Introducing Adobe, its graphic model, and its software

Prehistory: Warnock & Xerox PARC

Adobe Systems was founded in 1982 by John Warnock and Charles Geschke, two computer scientists who until then had been working at Xerox's Palo Alto Research Center (PARC).

John Warnock is a mathematician with an interest in the visual arts: he is a keen amateur painter. He studied mathematics and philosophy at the University of Utah. For his doctoral research, John Warnock was advised by two professors in the Computer Science Department, David Evans and Ivan Sutherland, who were pioneers of computer graphics and simulation systems and ran their own computer graphics business. His PhD described the 'Warnock algorithm', a recursive computational method to generate halftone pictures from complex geometry – which became significant in his later work on publishing systems.

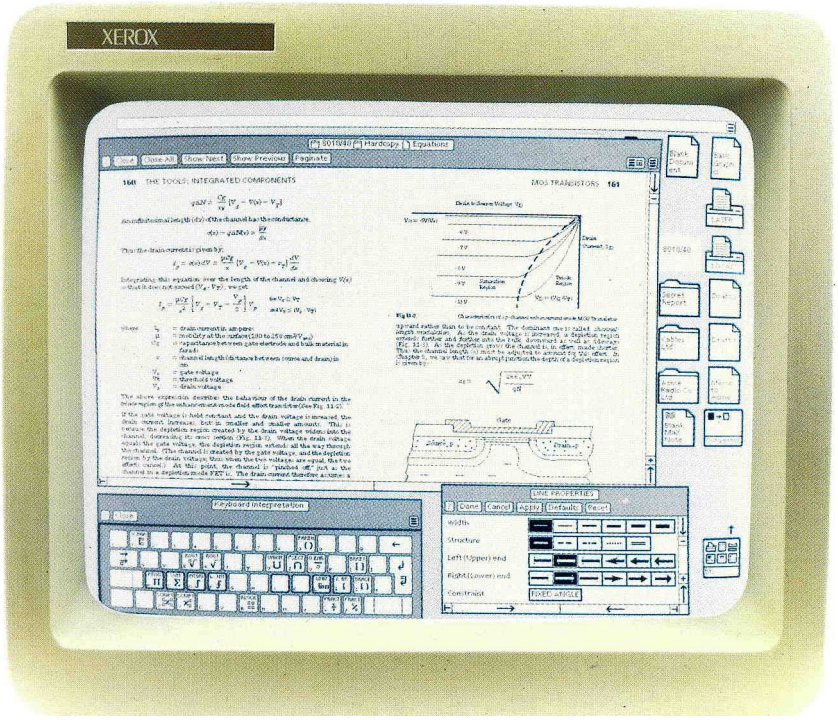
Warnock worked for a while in the San Francisco office of Evans & Sutherland, developing graphic simulation systems for training pilots and astronauts. In 1978, he transferred over to Xerox PARC, joining the Imaging Sciences Laboratory headed by Charles Geschke.

Xerox PARC was a remarkable laboratory in California's Silicon Valley. This is where the laser printer was invented in 1969; where LCD displays and optical discs were also developed.

Bob Taylor ran the PARC Computer Science Laboratory, which had a close relationship with (and recruited staff from) nearby Stanford University's 'Augmentation Research Center'. This was where Douglas Engelbart and his team had invented the computer mouse, email, videoconferencing and hypertext.

The PARC Computer Science Laboratory in turn was the birthplace of Ethernet, bit-mapped computer displays and the first computer systems to use the Windows–Icons–Menus–Pointer system and the desktop metaphor – things we now take for granted as standard for computers, but quite revolutionary then.

A key achievement of Xerox PARC was the Xerox Star 8010 graphical workstation, brought to market in 1981, conceived of as a networked document preparation system with a WYSIWYG ('what you see is what you get') document editor, and output to a laser printer. On the next page you can see what the screen of a Xerow Star looked like.



The Xerox Star 8100 workstation, introduced in 1981, was a pioneering desktop publishing system. However, a single workstation cost $16,000, and a viable networked system for a small office, with file-server and laser printer, cost about $100,000, so the Star was not a commercial success and this discouraged Xerox management from experimenting further in computing.

The ideas behind the Xerox Star were revealed to Apple boss Steve Jobs and Apple engineers during visits to Xerox PARC, and this led directly to Apple's development of the Macintosh.

Warnock and Geschke from Xerox to Adobe

At PARC, Warnock worked on the issue of **page description languages**. This was the problem that needed addressing: in the Star 8100, Xerox had a fine workstation for creating documents. Xerox also made laser printers. But how does the computer 'instruct' the printer which parts of the page to mark with black, which parts to leave white?

For the Star project, Xerox used a data format called Press, but this was thought to be too inflexible, and the **InterPress** project was set up to invent a successor to Press, based on the idea of an interpreted computer language. The elegant solution which Warnock worked on involved installing in the laser printer its own specialised computer, to which a set of drawing instructions could be sent, written in a special computer language. The embedded computer has the job of interpreting the instructions to create a **raster image** (one made up of laser-printer spots), and so it is called a **raster image processor**.

Frustratingly for Warnock and Geschke, Xerox management on the East Coast were not interested in developing and commercialising InterPress. They wanted Xerox to play to its strengths as a copier and printer company, and thought the whole Star project was an expensive failure (only 25,000 units were ever sold).

In 1982, Warnock and Geschke resigned from Xerox and set up their own little company in nearby Mountain View, with the aim of creating a printer-control system that would realise the potential of InterPress. They called their company Adobe Systems after the nearby Adobe Creek, and set to work creating a flexible, powerful and expressive page description language – PostScript. In parallel, they worked on devising the PostScript Interpreter (the software which creates a raster image by interpreting PostScript commands), and the hardware to run it on.

Adobe PostScript, and its fonts

The first thing to say about PostScript as a computer language is that although it is possible to learn the language and write programs in it, the job of writing programs that will result in printed pages is *not* a job for humans!

Suppose you create a page in a word processor, and you want to print it to a PostScript printer. When you print, choosing that printer calls up a PostScript **printer driver** – a program that your word processor uses to create a description of the page in the PostScript language. That PostScript file then travels to the printer, where the raster image processor interprets the PostScript commands, and prints the page.

In developing PostScript, John Warnock drew on some of the ideas from his PhD thesis, and some earlier work he had done on a simulation of New York harbour for an Evans and Sutherland project. His key idea was to describe all of the content of pages for printing not as collections of spots, but at a much more abstract level – as geometry.

Warnock was determined that this would even be true of typefaces. Xerox publishing systems had stored each character in their fonts as an array of dots – in other words the Xerox fonts were 'bit-maps'. This was very limiting. For example in a Xerox system one might be able to typeset text in 10 pt, using the 10 pt collection of dots, or in 12 pt, using the 12 pt collection of dots – but if you wanted 11 pt type and no matching collection of dot-patterns had been provided, you could not have that size. (Nor could you set type at an angle, or in another shade or colour other than black.)

Warnock decided that in the PostScript system, a font would be created as a collection of geometrical images, made up of straight lines and curves. In fact, a PostScript font would itself be a PostScript program, stored in memory chips on the printer's controller board, which could be called on when it was needed.

Here is a tiny PostScript program that selects a typeface, scales it to 20pt, gets it converted into bitmaps for that size ('setfont'), defines the co-ordinates of a starting point on a page ('moveto'), typesets a few words, and prints:

```
%!PS
/Helvetica-Bold findfont
20 scalefont
setfont
72 500 moveto
(A few words I want printed) show
showpage
```

PostScript, Macintosh and desktop publishing

While PostScript was being developed, much encouragement came from Steve Jobs, one of the co-founders of Apple Computer. Apple was at the time developing two new kinds of computer, the Lisa and the Macintosh. Both these designs evolved in a direction inspired by visits that the Apple engineering team had made to Xerox PARC: they adopted the bit-mapped display, mouse, windows, icons and so on that had been developed at Xerox PARC.

The Apple Lisa, launched in 1983, was in some respects a more sophisticated machine than the Macintosh; but at $9,995 it sold slowly – and it ran pretty slowly too. Steve Jobs championed the rival Macintosh design, which was released in January 1984, initially at a price of $1,995, and later on at well over two grand. Which is a lot, for a machine with only 128k of RAM and no hard disk.

The risk was that the Mac would be regarded as just a toy, not a serious business machine. It had a suite of free software bundled, all written by Apple – MacWrite, MacPaint and MacDraw; but at first it proved difficult to persuade software houses to port their business software over to Mac. Initial sales of the Mac were promising, but then began to fade.

Arguably, what saved the Macintosh was PostScript! Apple wanted to add a networked laser printer to their product range: it was central to their concept of the 'Macintosh Office'. Steve Jobs decided it would have to be a PostScript printer – indeed, the first PostScript printer. The PostScript interpreter board (that is, the computer inside the printer) actually ran faster and had more memory installed than any Macintosh. Because of this, and licensing fees paid to Adobe, the LaserWriter was expensive ($6,995).

However, in an era when local area networks were rare, and hard to set up, the Mac and LaserWriter came ready for instant networking. You could easily link a dozen Macs and a LaserWriter with LocalTalk cabling – this made the LaserWriter affordable for a workgroup, or for a graphic design studio.

The LaserWriter had a printing resolution of 300 dots per inch: that was not fantastic, but good enough for newsletters, flyers and other kinds of print job. But that was just a hardware limitation – the PostScript imaging system is resolution-independent. Adobe also had another customer – Linotype – who built a 2,540 dpi laser imagesetter with a PostScript RIP. This meant that you could use a LaserWriter as a proof printer, and get high-quality output from the same file by sending it to a PostScript imagesetting bureau.

And then came desktop publishing. The flexibility of PostScript printing, and the graphical interface of the Macintosh, encouraged developers to create page make-up software, especially when the Mac 512k came along with just enough memory to get the job done.

The term 'desktop publishing' itself was coined by Paul Brainerd, who developed **PageMaker** for the 512k Mac and released it to the market in July 1985. In the same year, Manhattan Graphics released **Ready,Set,Go!** and **QuarkXPress** came along in 1987.

Nor was the new method of publishing confined for long to the Mac. In 1986, another 'graduate' from Xerox PARC, John Meyer, released **Ventura Publisher** for IBM-compatible PCs. Later, when Microsoft developed Windows to a point where it was useful, PageMaker and QuarkXPress also came across to the PC.

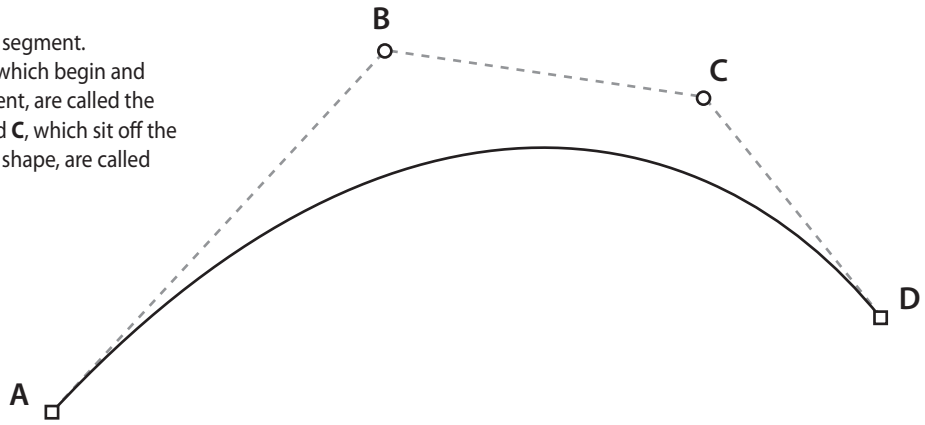
What had seemed a gamble when Warnock and Geschke gave up their jobs at Xerox PARC was now paying off. The PostScript bandwagon was on a roll. Linotype released their font library in PostScript format, and other manufacturers of laser printers and high-resolution imagesetters adopted the PostScript way of doing things. In this period, all of Adobe's income came from licensing the PostScript system and selling fonts – they did not have any software to sell to end users until 1987.

The Bézier curves as used by Adobe

Recall that on page 3 above we noted that John Warnock chose to make PostScript font characters geometrically, using straight lines and curves. To be more precise, he decided to use **Bézier cubic curves**. These are the curves which we draw with when we use Illustrator's pen tool.

Pierre Étienne Bézier was a French engineer who worked for over four decades at Renault, for whom he developed a computer-aided design automobile system called UNISURF. He needed to be able to work with subtle curves – not just arcs, which are segments of circles, but curves which vary their curvature along their length. It's important for vehicle body design, and it turns out to be very handy for drawing typefaces, too. Suitable candidates were curves formed by quadratic equations, and cubic equations. (But we are interested in how Adobe uses Bézier curves, so we will forget the quadratic versions!)

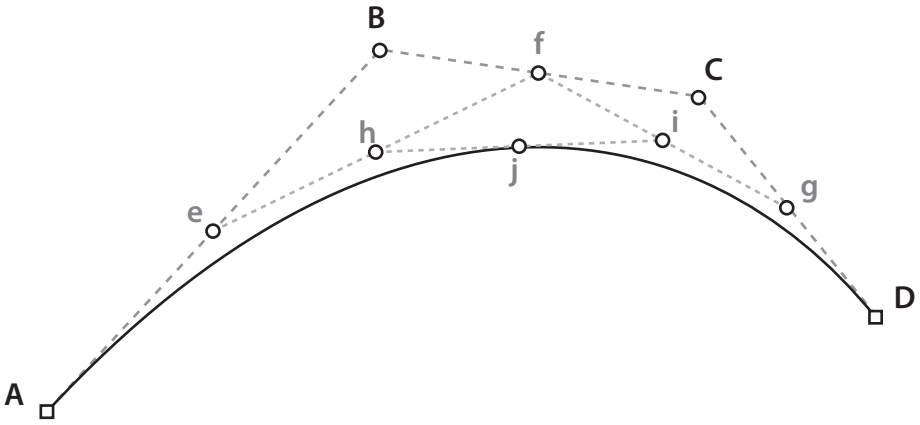
Cubic equations can be turned 'inside out', so that the parameters which determine the shape of the curve are represented as four points on a two-dimensional plane:



A Bézier cubic curve segment. **A** and **D**, the points which begin and end the curve segment, are called the 'anchor points'. **B** and **C**, which sit off the curve but control its shape, are called the 'control points'.

The location of these points is of course represented by their x and y co-ordinates, which are numerical values – and that is very tasty fare for a computer! To store this curve in computer memory, all we need are these four sets of co-ordinates. The curve in a sense 'does not exist', but drawing software like Illustrator can render it on the screen or on the printer whenever it is required.

The beauty of Bézier curves for computer drawing is that they are a perfect complement to having a graphical screen and a mouse pointing device. The mouse can be used to place the points on the drawing, and drag them around to adjust the shape afterwards. Though it does take time and practice before handling them becomes instinctive…



How four points are made into curves…

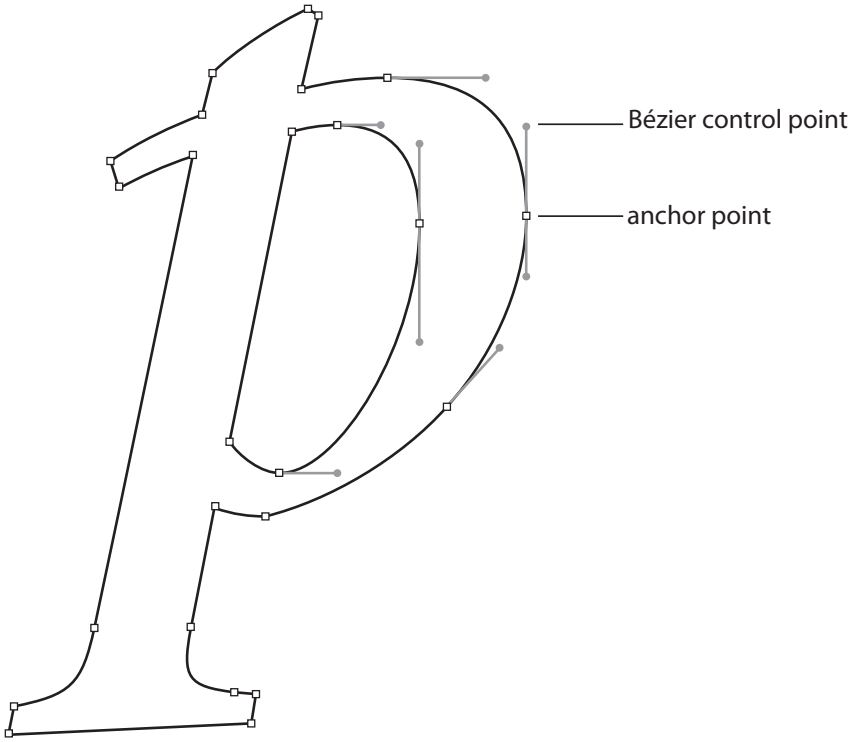
How can four points be turned by a computer into visible, printable curves? It's actually quite a trivial task and involves applying the same process again and again…

- For the computer, it is very easy to find a 'mid-point' between any two points. One simply calculates the average of the x and y co-ordinates of the two points, and that generates your mid-point.
- In the diagram above, the mid-point between **A** and **B** is **e**, that between **B** and **C** is **f**, and that between **C** and **D** is **g**.
- We can repeat the process to obtain mid-points between **e** and **f** (that's **h**) and **f** and **g** (that's **i**).
- Finally, the mid-point between **h** and **i** is **j**, and that point is sitting on the curve.
- Another way of looking at it is that we have divided the Bézier curve segment into two segments. **A**, **j** and **D** are anchor points, and **e** and **h**, **i** and **g** are the new control points.

We can now have the same process repeated to the two derived Bézier curve segments, to produce four – and repeat that again to produce eight, sixteen, thirty-two, sixty-four… This process of repeating the same function is called '**iteration**' and it was key to the solution that John Warnock came up with for his PhD thesis work.

The trouble is that iteration can literally go on for ever. But it doesn't need to. What we interested in is generating a polygon between the accumulating anchor-points – one that is sufficiently fine-grained that it represents the curve 'well enough'. And then you get the computer to stop obsessing.

When should it stop? It depends on the resolution of the surface that you are constructing the drawing on. For a computer screen, that's only about 80 to 100 pixels per inch. For a laser printer, it is six times higher and that requires running the iteration a few more times.



From font design to Adobe Illustrator

At Adobe, they created a drawing program so that they could draw the outlines of the type characters they were creating for the PostScript printing system, as shown above. Of course there is more to creating a computer font than just drawing the shapes – there is other software involved in that.

For typefaces to work properly, the 'counters' (spaces in the middle, like the gap in the letter ***p*** above) must be transparent. For this, Adobe uses a special form of path grouping, called a 'compound path'.

In April 1987, Adobe announced the launch of their first offering of boxed software. **Adobe Illustrator** was based on their font-drawing tool, with extra features added. At first it ran only on the Macintosh, and the first version could draw only in shades of grey; the following year 'Illustrator 88' came out and could prepare CMYK artwork for commercial colour printing.

There were sporadic attempts to port Adobe Ilustrator to other kinds of computer (NeXT, Irix, Sun workstation) but these were not commercially successful; and the first version which gained acceptance on Windows was version 7.

On the Macintosh, its main competitor was **Freehand**, created by Altsys and marketed by Aldus, then later by Macromedia. On the Windows platform, its main competitor was **CorelDRAW!**

Along comes Photoshop

Most of the computer applications which Adobe sells were not actually invented in-house. Photoshop was the brainchild of a PhD student at the university of Michigan – Thomas Knoll. Photoshop started as a project to display greyscale images on the purely black-and-white bitmapped display of a Macintosh Plus.

Thomas' brother John, who worked in movie special effects at George Lucas' Industrial Light & Magic, persuaded him to develop the software into a proper editor for raster graphics (that is, graphics which are made up as an array of pixels, as generated by a scanner).

John Knoll gave a demo of the software to staff at Adobe; they decided they would like to purchase the licence to distribute it. Photoshop 1.0 came out for the Macintosh in 1990 and rapidly became the graphic arts industry tool of choice for adjusting and improving digital raster images, making montages and fantasy images, etc.

In 1991, Version 2 (still Mac only) added a Bézier-curve Pen tool, at the time the only way to make a raster image of a non-rectangular shape was by saving it with a PostScript 'clipping path', and the pen was introduced to create these vector-based clipping paths.

In 1992, Photoshop 2.5 arrived, and a version for Windows was released. Version 3.0 which came out the following year was a highly significant release because it introduced Layers. At last it was possible to have a multi-layered Photoshop file, and this made montage creation much easier. Before that, once you had pasted a new element into the image, it merged with the background!

Photoshop started out as a program that worked purely with raster image data (clipping paths being the sole exception), but this has changed rapidly and significantly since 2000 (version 6.0). Type is now in vector format and remains fully editable; vector shapes can also be created on their own layers.

Photoshop users have long used the software's own format (.psd) to store their artwork, but have converted to another format for use in print, on the Web or in presentations and multimedia. This began to change when Adobe created their own page make-up software, InDesign, which can place native Photoshop files without them needing to be converted.

Acrobat and PDF make their appearance

The next major venture by Adobe Systems was the invention of the **Portable Document Format**, a project with which John Warnock was again very much personally involved. In many ways PDF is a re-implementation of the same computer graphics data model as PostScript, but in a way that creates relatively compact files for 'document exchange'. PDF files are easy to send over electronic network links, and view on any computer that has the Acrobat Reader software installed.

Whereas PostScript is a full programming language, PDF is more like a graphic file format. PostScript files have to be interpreted in entirety before pages can be presented for viewing (i.e., printed), but PDF files are organised hierarchically as nested sets of objects, which means for example that one page can be quickly viewed independently of the other pages. Nevertheless, everything that can be represented in a PostScript file (and given the demands of publishing, that means *everything* you might want to put on a page!) can also be faithfully represented in a PDF file.

Acrobat components

Originally, the Acrobat suite of software in 1993 had four components. **Acrobat Exchange** was the application for viewing PDF files and interacting with them; it was sold with **PDFWriter**, which pretended to be a printer driver, and could do a decent, convenient job of producing PDF so long as you used no PostScript graphic files (such as an Illustrator .eps file) in your publication.

Distiller was the 'professional' application for converting PostScript files into PDFs: it offered better control and could process all PostScript graphic contents. To create a PDF using Distiller, you would ideally set up your publishing application as if you were going to print to a PostScript printer, but get the PostScript saved to disk instead of sending it to the printer. Then, you would feed the PostScript file to Distiller to have it converted into PDF.

Adobe also supplied **Acrobat Reader**, a view-only application which could not make editorial changes to PDF files. This was initially sold at $50 or local equivalent. But Adobe had to re-evaluate its marketing model pretty quickly when take-up of PDF was extremely slow. It made a lot more sense to give the Reader away for free, to encourage people to download and install it, because having a large user base of free Readers out there is an incentive for document producers to save their documents as PDF for posting on Web sites.

Acrobat Catalog came along with version 2.0 (1994). It lets you build a search index for a collection of PDF files and search against them all.

Acrobat & PDF features and developments

Font simulation or embedding: If you transfer an electronic document in an original format such as Microsoft Word or Powerpoint, and the recipient doesn't have the same fonts that you used to create it, not only will they not see the document as you intended it, but font substitution may cause havoc with line endings and page breaks. To avoid this, Acrobat was designed to do one of two things:

- The font's name and 'metrics' would be captured (i.e. the width and stroke weights of all the letters), so that the Acrobat software used to view the file could generate a *simulation* using either a generic serif or sans-serif font. It might not be a faithful reproduction, but at least the type would occupy the same space. This was useful as a way to keep the filesize down at a time when bandwidth was limited, but later versions of Acrobat discontinued this method.
- Or, the font data could be embedded into the PDF, to guarantee that it would be used to render the document. Optionally, one might save on filesize through 'subsetting', which embeds only the characters that have actually been used.

Data subsampling and compression: This is the main reason why PDF files are smaller than the sum of all the files used to generate the original document. All of the text content of the file, and all of the geometrical elements such as boxes and vector graphics, have fairly standard, lossless data compression techniques applied to them.

The situation is more complex when it comes to raster-based images such as photographs, captured screen images, or scans of line drawings. Potentially these are quite massive chunks of data! In the process of setting up Distiller or another application to save a document as a PDF, you can choose various settings for image subsampling and compression.

- **Subsampling** is like using Image Size commands in Photoshop. It reduces the number of pixels in the image. As a result, the subsampled image will not have as much detail. For professional printing, you should not subsample colour or greyscale images below 300 dpi, or line-drawing scans below 1200 dpi. But if you are just sending a low-resolution proof copy or parking a downloadable version on a Web site, you might be happy to reduce the pixel resolution in exchange for a smaller file for downloading.
- **Compression** for raster images, if you choose to ask for it, can be done either in a 'lossless' or a 'lossy' manner. Lossless compression does not help you much with continuous-tone greyscale or colour

images such as photographs, but it can be effective for images that have flat colour (such as a screen-shot of a computer dialogue box) or for a scan of a line drawing.

- In the case of photographs, it is best to use the JPEG compression method, which offers a sliding scale of quality settings. At JPEG maximum quality, the images compress to about half their filesize and quality is not noticeably compromised for most images. If you apply heavier compression settings, the result will be a smaller PDF file – but you may regret the nasty little artefacts (like a scattering of dots) around edge details.

Another way of reducing the size of a file is to have all of the colour elements and images converted to the Red-Green-Blue colour model from Cyan-Magenta-Yellow-Black. This shouldn't be done for PDFs being sent to commercial print.

Security: Adobe was aware that people might want to send sensitive documents in PDF format, or might want to restrict what people can do with them. You can set two levels of password protection for a PDF document. One won't even let the recipient view the file unless they have the password. A separate password gives access to controls where you can say, for example, that people can read a document on screen but can't print it; or that they can read or print but not annotate it or change it in any way.

Annotations: Over the years the main Acrobat application (not the Reader) has added the ability to add 'sticky note' annotations and highlighting to PDF documents. This is useful for collaborative editing or proofreading.

Interactivity: From an early stage, PDF had hypertext-style linking and you could use the main Acrobat program to set regions which, when clicked on, you jump you to another page view within the same document. Later, support for external links (Web URLs) was added. It is now even possible to embed sound files or video clips into PDF files, or some kinds of three-dimensional images which can be grabbed with the mouse and rotated.

Graphics features: The Portable Document Format has gone through a number of revisions to support increasingly advanced data structures in graphic files. For example, in early PDF there was no support for transparency in graphics files, except that produced by clipping paths and masking. This made it impossible for PDF files to contain drop shadow effects, which depend on partial transparency.

Suitability for transferring jobs to printing services: The first few versions of PDF were OK for reading on screen, proofreading, or sending to an office printer, but PDF sadly lacked some of the features required for delivering a job for professional print output. These shortcomings were addressed around the turn of the century, and now most printers prefer to have publication 'artwork' sent to them as PDF.

One development that encouraged this trend was when printers stopped printing out films, taping them together into 'flats' on lightboxes and contact-printing the manually assembled films to litho press plates. Once they started acquiring direct-imaging platesetters instead, they needed a way to impose the page data into plate groups electronically. The 'object' data structure of a PDF file makes it ideally suitable for tearing apart into separate pages and organising into plate groups.

Sending PDFs to printing services involves taking responsibility and knowing what you are doing. All colour photos should be in CMYK format, for example, unless you are advised otherwise. More recent versions of Acrobat include 'Preflight' tools to check that files are OK to send. Standards bodies in the graphic arts have also developed subsets of the PDF standard that can be guaranteed printable – such as PDF/X.

Using PDF as a graphic file format: Over time, some publishing applications such as InDesign, FrameMaker or QuarkXPress have had graphics import filters added to them so that a PDF file can be imported as a kind of image file with mixed vector and raster elements, and with all of its fonts embedded.

Other ways of making PDF files

Initially there were only two ways to make a PDF file: print through PDFWriter, or print PostScript to disk and distill it. More recently, other methods have come along. The most straightforward method is if a computer graphics application has an inbuilt ability to save in PDF format directly. This is now true of InDesign, Illustrator and Photoshop.

PDF has been integrated into how the graphics rendering system for Apple Macintosh OS X (called 'Quartz') works. Mac OS X has a representation for graphics that is closely aligned with PDF, and when you go to *Print* in a Mac application, saving as PDF is one of your options. The Preview application is also a PDF viewer.

Thomas Merz of **PDFlib** in Germany developed some C programming libraries that generate PDF. There are database applications that have licenced Merz's libraries to generate PDF bank statements, for example. Serif Software also use PDFlib for saving PDF from their DTP software.

Adobe's need for InDesign

By the late 1990s, Adobe Systems had a well established reputation among graphic artists on account of Illustrator and Photoshop. Where Adobe was doing far less well was in the field of page make-up and multi-page document publishing software.

In 1994, Adobe Systems did a merger with Aldus Corporation, Paul Brainerd's company which had developed **PageMaker**. Effectively, Adobe took over Aldus, and in the process they acquired PageMaker. Now at last they had a page make-up program.

But PageMaker had been slow in solving some of the key problems in CMYK commercial colour printing, and when Quark brought out QuarkXPress 3.2 and then 4.0, it was clear the graphic arts crowd – designers, typesetters and commercial printers – were abandoning PageMaker in favour of Quark.

In 1995, Adobe also purchased Frame Technology Corporation, the developer of the respected **FrameMaker** software for publishing technical documents. Adobe and Frame had a long history of working together, and all of Adobe's software manuals were made using Frame-Maker. But this was not software that could be sold to graphic designers and magazine publishers.

Additionally, both PageMaker and FrameMaker were 'long in the tooth'. They both needed a total rewrite, but when software projects are that complex and that established, the effort of scrapping everything and starting to code from scratch is a horrifying prospect.

Adobe decided to produce a 'Quark killer' and the project codenamed K2 started work. It was released as InDesign 1.0 in 1999. Some of the key features of the program were as follows:

- The InDesign application is structured within Adobe's own graphic environment, making as little use as possible of system libraries from either Macintosh or Windows. For example InDesign writes its own PostScript and PDF output.
- InDesign consists of a core services module, and most of the tools are provided by plug-ins. This modular arrangement, rather than one monolithic application, makes code development and maintenance easier.
- Adobe aimed in particular at providing superior typography, with a multi-line composer to decide where line endings should fall, integration with OpenType, and Unicode encoding to support a very wide range of languages.

InDesign and the Creative Suite

It took a while for InDesign to 'find its feet'; the first couple of versions did not attract much market share. InDesign started to draw the attention of Mac-using designers when Version 2.0 came along in early 2002, because it could run under Apple's new-version operating system, OS X. Also, this version could handle true transparency in images, and could add drop shadows.

Then Adobe tried a new marketing tack to make inroads into Quark's market share. They already had mindshare in the graphic design world because of Illustrator and Photoshop; now they would bundle these programs up with InDesign as the 'Creative Suite' and sell the bundle at an attractive price. Adobe also offered generous terms to move up from PageMaker (which would not be developed further). InDesign also can open and convert existing PageMaker and QuarkXPress files, which is another way to smooth the transition to InDesign.

Initially, the Creative Suite concept was not much more than a marketing tool, but with later versions there has been more and more true integration between the CS products. For example, you can copy objects from Illustrator and paste then into InDesign, and components remain editable. Photoshop .psd files can be imported or pasted into InDesign documents – and so on.

The earlier versions of InDesign favoured the design of relatively unstructured, design-led documents such as magazines and posters. Later versions have added more structural features such as tabular data display, footnotes, Table of Contents and Index generation for books.

InDesign's underlying use of Unicode to store text, and its ability to work with Unicode-encoded fonts that have many character shapes (glyphs) in them, make it a strong contender for multilingual publishing. For example, it is easy to work in Greek or Russian, or with the appropriate keyboard support, Chinese or Thai. There is a plug-in system from a third party that enables typesetting in Indic languages. For right-to-left typesetting, however (Hebrew, Arabic) there is a special Middle East version of InDesign.

In the last few years, many design companies and publishing houses have made the switch-over to InDesign. It also helps that InDesign can natively write PDF files that are suitable for sending for commercial printing.