5

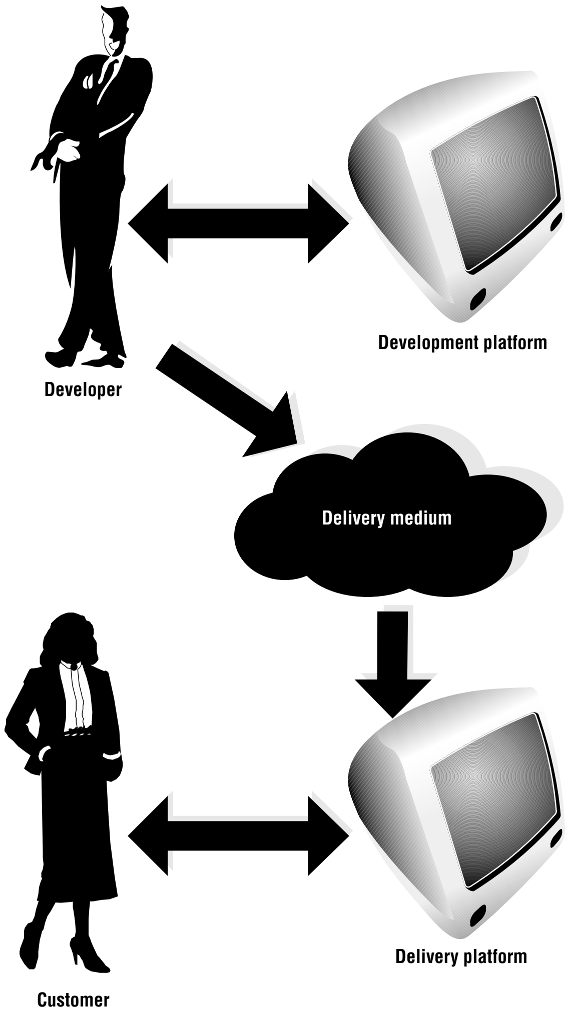
Platform parameters

Project manager's responsibilities

- To understand the implications of platform choice
- To discuss the chosen delivery and development environment with the client and advise on the best choice for the particular application
- To similarly advise on delivery medium

Introduction

The word ‘platform’ was traditionally associated with hardware: the computer platform. But a platform can also describe software as well as hardware, and it is increasingly used in this way. On the Web, the platform is made up of the browser plus computer, operating system and even a particular plug-in. For an offline application the operating system and computer is generally enough but sometimes more detail, such as the capabilities of the graphics card, may be important. A useful definition might be that the platform is whatever you have to specify in order to run the application. Often the specification will be extended to say what kind of display is needed, or



how big a hard disk, or how fast an Internet connection. This chapter will use the word ‘platform’ in this broad sense.

There are three options that will be discussed in relation to platforms. The first is the most obvious, and is the delivery platform: what does your user have to use in order to see your website (or application)? Then we come to the delivery medium: how do you get the application to the user? Finally we shall discuss the development and testing platform: what do you need to use in order to make the application in the first place?

Delivery platforms

Sometimes a client will come to you with a project and they will know exactly what platform they want to use. This may be because they use a certain machine or browser in their business, or because the target market for the application has mostly machines of a particular type. They may have other websites and have a set specification for those. Sometimes they will be a little more vague, and often they will ask for your advice. This chapter will not offer any advice for your particular project but it will help you to ask the right questions about the requirements, and to work out how to identify the positive and negative factors influencing the choice. The basic question we are asking here is ‘Which platforms are useful for what purpose?’

Online delivery

Delivery on the World Wide Web usually sidesteps many issues of the delivery platform because web browsers all take the HTML (and more) that makes up a web page and display it on the user’s computer. The Web is the multimedia delivery system that works on most computers because the browser takes care of the underlying operating system and hardware. It is an example of document-based programming, because the web page is a document made to certain standards, which is displayed by any browser that adheres to those standards. Despite the vagaries of differing interpretations of JavaScript (particularly between major browsers) and whether or not the browser understands ActiveX, the web page is the most cross-platform format.

There are several main issues affecting the delivery of a website over the Internet: speed of access, updating, security/payment, and the ‘unlimited’ size of the data space that can be provided online. You can restrict access, charge your users for access, and even keep track of who has accessed your information and when. (Some of these techniques spread themselves to CD-ROM, with software and fonts being sold by giving away encrypted versions on CDs and then charging for decryption.)

The speed with which your potential users can access your application is unpredictable. You may have users sitting at the end of fast permanent connections such as ADSL where download speeds are measured in megabits (not many megabits admittedly, but megabits nonetheless) and you may have users with 28.8 kilobit modems or even slower mobile connections. Unfortunately, speeding up the local link between the user and the network does not necessarily resolve the overall problem, because other links in the system will in themselves be unpredictable. Speed will depend on the bandwidth of the link itself (colloquially 'how fat is the pipe') and on how much other traffic there is on it. Also, the fatter the pipes become, the more data people want to pass down them.

Having a fast server, as well as a fast Internet connection, will help but at the end of the day your web page will be at the mercy of every possible bottleneck as the data flows between your site and its visitor; perhaps across opposite sides of the world.

On the Web, problems of download times for audio and video have been addressed by streaming the data for audio and video rather than copying it and playing it later. Streaming is the method whereby a steady stream of data is expected from the data source, and it is processed and displayed as it arrives (on-the-fly). The classic example of this is digital audio from a compact disk. The 16-bit digital audio data arrives at the decoder chips and is converted into analogue audio immediately. There is no storage of the data. If for some reason the stream of data is disturbed then you hear either clicks, where a single sample is misplaced, or pauses and hiccuping. A DVD player will be receiving a stream of MPEG-2 movie while it plays a disk.

A way around the problem of disturbances in the data stream is buffering and/or caching. For buffering, the data from the source is read into memory at the rate at which it arrives, and is read out of memory at the rate at which it is needed, hoping that you don't run out. In this way small discontinuities can be removed. To continue the CD example, some portable players have buffers so you can cope with knocks to the disk while jogging. With a cache the data is read completely into the cache memory and, once it has all arrived, the application accesses it. (In this case it's more like an MP3 player reading a song out of its RAM.) The disadvantage of a cache is that the application has to wait for all the data to arrive, although once the data is in the cache it can be accessed many times. The program or the data format can be designed to allow the data to be both streamed and cached, of course, but that is practical only if the data stream is fast enough.

So what are the pros and cons of using the Web as a platform? On the plus side you have a low cost of entry but with sophisticated possibilities. The application itself, the website, can be changed at any time either to update it or to fix errors and there is potentially an infinite amount of space to hold information for the website. The Web's reach is global and instant. On the minus side you have a heterogeneous user base and you need to decide how much you can cater for the differences in the users' systems. There are security implications which may or may not be important for your client, although unless you are carrying out online transactions this shouldn't be a real issue. Many of your users will be on relatively slow connections, which means you need to think carefully about the use of bandwidth-hungry media such as video.

Mobile multimedia

Mobile multimedia is likely to polarize into applications delivered onto a handset, with limited display size and keypad but great mobility, and 'bigger' devices such as PDAs where you might even get handwriting recognition or even a 'real' keyboard. Working mobile onto a laptop, while being bandwidth-dependent, doesn't really imply many other differences from conventional computer delivery. The services might be different and location based for example. As a delivery platform, mobile systems have many similarities with the Web but have the issues of bandwidth taken to an extreme since even though the stated aim of third generation systems is two megabits this will be slow in coming and may be very dependent on your distance from a base station. One potential benefit of mobile over the Web is the delivery medium through a telephone network because such a network will be set up to keep track of the user, for billing purposes if nothing else. The variation of performance of handsets might also be relatively small compared to the variety of platform parameters on the Web. Chapter 4 of this book has looked at mobile systems in some detail.

Interactive television and DVD

The arrival of digital television brings with it the possibility of interactive services. Initially the broadcasters are greatly concerned about the stability of the whole transmission system and take great pains to ensure that nothing could crash the set-top box or, in extreme circumstances, more crucial parts of the system. One result of this was the use of tightly-controlled proprietary software environments such as OpenTV and Liberate used on satellite and cable systems. An open environment based on MHEG (an open standard for multimedia) and Java is specified for terrestrial digital television.

Video-on-demand (VOD) may provide another market for interactive services on the TV or via ADSL to a PC, and if so it will provide a potential market for multimedia developers for the navigation (more so than content which is probably pre-existing). The way in which VOD develops as a platform will depend on which of the two proposed models for it is used, and where. You can say that in one model the VOD server pushes the application for display in the home on a television linked to the service by a set-top box. In the other model the set-top box itself pulls the application from the server.

The 'push' model works like this: the set-top box functions as an audio-visual version of the dumb terminal and basically passes user commands down the line to the server, takes the data the server sends, and displays it. This means that the application itself, usually something to run a movie, is actually being run in the server and not in the set-top box. This can be a crucial situation for the developer because the servers are often going to be very powerful processors running real-time operating systems totally unlike those you would usually find in multimedia. This is unless the servers are set up to run virtual machines that emulate other platforms. Since the set-top boxes are operating only as dumb terminals, including MPEG decoders, there is relatively free rein for potential manufacturers as only the display and communications need to be standardized, not the application environment because that is in the server.

The 'pull' model has the server operating as a data source, whether for downloaded data or for streamed data, and the downloaded application runs in the set-top box, which therefore has to be more than a dumb terminal. This raises a different problem both for the developer of applications and for the service provider. Since the set-top boxes are likely to be from more than one manufacturer, compatibility between the boxes and the applications is in question. In practice this can be overcome by offering only applications that the server knows the set-top box can support or that the set-top box itself knows it can support.

The biggest difficulty in producing interactive television is currently the high entry cost. The tools may cost money and/or need skilled but hard-to-find programmers and the service operator may charge a significant sum for access to their system. The positive aspects of iTV are that in the long term this is likely to be the interactive delivery medium with the highest reach among the general population. For some purposes, such as entertainment or public services, this is a very strong incentive.

DVD is easier since it is possible to produce DVDs on the desktop. But with both iTV and DVDs you are likely to have a particular kind of client, one with access to and experience of those kinds of distribution channels.

Computers and operating systems

When it comes to computers, the platform is not just the microprocessor or operating system. For a mathematical calculation program it may be sufficient to name the operating system, but for multimedia there are issues of screen resolutions and bit depth, sound parameters, the speed of the CD or DVD-ROM drive, the way that video is handled, the amount of RAM and the size of the hard disk … at least. There may be issues of whether the system is capable of multitasking (doing more than one thing at once) and whether you should take account of this.

In an ideal world every computer program would run on every kind of computer hardware. However, there are serious differences even between the way the basic microprocessors work, let alone differences between operating systems. Fortunately there are also similarities.

The 'standard' computer for many years has been the PC. Although 'personal computer' is a generic term, the initials PC have come to refer to Intel-based machines with the Windows operating system. Originally it was a disk operating system produced by Microsoft called DOS (for Disk Operating System) which defined the basic platform. The kind of screen available has been the subject of separate ‘standardization’ starting with screens designed for use with American television sets. Ironically, a screen resolution of 640 by 480 pixels (this is the active size of an NTSC television picture, of which more is said in Chapter 7) used to be one of the most standard things about computing systems, although this is now changing as higher screen resolutions have become commonplace.

PCs are made by a large number of companies, and it is this range of competing manufacturers that has led to the PCs becoming so inexpensive and widespread. Only Unix (especially its freely-available variant Linux) is as successful, but it remains successful in a niche in education, science and technology rather than on the business desktop. In multimedia, Unix variants such as Linux have most impact in web servers because many of the machines that run servers on wide area networks run one of these, and if you are producing CGI (Common Gateway Interface) programs for websites, then you might come across them.

With the introduction of Microsoft’s Windows the PC found itself a more friendly face, and Windows itself became the defining factor for the platform, which was vital, as the underlying microprocessors became more powerful and part of the Multimedia PC (MPC) standard.

Competition for Windows came from the Apple Macintosh, which had adopted a friendly windows (with a small w) approach from its inception. Only programmers drove a Mac from a command line; the users moved a pointer about and pressed virtual buttons on the screen. In some other niche areas there were companies such as Commodore (with the Amiga) and Acorn (whose Archimedes machine was firmly lodged in British education, but boasted the most bangs for the buck of any desktop machine of its day and was the world’s first RISC workstation).

Besides general-purpose computers there were games machines (from the likes of Sega, Nintendo, and Sony), short-lived home entertainment machines like CD-TV, 3-DO and CD-i, and set-top boxes for video-on-demand, Web-television and interactive television. As a multimedia developer your choice of offline platforms, for CD-ROMs and kiosks, encompasses these and more.

Criteria for offline choice

It would be nice to be able to say that, as a developer, you have the freedom to choose the platform best suited to deliver your multimedia vision. In fact the market is more likely to drive your choice, and often that points to whichever machine is prevalent in your target sector. Businesses have business machines, often not suited to entertainment techniques even if those techniques are appropriate.

You will have to research your sector and find a lowest common denominator for the machinery your customers have. This relates to factors such as:

- manufacturer and machine type;
- type and speed of processor (and therefore performance);
- amount of memory (RAM);
- size of hard disk (speed is less important but should not be forgotten);
- operating system (don't forget which version);
- CD-ROM/DVD-ROM drive (speed and capacity);
- access to online systems (local networks, Internet, World Wide Web, and so on);
- speed of network connection;
- resolution of the screen;
- number of colours on the screen;
- ability to handle moving video, and the multimedia architecture;
- sound handling (8- or 16-bit, mono or stereo, what compression?).

In some cases you may need to find out how often your users actually make use of their machine. This could be especially true in a business or training situation, where machines may be shared between people. It would be awkward for your users to spend half a day using your training package if someone else needed to use it every two hours to read the electronic mail.

Cross-platform chameleons

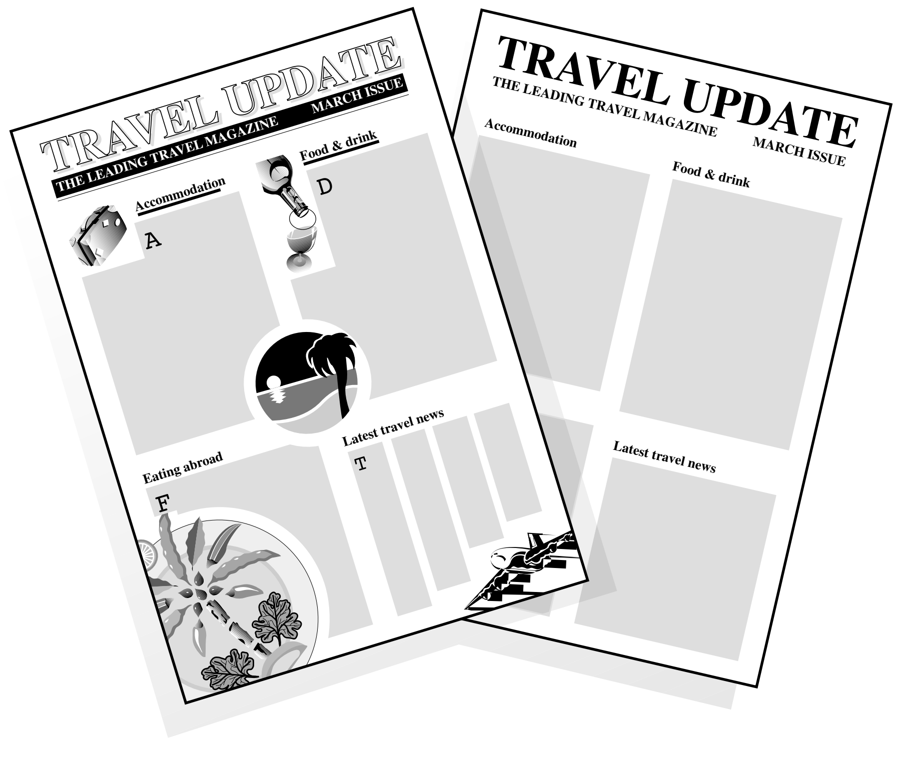
An alternative to choosing a single platform for an offline project is to produce the application for a number of platforms. This can be done by using an authoring system that produces versions of the application for several platforms. Macromedia Director is such a package, which will produce files that will run on Windows and on Apple from the same 'source'. At a lower level, there are libraries for graphical user interfaces, which can be used with C or C++ to run on different platforms with separate compilation. Director can also produce applications that will run with the Shockwave/Flash plug-in with a web browser.

It is possible for one computer platform to emulate another. A fast processor can run a program that appears, to the application, to be another platform entirely. The more powerful the platform, the more easily it can do this. There are Mac emulators for the Sun and Windows emulators for the Mac, where one machine pretends to be the other. It can even be possible (if rather strange) to run a program under emulation where the emulator is itself running under emulation.

A further refinement of this technique is the virtual machine. Here the application code runs in a specified environment. That environment is provided by a program, the virtual machine itself, that runs on the host machine. To run the application on a new platform you need only a new virtual machine. The ancestor of C, BCPL, ran in this way, and used a compiled intermediate code (called CINT code), which then ran on the virtual machine. This technique has a new lease of life through Java.

This technique needs fast processors, otherwise it is best suited to low-interactivity applications. This is because interpreted software (interpreted by the virtual machine) is likely to run more slowly than software compiled to run directly on the target machine. There is also the problem of the abilities and drawbacks of particular machines. A virtual machine has to have an audiovisual capability (since it is, to all intents and purposes, a platform in itself), and this will be the same as or less than the capabilities of its host.

If there is an incompatibility between the different platforms that can run the software then the software might have to run differently. A lesser platform may run the software in a less than optimum way. The way that the application (and possibly the virtual machine if there is one) copes with this is by degrading the performance of the application. Pictures may have fewer colours; movies may run more sluggishly. If this is done well, and possibly even invisibly to the user, it is called graceful degradation. Graceful degradation was inherent in the Web as it existed in the early days, and is relatively



Graceful degradation.

easily achieved today by providing versions of web pages for browsers with or without frames, with or without plug-ins, and so on. You add plug-ins, JavaScript or Java if you think that your Web pages need them and your viewers have the motivation, but you can be sure they will see something even without them. Offline multimedia has no such guarantee, and graceful degradation has to be programmed specifically rather than assumed.

A classic example of graceful degradation is illustrated in basic HTML. The browsers that display HTML documents have differing abilities, and it is recommended that any graphics used in documents are supplemented by words that will be displayed on browsers that cannot display pictures: although these are rare they are still around. In this way the document display gracefully degrades from graphics down to text only. If you are designing interactive applications that will run on networks you might have to account for extreme cases like this.

Delivery medium

We have already defined the delivery medium as the means by which you get your application to the user. If it is a Web page then the World Wide Web is that medium, and we shall come to that in a moment. Let's start by considering offline delivery.

Besides deciding on what computer platform the end-user will actually use to view your production, there is the question of how you will actually distribute the end result. To a certain extent this will depend on the size of the application, and might even have been specified up front. It is not unknown for small applications to be delivered on floppy disks, with a little help from compression.

Here are some of the options.

Floppy disk

Using a floppy has the advantage of using a standard medium, as you can assume most of your potential market has access to a floppy disk drive. However, the downsides are the capacity, the slow access times, and the relative difficulty of replicating in large numbers. Compressing the material on the disk means that it has to be decompressed onto the user's hard disk to run, but this also overcomes the slow speed of the floppy. It is, however, very easy to copy floppies to order if the quantities are small enough. Alternatively, small applications can be distributed easily using e-mail or the Web.

Compact disk and DVD

This medium has become so universal that it seems unlikely that it may ever lose its supremacy as a carrier. Currently it is splitting into an increasing number of different incarnations (Video CD, multi-session, CD-Plus, Photo-CD and the various formats of DVD). Back in 1984, when CD-ROM first appeared, it was touted as the data carrier to exceed all our requirements; but that was before digital video. DVD is a higher-density format than CDs; it makes use of a smaller physical structure on the disk and more layers of data to increase capacity, but otherwise it is effectively a 'turbo-charged' CD. Replication of CDs in quantity is very cheap (almost down to pennies), and the disks are very robust. Since a blank recordable CD now costs about a dollar (or less than a pound) each you can even consider 'burning to order' in small quantities and you can get self-contained machines which will copy CDs onto blanks in semi-bulk.

CD-ROMs usually adhere to ISO 9660, which specifies file structures. This is based on DOS but has extensions to cover other filing systems. Importantly, an ISO 9660 CD-ROM can be read on a large number of computers. Extensions to ISO 9660 have allowed longer file names and the ability to handle multi-session recordable CDs. Using ISO 9660 means that the computers can access the data on the CD: they cannot necessarily do anything with that data unless they have the appropriate application. DVD-ROM uses a system called universal disk format (UDF) to give multi-platform compatibility in the same way.

DVD-ROM disks (as distinct from DVD movie disks) are still relative newcomers and have been mostly used for large games and encyclopaedias but DVDs have either one or two sides and each side can have either one or two layers. A disk side can have two layers because the laser beam that reads the data on the disk can be focused at different depths in the transparent surface of the disk and so read two different layers. It is theoretically possible to produce a disk that has a CD-ROM upper layer with a DVD-ROM lower layer and so be backwards compatible to what is still a substantial installed user base.

CDs can be cheaply produced on the desktop for less than a dollar a time and the disks can be sent to pressing plants for replication. This is more difficult with DVDs since the recordable format is not practical for anything other than small DVDs. For movies the DLT tape format is usually used just as, once upon a time, Exabytes were standard for CD-ROMs.

Disk-based multimedia has been overshadowed by online systems in recent years. Now we also have the influence of DVD and interactive broadcasting and cable to consider, and this makes it uncertain as to what delivery platform will dominate. In practice, it is likely that evolution will find favour over revolution, and on- and offline systems will coexist. DVD-ROM evolves from CD-ROM, and broadcasting and the Internet will find that they have much in common. Added to this are mobile systems. Within the individual formats there will be evolutionary developments.

Hybrid delivery

Sometimes you can make use of both on- and offline delivery. Web links can be part of DVDs. Updating a CD or DVD from the Web allows a publisher to keep a product up to date between editions. Many encyclopaedias do this. Sometimes the application will update the offline data so that the updates are seamless; sometimes the application displays the new data in its own right as an addendum. This kind of application becomes less necessary as more users of the application are able to read all the data from the Internet.

Splitting data provides a good way of holding big assets such as movies but also allowing volatile information to be available from the website. This is easily achieved because web browsers will happily read files from hard disks or DVD/CD-ROM as long as the web pages are written with relative file paths or addresses.

Websites and server/browser balance

On a network, where there is software running on the user's local computer (the client) and it is working closely with software running on a server, the platform issues can become more complex. For the purposes of our discussion here we shall be dealing only with the Web, but sometimes similar issues can develop in any client–server application.

There are four software components in the web page chain: the page itself, the browser, any software that dynamically generates the pages (probably using a server-side include system like PHP or alternatively a separate gateway (CGI) program), and the server itself. The web page runs on the browser, and the dynamic page generation software runs on the server. HTML, Java, JavaScript and plug-ins are part of the page/browser combination.

When a web page is requested by a browser a set of HTTP (hypertext transfer protocol) requests are sent to the server. Besides the information about the requested page or asset, the browser will also pass information to the server telling it such things as the type of browser and platform, the referring page (where you clicked), the network address of the browser's machine, and possibly a string called the environment variable. All this information is available to any programming running on the server.

Sometimes you will need to decide whether a particular task is best handled by HTML or a server program. On the author's infrared photography site, Invisible Light at http://www.atsf.co.uk/ilight, there are a number of photographs that are displayed in individual web pages. When a viewer clicks on a link to display one of these pages a CGI program (written in Perl) is executed, which takes as its input the picture name that is passed in the HTTP request as a parameter. You will see that the URL requested has the extension .pl at the end rather than .html. The Perl builds the necessary HTML for a web page to display the photograph, reading the image size from the picture's JPEG file itself, configuring GIFs to form a frame around the image and incorporating a caption from another file. The resulting HTML is sent back to the browser exactly as if it had been a static file. In this way there is no need to have a separate HTML document for each page: the page is generated on demand. So the programming removes the need to have a lot of HTML documents on the site, and this is the essence of dynamic web design and a very simple example of a content management system. The Invisible Light software happens to be written in Perl but the same thing could be achieved by other means. Sometimes you will have a free hand in choosing what software to use. In this case Perl was used because the web space rented for this particular site supports it.

As has already been mentioned, a very large site with hundreds of pages or more will almost certainly have to be generated by a database in order to be maintained and updated – in other words, dynamic.

You should not confuse the size of the design task with the number of pages on a website. Database-driven websites may have hundreds of pages, but those pages will have a small number of different layout templates into which the database 'pours' the content. For example, in the casting directory on the Internet for the UK company *The Spotlight* there are over 20,000 possible pages available – one for each actor – but there are actually only three different Web page layouts used for the database: one for the search card, one for the list of hits resulting from a search, and one for the individual actor's information and photograph. A relational database generates all the pages based on HTML templates.

Another decision is whether a particular task can be carried out at the client side or the server side. For example, if you want to modify the web page according to the browser being used you can do this either by using JavaScript in the web page to dynamically rewrite the web page itself or by using a CGI program on the server. The advantage of JavaScript is that the load on the server is lighter, but the disadvantage is that the HTML is larger because of the JavaScript embedded in it and support of JavaScript is inconsistent between browsers and operating systems. The advantage of doing this using a server-side application is that the results appear to be normal HTML to the browser, and so the solution is more reliable. However, running a lot of CGI programs can be a problem if the site is heavily visited in which case one option is to separate the dynamic generation of the pages from the requests to view them. The CGI can be used to generate HTML files and save them on the server where they are accessed in the usual way. The pages might be generated at regular intervals using a timer or they might be generated every time the content changes. The latter seems more sensible but it is possible for you to have no real idea of when some of the content is changed since it might be being pulled in from an external source.

Sometimes the choice of where to generate the dynamic content is made for you. If you want the web page to reflect the time of day where the user is situated – remember they could be anywhere in the world – then only the browser knows this, and so client-side is your only option. Conversely, if you want to show visitors what number visitor they are then this has to be done at the server because only the server can keep a count of all visitors. There are also many web surfers who do not enable JavaScript or Java, or who are prohibited from using them on a business network.

CGI programs can be written in any language that runs on the server, and databases are a common form of program run using CGI. In practice, server owners may restrict you to using server includes such as those written using PHP or using interpreted languages such as Perl rather than running compiled C code because this dramatically reduces the risk of accidental damage to their server by your code. On the other hand, a compiled C program could be less prone to hacking.

Platforms for development and testing

So far, this chapter has dealt with the delivery of your application to the end-user: your customer. You will also be making choices about the platform or platforms you use to design and build your application or your web pages and to test it.

You do not have to develop your application on the delivery platform. This is especially true for a cross-platform offline application, since the usual practice would be to develop the application on one of the group of delivery platforms and test it on them all. Even low-level code can be produced on a different computer using a cross-compiler.

It is more likely that you will use one consistent platform for your asset creation and manipulation for every application. Even though the IBM PC 'standard' has been the most common delivery platform for multimedia and the majority of web surfers use PCs, many multimedia developers have used Apple Macintoshes for their asset work and have moved the assets across to their delivery platform during integration. The reason for this approach is that the best tools tended to appear first on the Macintosh and that, for the graphically minded, the Mac was already the platform of choice. The wider availability of tools and the more 'corporate' policies of larger development agencies means that this is much less the case than it was but you still hear managers saying that they'd like to 'wean the designers off their Macs'.

Moving assets between platforms needs to be done with some care and attention to quality and parameters. For example, screen gamma is different on different platforms: the same image looks lighter on a Mac than on a PC. This and other asset formats will need to be checked and tested on both platforms. If you are producing an application for more than one platform, especially if you store your assets in a single common format, this will be even more important.

In fact, it does not matter whether you are a Mac fan or a PC fan or a Linux fan or whatever. The point is that you can retain your platform of choice for asset creation and manipulation even if your client or market wants a particular application delivered on something else. The only limiting factor is that the asset creation platform must have a display that matches or exceeds that of the delivery platform. It is clearly no use whatsoever to try to do colour graphics on a black and white machine, or to use an 8-bit audio system to produce sound for a system with CD quality 16-bit sound.

Similarly, you should create and manipulate your assets in the highest convenient standard and convert down, if necessary, at the last moment. This will not only give you the option of porting the assets to other delivery platforms if required, but will actually help you to keep the quality as high as possible.

For online authoring you can use almost any computer since an HTML page is simply a text file. But just as any browser should be able to read the web pages, so you have a choice of the ways you author them. Some people like to handcraft the HTML in a text editor or word processor, while others use one or more of the many WYSIWYG packages for web page layout. The production of web page assets is very much like offline work. The same packages can be used to produce the images, sounds and movies, and even animations and interactive mini-applications. Again, you can choose to produce your HTML and assets on whatever platform you can use.

An exception to this is production of websites with JavaScript. The vagaries of JavaScript implementation mean that you need to be able to test for both main browsers (plus others you want to support) on both platforms. This problem has been so acute that many developers are refusing to guarantee complete cross-platform compatibility of sites with JavaScript. The great thing about standards is that you have so many to choose from!

It is certainly true that your design platforms have to include your delivery platform so that you can, at the very least, test the performance and carry out debugging. This can mean that you have to have access to every possible configuration of platform that your customers will have. This is no trivial task. With a website you can test with a validator to check that your code is correct, but you should also look at the pages on as many combinations of browser and computer as you can. (There's more on this in the *Testing* chapter in Book 1 Chapter 11.)

With an offline application the problem can be significantly worse because you are likely to make more direct use of the computer's facilities. To show the magnitude of this problem, let's consider variations of a computer:

- different models
- different amounts of RAM
- different screen resolutions and numbers of screens
- different sizes of hard disk
- common extensions to the basic system
- different versions of the system software
- different CD-ROM drives
- different network configurations.

With that list, and assuming five possible options for each category, the number of possible configurations is $5 \times 5 \times 5 \times 5 \times 5 \times 5 \times 5 \times 5$, which is 390 625. This is not really practical, and you will test for the most likely problem areas such as system software versions and amounts of RAM.

Author once and deliver everywhere

There are delivery platforms for convergent media that seem as similar as the proverbial chalk and cheese. We either have to prepare to produce different applications or different formats of web pages for each of them or we need techniques that allow us to author once and deliver on many platforms. XML, which is discussed in Chapter 2, *The Internet*, is one part of the solution to this: it allows the content structure to be separated from the way it is laid out, so that the layout can be changed depending on the delivery platform.

Fortunately the range of platforms available to us is increasing just as techniques are developed to cope with them. Eventually content and display will be so separated that we won't even worry about the delivery platform, in the same way that a television programme maker does not need to worry about the type of receiver the audience is using.

THEORY INTO PRACTICE 5

Experience is the best teacher on moving applications from one delivery platform to another. You can learn a lot by talking to people you know who have direct experience of this and by discussing the detailed problems they may have encountered.

For online, set up as many computer/browser configurations as you can and look at some websites with them to compare the results. If you have PCs, Macs and Web-capable televisions you should also compare the brightness of the screen images.

For offline, look at a multimedia application that is available on more than one platform and compare the versions.

Summary

- The choice of delivery platform will usually be decided by either your client or your target market.
- You do not have to do all your development on the delivery platform, especially when it comes to working with your assets.
- Each delivery platform has its strengths, weaknesses and special considerations that you need to take into account.
- It is possible to develop an application that will work on more than one platform, but moving from one platform to another can lead to changes in performance, and you should be aware of what is likely to happen.
- When planning web gateway programs you should consider the balance between the browser and the server.

Recommended reading

Siegel D. (1997). *Creating Killer Web Sites*, 2nd edn. Indianapolis, IN: Hayden Books

Vaughan T. (2001). *Multimedia: Making it Work*, 5th edn, Berkeley, CA: Osborne McGraw-Hill

MHEG was initially an initiative of CCETT in France and information on it can be found on their website at
http://www.ccett.fr/mheg/overview.htm
and the Digital Television Group have reference material on use of MHEG in television in the reference section of their website at
http://www.dtg.org.uk/reference