Common questions about dviwin

Hippocrates Sendoukas
University of Southern California
Dept. of Finance & Bus. Economics
March 15, 1993

This document attempts to answer the most common questions that I received about dviwin; please read this file about any problems, and if you cannot find the answer here, do not hesitate to contact me. My e-mail address is "sendouk@scf.usc.edu"

When I open a dvi file I get only an empty page. Why does this happen?

This indicates that the program cannot find any fonts; it usually happens if you did not install any fonts in your hard disk or the base directory specification is incorrect. If you are not sure about setting up the fonts, please read the manual. If you select the "Log" entry in the main menu, you will see the log file where the program indicates which fonts it cannot find. It is always a good idea to check the log file whenever something goes wrong. If you want to know exactly which font files the program is attempting to use, check the "Trace Fonts" entry in the "Options" submenu *before* opening the dvi file. *If you are upgrading from version 2.0, make sure that you change the font directory specification*. If for example your base directory is "c:\tex\fonts", you should change it to "c:\tex\fonts\$r". I am sorry for this incompatibility, but the new method is much better.

The program finds the fonts, but the letters are too big and on top of each other.

This happens when you try to preview at a screen resolution (eg., 100dpi), but you only have fonts for the printer (eg., 300dpi). If dviwin cannot find the correct fonts at 100dpi, it tries all neighboring resolutions until it finds a match. If the matching font is much larger than the requested font, you will end up with overlapping characters, because the positioning is still based at 100dpi. You can find a decent set of screen fonts in the files "dvivga2.zip" through "dvivga8.zip" in the directory pd1:<msdos.tex> of wsmr.simtel-20.army.mil, or the directory pub/msdos/tex of oak.oakland.edu. Unlike the first release of the program, the new version looks only for fonts within three magsteps of the original font size, so if you preview at normal screen resolutions but you have only printer fonts, nothing will be displayed.

I am still unclear about the font setup. Can you give an example?

My base directory is "c:\usr\texfonts". Under this directory I have subdirectories called "70", "76", "84", "92", "100" and so on. Each of these subdirectories contains a full set of PK fonts for the particular resolution. Therefore, when dviwin looks for cmr10 at 100dpi, it should look for the file "c:\usr\texfonts\100\cmr10.pk"; this is accomplished by telling dviwin that the font directory is "c:\usr\texfonts\$r". The same specification will also work if you want to use "fli" files in the base directory (c:\usr\texfonts).

I am using another dvi driver and the names of the subdirectories are c:\tex\pkfonts\100dpi, c:\tex\pkfonts\110dpi, etc. How can I instruct dviwin to use these directories?

Specify the font directory as "c:\tex\pkfonts\$rdpi" and everything will be fine.

The program cannot find the fonts lcircle10 and lcirclew10. Why does this happen?

These names exceed the DOS limit of 8 characters. If you try to create such files under DOS, the names will be truncated to "lcircle1" and "lcirclew" respectively. Dviwin will find these files in the PK format, but you may encounter problems if you store them in a font library without any name remapping. The easiest way around this problem is to add the following lines to the font substitution file:

```
lcircle10       ->      lcircle1
lcirclew10      ->      lcirclew
```

If you use the dvivga fonts from the simtel archives, you will find that they have renamed these two fonts to "circle10" and "circlew1" respectively. In that case, change the two substitution lines as following:

```
lcircle10       ->      circle10
lcirclew10      ->      circlew1
```

The file "dviwin.sub" already contains the last two substitutions. so you don't need to change anything if you use the dvivga fonts.

The program cannot find some fonts even though they are installed on my disk.

If the font directories are specified correctly, this may indicate that there are not enough file handles in the system. Make sure that you use a value of at least 50 for the FILES statement in your config.sys file.

I have installed the "share.exe" program and I am getting sharing violations. Why does this happen?

The basic idea behind share.exe is to arbitrate file access among different processes. Unfortunately, the implementation is not that great and sometimes it complains even if two or more processes try to read the same file (this is wrong: it should complain only if a program tries to read a file while another is writing it). You can placate "share.exe" by setting the attributes of the files in question to "Read Only".

I have PK fonts at 118dpi and I don't want to waste space by installing new fonts at similar resolutions. Is there a way to use my fonts?

The first version of dviwin could use only the resolutions appearing in the "Size" submenu. The new version lets you add two custom resolutions to the submenu, so you can use any fonts that you like. You can select the entry "Custom Sizes" from the "Options" submenu, and you will see a dialog box where you will specify the resolutions that you want. As soon as you do this, you can select any of these resolutions from the "Size" submenu.

Why do you use these particular font resolutions? they do not follow the standard geometric progression exactly. What if I want to use the proper progression?

I decided to use these fonts in order to remain compatible with other drivers from the Beebe family, and because the appropriate fonts are already at simtel, so anybody can download them. It would be an enormous waste of space and bandwidth if I required everybody to obtain new

fonts. The progression is sometimes off by 1dpi, but this does not really affect the viewing or printing quality. There is a workaround though: if you use the custom sizes, the program does not make any assumptions about your fonts, so it uses the standard geometric progression; suppose for example that you use the *built-in* resolution of 100dpi: if you need to load a font at magstep1, the program will look for the font at 121dpi (instead of 120dpi) for compatibility purposes. If on the other hand you use a *custom* size of 100dpi, the program will look for 120dpi when it needs a font at magstep1. This method should give you the most flexibility.

The startup time seems a bit longer than before. Why does this happen?

When you start the program, it looks in the base directory for font libraries. If it finds any, it makes a list of all fonts in all libraries. This information is used when searching for fonts for your documents; in essence, you spend about one extra second upon startup but you save more time, because there will be less disk accesses during the processing of your documents. If you don't use any font libraries, the startup time should be a bit shorter because it does not allocate as much memory as before.

Why do you open the font libraries more than once?

The program uses a separate file handle for each font even though they may be in the same font library. This lets it exploit the file buffering better and speeds up the font reading somewhat.

The printer output is extremely small. What is the problem?

The problem was that dviwin would send the bitmap to the printer at the *current* resolution. Therefore, you needed to switch to the appropriate resolution for the printer before issuing the "Print" command; this was done for flexibility purposes. Since however most people are annoyed by it, I decided to use another method: the "Print" dialog contains a combo-box that lets you select the resolution used for the printer. This value defaults to "Auto" which means that the program will use a resolution which is as close as possible to the printer's resolution. This will work even when you change the default printer; it is also independent from the previewing resolution. Therefore, you will not need to do anything under normal circumstances. If however you want to produce magnified or reduced output, you can still do it with minimal fuss.

When I try to print on a laser printer, the output is weird: the text comes out at the proper size, but it does not fill the page and there are some extraneous characters.

This is almost certainly due to insufficient memory in the laser printer. You will need about 1M of RAM at 300dpi and 4M of RAM at 600dpi. This behavior is typical of HP printers; Apple printers reset and print a test page when the memory limits are exceeded.

The program is too slow when printing. Are there any improvements in this version?

The lack of speed was due to two problems: the first one was that the program was using too much memory and Windows would swap excessively if you had 4M of RAM or less. The memory requirements of the new release have been drastically reduced and it can handle even 400dpi resolutions without significant swapping on 4M machines (you will need at least 6M of RAM for 600dpi printing). The second problem is that the program sends a bitmap to the printer; this

method can be slow on high resolution printers; I made some optimizations, but the bottleneck is the interface of the computer to the printer. I run some tests on a 386/25 machine with 4M of RAM; when pressing the PgDn key at 300dpi, dviwin 2.0 needed 15 seconds to display the next page; the new version needs 2.5 seconds; this difference is due only to the reduced memory requirements and the lack of swapping. When however you compare the printing speed between the two versions, the difference is much smaller, because of the computer-printer bottleneck. I will work more on this, but I don't see any obvious way to improve it.

Is there a way to print two TeX pages on a single physical page?

The easiest way to accomplish this task is to use the program "dvidvi" (or dvi2dvi) in conjunction with the driver. There are two common page orderings: the first one is to print pages [1,2] on the first physical page, [3,4] on the second, [5,6] on the third and so on. This can be used for program listings. The second useful page ordering is to shuffle the pages for making a booklet; if there are N pages in the document (and N is even), you will want pages [N,1] on the first physical page, [2,N-1] on the second, [N-2,3] on the third and so on. Then you can just put staples in the middle, fold the paper and the booklet will be ready. In both situations you will also want to print the odd physical pages in one pass and the even ones in a second pass; in this way, you can re-insert the paper from the first pass to the printer to get double sided printing (keep in mind that some printers are not really good at this, so you may need to do it with a copier). Suppose now that the input file is "xyz.dvi" and you want to use the first ordering (for program listings). You can issue the commands:

```
dvidvi  4:0,1(8.5in,0in)  xyz.dvi  xyz1.dvi
dvidvi  4:2,3(8.5in,0in)  xyz.dvi  xyz2.dvi
```

These commands will produce the files "xyz1.dvi" and "xyz2.dvi" which contain the reordered pages. If on the other hand you want to use the second ordering (for booklets), you can issue the commands:

```
dvidvi  4:-3,0(8.5in,0in)  xyz.dvi  xyz1.dvi
dvidvi  4:-1,2(8.5in,0in)  xyz.dvi  xyz2.dvi
```

You will again get the files "xyz1.dvi" and "xyz2.dvi" but the page ordering will be different. For A4 paper, use the dimension "210mm" instead of "8.5in" in all the above commands. Now start dviwin and select a Ledger page size (17in by 11in); for European paper, select the A3' size (420mm by 297mm: it is the transpose of A3). There are two ways to proceed from here: if the printer can handle such a large paper size, just print it at the appropriate resolution and reduce the output with a photocopier; this will give you great results because the effective resolution will be higher. If however the printer cannot handle such paper, we can reduce the output with dviwin; from the "Print" dialog, select a resolution between 65% and 75% of the actual printer resolution. In the printer "Setup" dialog, select Landscape printing, and in the Alignment dialog select automatic page positioning. If for example you use a laser printer capable of handling only Letter or A4 paper at 300dpi, you will probably use a resolution of 208dpi (you can go to 228dpi if you use a wide margin format such as LaTeX). This arrangement will let you print the file in the desired manner.

Up to this point, we assumed that the original dvi file was intended for normal printing; this has the advantage that you do not need to change anything in the TeX file. If however you are willing to modify the TeX file for this printing method, you can make the text a bit narrower and taller to fit as much of the page as possible; you can also experiment with the parameters of dvidvi. You can find dvidvi in the directory pub/os2/all/unix/tex of ftp-os2.nmsu.edu; dvi2dvi can be found in the

directory pd1:<msdos.tex> of wsmr-simtel20.army.mil or pub/msdos/tex of oak.oakland.edu.

The documentation states that dviwin supports devices with square pixels; I have a dot matrix printer with non-square pixels. Can I use dviwin to print on this printer?

I use square pixels for simplicity because most screens and higher resolution printers use them. You have two options when using a device with non-square pixels: the first and easiest approach is to use an approximate resolution (between the horizontal and vertical resolutions of your device) and select the automatic option from the Printer Alignment dialog box; the resulting output will not be correct, but it will be fine for a quick draft. If however you want to get the best possible output from a dot matrix printer, you can follow these steps:

1. Get the appropriate PK fonts (or font library) for the desired printing mode.
2. Add a custom resolution corresponding to the horizontal resolution of your printer.
3. Select manual alignment from the Printer Alignment dialog box. At this point you have to enter the appropriate horizontal and vertical offsets; I would suggest the following values for the three most common printer modes (with non-square pixels):

Printer Mode	Horizontal	Vertical
120x144 (9-pin)	30	72
240x144 (9-pin)	60	72
360x180 (24-pin)	90	90

You may need to fine tune the above values, but they should be quite close to the optimal. The vertical resolution of the first mode is higher than the horizontal, so you may also want to tell dviwin that the paper is a bit taller (use for example Legal instead of Letter size). I understand that the above procedure is a bit involved, but you need to do it only once, and only if you want to use a dot matrix printer for final output.

When I preview a document, I cannot see the entire page; I see instead the top part repeated many times. Is this a bug in dviwin?

This is a bug in the video driver; it cannot handle bitmaps that are larger than 64K bytes. This problem appears with Trident, Paradise and Diamond Stealth drivers (and probably others as well). I devised a method to circumvent the bug, so you should not have any problems with the new release. You can try to find updated drivers for your video card in the directory /pub/pc/win3/drivers/video at ftp.cica.indiana.edu; the problem is that the drivers there are not that new; your best bet is to contact the manufacturer of the video card and complain about the driver quality. If enough people complain, they will fix it.

I am still a bit unclear about the graphics capabilities. Can you explain them better?

The graphics capabilities are not that complicated. We have two ways to import graphics: the first one is without any graphics filters, and the second one with graphics filters. The first case works only with "standard" metafiles. The second case works with "placeable" metafiles, or whatever graphics files are supported by your filter.

Consider the first case (without filters). You make a graph for example with Excel, copy the graph to the clipboard and then run the "clipmeta" program to save the metafile [make sure that you specify a standard (or plain) metafile format in the "Save As..." dialog]. Let's say that you saved

the graph in the file "abc.wmf". Now put the commands

```
\special{anisoscale abc, x-size y-size}
\vskip y-size
```

in your TeX file. x-size and y-size are the desired dimensions of the graph (look at the demo.tex file). When you view or print the file, the graphic should be fine.

Consider now the second case (with the filters). We will do the same example as above, but with two different graphics files. Let's say that you want to import a "placeable" metafile and a TIFF file. By the way, even though TIFF files are standard in desktop publishing, they are not particularly good, because they cannot be scaled very well (the same applies to any bitmap format). Metafiles on the other hand are great because they are small, can contain most drawing or graphics commands and can be scaled very well. Anyway, let's talk about the technique for using the two graphics files. The first and most important requirement is that you have the appropriate filters. If for example you have Word for Windows, you will need the files "wmfimp.flt" and "tiffimp.flt". You will also need an entry in your win.ini file as:

```
[MS Graphic Import Filters]
Windows Metafile(.WMF)=c:\binw\filters\wmfimp.flt,WMF
Tagged Image Format(.TIF)=c:\binw\filters\tiffimp.flt,TIF
```

These lines instruct dviwin to use the filter "wmfimp.flt" for files with a "WMF" extension, and the filter "tiffimp.flt" for files with a "TIF" extension. I put them in the directory "c:\binw\filters" but you can put them anywhere you like. Word for Windows adds this entry in "win.ini" upon installation, so you may already have it.

Once this step is done, just put the \special{} and \vskip{} commands in your TeX file, and everything should work out fine. The beauty of this setup is that it is extensible: if tomorrow we decide to use a new graphics format, we only need a new filter, and all applications will benefit. If you have Word for Windows, Toolbook, Powerpoint or Pagemaker, the filters are already in your hard disk.

The capability for various graphics files is pretty nice: most of the time however you can use plain metafiles with no loss in functionality (provided that you use a Windows program to generate the graphs). Therefore, you do not really need the filters for everyday work. I just thought that this capability would be nice.

I tried to import an HPGL or a PIC file using the filter supplied with Word for Windows 2.0, but nothing is displayed. What is the problem?

This is a bug in the graphics filter: it is supposed to set the window limits when importing the graphics file, but it doesn't do it, and Windows draws the entire graphics file on a single pixel. Word for Windows knows about this problem and fixes it silently. I did the same thing in the new release, and these files now display properly. Keep in mind that some of these filters are not that great: I tried for example to import a TIFF file and the filter works only with uncompressed files. Microsoft claims that the filter works with compressed files too, but I never managed to import such a file. The PCX filter cannot import a file containing more than 256 colors, and even then, the current video must match the video mode used when creating the file.

I tried to import a Postscript file but I can only see an empty rectangle.

The problem is that the EPS filter does not really import a graphics file. It just creates the empty rectangle, so that you know where the graphic will be; the actual translation of the postscript file is delegated to the translator inside the postscript printer. Therefore, there is currently no easy way of importing such a file.

When I insert a metafile in my document, the text inside the graphics file looks ugly. Is there a way around this problem?

This usually happens if you use a non-scalable font in the metafile. Try using the "Times New Roman" font instead. Microsoft also sells a larger collection of TrueType fonts; if you have it, you can get better results by using the "Century Schoolbook" font which blends better with the cmr fonts.

Why don't you distribute the graphics filters with dviwin?

The graphics filters are commercial programs; I would have to license them from their authors to distribute them with dviwin, and that is clearly infeasible for a free program.

How does the refresh capability work?

When you switch to another program, dviwin closes the dvi file but keeps the time and date information of that file. When you switch back to dviwin, it checks the time and date information of the dvi file, and if it is different, then it reloads the file but keeps you at the same position as before. It also tries to re-use as many of the old fonts as possible, so the reloading should be much faster than the initial loading. This approach is extremely useful if you wish to integrate the editing and previewing; suppose that you edit a file, run TeX and start the previewer. If you see something wrong, you only need to switch to the editor and rerun TeX; as soon as you switch to the previewer, it will load the new version of the file automatically (and quickly). You can also automate the process even further if you use a good editor and a batch language for Windows. Another advantage of this approach is that it does not depend on DDE or other Windows intricacies, so it will work with any version of TeX.

What are the advantages of the font cache? how large should it be?

Windows is quite slow in accessing the disk (much slower than DOS); this is a big disadvantage for dviwin which needs to access the dvi file as well as the fonts. Therefore, I decided to keep old fonts just in case we may need them again in another document. When you close a document (either explicitly or by loading another one), the old fonts are not discarded; they are kept in the font cache. If the new document needs any of the fonts in the cache, it will take them from there, so the loading will be much faster. You obviously need to set a limit in the cache size (otherwise you will run out of memory), but this limit depends on the type of your documents as well as the available memory. A cache size of 10 should be a good initial value; then you can observe the number of fonts in the cache from the "Cache Size" entry in the "Options" submenu; a relatively good heuristic is to adjust the cache size so that it is usually full. You will need of course to take the amount of available RAM into consideration. The program clears the cache completely when printing; at that point, you probably need all the RAM that you can get.

I start dviwin and load a document. The cache size is 10 but I see no fonts in the cache. How is this possible?

Only *currently unused* fonts are in the cache. If you start dviwin and load a file, it will read all the fonts required by that file, but all of the fonts are "in use". Therefore, the cache will be empty. If you load a second file (or a revised version of the same file) which does not use all the fonts from the first file, then you will see some fonts inside the font cache.

Why do you require separate screen fonts for previewing? Is there a way to use the printer fonts to save some space?

You can do this trick by reading the printer fonts, keeping the bitmap at that resolution and producing a small bitmap by reducing the big bitmap by an integer factor. There are two common methods for the reduction; the first one is to split the big bitmap into N by N squares (N usually ranges from 2 to 5), count the number of black pixels within each square and if this number exceeds a prespecified threshold, set the corresponding pixel in the small bitmap to black; otherwise, you set it to white. This method is used by xdvi, but it does not look good on normal VGA resolutions (you cannot beat Metafont's algorithms with this simple scheme). The second approach is similar to the first one, but it does not constrain the small bitmap to black and white values; it just assigns a gray level to each pixel in the small bitmap according to the number of black pixels in the corresponding square of the big bitmap. This method is used by dviscr and produces excellent results on low resolution devices because it increases the color resolution to compensate for the loss in spatial resolution. The main problem with this method is that grayscale bitmaps take more space than monochrome bitmaps (at least 4 times more memory), so Windows would probably start swapping and render the method useless; you will essentially use the disk space for swapping instead of keeping the actual fonts. Apart from this, Windows displays these bitmaps slower than monochrome bitmaps, so even simple scrolling would take more time. Another issue with the grayscale fonts is how you should handle color displays: it is quite easy to produce four gray levels under any Windows display, but if for example the background is blue and the foreground is yellow, the program would need to find several colors between these two colors (in a visual sense); therefore, you would need at least a 256-color video driver, and even then you may get visual artifacts with some color combinations. I do not want to constrain the program to use black letters on a white background because the display flicker is most visible on a bright background.

I will try to work more on this approach, but it is not that easy. Considering the fact that screen fonts require much less space than printer fonts, you will need about 1.4M of hard disk space for a reasonable set of screen fonts. I don't know if that space is worth the effort; I would prefer to spend the disk space on the fonts instead of waiting more for the display.

Why is the close-up view so coarse?

The close-up view is obtained by taking the actual bitmap and doubling the bits in both axes; therefore, it is not surprising that it is ugly. This is the reason for providing a smoothing option; I find that it looks better when you select medium smoothing. The alternative is to open new fonts at higher resolutions and typeset the page again, but this will take enough time to be unusable. xdvi produces a fine close-up view because it keeps the fonts at printer resolution internally, so it just displays them without any reduction (look at the previous question for more on the reduction issue).

You mention that there is a limit of 17 open font files. Why? Are there any other limits?

The C compiler imposes a limit of 20 open files for buffered I/O; apart from the fonts, dviwin opens three files: the dvi file, the log file and the font substitution file. Therefore, there are 17 files available. We can use a higher limit with unbuffered I/O, but the net speed effect is negative. There are a few other limits too:

1. The maximum number of Windows fonts inside a *single* metafile is 20.
2. The maximum custom resolution is 3000dpi (this is just a sanity check).
3. The unpacked raster for a single character cannot exceed 64K bytes.
4. The maximum number of pages in the dvi file is 16384.
5. The maximum size of the TeX stack is 2048.

The last three limits come from the 64K segments (ie., there are no preallocated arrays). I think that all these limits are sufficiently high as not to be binding.

Is there a way to search for text inside dviwin?

There is currently no way to do this: I am planning to add this capability in the future, but it will accomodate only simple strings; you will not be able to search for "{\bf test}" or "n^2" because this would require a full TeX parser which is beyond the scope of a dvi driver.

Can I use virtual fonts with dviwin?

The current release cannot use these fonts. I will investigate them, and they will be supported in a future release if possible.

Is there a way to use TrueType fonts in my TeX documents?

I have some ideas about producing PK fonts from any TrueType font. The main problem is how to produce a TFM file; I would appreciate if any reader can show me how to do this. Once the TFM files are in place, you will only need a few macros to set up your document. Keep in mind however, that these fonts will not be as common as the cmr family, so you will lose some portability.

Why do you support only Windows 3.1 instead of 3.0 as well?

The first release (2.0) had a problem when running under Windows 3.0. I am using a Microsoft-supplied library (ctl3d.dll) which was supposed to work under Windows 3.0. The problem is that this library did not actually work under the older version; my fault was that I did not test it myself. Eventually, I received a new version of the library that works properly under Windows 3.0, so this problem has been solved. However, I found a more serious problem with Windows 3.0: when you import a graphics file through the \special{} command, the driver allocates a bitmap and asks Windows to draw the graphics file into the bitmap. The problem is that the bitmap will often be larger than 64K bytes (especially when printing) and Windows does not display the graphics file properly (it does not report any errors either). There are some ways around this bug, but they take time, and I think that it is unfair to penalize all users for this bug in Windows 3.0; given the amount of bugs that exist in Windows 3.0, the best policy is to upgrade. If you still want to use the driver under Windows 3.0, you will only need another Microsoft library (commdlg.dll) which I can send you via e-mail.