Getting To Grips With Palm

by Jim Cooper

This is the sort of chance egomaniacs would die for: able to pontificate at length on a favourite subject, with no hecklers or interruptions. Lucky for all of us that I'm not like that, eh? However, I will admit that this is a personal and biased view of the Personal Digital Assistant (PDA) market and absolutely does reflect my own experience and blind prejudices, where appropriate.

We'll start by looking at PDAs in general, then have a quick look at the major operating systems, before going on to see what's involved in programming the little devils, and what our Delphi experience can bring to the party.

I suppose most people are aware of the recent slowdown in PC sales, which is one reason why many hardware manufacturers are so interested in the way PDA sales are taking off. People are starting to realise the potential of small computing devices, especially when they are combined with wireless and mobile phone technologies. This is also why the mobile phone manufacturers are so interested, at a time when many of their major markets are becoming saturated too. As we'll see, I'm not convinced everyone understands these devices and their uses.

Just so we're all reading from the same page (Geddit? You're reading a magazine and all of you are looking at the same page?), let's define what distinguishes a PDA from other computing devices. Most importantly, they're small and light enough to carry everywhere. Secondly, they can have additional third-party software installed, unlike electronic organisers. Data can easily be synchronised with a PC or server. They usually have very low power consumption, perhaps going weeks between battery charges or changes. They don't have a keyboard, but instead rely on a pen or stylus and handwriting recognition for input. They can be turned on and off instantly, with no lengthy boot process. As a consequence of the size and power restrictions, PDAs usually have lower speed processors, as well as limited memory and other storage. However, this can often be augmented with hardware addons, which might also include modems, keyboards, digital cameras, GPS systems, and so on.

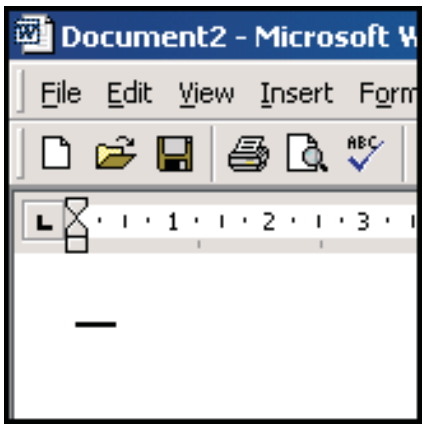
PDAs are the smallest general purpose computing devices. They are often grouped with Palmtop computers, like some HP Windows CE machines, and the Psion range. These machines typically have larger screens, small keyboards and a much shorter battery life. The next larger machines are subnotebooks running WinCE, or one of the Windows variants, and then full size notebooks. So while Psions are often compared with Palm devices, for instance, it's like comparing apples and oranges. Or maybe mandarins and grapefruit.

There are other options. Nokia and Kyocera, to name but two, make phones with an built-in PDA, for instance. It is also possible to go in the opposite direction, and get plug-in GSM phone modules for PDAs (Omnisky is running a UK trial of its Palm wireless service at the time of writing). It seems inevitable that mobile phone, wireless LAN and Bluetooth technologies will combine with PDAs at some point, but it is not yet clear how. It may even be different in different countries. In the US, for example, you can use a Palm VII for unlimited wireless internet access for a fixed monthly fee. It appears likely that in the UK mobile phone call charges will still apply. Combined with the slow progress in 3G mobile phone networks and working Bluetooth appliances, it had been suggested that putting free-to-access wireless LANs in public areas like airports, shops and so on will actually be more useful in the short term.

Palm It Off

The two main features of PDAs are both strengths and weaknesses. PDAs are small, so they're easily portable and usable in situations where larger computers are not. For instance, you can buy sealable plastic bags that will snugly contain a Palm and allow it to be used underwater by divers. However,

➤ *Figure 1*



About Perspectives

The 'occasional' Perspectives column aims to give readers a wider perspective than just Delphi and Kylix, allowing us to keep an ear open to other development technologies, with input from a variety of contributors. Do you have an idea for something we should cover? Email me at chrisf@itecuk.com.

Chris Frizelle, Editor

this also means that the amount of data that can be displayed is tiny. Most Palms have a 160x160 pixel screen, and PocketPC devices are only a little larger at 240x320. Figure 1 is a screenshot of a Palm size screen with Word running.

So you must be careful not to waste valuable screen real estate. You should only ever display what is absolutely necessary at any given moment. It is for this reason that the PalmOS does not support disabled controls, and that menu bars only appear when a button is pressed. Because so little can be displayed, having 5,000 customer records on a PDA might be possible, but the application needs to make finding the right one easy. Scrolling through a list showing 12 to14 at a time is going to be tedious.

On the other hand, collecting information on 5,000 customers and uploading that data to a PC or server database may be an appropriate use of the technology. This depends on how easy it is to enter the data, which brings us to the other feature that has both good and bad points: no keyboard. The device can be smaller, but data entry is slower, even though handwriting recognition is pretty good on most PDAs. On Palms it is necessary to learn Graffiti, a standard way of writing that uses unique strokes for each character, rather than the PDA recognising your handwriting. You can also pop up keyboards on the screen, and tap to enter letters, numbers and punctuation. Whatever method is used, though, is generally slower than using a full-sized keyboard, and is therefore not suited to high volume data input. You could write a novel on a PDA, but it would be arduous. For large amounts of input, other methods are needed. These include collecting input that only involves tapping on the screen with little or no writing, and using bar code scanners or portable keyboards. You might also be able to use plug-in digital cameras or voice recorders.

The point is that it's wrong to think of PDAs as very small PCs. They have restrictions that can seem quite severe to anyone used to the power of a desktop or laptop computer. However, used appropriately, they can be amazingly useful devices.

A Palm In The Hand Is Worth Two PCs In The Pocket

Currently the market is dominated by PalmOS devices, with about 85% or so of the market, with PocketPC (the handheld version of WinCE) and EPOC (Psion) devices making up most of the remainder. I'm not going to talk about Psions any further, because their market share is tiny, and getting tinier, and Psion hasn't released new machines for some time. Despite all the talk of alliances with phone manufacturers, few products have been forthcoming, and it seems unlikely to me that Psion can ever get back the market it lost. I'm also not going to talk about Linux devices. Maybe things like the Agenda and Samsung Yopy will take over the world, but not for a while yet.

For the remainder of the article I'll use the terms 'Palms' or 'Palm devices' to refer to anything running PalmOS, such as Palm branded machines, Handspring Visors, Sony Clies and so on. I will similarly refer to 'PocketPCs' generically.

PalmOS is the most mature platform. Palm has succeeded because the company recognised the essential qualities of truly portable devices right from the start. Palms are small, light, responsive devices. They can be turned on and off at any time without data loss, and without waiting for the machine to reboot. The user interface is miserly with screen real estate, and the OS does not include unnecessary frills like multimedia support. Programs are tiny, by PC standards, usually weighing in at less than 64Kb. With version 3 of WindowsCE, Microsoft is finally starting to learn the same lessons, with the result that the latest devices are comparable to Palms. It is even possible to transfer data between both types of machine.

Comparing hardware specifications across platforms is difficult. A Palm may seem underspecified with a 20MHz processor and 8Mb of RAM, compared with a 233MHz processor and 32Mb RAM in a typical PocketPC device. However, the PalmOS uses hardly any of that memory when it's running (a few kilobytes as opposed to megabytes for PocketPC), is far more lightweight, and the applications are mostly orders of magnitude smaller. The faster processors and larger memories of PocketPC machines are necessary to achieve the same perceived speed as a Palm. In typical operation, users will not notice much difference.

Each OS has its own strengths and weaknesses that you should be aware of. PalmOS strengths include the fact that it's the most mature platform, and the hardware is generally cheaper, particularly the Visor range and the new Palm m100 series. This is largely due to the use of cheaper processors, smaller amounts of RAM, and monochrome screens in most models, and means the price differential is likely to continue. Synchronisation of data with Windows, Macintosh and Linux systems is very easy, using either existing or custom solutions. Application development can also be done on Windows, Macintosh or Linux, and there is a greater choice of development tools and languages. Many people don't realise that there are in fact more hardware manufacturers by far for the PalmOS than for PocketPC, and Hewlett Packard is considering switching to either PalmOS or Linux as I write. Finally, there is an enormous range of programs, mostly cheap or free, plus a large and rapidly growing set of hardware addons.

The first major weakness is that memory is limited. Currently 8Mb is the most available out of the box. While other solutions are becoming available (Compact Flash and SD cards, for instance), this means that the price advantage over PocketPC devices is eroded. Although this is not usually a problem for storing applications (I have a Palm III with 2Mb of RAM and I've never come anywhere near to filling it), it may well be a problem for data storage. The second problem

is that the processor is slow, so number crunching is better suited to the more powerful PocketPC processors. Usually Palm applications offload intensive processing to a PC, and then retrieve the data from there. This is not always the most desirable option and adds to the application complexity. Also, the screen size is almost always smaller than on PocketPC devices, although the latest model announced by HandEra (the manufacturer previously known as TRG) has the same 240x320 pixel resolution. Some may consider that monochrome screens on most models is another drawback, but this does contribute to lower cost, less weight and longer battery life, and colour doesn't add that much to the experience of using a Palm. Having said that, I must confess my next Palm will be a shiny new colour m505, but my reasoning is that in my line of work I need to be familiar with the latest version of PalmOS, and the implications of supporting colour devices. That's my flimsy and transparent justification, and I'm sticking to it.

In the other camp, the development tools from Microsoft are free, but there isn't much choice if you don't like them. You also need to compile different versions of your program for devices with different processors (which is unnecessary for the PalmOS) and you are limited to developing on Windows. The larger screens (usually colour) and faster processors mean the devices are bigger and heavier and can have very short battery lives. A recent *PC Magazine* test showed that a Handspring Visor Deluxe had 12 times the battery life of a Compaq iPaq. If you need to do a lot of number crunching, or store large blocks of data, this might sway you towards Microsoft's offering. However, there is a lot of excess baggage in the PocketPC operating system. How many people really need to show video clips on a PDA? In any case you can do that on both platforms. Bear in mind what we discussed earlier: a PDA is not a small laptop, and applications that treat it as one are likely to be unsuccessful.

I hope my prejudices are showing by now. I would recommend that 90% of the time you should be writing for PalmOS; however, you should give Microsoft some encouragement, because it performed the invaluable service of giving Palm a great kick up the behind when it seemed to be getting very complacent.

Sweaty Palms

By now some of you will be wondering whether you can program either Palm or PocketPC devices in Delphi. The short answer is no. The longer answer is yes. Before I start sounding too Charles Dickens here I had better explain. There are two tasks involved in programming for either platform. One is to write the actual application that will run on the handheld of your choice. Since you are targeting different operating systems and different processors, Delphi can't do that. The other task is to write the software that synchronises data on the handheld with that in a PC or server database. This you can do in Delphi, because you're writing Windows code.

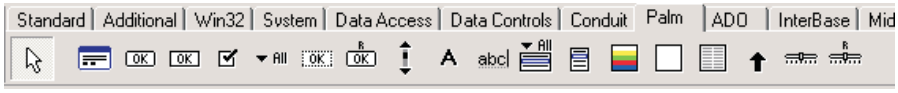
From now on, I'm only going to talk about Palm programming, because it's the most important platform, it's where I have most experience, and space is limited.

Palm calls the synchronising process a *HotSync*, it is controlled by the HotSync Manager (HSM). When a user presses the `HotSync` button on the cradle connected to the PC, the HSM starts and checks to see what *conduits* have been registered. There are a couple of system conduits that install new software, and backup databases which have no other conduit, but usually each conduit is responsible for synchronising the data in a Palm application with that in a PC application's database. It will determine what happens when, say, both the PC and Palm records have been modified. Will one overwrite the other, or will a duplicate record be created at each end with the new data in it? There are many other possibilities that need to be covered: deleting on the PC and modifying the same record on the Palm, for example. The situation is slightly complicated by the fact that Palm records can usually be marked as *archived*, meaning that they should be removed from the Palm, but not from the PC. These records should not reappear on the Palm during later syncs, either.

In addition, Palm databases are not databases in the sense that you're probably used to. Unless you provide it, they have no metadata, no fields or columns, and no querying or filtering is available. A Palm database consists essentially of a list of variable length records that are each just a big blob of bytes. How you interpret those bytes is up to you (obviously the conduit and the Palm application must interpret them the same way). It is perfectly possible to have completely different things in each record: a bitmap in one, three strings in the next, and so on. Another factor is that while each Palm record has a unique identifier called a *local id* within each database, these are assigned

➤ *Table 1: Palm Tools.*

Palm Conduit Development Kit For C/C++, Java and ActiveX	www.palmos.com	Free
EHAND Palm Access Active X	www.ehand.com	Commercial, runtime licence may be required
ConduitDB (BDE based solution)	www.envicon.de	Commercial software
Paul Gilbert's components (VCL components)	www.whyzero.com/~paul/	Freeware with source
TurboSync (VCL components)	www.tabdee.ltd.uk	Commercial



➤ *Figure 2*

by PalmOS, which means that you need to be careful matching Palm records to PC records if there are multiple users of the PC database.

Fortunately there are several tools available to help Delphi programmers deal with these issues. One thing you will definitely need is the Conduit Development Kit (CDK) from Palm's official developer website (www.palmos.com). The documentation is invaluable even if you never directly call any of the APIs. It also contains some free ActiveX controls that should work in Delphi. At the time of writing (v4.0.1) they don't work under Windows 2000, though. EHAND also makes ActiveX controls that work within Delphi. For native Delphi components, you should try my company, Paul Gilbert and Envicon. All the URLs are in Table 1. If you're in need of a conduit, check them out and see which one suits your needs best.

Hairy Palms

So what about writing a Palm application? There are a plethora of tools and languages available, from the free gcc C/C++ compiler, the very cheap PDA Toolbox, through to the very expensive enterprise priced tools like ScoutBuilder. The 'standard' Palm development environment is CodeWarrior from Metrowerks. Although other versions of CodeWarrior support Pascal, for Palm development you are stuck with C/C++. Metrowerks has to be guilty of resting on its laurels, too, as CodeWarrior has barely changed in years, and isn't all that friendly to use (think Visual C++ and you'll get the idea). Other languages vary from SmallTalk, Forth, Python, and various dialects of Basic, to proprietary scripting languages.

Good starting places if you're looking for a programming tool are the Palm developers' website (also good for documentation) and PalmGear (www.palmgear.com).

There are a few things to bear in mind during your search. The first is cost. Many of the tools are free or very cheap, and often there are trial versions available, even of the very expensive tools. Keep in mind that some tools have runtime licensing fees. The next thing to consider is my pet hate. Most of the programming tools require a runtime library on the Palm. This really bugs me, as I can foresee the day when some poor user has a device full of runtimes and no room for data. The worst offender has to be Java. The KVM is huge, by Palm standards, and it doesn't work very well into the bargain. For corporate work, runtime libraries may not be a big deal if you're in control of what gets loaded onto the handhelds. However, the other drawback is that having a runtime usually means you can only get access to those Palm APIs that have been included somehow. Some tools allow you to extend the runtime libraries using C/C++, but still, you should check that your chosen development tool supports all the things you want to do. Another option you may be able to consider is not to write an application at all, but to use one of the third-party database tools like HanDBase or MobileDB. Again, have a search through PalmGear for all the options. The upside of all these sorts of tools is that developing applications is usually much faster than writing in C/C++.

Palm Delphi

So far, the only mainstream tools that allow access to the full API and compile small, standalone, native applications are the C/C++ compilers. They are very soon to be joined by a more interesting option for Delphi developers: PocketStudio. This is a Pascal compiler and IDE that will do anything that CodeWarrior will, and generate applications of similar size and performance. The current beta version even integrates into the Delphi IDE, and

uses Delphi's forms designer and Object Inspector to design Palm screens. Figure 2 has a shot of the page for the Palm controls.

This may not seem many, but the PalmOS doesn't have a lot of controls, and some of them have confusing names. The equivalent of an edit box is called a *field*, and a grid is called a *table*. A programmer-defined user interface element is called a *gadget*, and an example is the view of your appointments by week or month. If you've never used a Palm then you have no idea what I mean, but fortunately there's a free way to try one out. Palm provides the Palm OS Emulator (POSE) on its developers' website. You need a ROM image to run it, that is, a copy of the Palm memory that contains the operating system, but you can download one from a real device, or join the developer program and get access to as many as you like. If I choose POSE as the host application under the Delphi `Run|Parameters` menu, and set a couple of parameters, I can hit `F9` and get my PocketStudio application compiled, loaded into POSE and running ready to test. POSE behaves exactly as a real device. All the hardware buttons work, you can HotSync with it (without needing any cables) so you can test conduits too, you can try different versions of the operating system, and different hardware models and memory configurations. It has various profiling and debugging aids, including *gremlins*, which are a series of simulated stylus taps all over the screen (biased towards actual controls) that you can set running, and will produce a report of the bugs found. POSE works with applications developed with any Palm programming tool.

The final point I'd like to make about PocketStudio is that it is not currently the equivalent of Delphi for the Palm, more of a Turbo-Pascal. Anyone who programmed old-style Windows applications, where you had to write your own event loops and use handle and memory locking, will recognise the equivalent features in a PocketStudio program. The code is much like a C program written in Pascal. You can see examples at www.pocket-technologies.com. This may sound a bit scary, but it actually isn't that hard, as there's a high degree of code reuse possible, and Palm programs are usually very small. The Palm APIs are well documented and organised. In any case, Pocket Technologies has a number of enhancements planned and in development that will turn PocketStudio into the sort of tool we're used to. Even as things stand, it's my Palm programming tool of choice. The latest estimate I have for a release date is sometime in July 2001.

Parmesan Cheese

To top off the article (*now* you understand the section heading, eh?) we'll look at wireless communications. This threatens to be the next big technology, and it may well be, but it could all too easily turn out to be the next WAP.

Surfing the net wirelessly on your handheld is certainly more pleasant than on a mobile phone, but you can still only see a limited, often useless, portion of a page, and graphics can download at a glacial pace. It still remains to be seen how well web pages and other software will be designed for PDAs. Having said that, some applications seem natural fits. Something like a Symbol SPT1740, with built-in barcode scanner and wireless LAN card, talking directly to your company database, would be a real boon during a stocktake, for instance. It's here that the small screen size may actually be a benefit, because not much information needs to be sent to the handheld at a time. ASTA has a new website (www.astawireless.com) specially devoted to writing these sorts of applications using shared libraries it has developed for use with several of the major Palm development tools. PocketStudio will be supported as soon as it's released. The demo application is surprisingly fast, and allows practically any sort of access to an ASTA server sitting on the other end of any TCP/IP connection, including sending SQL queries, and finding out what time it is in Boise, Idaho. I tried it out using POSE and a mobile phone connected to my laptop's serial port. At only 9,600 baud it ran at a more than useful speed over an internet connection, easily fast enough for your killer app to be written now, without needing those long-awaited 3G mobile phone networks. It gives you almost too much power, as you still need to remember that your application needs to be designed to make the 100,000 records in your database useful. You need to heavily constrain the result sets without making the program seem too clunky to use. Design is everything!

Lastly, I should also mention that you can use Delphi to develop HTML, WML or PQA (Palm's version of web clipping) pages that you can access via a Palm browser.

Palm Reading

We've come to the end of our brief sojourn in the world of wee dinky computers. There are loads of tools and resources that I haven't mentioned, and the number seems to be growing weekly. There are also quite a lot of good Palm programming books available now, and I've listed a few below, but again, the list is not exhaustive. We all know Delphi is a great tool for developing big applications, but it can also be put to excellent use creating little applications on small platforms.

Bibliography

Rhodes, Neil and McKeehan, Julie, *Palm Programming, The Developer's Guide*, O'Reilly & Associates, 1999. ISBN 1-56592-525-4.

Foster, Lonnon R, *Palm OS Programming Bible*, IDG Books Worldwide, 2000. ISBN 0-7645-4676-7.

Mann, Steve and Rischpater, Ray, *Advanced Palm Programming*, John Wiley and Sons, 2001. ISBN 0-471-39087-9.

Jim Cooper works for Tabdee Ltd, Reading, UK. You can contact him at jcooper@tabdee.ltd.uk