

!Drawf

This is the manual for !Drawf, a Wimp front end for `mkdrawf` and `decdrawf`. If you don't know what those are, you should not be reading this. On the other hand, you don't need to be expert in the use of `mkdrawf` before reading this; if you come across an allusion to a feature you've never heard of, just ignore it or look it up in the `mkdrawf` manual.

Since the three programs are distributed together, their version numbers should always agree. This manual describes version number 3.08 of !Drawf; you can tell what version you have by consulting the *Info* item on !Drawf's menu, producing something like:

About this program	
Name	Drawf32
Purpose	Make & decode drawfiles
Author	Gareth McCaughan
Version	3.12 (06/06/2023)

but hopefully with a more recent version number.

Legal rubbish

This program has the same status as `mkdrawf` and `decdrawf`; see the main manual for a precise statement. Roughly, you are permitted to copy and distribute this stuff provided you don't (1) make any changes or (2) charge for doing so. If you want to do (1) or (2), get in touch with me. Of course you can make changes to your own copy; but you must not (without my permission) distribute the changed version.

A potted summary

Drag a text file to the !Drawf icon to get it converted to a drawfile by `mkdrawf`; drag a drawfile to the !Drawf icon to get it converted to a text file by `decdrawf`; the *Options* window (accessible from the menu) lets you specify where `decdrawf` puts some kinds of data and where `mkdrawf` gets certain other kinds of data from. Error messages get sent to a throwback window, if you have the Desktop Development Environment installed; into a normal text-editor window, if not.

Basic use

When you run !Drawf its icon should appear on your icon bar. The icon looks like this:



which is a rather feeble attempt to indicate that it turns text files into drawfiles and vice versa. Dragging a file to this icon will have one of three results: if the file is a text file, !Drawf will try to invoke `mkdrawf` on it, producing a drawfile; if the file is a drawfile, !Drawf will try to invoke `decdrawf` on it, producing a text file; otherwise, !Drawf doesn't know what to do with it and therefore takes no notice.

Supposing the file to be of a kind !Drawf recognises, it should then pop up a window looking like this:

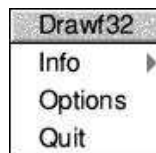


(of course, the icon might show a drawfile instead of a text file). You should either enter a full pathname in the writable icon at the bottom of this window and click OK, or else enter just the leafname and drag the file icon somewhere. (The “somewhere” is allowed to be, for instance, a text editor window or !Draw’s iconbar icon.) The one thing you should *not* do is to drag the icon to the directory from which the file originally came; that would make the output overwrite the input, and you probably don’t want that.

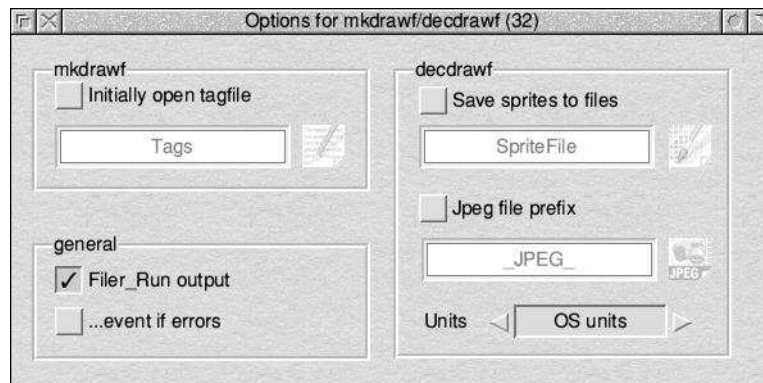
There may well be a delay before the window appears, though; all the actual processing of the file is done before the window appears. Also, any error messages produced by mkdrawf or decdrawf should appear in a window (either a normal text editor window, or a throwback window, depending on whether you have the DDE installed) at about the same time as the save window appears.

Setting options

From !Drawf’s menu



you can get to the *Options* window



in the obvious way.

The purpose of the buttons in the *general* section of the window is probably pretty clear: if *Filer_Run output* is selected, the drawfile or textfile produced will be run (so that it appears in a window) unless mkdrawf or decdrawf reported any problems while making it. If ... *even if errors* is selected, it will be run even if there are errors; you are warned that this is often completely unhelpful, especially in the case of mkdrawf.

It may well not be obvious to you what the rest is for. The full answers are in the mkdrawf manual, but briefly: You can recompile mkdrawf so that it can use “tagfiles”, which are a bit like RISC OS 3’s message files but simpler, and if you choose mkdrawf will open a particular tagfile when it first starts up; decdrawf usually represents sprites and JPEG image objects as great lumps of hexadecimal data, but you can tell it to write this stuff to another file instead.

When one of these is selected, you can edit the contents of its writable icon; alternatively, dragging the file icon at the right-hand side to a Filer window will fill in the pathname in much the same way as if you were saving something (except that nothing actually gets saved).

Finally, `mkdrawf` and `decdrawf` can use various units for describing dimensions; by default they use *points* (a point is 1/72 of an inch), but this can be changed. To make `!Drawf` ask `decdrawf` to use any given unit, change the units specified at the bottom right.

There are a few traps of which you should be aware...

1. You need to set whatever options you want *before* dragging your file to the `!Drawf` icon. It's no use dragging the file and then fiddling with the options window; by that time it's already too late.
2. `!Drawf` makes no attempt to check that the place to which you have dragged one of the icons is actually a Filer window. Unfortunately, if it isn't then attempting to do anything which uses the option is likely to cause trouble. So don't do that.
3. RISC OS is in many ways a direct descendant of the operating system on the old BBC micro. One way in which this shows is in the absurd 256-character limit on command lines. This applies even when the command is being issued by another program. Furthermore, violating this limit causes my machine (I haven't tried it on any others) to lock up completely until it's reset. So, `!Drawf` checks whether the command line it constructs for running `mkdrawf` or `decdrawf` is too long. It is quite likely to be too long, if you are saving sprites or JPEG images to separate files. If you have the DDE, this ceases to be a problem and `!Drawf` doesn't do the check.

That's all, I think. I did say it wasn't complicated.