APPEARANCE MATTERS

'Microsoft have consistently underestimated the importance of appearance and so have not been of service to the publishing industry' said John Warnock during his Lovelace Lecture. ***Conrad Taylor*** reports.

John Warnock gave the British Computer Society's 2004 Lovelace Lecture, entitled 'The Invention of PostScript and Acrobat — Small Innovations that had a Big Impact' before an audience of members of BCS, at a conference centre in West London, on Thursday, 13th May 2004.

He spoke immediately after having been awarded the Lovelace Medal — named after Lady Ada Lovelace, pioneer programmer and Charles Babbage's assistant — by the BCS's then President, Professor Wendy Hall of the University of Southampton.

History

In talking about the development of PostScript and the Portable Document Format, John Warnock was at pains to emphasise the continuity of the ideas about computer imaging that he has been working on for thirty years, and the way in which each of his projects has fed into the next one.

Thus, although PostScript was launched as a functioning page description language in 1984–5, John traced the origins of it back to a simulator project he was hired to complete for Evans and Sutherland.

Professors David Evans and Ivan Sutherland, pioneers of computer graphics, had been John's supervisors while he was doing his PhD at the University of Utah (*A Hidden-Surface Algorithm for Computer Generated Half-Tone Pictures*–1969) and brought him in to help complete a contract to build a training simulator for the New York Harbor Authority.

The idea behind the simulator was to build a replica of the bridge of a supertanker, replacing the windows with three computer screens. These screens were to show a simulated image of the entire harbour zone of New York, showing all the surrounding landforms and buildings, jetties, buoys, etc. Evans and Sutherland had agreed to deliver the project in a three and a half year time-span, but at the time John was hired, nothing had been done on the project and there was only one year left. Seemingly, this was an impossible task.

A team of four were set to work to complete the software and database side of this project: John Warnock, John Gaffney, P. J. Zima and C. Barton. Meanwhile E&S were, in the same timeframe, to build six racks of specialist hardware to drive the large displays.

This, said John, was the project where he really appreciated the value of 'late binding'. They could not wait to develop the rendering software before starting the huge task of building a detailed database of the terrain, so they opted for a data format that was essentially a huge text file. John Gaffney masterminded this side of the project, using a large and often unreliable digitisation plotting table to capture from maps the geographical coordinates of each feature of the terrain, and adding a third figure to designate the height. All this data was written to a text-based database file.

Meanwhile, John Warnock was hard at work building the 'virtual machine' that would calculate and render the harbour views from this database. This is where the roots of PostScript can be detected. The interpreter ran on a DEC PDP-11 computer with only 32K of memory, and had to be very efficient in the way it managed resources.

Almost miraculously, the project was delivered only one month late.

John Gaffney and John Warnock went on to collaborate on another E&S project, a real-time space shuttle simulator for NASA to help train astronauts in the use of the shuttle's manipulator arm for moving loads in and out of the shuttle cargo bay. For this project, they decided to create a fully functional programming language with a stack-oriented approach to the management of resources. This language, which they called The Design System, also points to the future development of PostScript.

The Xerox years

The next twist in Dr. Warnock's career came when he was hired to work at the legendary Xerox Palo Alto Research Centre (PARC). He was put under the management of Charles ('Chuck') Geschke, who was soon to become his partner in the founding of Adobe Systems. John worked together with Martin Newell and Doug Wyatt on the task of developing an imaging language that Xerox could use to drive its innovative laser printers.

The interim version of their efforts was known as JaM (for **J**ohn **a**nd **M**artin), and was designed as a device-independent imaging language, floating above the world of 72 dpi screens and 300 dpi printers then inhabited by Xerox technology, and anticipating the development of progressively higher imaging resolutions.

JaM can, in fact, be seen as the PostScript language and imaging model but without the outline font handling mechanism: Xerox fonts were all stored as hand-tuned bitmaps at a series of fixed sizes and orientations.

This all started to come together in around 1982, implemented in MESA on Xerox Alto and Dorado computers. However, John and Chuck were not best pleased by the way the project developed to meet Xerox management's demands. Their ideas were eventually implemented by Xerox as the page description language InterPress — a strange and messy mix of declarative and interpreted language.

The face of PostScript

As is now well known, John Warnock and Chuck Geshke decided to leave Xerox together, and set up a small company based in Mountain View and named after a local stream, Adobe Creek. Their initial idea was to compete with the likes of Interleaf and Texet by developing workstation publishing software, but they were advised by friends that their true expertise lay in their ideas about how to develop a device-independent document imaging model and they should concentrate on that.

So JaM came off the shelf, so to speak, and became PostScript, with the additional help of Doug Brotz, Bill Paxton and Dan Putman. The problem which the team now had to address was how to handle fonts. Up until that time, everyone had failed to devise a system that would produce raster fonts for printing from outline data 'on the fly'. Xerox had failed; even eminent computer scientist Don Knuth had failed; and so the conventional wisdom was that it was impossible. Every other system used hand-tuned bitmaps.

The problem, as John proceeded to demonstrate with diagrams, is that a simplistic process of 'turning on' pixels that fall inside, or are touched by the font outline, leads to unfortunate aliasing problems.

Showing a letter 'm' for example, John showed how one might end up with two vertical stems, or 'staffs' as he called them, at two pixels wide, while the third was rendered three pixels wide. Diagonal lines might be rendered with odd wiggles in them; rounded lowercase letters might project above and below those with flat tops and bottoms; and serifs would not be rendered consistently.

Unlike previous attempts to solve this problem, which had tried to come up with more sophisticated scan-conversion algorithms, the Adobe team decided to ‘cheat’ by moving the goalposts — more accurately, the letter boundaries — in such a way that the resulting scan conversion would be more acceptable. According to John, this was the first time a member of the team had spoken publicly about how they did this: ‘We kept this as a dark secret for 22 years.’

Each PostScript typeface was assigned sets of ‘blue hints’. The ‘blues’ were critical zones within the vertical dimension of the face, such as the baseline and the x-height, where it was important to normalise the placement of pixels to guarantee evenness of letter-height. Then, each letter was given ‘yellow hints’ for each of its vertical stems, to ensure that these would be rendered consistently.

The bitmaps for PostScript fonts are generated when they are needed, by the PostScript Raster Image Processor (RIP) for that printing device. The RIP first retrieves the font’s outline coordinates and scales them to the required dimensions, then uses the blue and yellow hinting information to move the boundaries relative to the pixel grid which is appropriate to that particular printing device’s resolution. Only now does scan conversion take place, and the alphabetic bit-maps, thus generated, are cached and used as required to build the display for the page.

John also described some other subtleties in this process, such as the tweaking of 45-degree lines, and an ‘erosion’ procedure, which stops small letters from being rendered too bold.

The launch of PostScript

Support for the PostScript project came from Steve Jobs, co-founder and CEO of Apple Computer, who visited the little start-up company and saw what they were doing. Steve decided that Apple should build PostScript into a laser printer to go with their Macintosh computer (launched in January 1984), and gave Adobe a million and a half dollars as an advance on royalties to finance the development of the PostScript RIP for the Apple LaserWriter.

Burrell Smith of Apple and Dan Putman of Adobe started to design the Apple LaserWriter controller board in 1984. Equipped with a Motorola 68000 processor, like the Apple Macintosh itself, this board was to be, in effect, Apple’s most powerful computer, with a megabyte and a half of RAM at a time when RAM was very expensive indeed.

Even more nerve-racking was the building of the ROM chip to contain the printer’s operating system and PostScript interpreter program code: half a megabyte of masked ROM, more than had every been attempted anywhere before, to the best of John’s knowledge. There was absolutely no margin for error.

Nor, at first, did it look as if there was any margin for profit, despite the high price of $7,000, which provoked such shocked incredulity not only in the market at large but even within the dissenting ranks at Apple Computer.

The cost of all that RAM meant that the printer was going to be sold at a loss, perhaps only justified as part of Apple’s larger strategy to promote the Macintosh Office vision of networked Macs sharing a superior printer. Then in the week before the LaserWriter shipped in early 1985, the cost of RAM took a tumble and the printer won some profit margin.

Far from being an expensive mistake, the LaserWriter was an immediate hit with graphic artists and publishing professionals. Combined with innovative publishing software, such as Paul Brainerd’s PageMaker, it created Desktop Publishing and saved the Macintosh.

The importance of PostScript in publishing was soon to be confirmed when the second licensee was Linotype, who commissioned Adobe to provide their L100 imagesetter with a PostScript RIP, soon followed by the highly successful 2540 dpi Linotronic 300 imagesetter.

In 1987, IBM chose PostScript as their print imaging model; Hewlett-Packard followed ‘in a nanosecond,’ John said. Success also brought competition, with seventy clone competitors to the PostScript interpreter by 1988.

Competition also came from within Apple Computer, where an influential group pushed for the development of the TrueType font format; and in 1989 Apple and Microsoft announced that they would develop a whole alternative imaging model dubbed ‘True Image’. John seemed gleeful as he pointed out that the True Image project produced only one printer — and it didn’t work.

As for TrueType, that of course is a whole different story — and not one that John went into.

The surprisingly early roots of PDF

John next turned to that other useful little invention, the Portable Document Format. This also has longer roots than might at first be imagined; John dated the origins of the idea to 1984.

Steve Jobs — ever the showman — wanted to have some sample documents that could be printed out on stage at the LaserWriter launch event.

Lacking, at that time, a sufficiently sophisticated page make-up program, Adobe created hand-coded PostScript to print a number of documents, the most visually and politically impressive of which was to be the IRS tax form. Applying his elegant programming style, John created the PostScript for this form with efficient subroutines. The trouble was, it took two and a half minutes for the LaserWriter to process that code and print the page.

Steve Jobs insisted that watching a LaserWriter’s little green light flashing for two and a half minutes was going to seriously undermine his performance, however wonderful the final printed result. Couldn’t John get the thing to print faster?

John, therefore, went back into the code, unpacked the clever subroutines and loops, and flattened them out into a static description of the page; and this, essentially, is what PDF was to become. The immediate benefit was that the IRS tax form sample now printed in 12.5 seconds.

In 1991, just as the Internet was opening up and the Web was about to burst onto the scene, John Warnock started to conceive a method for sending fully-formatted, graphically rich documents across the Internet.

Al Gore had been making popular the metaphor of the internet as the ‘Information superhighway’, and in one of his early presentations about what was then called the Carousel project, John said that the so-called superhighway was like a link between walled cities, with no vehicles in which to make the journey. The Portable Document Format (PDF) was to be an implementation of the Adobe imaging model designed to make those documents travel safely and arrive intact.

And so the Adobe team got to work. The first demonstrations of the technology were created by Alan Wooton and Mike Pell. PostScript veteran Doug Brotz figured out how to write the Distiller application to convert PostScript files into PDF, and the Englishman Peter Hibbard figured out the file structure with its object-oriented, nested structure and its end-of-file table of references to make the whole thing hang together.

Acrobat was announced to the world in general in 1993. It has been enormously successful. There have been over 600 million downloads of the free Acrobat reader, and there are an estimated 60 million PDF documents on the Web. Acrobat is now Adobe’s largest product, and also the one that is showing the fastest growth.

There have been changes to both Acrobat and to PDF since it made its debut, but John did not have the time in this brief lecture to go into that detail. The only thing he did mention was the effort that had gone in the early developments of Acrobat into the font-substitution mechanism using a generic sans-serif and serif face in Multiple Master font form. At the time that seemed significant because by having the fonts substituted rather than embedded, one could trim down the file-size; in an era of 9600-baud modems, that seemed worth doing. Now, everybody embeds the fonts anyway.

Q&A

John then took and answered about twenty questions from the audience. Some of these were more about commercial matters, but there were also some with a technical bent. For example, John was asked about the typeface Optima, which seemed to have two sets of outline data in it.

‘True’, said John, ’because the gentle curves on the stems of Optima are a nightmare at low resolutions, so they used one set of outlines for small sizes at low resolutions and another for the larger sizes.’

John was also asked whether PostScript and PDF files had archival qualities: whether they would still be readable in a hundred years’ time, for example.

John saw no reason why not, provided that the knowledge was kept alive of how to write an interpreter for whatever future computing platform would be required to read them. Ed Taft at Adobe has the responsibility for watching over the programming teams who work on PostScript and PDF to make sure that the file structures are kept clean and well-structured.

Someone wryly commented that PDF might be platform-independent, software-independent and resolution-independent, but an A4 document wouldn’t print well on US Letter paper.

‘True’, said John, ’but PDF gives better visual results than any of the alternatives.’

I myself asked John to comment a little on the problem of transparency in the PostScript imaging model. The inability of PostScript to define objects as being transparent (e.g. to produce a better irregular outline around hair in a cut-out photo, or a drop shadow) had been as a result of the design decision not to require PostScript RIP to retain ‘state’ in the interests of efficient memory management.

In PostScript, as objects in the Display List are rasterised, they are immediately merged with the virtual page; and the memory, thus released, is recycled for the next task. Processing transparency in an imaging model requires holding the state of objects potentially a hundred levels or more apart in the stack.

At high resolutions, the memory requirements are enormous. Processing transparency in the Adobe imaging model has, therefore, been possible only very recently, as memory costs have fallen.

Did John have any regrets? Was there anything he might with hindsight have done differently?

‘Well, it’s hard to argue with success,’ said John, perhaps a little too complacently; but he did add that Adobe should have published the specifications of the PostScript Type One font format and its encryption techniques earlier, instead of in 1989 in reaction to the Apple-Microsoft threat from TrueType. That could have forestalled what John considers the unnecessary complication of TrueType.

Why hasn’t Microsoft taken over PDF or defeated it with its own alternative? ‘Because’, said John, ‘it seems that Microsoft “just doesn’t get it”—they have consistently underestimated the importance of appearance and so have not been of service to the publishing industry.’