering aspects of our discipline, citing Douglas McIllroy's meticulous scholarship in his Bell-Labs technical 'Trilogy on Raster Ellipses and programming Methodology':

1. 'Getting Raster Ellipses Right'
2. 'Math before Code: A soundly Derived Ellipse-drawing Algorithm' and
3. 'Ellipses Not Yet Made Easy'.

"Beware of programs with imprecise specifications". An engineer expects to first set up a precise definition of his intentions, in this case using Freeman chain coding to specify the correct sequence of incremental moves that defines the intended "best" incremental approximation to an ellipse, before setting up an algorithm and resorting to computer code.

That may be good enough for an engineer, but science doesn't work like that: Computing is, I believe, an experimental science, and it is, in my view, perfectly legitimate to scribble out a programming hypothesis for subsequent check and validation on a computer, and to worry about the mathematics and meaning involved after the experimental validation. Like Jack Bresenham, I came into computing many years ago. I wrote my first computer program (for the pilot assembly EDSAC) some 50 years ago, so I hope my audience today will forgive me if I, too, indulge in some reflection on the past. We've certainly come a long way since those early days. I entered the field, however, with a scientific background, and although science seems to have become less popular in recent years than once it was, I still admit to be, by profession, a computer "scientist", and I feel that the "science" is still a paradigm that has much to offer to our discipline.

WSCG'2005 Conference proceedings ISBN 80-903100-7-9 January 31- February 4, 2005, Plzen, Czech Republic.

The beauty and elegance of the equations of Physics: Newton's "force is equal to the rate of change of momentum", special and general relativity (including the now infamous $e = m c^2$), Maxwell's and Schroedinger's equations, Dirac's model of the Hydrogen atom and many other classic examples. Engineers, too, share in this search for simplicity - the design of a Brunel bridge, for example, or the architecture of Sir Christopher Wren.

Modern software, by contrast, is plagued by spaghetti codes that need frequent patching to correct various glitches or to avoid attacks by the schoolboy authors of worms and viruses. Could it be that the reason source code is so often protected is not so much because of its commercial value but to protect its authors from the lampooning that they would be exposed to by competent computer scientists if it were ever to be published - the "if your bridge looks weak, nail on another bit of wood" philosophy?

In the early days the limited hardware, with random access storage often measured in words, rather than kilo, mega or giga bytes, imposed a discipline of its own, and having to sit up late at night to complete a single sample run certainly encouraged the writing of good, efficient code!

I was inspired when I read Jack Bresenham's article in the IBM Systems Journal of 1965. At the time it was to me the most original algorithm since Euclid's (to which we now know it is closely related, e.g. Clive Castle's 'algorithms for the even distribution of entities', Computer Journal 1986). Bresenham's algorithm draws the "best" approximation to a straight line with a gradient defined by two constants "b" and "a", and involves only one addition and one test for each output move generated. This led me to wonder if I could generate useful curves efficiently if I added extra instructions each side of the loop, and the obvious thing to try seemed to me the addition of instructions like "b - K1→b, a + K2→a" somewhere in the left hand branch of the loop and similarly "b - K4→b, a + K3→a" in the right. (The signs were chosen as

I was thinking of making a get smaller and/or b larger to generate my curve.)

At the time I had in mind something like a slowly varying approximation, as I realized that I could resort to the use of calculus if my four curvature constants K1 thru' K4 were small enough. But then, when it turned out that, with K2 set equal to K4, the algorithm generates the "best" approximation to the arc of a conic section (though I had to generalize the interpretation of "best" from Bresenham's original definition) I felt that I might have discovered something very interesting indeed. (With K2 not equal to K4 the algorithm is now known to generate a curve which spirals in or out logarithmically, though it took me many years to make that discovery, and it involves a further relaxation of what we must accept as a legitimate meaning for "best".) It ran like a bat out of hell, and could drive an incremental graph plotter flat out even with the primitive computer we then had installed at Brunel. The first operator to try it actually switched the machine off, as she was convinced something must have gone wrong; she was used to seeing just three of four increments per second.

One particular feature of my algorithm excited the scientist in me: It could follow any conic section, a hyperbola as well as an ellipse, and it had no restrictions on the orientation of the axes of the figure. I can't imagine any worthwhile scientific theory that relies on the choice of axial orientation to be meaningful.

There are many problems remaining, however. As Jack put it in his introduction to this conference two years ago: "Ellipses are a shape often done with degenerate instances unaccounted for; that is, they fail in certain instances. Comprehensive testing and a thorough understanding of an algorithm's minutia is always essential." Vaughan Pratt offered a fix in 1985 (and also solved a problem of all integer working that I had given up on), but this involves extra work in the loop which is a nuisance and seldom required in normal use. Also we need to monitor the sign of b and a, with remedial work required if either of these two gradient defining variables become negative, if we are to extend the algorithm sensibly to draw a complete ellipse, for example.

Many talented computer science researchers have worked to try for an elegant and simple solution to at least some of these problems, but I am now driven to wonder if Douglas McIlroy is right in saying that "there is no Royal Road to programs". How very disappointing if this is so! However, our field is young and, like Jack I would like "to encourage anyone not to be discouraged by earlier problem solving attempts that may have been less than successful; keep trying and likely it will ultimately be successful".

Following the thinking of C.P. Snow, I have sometimes argued that the information revolution deserves to be accorded the status of a "third culture". But for this to be achieved there needs to be an algorithmic equivalent of general relativity, and which Einstein described as being "so elegant and beautiful that it must be right", or a Maxwell's or Schroedinger's algorithm, or a beautiful design in software engineering to match the best works of Sir Christopher Wren or Isambard Kingdom Brunel. Maybe some such thing will be presented to us at this conference!

2. REFERENCES

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

[2] Freeman,H.: On the encoding of arbitrary geometric configurations, IRE Trans.EC-102, pp.260-268, 1961.

[3] Pitteway,M.L.V.: Algorithm for drawing ellipses or hyperbolae with a digital plotter, Computer Journal 10, pp.282-289, 1967

[4] Pratt,V.: Techniques for Conic Splines, Computer Graphics (SIGGRAPH) 19, pp.151-159, 1985.

[5] Bresenham, J.E.,Earnshaw,R.A., Pitteway,M.L.V.: Fundamental algorithms for computer graphics, Springer-Verlag, 1985.

[6] Bresenham, J.E., Earnshaw,R.A., Forrest,A.R., Lansdown,R.J., Pitteway,M.L.V.: Theoretical foundations of computer graphics and CAD, Springer-Verlag, 1988.

[7] Castle,C.M.A., Pitteway,M.L.V.: Algorithms for the even distribution of entities, Computer Journal 29, pp.574, 1986.

[8] McIlroy,M.D.: There is no royal road to programs: a trilogy on raster ellipses and programming methodology, Computer Science TR155, AT&T Bell Laboratories, 1990. (see the internet: http://cm.bell-labs.com/cm/cs/estr.html.)

[9] Golipour,M.K.: A new general incremental algorithm for conic sections, 2004.

Choose the appropriate octant and set the initial values for b, a, d, K1, K2, K3 and K4. Then

For $d < 0$:	For $d \geq 0$:
square step	diagonal step
$b - K1 \rightarrow b$	$b - K4 \rightarrow b$
$a + K2 \rightarrow a$	$a + K3 \rightarrow a$
$d + b \rightarrow d$	$d - a \rightarrow d$

If b or a < 0 change octant

Repeat until done

Bresenham's algorithm with curvature parameters.