discussion

Drawing the line

JOHN LANSDOWN *talks to* JACK E. BRESENHAM

Although they might not be aware of it, virtually everyone who uses an incremental plotter or a raster screen for making drawings uses an algorithm which was derived in the early 1960's by a young American engineer and computer scientist, Jack E. Bresenham. His procedure gives the best approximation to a straight line that you can have on devices which only draw in a limited number of directions (like plotters), or with spots of light, (like raster displays). It does the approximation with minimal operations – only addition, subtraction and sign testing are used – which makes it very fast and easy to incorporate in hardware. Last year Dr Bresenham was in the UK for a conference on '*Fundamental algorithms of computer graphics*' and John Lansdown took the opportunity of discussing with him the invention of what must be the most fundamental of all computer graphics algorithms.

JL *Jack, it's now more than twenty years since the first publication of the line drawing algorithm that bears your name. Did you have any feeling then that you were inventing something which would have such a lasting influence on the way we do computer graphics?*

JEB No, John, that all came as quite a surprise to me, albeit a pleasant one – but years later. At the time I simply had a problem to solve. I was working at IBM in San Jose, California in about 1961. We had a small incremental plotter which one of the engineers, Kemp Allen, had attached to the 1407 console interface of the 1401 system in such a way that the computer thought it was a typewriter. I had just transferred to the Computation Laboratory where Dr Gene Lindstrom, an excellent computer scientist, was in charge. He was the sort of manager who gave his people freedom to do things and our job was to support the other engineers in all sorts of ways. Shortly after I came in, Gene suggested we look at the machine code to drive the plotter. We hoped that the engineers would then be able to use the plotter to draw curves and not have to look through so much raw data. I worked under Al Mitchell who had the plotter as part of his responsibilities and he gave the coding job to me. My colleague, Dave Clark, to whom I owe a great deal, got me started on the practical aspects of the problem.

The 1401 computer was slow compared to the larger IBM machines of those days and very slow in comparison to pretty much everything these days and, it soon became clear that, if we had to do multiplication or division in the line drawing, we couldn't run the plotter anywhere close to its rated speed. As I analysed the processes that had to be carried out I tried making use of information gleaned from three courses I'd recently taken at Stanford University under Dr Gerry Lieberman, although I'm not sure he'd recognise the connection between what I did and what he taught me.

JL *What were these courses?*

JEB Applications of switching theory, engineering statistics and statistical inference. In switching theory, we learned about Boolean logic; in engineering statistics, we learned how to fit a regression line to scattered data; and in statistical inference, we learned the techniques of taking integrals of absolute values using sign testing. I thought that what I was trying to do was something like the reverse of taking a regression line.

JL *Instead of fitting a straight line to data – finding out which data were needed to match a line?*

JEB Yes, and then working out an error measure to test how close I was to the line I should really be drawing. I set about doing this but, in the first cut, I got it wrong. What I did was to track the line as if I were steering a ship; whenever I got off course, I'd make a correction to pull the ship back to the line. This didn't work out at all well. I was also having troubles with accumulated round-off errors and I soon realsised that I couldn't use fractions and would have to resort to sign testing for speed. After this first false start, the method developed quickly and soon worked in the way that we wanted. Of course, in writing up the work, I discovered further simplifications which should have been obvious from the start. The mere fact of describing the work to others and documenting it to the standards that Gene and Al had laid down showed me ways of simplifying and refining. But the need to obviate multiply and divide operations really was at the heart of the operation. Necessity here truly was the mother of invention.

Another thing that helped was the visit to San Jose of Ken Iverson who came out to give a talk on his programming language APL. At that time the language had not been implemented on a computer but was simply a notational method. I remember during his lecture seeing him put up his 'floor' and 'ceiling' operators and realising that these were exactly what I needed to simplify my method.

JL *So this is another example, and there have been many in the history of ideas,*

where a good notation spurs you on and is almost the key to solving the problem.

JEB I really believe that very strongly. Good notation often enables you to express a problem tersely enough to contain a larger view of it and to manipulate it consistently. A proper notation is essential.

JL *Were you under constraints to get everything working in a tight time-scale? Was someone saying, 'I want this by next Thursday'?*

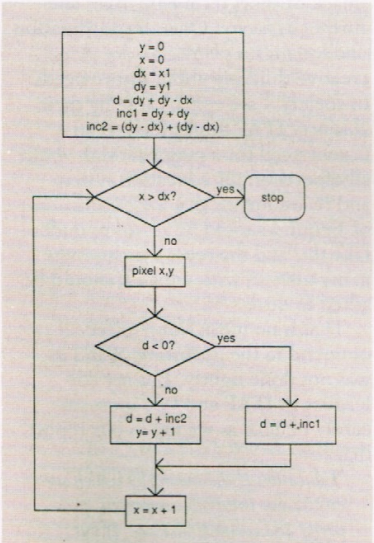
JEB It wasn't quite that sort of time constraint. There were time constraints because we wanted to get the plotter into use; but the atmosphere that Gene and Al fostered was to be responsive to needs of time but to do things right and to do things properly.

JL *And when you'd done it, what did you think?*

JEB I simply thought that this was an interesting application and, as I'd only been out of university for two years, I was glad to have something useful to publish. One of the things that Stanford teaches you is to publish in order to reinforce the fact that one always builds on the work of others. If your predecessors didn't publish you couldn't do that. IBM encourages publication for the same reason.

JL *At that time your description of the algorithm dealt with plotter drawings because it was written before we had any raster devices. When did you realise that your method would be appropriate to raster devices too?*

JEB Well, I'd stopped being active in graphics from the Fall of 1962. It was in the mid 1970's that I happened to see reference to my algorithm in a paper on raster graphics. As similar needs are present in both plotter and raster work, it seemed a fairly natural development but I'd never thought of my work being used at all let alone on raster graphics.



Bresenham's Algorithm: for line from (0,0) to (x1,y1) with x1 > y1 > 0

JL *But aren't you surprised that, although there have been some minor modifications by both yourself and others over the years, your early work still remains the basis for almost all our straight and curved line drawing methods?*

JEB I am surprised that people continue to reference my paper directly. Usually in computing, as you know, these things get lost in antiquity and who developed what tends to be forgotten. But, as long as one is trying to address memories bit by bit and do drawing incrementally, then there seems to be little else to do than to follow the general principle I adopted for the plotter. The need simply to do a sign test and a single add makes for a fairly tight loop and, whilst the minor changes that have been introduced are interesting, there seems to be very little scope for making major alterations. Of course, for a real implementation where you might need things like thick lines or exclusive-or, you have to make alterations to suit the context of the problem. So I am not surprised that the method is still used – only that references are still made to the original.

JL *Having devised the most fundamental algorithm in computer graphics you then left the field altogether?*

JEB I don't know about its being fundamental, John, but except as a hobby, I went on to other things.

JL *But as we have seen at this meeting, there are some new and interesting things to discover and say about drawing lines with computers. Are these developments a revelation to you or did you always know that there was more to the problem than, as it were, meets the eye?*

JEB Some of the things we've seen here have been a revelation to me – although, of course, people have been keeping me in touch with their ideas as they develop. Engineer Bron's image processing approach to establishing the patterns in the lines interested me greatly. His top-down, macroscopic method of revealing the patterns was fascinating. The work of Mike Pitteway and Clive Castle is also far-reaching and helps us all to understand the structure of the different patterns that emerge.

JL *What you didn't tell us is whether or not the engineers in San Jose actually used the plotter to do their drawings after you'd worked out the method to help them do so.*

JEB Initially there was some reluctance because they thought that the curves wouldn't be drawn smoothly enough. So one Saturday, using grid paper, I digitised the Playmate of the Month from Playboy magazine and put this into the machine as one of our 'test figures'. When the engineers saw the results they were convinced that acceptable curves could be drawn – of course I made the figure quite small so that the sampling points seemed very close together. But after that there were no problems.

Fortunately, too, some of them were engaged on cement kiln simulations where vast quantities of numbers were being produced from the battery of simultaneous differential equations that they were using. They were having a hard time searching through these for troughs and peaks of performance. Plotting the output got over the difficulty.

What also helped is that, for printing 7094 output tapes, we were using on the 1401 what turned out later to be called 'multiprocessing' and 'multiprogramming'. On the 1401 we had a method of outputting a single character to, and then releasing, the typewriter port – which, in our case, was the plotter. We were also using the 1401 to output the numerical results of the programs on to a 1403 line printer and we could test for 'printer busy' and 'not busy', release the printer and then execute code. So we put in a test loop to give priority to the 1403. Then, as soon as we had given a 1403 command in the print program, we would output a character to the plotter and return to check if the printer was ready for more results. Thus the plotter output was essentially free – although the San Jose IBM engineers weren't charged quite on that basis!

JL *Your presence as one of the organisers of this conference, as well as your 1982 paper in The Computer Journal, shows that you are still extremely interested in algorithms for computer graphics – but I understand that your current employment is not in this area.*

JEB Right now I'm working on the Display Products Business Unit HQ staff in the IBM Communications Products Division looking at our strategy and products in the general business line – as well as dealing with issues that come up in some of our laboratories. My periods of active graphics work were the 6–9 months in 1962 and the three-and-a-half years when I was fortunate enough to work for Mike Davis at IBM Hursley. The Hursley time was the highlight of my career; I just enjoyed every minute of

it and learned something new each day from the remarkable group of engineers he has there taking algorithms and putting them in hardware, software, microcode and so on.

JL *When was this?*

JEB I was there from March 1981 to August 1984. The last product I worked on was the IBM PC 3270 GX 1000 x 1000 colour graphics display.

JL *If, rather than a software method, you had invented and patented a piece of electronic hardware that fitted inside every computer graphics display and plotter, you would now be a very rich man. Do you ever think of that?*

JEB I hadn't thought of that. In those days, software wasn't looked at for patents – but IBM has recently been taking a retrospective look at some of their activities in graphics and, at the end of 1984, gave me an Outstanding Innovation Award for my 1962 work.

JL *Has the last word yet been said about drawing lines by computers?*

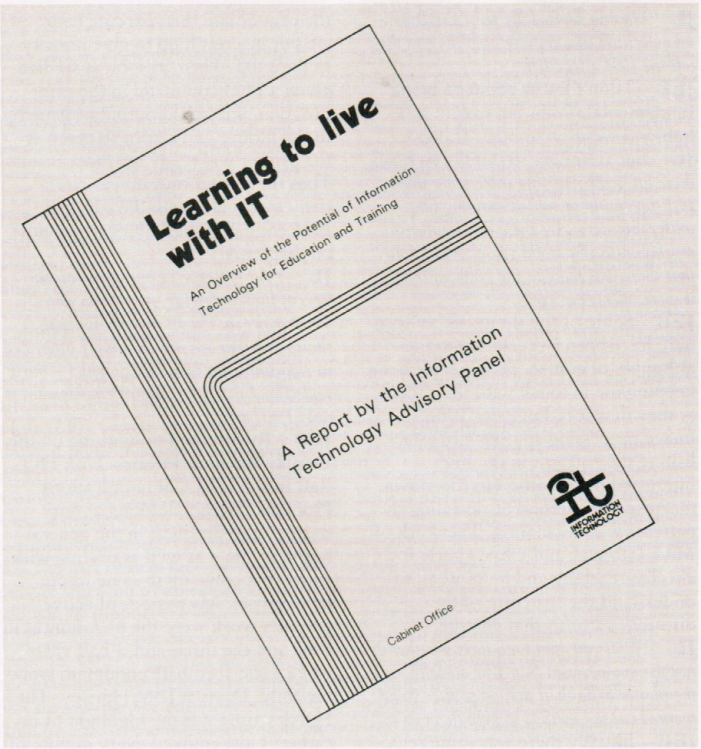
JEB I should think we'll always go on progressing with new and better techniques. One thing that always happens in engineering is that new methods and technologies arise and others become obsolete. As the technology changes, people will need to invent other things. As an example, in the mid-1970s, I was working on the HQ staff in White Plains and a friend of mine in Kingston told me about the gas plasma panel on which IBM was working – now, of course, a fully developed product. In this, they could address a whole scan line of pels [IBM for 'pixels'] in the same time as they could address a single one, because they were using shift registers. Thinking about this, I came up with a run-length algorithm to calculate all the pels in one slice simultaneously, thus taking advantage of the feature. (Unknown to me, Reggiori at New York University, had actually worked out an essentially similar algorithm some time before this.)

I think that the best algorithms that are developed are the ones that are matched to the technology. We'll see modifications for such things as anti-aliasing and for new technologies such as laser printers, raster and ink jet plotters. The video RAM, too, will need algorithms which take advantage of its ability to bypass bottlenecks in a single buffered display – by addressing the bit planes in the write cycle rather than in the refresh period. New algorithms will have to be invented to take full advantage of any new technology. There's plenty to be done yet.

Reference

The proceedings of the conference, in which some of the work referred to is included, are edited by Rae Earnshaw and published by Springer Verlag under the title *Fundamental algorithms for computer graphics* (1985).

IT and education *continued*



academic disciplines are becoming increasingly inappropriate as new combinations of knowledge are required in our developing society.

There are, of course, many specific ideas and initiatives in education currently being pursued. Most are commendable in themselves but they all smack of short term palliatives and further *ad hocery*. Overall coordination and long term planning – let alone creative thinking about future needs of society – are remarkable by their absence. ITAP thinks there is an urgent need for a major rethink about all aspects of our education system and recommends that a Commission of Enquiry should be set up to undertake this and the report outlines the many matters with which it should be asked to deal.

That is no mean thing to recommend to the government and it was not done lightly. But my colleagues in ITAP and I do most sincerely believe as we say in our report that

'Education is the means by which we ensure our future development, prosperity and cultural identity. If our education system decays our nation will decay with it.'

Figure 1Computer Bulletin by The British Computer Society volume 2 Part 4 December 1986

A special issue celebrating Stanford's Centennial p.158-159 by William Rodarmor "Throwing the Computer a Curve"

WIZARDS

With the help of a centerfold, Jack Bresenham showed that computers can draw curved lines.

Throwing the COMPUTER a Curve

by William Rodarmor

THE UNIVERSE IS FULL OF MAGICAL THINGS PATIENTLY WAITING FOR OUR WITS TO GROW SHARPER.

Eden Philpotts

Drive a car and you may not know who to thank for inventing the wheel. But draw a graph on a Macintosh, or play a game on a PC, and you owe a prayer of thanks to retired IBM engineer Jack E. Bresenham, MS '60, PhD '64. Almost 30 years ago, he developed Bresenham's Algorithm, the fundamental technique which enables you to quickly and easily draw lines and curves on a computer graphics screen. A modest man, Bresenham claims his algorithm is simple. But because of its elegance, his method hasn't been superseded; it's still widely used in computer graphics. In recognition of his work, IBM gave Bresenham a $25,000 award for innovation in 1984. The company gave him another $35,000 corporate award in 1989–two years after he had retired from a 27-year career with Big Blue.

Bresenham was raised in Clovis, N.M., and is an easygoing, down-home kind of guy. His colleague Karl Guttag, a graphics strategy manager at Texas Instruments in Houston, still relishes the memory of the day he dragged a sales trainee up to Bresenham; Guttag made the young victim explain line-drawing to the grand old man of computer graphics. Bresenham was as attentive then as he is to the students whom he currently instructs in computer science at Winthrop College near his home in Rock Hill, S. C.

So what kind of lines lead to computer graphics fame?

First of all, straight ones. In 1962, when the 25-year-old Bresenham was working at IBM's San Jose computation laboratory between stints at Stanford, his supervisor presented him with a difficult assignment. One of the computers was hooked up to a small incremental plotter–a cylindrical drum connected to a ballpoint pen. The plotter was used to draw lines, curves, and other graphics. But for the computer to instruct the plotter to move the pen, it had to complete a laborious series of calculations using multiplication and division. For computers, as well as for people, division can be many times slower than multiplication, which, in turn, is far slower than simple addition. To speed things up, Bresenham reasoned, the computer should rely on addition and subtraction.

To understand exactly what Bresenham did, take a bird's-eye view of the line-drawing process. Relax, you don't need to be a mathematician to understand it. Say you want to draw a slanted line that slopes up to the right at a 20-degree angle. With paper and pencil, you'd pick the two endpoints, slap down a ruler, and simply connect the dots.

But on a computer screen, you have to connect an entire series of points to produce the line you want. Unfortunately, your choices in this dot-to-dot

puzzle are limited. From any given spot, you have to decide whether to move to another point directly above you, *or* to one directly to your right, *or* to a point which lies at a 45-degree angle from you; zooming along at 20 degrees isn't an option.

"Think of a chess board," says Bresenham, "with the king piece in a central square." Because you can't move the king along at a 20-degree angle, you have to settle for moving it one square at a time to the right, then diagonally a square, then right again, etc. Each zig-zag move introduces a slight error from the intended path. Such small errors produce an image which looks more like a staircase than a line. Bresenham's task was to compensate for these errors so that, on a computer screen, the staircase looked straight.

In analyzing the problem, Bresenham used knowledge gleaned from statistics courses he had taken from Professor Gerry Lieberman, who later became Bresenham's PhD advisor. The young computer whiz then turned his new-found knowledge on its head.

After some false starts, Bresenham developed a technique that gives the best approximation of a straight line that you can get with plotters and other devices that draw in a limited number of directions. According to his friend and IBM colleague, Howard Funk, Bresenham's method uses simple addition and subtraction to keep track of the amount by which the pen is above or below the line you want to draw, and then tells the plotter which way to move the pen to come back to that line. It is simple, elegant, and revolutionary.

Despite the breakthrough, Bresenham's fellow engineers weren't convinced that his line-drawing technique could produce smooth curves. So Bresenham turned to *Playboy*'s 'Playmate of the Month' for help. "I laid thin graph paper on top of the young lady," he says. He then traced the photo's outline and fed the tracing into the computer, which instructed the plotter to copy the figure. When his co-workers saw the results, they were convinced that Bresenham's method could produce what he calls "acceptable curves."

Although he had crossed a major mathematical hurdle in graphic displays, Bresenham didn't give too much thought to what he had done; it wasn't until the 1970s that he learned his work was being widely used in the computer industry. "I was looking through a magazine and saw that someone else had written an article based on my work," he says.

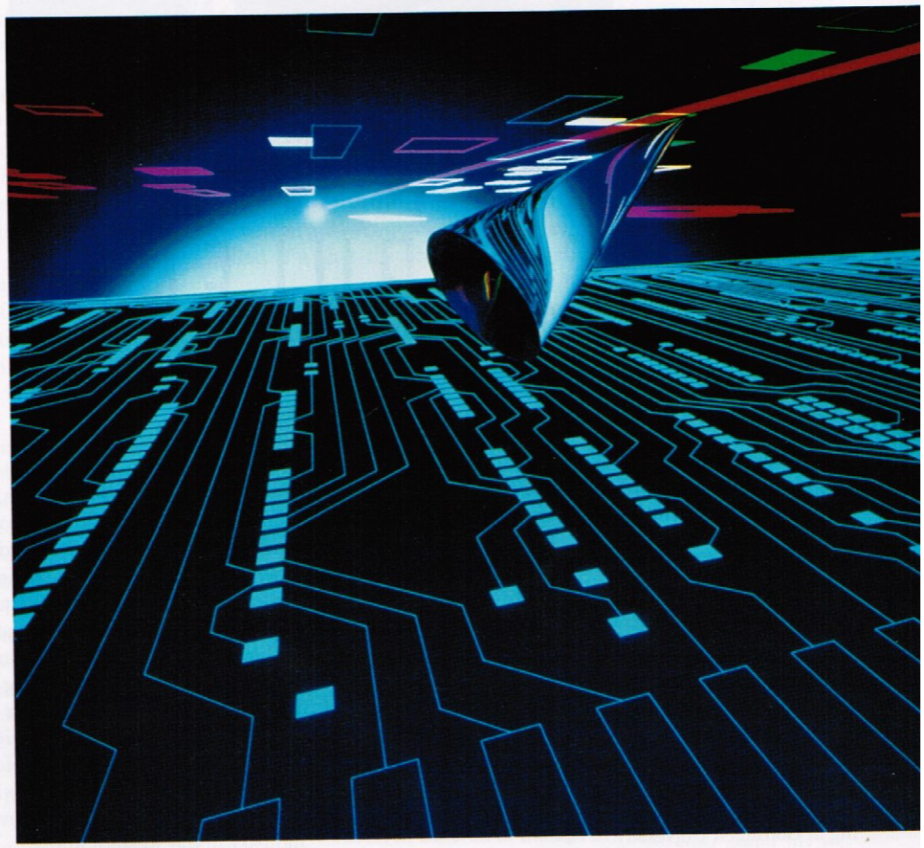
Things haven't changed. Search the technical literature today and you'll find Bresenham's Algorithm cited a lot. "The number of references to his work is enormous, and the remarkable thing is, they're current," Funk says. "Algorithms are replaced from time to time, but the good ones stick around." Despite some minor modifications by Bresenham and others, his algorithm remains the basis for almost all straight- and curved-line drawing methods.

Bresenham says he is slightly surprised by this. "A lot of work gets obscured or people re-invent it and it gets a new name," he says. "To have lasted 25 years is a bit unusual."

In practical terms, Bresenham's Algorithm has allowed computer hardware designers to use very simple processors–the brains of a computer–and run them at high speed. By the late 1970s and early '80s, computer chips had Bresenham's Algorithm hard-wired into them. Today, his work is used in products made by such major manufacturers as Texas Instruments, National Semiconductor, Intel, Advanced Micro Devices, and, of course, IBM.

The result? You now know who to thank when a computer hands you a line. ⊟

William Rodarmor is the managing editor of California Monthly, *the UC-Berkeley alumni magazine.*



In the world of computers, algorithms are almost always updated. But Bresenham's breakthrough computer graphics algorithm has remained a standard for more than 25 years.

PHOTOGRAPHY BY WESTLIGHT© ORION PRESS COMPUTER GRAPHICS

Stanford Centennial magazine

A special issue celebrating Stanford's Centennial p.158-159 by William Rodarmor "Throwing the Computer a Curve"