SwiftJPEG v1.13 User Guide
07-Aug-2012

Contents

Main contents

Appendices

Introduction

SwiftJPEG is a viewer for JPEG images. These are a particular kind of image common on the World Wide Web and used by most digital cameras.

The application runs on RISC OS 3.1 through to RISC OS 5. To run on RISC OS versions 3.5 and earlier you have to load some support modules (described later). RISC OS 3.6 and later need no additional software.

Why use it?

SwiftJPEG provides a fast and convenient way to view JPEG images without using a lot of the computer's memory. On a RISC OS 3.1 machine with the relevant modules loaded, SwiftJPEG provides a chance to view images that would normally be far too large to fit in memory using the conventional mechanism of decompressing the image using something like ChangeFSI.

On RISC OS 3.6 upwards, the Draw and Paint applications can both read JPEGs directly. Draw keeps the image as a JPEG, but Paint decompresses the image completely, which means a large JPEG can require a very large amount of RAM to be viewed. Neither application is designed purely as a viewer for images; both are designed as editors for other kinds of material. SwiftJPEG fills this gap.

More about JPEG images

JPEG images use a form of lossy compression. This basically means that the quality of the JPEG image can be chosen when it is created, with a lower quality image producing a smaller file. JPEGs are designed to be used with photographic images; the sort of information that the lossy compression mechanism throws away as the quality is decreased is least noticed with photographic images, and most obvious with things such as line art or screen shots of computer programs.

JPEG images should have filetype **&C85**.

Starting SwiftJPEG

To start SwiftJPEG, double-click on the application. It can be run directly from a read-only medium such as a CD-ROM or archive, though if you want to be able to save Choices settings, you must copy the `!SwiftJPEG` application directory to a writable location such as a hard disc. SwiftJPEG can also be started by double-clicking on a JPEG file once the Filer has 'seen' SwiftJPEG.

Viewing images and the File List

When SwiftJPEG is running, you can drag JPEG images to the icon bar icon to load them. They do not have to have the correct filetype (&C85) set first. Double- clicking on an image which does have the correct filetype will usually load it too, though if you have any other applications running which can also load JPEG images this way, one of those might catch the image first.

A collection of images can be loaded by either selecting a group in the Filer and dragging that selection to the SwiftJPEG icon bar icon, or by dragging a directory to the icon bar icon. In addition, by default, if you just drag on a single image SwiftJPEG will search the rest of the directory from which the image came for any other files of filetype &C85 and will queue these for viewing too. It'll still load the image you asked it to load first. When you drag on a directory, SwiftJPEG searches the contents for any files of type &C85 and looks in any subdirectories it finds as well, so you can load a complete collections of images very easily. With directories, once SwiftJPEG has queued all the JPEG files available, it will load the first one it found.

When you drag a single file or a directory to SwiftJPEG, it asks RISC OS for a list of the other files it adds to the file list. They are returned in an order which depends upon the filing system (typically in ascending alphabetical order, but not always). If you want to add a group of files in the exact sort order as you seem them in a Filer window, select the group of files you want to view and drag that group to SwiftJPEG directly. In this case, the Filer itself tells SwiftJPEG about each file in the group and always lists them in the same sort order as you see in the Filer window.

All queued files are held in the *File List*, which is kept for as long as SwiftJPEG runs or until you next try to load one or more new files by dragging them to the icon bar icon or double-clicking on them. You can easily move through the file list to browse through the various queued images. Navigation through the file list is described later.

Useful tip

SwiftJPEG supports two useful modifier keys that you can hold down when dragging images or directories to the icon bar icon:

- `Ctrl` (Control) – only add the specific image that was loaded to the queue; do not add the contents of the directory in which it lies (handy if it takes a long time to search for other files, e.g. because the directory has a great many images in it).
- `Shift` – append the image details to the existing file list (normally, if images have been shown, any new images dragged to SwiftJPEG are loaded into a new list). When using this modifier key, if you drag on a single image then only that image is appended onto the list – an automatic directory search is never done.

By holding down both keys when loading images, you can build up a slideshow containing a series of just the specific images that you drag to SwiftJPEG to be shown in the order that you dragged them.

The Display window

Using the default settings, whenever SwiftJPEG loads a JPEG image it shows it at 1:1 scaling in the Display window on the RISC OS Desktop:

Clicking with SELECT (the left mouse button) on the icon bar icon will bring the window to the front of the stack, in case it has been hidden by other windows. If you close the window, the JPEG image is removed from memory but can easily be reloaded using the main menu (see later). If you click on the window with SELECT or ADJUST (the right mouse button), it will gain input focus and the following keys can be used:

- Right cursor – go forwards in the File List to the next available image, if there are any
- Left cursor – go backwards to the previous available image, if there are any
- Down arrow – skip forward 10 images or to the last available image, whichever comes first
- Up arrow – skip backwards 10 images or to the first available image, whichever comes first
- Escape – close the window and remove the JPEG image from memory
- F, Enter or Return – view the image in Full Screen mode (more on this later)
- F3 – save a copy of the current image
- I – open the picture information window as a menu (more later; press Escape once to close it)

If you press MENU (the middle mouse button) the main menu opens. This can also be opened by clicking MENU over the icon bar icon. Dragging with SELECT from within the Display window to a Filer window saves a copy of the image using the same leaf name as the original. This is really useful if you are previewing a set of images, only wishing to keep some of them. You can have a Filer window open for the directory of the images you want to keep, flick through the original images with the arrow keys or main menu (see below), and drag directly from the Display window to the Filer window to save copies of those that you like.

Menus

Main menu

The main menu is shown below:
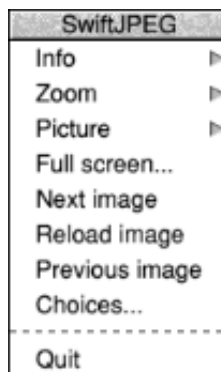
It contains the following options:

1. **Info**

   Move the pointer to the right to display some information about the version of SwiftJPEG in use.

2. **Zoom**

   Move the pointer right to open a small window that lets you zoom in or out of the current image. The window is described in more detail later.

3. **Picture**

   Move the pointer right to display the picture menu (see below). Greyed out if no image is loaded.

4. **Full Screen...**

   Shows the currently loaded image in Full Screen mode. A large amount of SwiftJPEG's functionality is wrapped up in Full Screen mode, which provides a flexible set of slideshow and viewing options described in full in their own section later.

5. **Next image**

   Go forwards in the File List to the next image relative to the one currently being viewed, or recently loaded. Greyed out if there is no next image to load; this is usually because you have reached the last valid item in the File List.

6. **Reload image**

   Reload the most recently loaded image. Greyed out if no images have yet been loaded, or if there is no valid item available to be loaded.

7. **Previous image**

   Go backwards in the File List to the previous image relative to the one currently being viewed, or recently loaded. Greyed out if there is no previous image to load; this is usually because you have reached the first valid item in the File List.

8. **Choices...**

   Open the Choices dialogue box. This is explained in more detail later.

9. **Quit**

   Shut down SwiftJPEG. This option is only available when the main menu is opened from the icon bar icon.

Picture menu

The picture menu is accessed through the main menu and is shown below:

```
 SwiftJPEG
 Info            ▷
 Zoom            ▷   Picture
 Picture         ▷   Info     ▷
 Full screen...      Save     ▷
 Next image
 Reload image
 Previous image
 Choices...
 - - - - - - - - -
 Quit
```
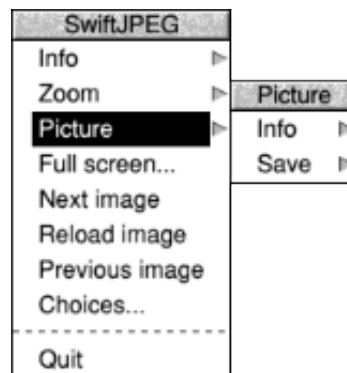
The two options are:

1. **Info**

   Move the pointer right to open the Picture Information dialogue box. This gives a description of various attributes that the image has, such as its size and the date stamp on the file:

| JPEG info	
...to Egilssel.egilssel_columns/jpg	
Size (bytes):	430624
Width (pixels):	1600
Height (pixels):	1200
Camera make:	NIKON
Camera model:	E700
Picture date:	17:02:36 13-Aug-2000
File date:	15:38:44 20-Aug-2000

Digital cameras and JPEGs

Almost all digital cameras write extended forms of JPEGs sometimes called 'EXIF images', after the technical name of the area where extra data is stored.

The first four lines give the image pathname, size in bytes, and height and width in pixels. These are always filled in.

The next three lines (camera make, model and picture date) come from information contained within the JPEG itself. Some JPEG images, but not all, contain this extra detail. It is particularly common in images from digital cameras. 'Picture date' usually tells you when a photograph was taken. Since it is contained within the image itself, the picture date is independent of the date stamp on the file, which is an external property maintained by the file system on which the picture lies. You can make copies of the file, for example, and they will be stamped with the date on which they were copied, without affecting the internal record of when the image was created. If any of the three fields are not present in the JPEG image or are in a format which is not understood, 'Unknown' is shown.

If you modify an image in some photograph editor application, you may destroy or change the EXIF information. For example, the "Picture date" may be updated to the date when the image was edited, rather than the date when the photograph was taken.

The last entry in the dialogue box gives the date stamp on the file itself. This lets you know when the image was stored on disc, or copied, but is not necessarily a reflection of when the image was originally created. The information is always filled in.

Some versions of the RISC OS JPEG plotting software cannot understand EXIF images. SwiftJPEG translates the images after loading them so that they can still be viewed. The file on disc is not altered in any way and saved copies will be identical to the original, including the EXIF data.

2. **Save**

Move the pointer right to open the Save Copy dialogue box. This is a standard RISC OS save box; drag the icon to a Filer window to save a copy of the file, typing in a new filename if you want one, or type in a complete pathname and press Return or click on the OK button to save the picture.
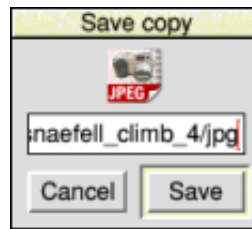
If you just want to make a copy of an image in a new directory, you can drag straight from the Display window to a Filer window to save a copy using the same leafname as the original image.

The Zoom box

The zoom box is accessed from the main menu:
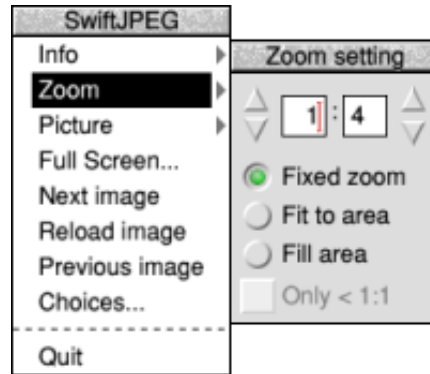
The three radio buttons provide options as follows:

- The **"Fixed zoom"** setting, if selected, will enable the writable fields and 'bump arrows' at the top of the window. These let you set a fixed scale factor of 20:1 (twenty times larger than normal) down to 1:20 (one twentieth of the normal size).

- The **"Fit to area"** setting selects a select a scale factor that makes the image occupy as much of the Desktop as possible without any of the image being cropped (so you don't have to scroll to see any of it). The Display window will be resized to as large as is required to fit in the image. If you have configured your machine to stop windows covering the icon bar, SwiftJPEG will obey this setting.

- The **"Fill area"** setting selects a scale factor that makes the image occupy as much of the Desktop as possible, filling the whole screen with the exception of the window tools around the Display window. For most images, this will result in cropping either horizontally or vertically, so you will need to use the scroll bars to see the whole image.

> *Making changes last*
>
> Whenever you use the zoom box, the settings you make apply just to the current image being viewed.
>
> If you want your changes to apply to all images, use the Choices dialogue box (described later) to specify the zoom settings.

At the bottom is a slightly cryptic option button, "Only < 1:1". This restricts the scale factor used for the "fit" or "fill" options so that an image is only ever shrunk down to fit on screen, but never scaled up. When JPEG images are scaled up, they become blocky (you can more easily see the individual pixels that make up the image).

You may not like this blocky effect, but may still want to be able to see the whole image without having to scroll around. If so, turn on this option. Now, small images will not be zoomed in to fit or fill the available viewing area, but large images will still be zoomed out so that you can see the whole picture.

Limitations with zooming

The zooming done by the OS JPEG plotting routines is fast but not high quality. If the screen mode for the Desktop supports less than 16M colours and in particular 256 colours or less, you will see that images scaled up above their normal size look very "speckled". This is because the OS does something called *dithering* to make an image look better in a lower number of available colours on screen, but this relies on pixels being small and hard to individually make out. The first picture on the left shows a section of sky from a digital photograph dithered down to 256 colours – it looks fairly smooth.

When an image is scaled up, the OS scales the dithered pixels too, so they become very easy to see. It would be better if the OS scaled the pixels from the JPEG image first, *then* dithered the result as if it were one much larger image, but at the time of writing it does not do this. The second picture on the left shows a zoomed section of the same sky picture from above.

When the OS routines zoom out of (scale down) an image they just stop plotting rows or columns as required to make the image fit with the requested scale factor. No attempt to spread errors from neighbouring rows or columns is made. This can create a "stepped" look to sloping edges and have an odd effect for patterned parts of images, such as chequered fabrics or metal gratings. If you are in a mode with less than 16M colours, then again the dithering issue crops up. This time, dithering pixels that were being used to simulate more colours than are really available get removed as rows or columns don't get plotted. Images will often take on a mottled appearance, as shown in the third image on the left.

Again, it would be better if the OS scaled down the original image, then dithered the result, but this is not how things work at the time of writing.Ultimately, if you are in a display mode with 16M or 32K colours available, scaling works reasonably well. In 256 colours or less, the display quality will be fairly poor, particularly if an image is scaled down (zoomed out) – you may be best off sticking to 1:1 scaling.

Above: 1:1 zooming
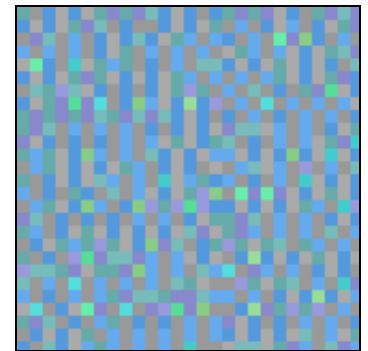
Above: 6:1 zooming
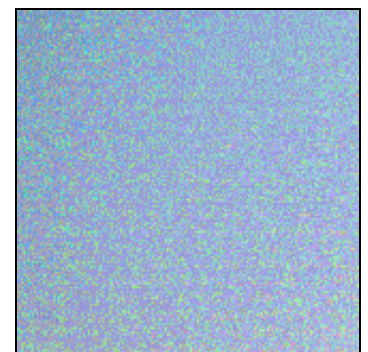
Above: 1:3 zooming

Full Screen mode

SwiftJPEG includes a way of viewing images full screen, without the Desktop visible. It can be told to operate a slideshow, waiting for a predefined number of seconds before advancing automatically to the next image. You can set up fades on RISC OS 3.5 or later; the drawing of the JPEG is hidden, then the image is faded smoothly in from black or 'flashed in' from white. It can be faded or 'flashed' back out again too; both actions are independently configurable with a range of different fade speeds available.

Some points to note:

- Images are shown outside of the Desktop, and the machine single tasks – other applications do not carry on running.
- The screen mode used to display the image may be changed from that in which the Desktop is displayed. Behaviour is completely configurable.
- Images can be zoomed in and out whilst in Full Screen mode, though the zoom settings are restricted to some predefined values.

The keys defined for Full Screen mode are as follows:

- `F`, `Enter` or `Escape` – return to the Desktop.
- `Right cursor`, `SELECT` – go forwards in the File List to the next available image, if there are any.
- `Left cursor`, `B`, `ADJUST` – go backwards to the previous available image, if there are any.
- `Down arrow` – skip forward 10 images or to the last available image, whichever comes first.
- `Up arrow` – skip backwards 10 images or to the first available image, whichever comes first.
- `I`, `MENU` – show some information about the current picture.
- `Z` – change zoom mode. Cycles through zoom settings – the fixed zoom ratio set up in the Choices dialogue box, fit to screen without cropping, fill screen completely cropping if necessary, and a special 1:1 mode if the fixed zoom ratio is not already itself 1:1.
- `P`, `Space Bar` – pause a slideshow, when a slideshow delay has been set up in the Choices.
- `Q` – if fading is turned on, pressing `Q` jumps to fully faded in or out so you don't have to wait for the fade to complete.
- `0-9` – set a fixed zoom size according to the table below:

> *Useful tips*
>
> In Full Screen mode, any key that is not already used for something else will move forwards through the File List to the next image.
>
> If you overlay picture information using the I key, you can remove it by forcing the image to redraw: use one of the number keys listed in the table below.
>
> When no slideshow timer is set (the slideshow delay in the Choices dialogue box is zero), the P and Space Bar keys move onto the next image.

Key	Zoom up	Key	Zoom down
1	1:1	6	1:1
2	2:1	7	1:2
3	3:1	8	1:3
4	4:1	9	1:4
5	5:1	0	1:5

Various overlaid items are briefly shown in the bottom left hand corner of the screen in response to some key presses. They indicate that a particular change has been made, as follows:

Normal zoom (1:1)

Zoomed in (>1 : 1)

Zoomed out (1 : >1)

Fit to screen (no cropping)

Fill screen (cropping if necessary)

Slideshow paused
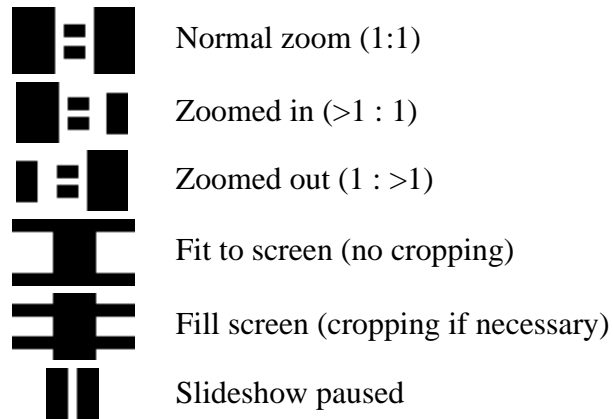
You cannot scroll around an image in Full Screen mode, to view parts of it that have been cropped off the edges of the screen due to a particular zoom setting. That's what the Desktop is there for!

Many features of Full Screen mode as described above can be changed using the Choices dialogue box, described in the next section.

The Choices dialogue box

Accessed from the main menu, the Choices dialogue box provides an extensive array of options that modify SwiftJPEG's behaviour. To help make those options less bewildering, the dialogue box is separated into several groups as shown below:
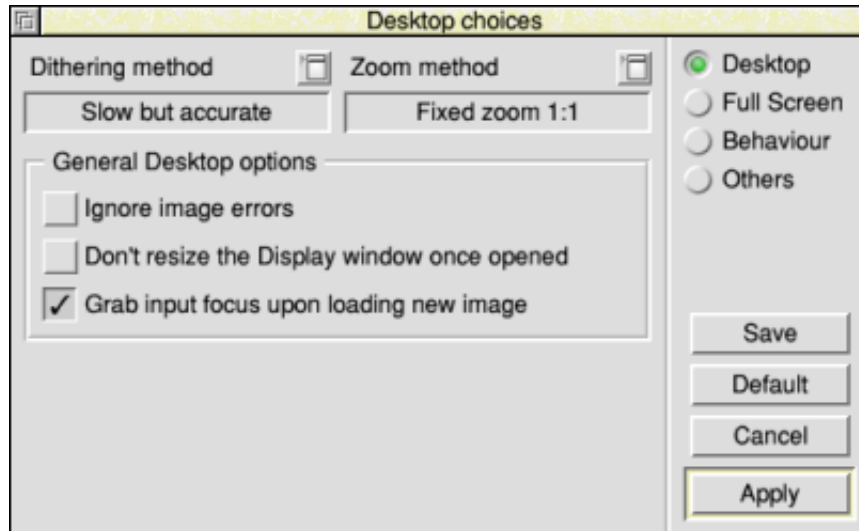

Select a group by clicking on the radio button next to its name, in the panel on the right hand side of the window.

"Desktop" section

This section contains options that change how SwiftJPEG behaves when showing images on the Desktop. At the top left is the *dithering* menu. Dithering is a way of showing images with lots of colours in screen modes with fewer colours in a way that attempts to give the illusion of more colours being displayed at one time than is actually the case. For example, if the screen mode only had red and yellow colours available, orange could be simulated by making a pattern of alternating red and yellow pixels. If the pixels are small, the effect is quite successful.

The RISC OS JPEG routines provide three dithering settings – a high quality mode, which gives good results but is quite slow; a medium quality mode, which produces average to poor results but is quite fast; and no dithering, which gives the fastest display but low quality output.
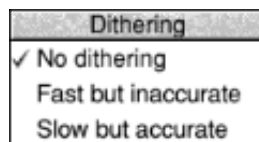
In 16 million colours, no dithering is needed because the screen mode can already show as many colours as it is possible for a JPEG image to contain. Set the dithering menu up for your preferred arrangement when in lower colour depth modes, or ignore it if you always work in 16 million colours.

To the right of the dithering menu you can set up the zoom method used by default when new images are loaded. The options are analogous to the Zoom box shown from the main menu or an image view:
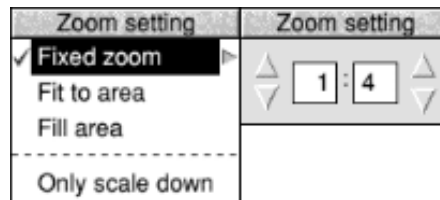
You can select a fixed zoom setting, fit the picture to the screen without cropping, fill the screen, cropping if necessary, and set an override which will only shrink, but never enlarge images (" **Only scale down**" is analogous to the Zoom box "**Only < 1:1**" option).

Underneath are three other options:

1. The "**Ignore image errors**" option suppresses errors that occur when trying to load a picture in the Desktop. It can be useful if you are browsing through a group of pictures and don't want to be interrupted by error boxes if one or two of them cannot be shown for some reason. Normally this setting is disabled, so you get told about problematic images straight away.

2. The "**Don't resize the Display window once opened**" option stops SwiftJPEG from resizing its window when you load a new image. If the window is closed and you load an image by double-clicking on it or dragging it to the icon bar icon, the image view window is opened at the best size to fit your image and the zoom settings set up by the zoom method menu (see above). If the "Don't resize…" option is turned off, if you load another image with the display window still open, it will be resized to match that image. Alternatively, if the "Don't resize…" option is turned on, the next image loaded into an already open window will be scaled or cropped to fit the current window size as best as possible.

3. "**Grab input focus upon loading new image**" makes SwiftJPEG claim "input focus" (the ability to receive key presses in the Desktop) whenever a new image is loaded. This is the default behaviour so that you can use the various key shortcuts to work through a File List, enter Full Screen mode and so-on. However, sometimes you might want the input focus to stay where it is, for example, in a text editor where you're writing a description of a set of images you wish to browse through.

"Full Screen" section
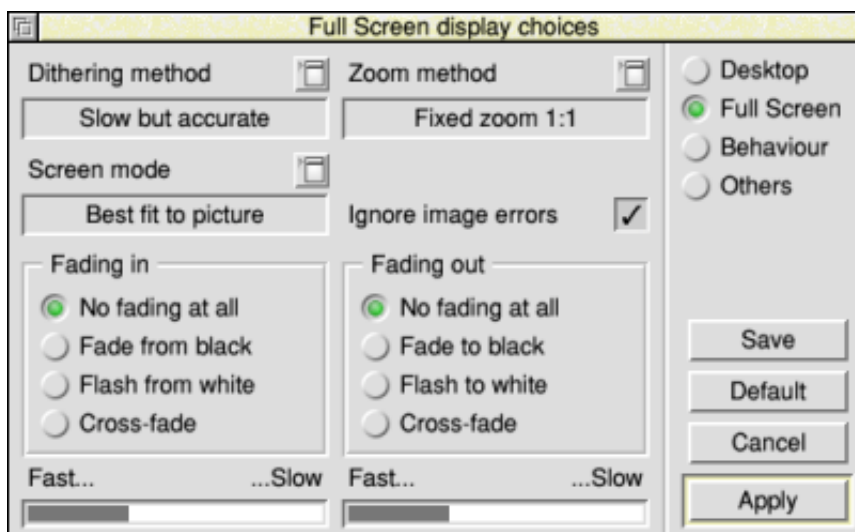
This section contains options that change how SwiftJPEG behaves when showing images in Full Screen mode. At

the top left is the *dithering* menu, which works exactly the same as the one in the Desktop section but, obviously, only influences images plotted in Full Screen mode. To the right of this is a zoom menu, which again works just like the one in the Desktop section but only for Full Screen mode images.

Underneath the dithering menu is the *screen mode* menu. This controls the way in which SwiftJPEG picks a screen resolution and colour depth for each image it displays. The best choice depends on your machine's capabilities, the kind of images you are viewing, and the reason why you are viewing them:
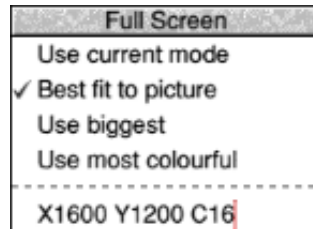
- "**Use current mode**" shows all images in the same mode as the Desktop. This avoids the need to change screen modes at all, and can be particularly useful if you have a monitor with a very restricted range of resolutions such as an LCD panel.

- "**Best fit to picture"** is the default setting. First, a screen mode which can show as many pixels horizontally and vertically as the image contains is found, with the screen mode being as close to the image size as possible but no smaller than it. The image is never cropped off the edges of the screen, unless it's such a big image that there is no mode large enough to show all of it. Then the largest number of colours available in that mode is selected. If you have set up a fixed zoom setting, then the resolution of the mode is chosen according to the image's size *after* scaling. For all other zoom settings, the resolution of the image at 1:1 scaling is used as the basis for screen mode selection. Only square pixel modes will be chosen.

  This selection is a good 'all-rounder', matching the screen to the image, but it does mean that there could be lots of mode changes if your images have very varied sizes, and sometimes odd modes may be chosen that you wouldn't normally use which your computer does support, but don't work very well with your monitor (SwiftJPEG has no way to assess how well a mode might be displayed so treats all modes equally in this respect).

- "**Use biggest**" finds the largest resolution (X and Y pixels) mode, and uses the largest number of colours that your machine supports at this resolution. Again, only square pixel modes will be chosen. This avoids the need for many mode changes and can be useful when you have a lot of very large images to view, with a few smaller ones scattered around the collection.

- "**Use most colourful**" finds the screen mode with the largest number of colours (which will usually be 256 on a RISC OS 3.1 machine, or 16M on anything newer). Then the mode with the greatest X and Y resolution in that colour depth is chosen. Once more, only square pixel modes are chosen. This is a good setting if you want to see the images in the best possible quality (in terms of colour depth) at all times, even if the image might get cropped. It makes a lot of sense when used in conjunction with the "fit to screen" or "fill screen" zoom options.

Finally, you can type in a mode number (RISC OS 3.1 upwards) or mode selector string (RISC OS 3.5 upwards only) if you want to specify an exact mode to use. This is most commonly used when using image double banking

– more on this later.

To the right of the screen menu is an option button that lets you choose to show or ignore image errors. As with the dithering and zoom method menus, this works the same as the similarly named option in the Desktop section but only takes effect for images loaded while in Full Screen mode.

The bottom half of the Full Screen section is dedicated to fading or cross-fading between images. It is greyed out unless the ScreenFX module is present. This module is available from the SwiftJPEG download page at `http://www.riscosopen.org/`. At the time of writing the module requires RISC OS 3.5 or later, so on RISC OS 3.1 the fading section is not available.

Except for the cross-fade options, you can set independent fade in and fade out effects. When you select cross-fade in either the fade in or out section, the cross-fade option in the opposite panel is selected automatically. This is because during cross-fading, images fade in and out at the same time; for the duration of the fade, both images are visible simultaneously, to varying degrees. There are some more detailed notes on cross-fades below, but for now, please note that it requires a 16M colour screen mode to work. If you try to cross-fade in a Full Screen mode with fewer colours, SwiftJPEG will raise an error.

When any fade-in option is selected other than cross-fade, the JPEG image is plotted with the screen set to black, then (for "**Fade from black**") 'faded in' to normal brightness, or (for "**Flash from white**"), from a full white screen to a normal brightness. If you set a fade-out option other than cross-fade, then when you move to another image or exit Full Screen mode, the display will be faded back to black (for " **Fade to black**"), or (for "**Flash to white**") to bright white then quickly back to black. These options are most useful if you are creating some kind of slideshow and want to have smooth transitions between images.

Cross-fading works by making clever use of the same hardware facilities that allow fade in/out and flash in/out to work, only it programs a rather more complicated transition that actually uses just one 16M colour screen which contains *two* images at the same time. A 16M colour image uses 24 bits of information to describe each colour of each pixel (dot) making up the image. When cross-fading, SwiftJPEG divides this in two. It dithers down (see earlier for information about dithering) each of the two images to just 12 bits per pixel, so two images can fit in the space that one would normally occupy; you do lose some quality because of the dithering though. A "speckled" effect can be seen, more noticeable on images with large areas of very similar colours. SwiftJPEG programs some hardware display settings so that only one of the two images physically written to the screen is actually visible, and visible at full brightness. Cross-fading is achieved by simply changing the hardware display settings gradually from the setting that shows one of the images, to the setting that shows the other. The result is smooth and usually impressive transition from one image to the next. Since dithering is always used, the higher resolution of screen display your computer can manage in 16M colours, the better. Iyonix users do very well here, since the Iyonix can manage very high resolutions at this colour depth.

> *Recommendation*
>
> If you turn on cross-fading, select the "Use most colourful" option in the "Screen mode" menu. This is the setting most likely to work for the widest possible set of machines and images.

Finally, the sliders in each section let you control the speed of fading – to fade faster drag the slider down to the left; for a more gentle fade, drag the slider towards the right. For cross-fades, both sliders become "tied together"

so whichever one you drag, the other will always be updated to match it. During any fade you can press Escape to cancel the fade and jump to its end result, which is useful if you get tired of waiting for it to complete!
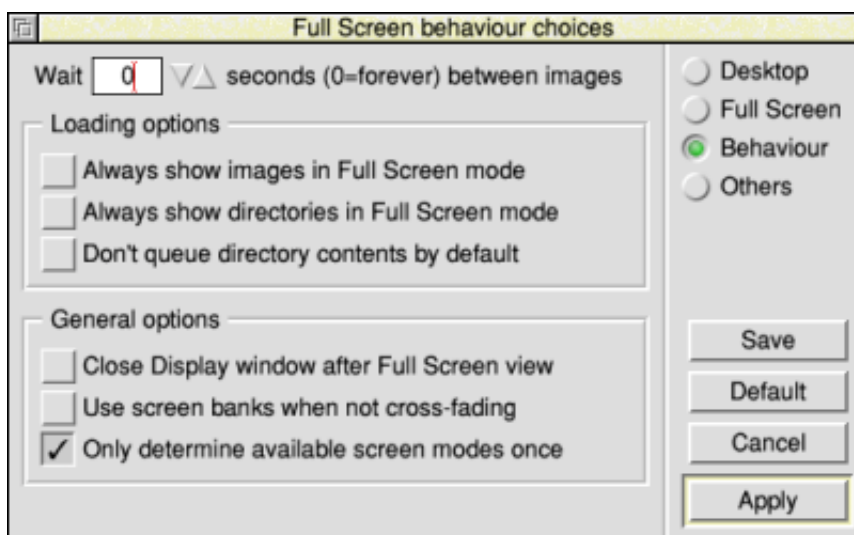
SwiftJPEG v1.10 and earlier attempted to maintain changes to the screen gamma settings made by the application "Gamma" by Rob Davison, rdavison@xtra.co.nz (if you don't use this application, you can ignore the rest of this paragraph). If the Gamma application was loaded before SwiftJPEG then SwiftJPEG took note of any settings made in that application and applied them to the fades. However, the application is no longer maintained and not 32-bit (e.g. Iyonix) compatible. SwiftJPEG v1.11 and later use the ScreenFX module for fades and cross-fades. At the time of writing, this does not take note of changes made by the Gamma application. There are at least some attempts made to restore the previous settings when Full Screen mode is cancelled and the Desktop view restored.

If you're still curious about the Gamma application and don't mind full screen mode fades ignoring its settings, then at the time of writing, you can find a copy at the AcornSearch FTP archive:

```
http://www.geocities.com/SiliconValley/7320/softw.htm
http://www.geocities.com/SiliconValley/7320/archives/gamma.zip
```

"Behaviour" section

This section contains options that change how SwiftJPEG behaves in general, when loading images, or when showing them in Full Screen mode or in the Desktop. At the top of the section you can choose a timer value to use for slideshows in Full Screen mode. Zero waits for a key press before moving onto the next image. Non-zero values give the number of seconds for which to wait before moving to another image automatically. The slideshow advances forwards through the images, or runs backwards if you press a key to view a previous image during the show – so you can have the slideshow advance forwards or backwards through the images as you see fit.

Underneath there are three options in a group. These alter the behaviour of SwiftJPEG when it loads something:

1. If "**Always show images in Full Screen mode**" is turned on, then whenever you load an image or group of images selected in the Filer, SwiftJPEG will change into Full Screen mode straight away. Otherwise, SwiftJPEG shows images in the Desktop unless you explicitly enter Full Screen mode (e.g. by double-clicking on the image view window). This setting has no effect on what happens when you drag a directory to SwiftJPEG.

2. If "**Always show directories in Full Screen mode**" is enabled, then when you drag a directory to SwiftJPEG, any JPEG images found inside it will be shown in Full Screen mode rather than queued for display in the Desktop. This setting has no effect on what happens when you load an image or group of images selected in the Filer.

3. If "**Don't queue directory contents by default**" is turned on, then when you load a single image from a directory, SwiftJPEG will *not* automatically try to find any other images in the same directory. Otherwise, it will do; for directories containing lots of files or slow filing systems this may take a long time, which is why the behaviour is optional. You can use the `Ctrl` (Control) key when dragging an image to SwiftJPEG to reverse the value set for this option for just that one image, without having to go into the Choices dialogue box and make a more permanent change.

Finally, the section holds another three grouped options that have various miscellaneous effects:

1. Using "**Close Display window after Full Screen view**" you control whether or not the image most recently shown in Full Screen mode stays loaded and visible in the Display window, or is closed, when you return to the Desktop.

2. If "**Use screen banks when not cross-fading**" is set, *screen banking* is used to cache images in Full Screen mode so that the change from one to another is near-instantaneous. This option is only available when cross-fading is *not* selected in the Full Screen section. For a more detailed explanation, see below.

3. "**Only find available screen modes once**" means SwiftJPEG will only find only find the biggest and most colourful screen modes available on your machine once, rather than trying to find them each time an image is shown in Full Screen mode. Usually this can be left on, but if you have a machine where the available modes change a lot at run-time turning the option off will let SwiftJPEG recheck for any new modes without having to be restarted.

The second option in the last group of three concerns the use of screen banks. RISC OS supports the idea of having more than one copy of your screen display. One copy can be redrawn whilst the monitor is displaying another, so the action of redrawing is hidden. When fade in and out is turned off, you see RISC OS plotting each image within SwiftJPEG (it 'rolls' up the screen). Using screen banks, the plot is hidden and the switch from one image to another is virtually instantaneous. Even with fading enabled this is useful, because it prevents there being a delay after one image has been faded out and the next one is being drawn – the next one is *already* drawn and just waiting to be faded in.

SwiftJPEG uses the time while you're looking at a 'current' image to load and draw the next one in the screen bank you can't see. Thus, you don't have to wait around when you want to move to that next image – it's instantly available. SwiftJPEG always assumes that if you just went to a 'next' image, it should load the next one after that too, or if you went to a 'previous' image, it should load the one just before that as well. However, if you change direction – say you were viewing images forwards, then asked to go backwards, or vice versa – there will be a delay, as the wrong image was cached in the screen bank, so SwiftJPEG has to throw it away and plot the correct one.

There has to be enough video RAM available to store two copies of your screen mode for banking to work, and the screen mode cannot change between banks. If you ask SwiftJPEG to find the best fit mode to the picture, banking can't work as the screen mode may change between images. If you ask SwiftJPEG to use the most colourful or

biggest mode, on some machines banking *might* work but usually it won't – your machine's video RAM is probably already almost full with just one copy of the highest resolution or most colourful mode that it is capable of displaying. This is where the ability to specify your own fixed screen mode comes in handy – if your machine can do one bank at 800x600 at 16M colours, say, it'll be able to do two banks at 800x600 in 32K colours.

Some extension graphics cards don't support screen banking – some versions of the Viewfinder firmware do not, for example. Screen banking is also less useful on machines with more limited graphics hardware as you might have to make heavy sacrifices on image quality to be able to fit two screen banks into your machine's available screen RAM. This is why it is turned off by default – image quality is prioritised over speed of display. That said, if SwiftJPEG cannot use screen banking, it just falls back to its normal way of displaying images. There are no ill effects from trying to use banking but not being able to, so if you like the fast switching effect, you can leave it turned on.

"Others" section
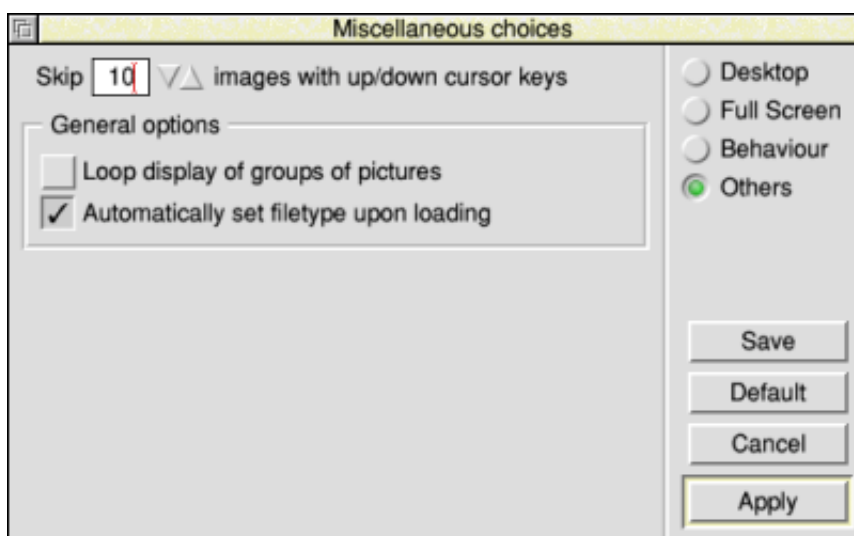
This last section contains options that don't really fit anywhere else. First, you can set the number of images that are skipped when pressing the up or down arrow keys rather than the left or right arrow keys to move between images in the File List. This affects both the Desktop and Full Screen mode. Underneath are two options:

1. "**Loop display of groups of pictures**" makes SwiftJPEG loop around through a File List indefinitely. It allows you to create a slideshow that repeats forever, or flip through from the start to the end of a group of images very easily (just go "backwards" from the first image). The option is turned off by default.
2. "**Automatically set filetype upon loading**" is turned on by default, making SwiftJPEG set the file type of any JPEG image that it successfully loads and displays but does not already have the JPEG filetype of &C85. If you want your files to be left with a different filetype, make sure you turn this option off.

Buttons

The four buttons at the bottom right of the window work as follows:

1. **Save**

   Permanently save the choices inside the !SwiftJPEG application directory and close the window.

2. **Default**

Restore the values shown in the window to default settings. You can then accept the defaults with the "Save" or "OK" buttons, or discard them with the "Cancel" button.

3. **Cancel**

Throws away any changes you may have made in the Choices dialogue box and closes the window.

4. **Apply**

Accepts the settings in the Choices dialogue box and closes the window. The settings are not saved permanently – quitting and restarting SwiftJPEG would lose them. To store changes permanently, use the "Save" button.

The ScreenFX module

As mentioned earlier, the ScreenFX module is required to support fades and cross-fades in Full Screen mode. This module is available from the SwiftJPEG download page or you can check for newer versions on the RISC OS Open Limited web site:

> http://hipposoft.pond.org.uk/
> http://www.riscosopen.org/

At the time of writing the module requires RISC OS 3.5 or later and must be installed in the following location:

> `!System.350.Modules.Video.ScreenFX`

If you do not have the `350` directory, the `Modules` directory inside it or the `Video` directory inside that, create them. ScreenFX is a relatively new module built with relatively new development tools and requires similarly new run-time support. As a result, users of any version of RISC OS that do not include a modern Shared C-Library module (RISC OS 5.03 or earlier and most RISC OS 4 versions) need to download the "System/zip" file from the Castle 32-bit page at:

> http://www.iyonix.com/32bit/system.shtml

The above page contains complete details describing the software and how to install it.

> *Not using ScreenFX?*
>
> If you do not want to go to the trouble of installing the updated C library and ScreenFX module, you can run SwiftJPEG without full screen fading support by commenting out the line that loads it in the application `!Run` file. To do this, add a vertical bar character ("|") at the start of the line. In versions 1.11 and later, you should do this on line 25.

Using RISC OS 3.1 or RISC OS 3.5

The RISC OS module "SpriteExtend" was extended in RISC OS 3.6 and upwards to provide the JPEG plotting routines upon which SwiftJPEG relies. To use SwiftJPEG on an earlier version of the operating system it is necessary to either load a later version of the module, or alternative support modules that provide the same facilities.

RISC OS 3.5

There are two fairly simple choices:

1. Use a new version of the SpriteExtend module. Version 0.99 is in fairly common circulation as it was distributed on the Acorn FTP site for a while, and so is still available in the FTP site archive (see link below). Later versions may also work.

   ```
   http://acorn.riscos.com/riscos/releases/spriteextend.arc
   ```

   This is a simple and self-contained approach that works well.

2. Use the JPEGRender and JPEGMsgs modules. These provide the JPEG plotting facilities for RISC OS 3.1 or 3.5. They were distributed with the Acorn web browser Browse, so if you can get hold of those copies, do so – version 1.02 of JPEGRender in particular. In fact v1.02 is a little better than SpriteExtend 0.99, and you won'' be replacing a core system module just for the JPEG features with this approach. An older version of the modules can be found in a universal !Boot archive from the same site as SpriteExtend:

   ```
   http://acorn.riscos.com/riscos/releases/UniBoot/
   http://acorn.riscos.com/riscos/releases/UniBoot/uniboot.zip
   ```

   The modules are in "`!Boot.Resources.!System.310.Modules`". These older versions aren't quite as good – no better than SpriteExtend 0.99, so the odd image that might work with the newer modules won't work with these – but still do the job by and large.

RISC OS 3.1

The choices are slightly different for this early version of the operating system.

1. Again, use the JPEGRender and JPEGMsgs modules. These are easily the best approach for RISC OS 3.1, though dithering will not work in 16 colour modes or less.

2. You could alternatively try to load a new version of the SpriteExtend module, such as v0.99 originally from the Acorn FTP site. However, this version of the module assumes an operating system feature called *Dynamic Areas* is available; RISC OS 3.1 does not have this feature. To make the JPEG plotting routines work, you need to use Justin Fletcher's Dummy Dynamic Areas (DDA, or 'Dummy') module, version 1.07 or later. This is a patch which provides the relevant features, though you cannot use a RAM disc (RAMFS) when it is loaded. Some JPEGs that might work with JPEGRender may fail with this approach, but on the plus side the DDA module keeps all of its working data in the RAM disc area, whereas JPEGRender dumps it in the RMA. RISC OS 3.1 users are normally all too aware of the module area

filling up or becoming too fragmented, so you may prefer to try this alternative method for that reason alone.

```
ftp://arcade.demon.co.uk/data/075000/077100/077108
```

Where do I put the modules?

On either operating system version you will need to put the modules in " `!System.310.Modules` " (new `!System` structure) or "`!System.Modules`" (old `!System` structure). The filenames are "`JPEGRender`" for the JPEGRender module, "`JPEGMsgs`" for the JPEGMsgs module, "`Dummy`" for the Dummy Dynamic Areas module and "`SprExtend`" for SpriteExtend.

Other issues for older operating systems

Note that on RISC OS 3.1 SwiftJPEG uses the system sprite area to store the File List. A side effect of this is that only one copy of SwiftJPEG can run at a time on RISC OS 3.1 – RISC OS 3.5 upwards does not have this restriction. You may notice a sprite called "`sjpg_filelst`" in the system sprite area; this is where the File List gets stored. **Don't tamper with this sprite!** SwiftJPEG will delete it on exit.

Full Screen mode and ViewFinder

Note that on RISC OS 3.1 SwiftJPEG uses the system sprite area to store the File List. A side effect of this is that only one copy of SwiftJPEG can run at a time on RISC OS 3.1 – RISC OS 3.5 upwards does not have this restriction. You may notice a sprite called "sjpg_filelst" in the system sprite area; this is where the File List gets stored. Don't tamper with this sprite! SwiftJPEG will delete it on exit. 12. Using SwiftJPEG's Full Screen mode and ViewFinder If you are using SwiftJPEG with the ViewFinder graphics card in the Desktop, things should work as for any other Desktop application. However, if you want to perform Full Screen mode operations, please note that the level of integration that ViewFinder provides with the OS is insufficient for SwiftJPEG to be able to determine best available screen modes. It will tend to select only modes that were available under RISC OS. When returning to the Desktop, SwiftJPEG will have correctly read the previous mode information, but will not be able to restore it unless it could be displayed without your ViewFinder card present; you will be left in the last selected Full Screen mode.

It is possible to specify ViewFinder-only resolution, colour depth and frame rate combinations using the mode string in the "Full screen mode" menu in the General Choices window. Unfortunately, the operating system will claim that these modes are not available when SwiftJPEG checks for the string's validity, so you must only directly specify screen modes that would work without your ViewFinder card being present.

If you ask SwiftJPEG to enter modes of 4 or 2 colours, it will do so, but under certain ViewFinder card types no usable display will be produced as some do not support 2 or 4 colour modes. If your previous Desktop mode was suitable for ViewFinder only, you will then end up stuck in the 2 or 4 colour mode when you go back to the Desktop. You'll need to use the command line to restore a usable display (e.g. F12, `WimpMode X800 Y600 C256 F60`").

If ViewFinder in future presents a deeper integration with RISC OS, these problems should go away by themselves. It should be noted that this lack of integration is not a bug in ViewFinder – it's simply not implemented, and if it were added in future, it would be added as a feature addition rather than a bug fix. In the mean time, the best bet is usually to select a resolution and colour depth you think would be most appropriate for the batch of JPEGs you wish to view, put the Desktop into that mode, and select "Use current mode" in the "Full Screen mode" menu in the Choices window.

These comments refer to using SwiftJPEG v1.10 with ViewFinder firmware version 1.47. At the time of writing, I have not tested with earlier or later versions of either.

Miscellaneous notes

Use of `Alt+Break`

If you use Full Screen mode with bank switching enabled and often kill applications using `Alt+Break`, you should read this note.

A small ARM code routine is installed whilst a JPEG is plotting, which sits looking for key presses whilst the machine is otherwise busy doing the image plot. This allows key presses to be recognised in Full Screen mode when an in-the-invisible-bank banked image plot is happening; you see an hourglass appear until the plot completes, then the key press gets acted upon. Because SwiftJPEG is a Wimp application the period for which the handler is installed *must* be minimised – `Alt+Break` could kill the application with the handler still present, and crash the machine. This does mean that for large images (in terms of file size) which may take some time to load, there is still a period when a banked image is *loading* but is not yet *plotting* when key presses may be ignored. So if key presses seem to be unresponsive when using image banking, check and see whether this is just because you have large image files that your machine is taking a while to load (you should find that with image double-banking turned off, keys are very responsive once the JPEG image has finished plotting).

Caveat – ***never use `Alt+Break` to kill SwiftJPEG when it is running in Full Screen mode*** unless you are *sure* that it is *not* in the middle of plotting an image off-screen. If you use `Alt+Break` whilst a banked JPEG plot is in progress, the interrupt will happen immediately after the plotting SWI completes but before the handler gets removed. If you choose to kill SwiftJPEG, ***your machine will crash***. It is perhaps a shortcoming of RISC OS that utilities like this require implementation within modules rather than applications to run safely.

Troubleshooting – stripy images and ColourTrans aborts

If you find certain JPEGs do not plot properly – they are stripy – then they've probably been rotated, using a flag in the image header. SpriteExtend 0.99 and JPEGRender up to 1.02 do not handle this properly. However, a patch is available that fixes the problem. Search on Google Groups (`http://groups.google.com/`) for details.

Some people reported aborts in ColourTrans with RISC OS 3.1 under Virtual A5000. I have tested JPEGMsgs 0.01/JPEGRender 1.00, JPEGMsgs 0.01/JPEGRender 1.02, and SpriteExtend 0.99/DDA 1.08 beta with the ROM ColourTrans and all worked. I then soft-loaded ColourTrans 1.57 as well; now only the newer JPEGMsgs 0.01/JPEGRender 1.02 combination worked – both the older JPEGRender and SpriteExtend failed with an abort. If you see aborts, check to see if your ColourTrans module has been updated, for example by other installed software.

Contacting the author

Snail mail:

    Andrew Hodgkinson
    52 Beaulands Close
    Cambridge
    CB4 1JA

E-mail:

    `ahodgkin@rowing.org.uk`

On the web – SwiftJPEG:

    `http://hipposoft.pond.org.uk/`

On the web – personal, hobbies, ventures:

    `http://pond.org.uk/`
    `http://nines.rowing.org.uk/`
    `http://www.riscosopen.org/`

Thanks

Hipposoft would like to thank (in no particular order!):

Graeme Barnett and Neil Coffey for feedback relating to SwiftJPEG in the early stages, and particularly Stuart Halliday and Stephen Borrill for various suggestions and bits of info; and everyone else who has e-mailed me about various things since then, including Stuart Bell who made me realise I shouldn't completely drop the Junior manual format in favour of HTML (nearly ten years on, I've finally crossed into the world of PDF!), and was instrumental in getting SpriteExtend released for RISC OS 3.5 users.

John Sullivan and Ben Blaukopf for help on user interface and some programming issues, and for ideas on improving the HTML manual.

Chris Cox of Acorn for a very positive response to my request for a distribution of the JPEG-aware SpriteExtend for RISC OS 3.5 machines, and Charles Whyte of ART, to whom the request eventually got transferred (unfortunately it required a full software license and I couldn't afford the fee); and of course Tim Caspell who later pursuaded ART to do the public release on the Acorn FTP site!

Ragnar Hafstað and Dick Alstein for !BasCrunch v1.02, which was used to compress the original BASIC source.

André Timmermans for pinning down the Select 2 / 4.32 JPEG_Info call bug.

Terry Blunt, Bernard Veasey, Richard Hallas, Paul Sprangers, Daniel Ellis, Chris Walker, Stefan Bellon and Dave Higton for putting up with beta versions and providing invaluable suggestions, requests and bug reports.

Castle for opening the RISC OS sources, including the previously closed ScreenFX module on which SwiftJPEG now relies.

An ever-increasing number of others for feedback, feature requests and beta testing.

Development history

Version 1.13 (07-Aug-2012)

Fix for 'Outside File' errors found in JPEGs from an HTC Desire (with test images from and reported by Dave Higton).

Version 1.12 (01-Aug-2007)

Fix for CompoSupport module integration problems.

Version 1.11 (30-Jul-2007)

Rewritten, pane-based Choices window. Ultra-funky 12bpp cross-fade support. More control over the image window behaviour in the Desktop. Various bug fixes and tweaks. An incremental release despite the number of years (!) between this and the previous formal release, due to lots of other things taking priority over development. Feature requests and bugs still being tracked on pond.org.uk, hopefully to get implemented at some future date.

Version 1.10 (11-Sep-2004)

Incorporating changes from 1.10 beta versions, plus a rewritten PDF manual, slight rearrangement of the Choices window template and change to the !Run file to load SpriteExtend 0.99 as well as JPEGMsgs/JPEGRender/Dummy. Version is feature complete according to informal specification of new features based on feature requests and ongoing beta testing.

Version 1.10-B5 (June 2004) Known problems

Menus aren't all closed when a new image is loaded. Since it doesn't seem to be possible to close a specific single menu, this does mean that the main menu, if open, ends up "behind" the display window if a new image is loaded using, say, the cursor keys. It's the lesser of two evils as it allows Director menus and similar to stay open.

Version 1.10-B5 (June 2004) New features

Position of mouse pointer is restored as well as the screen mode when exiting full screen view (this might be considered a bug fix).

The Zoom window for a loaded image now carries three radio buttons, which when the window is first opened after loading a new image will reflect the zoom settings for the Desktop as specified in the Choices window. You can modify the zoom setting for just the loaded image using the radio buttons and the zoom values if you wish, and an "Only < 1:1" option button - which means "don't automatically scale up above 1:1" - is present and used for the "Fit to area" and "Fill area" choices. This all matches similar options available in the Choices window. When you reload the image or load a new image, it will revert back to the Choices settings. When the temporary zoom setting is changed, the Display window resizes to fit the new settings, both increasing or decreasing size as required. In earlier versions the window tried to maintain the previous size which wasn't usually what you wanted it to do. Note that any temporary zoom setting in the Desktop is reset to the Choices values if you go into and back out of Full Screen mode.

Double-clicking with Select or Adjust on the Display window enters Full Screen mode, just as if "Full screen" were chosen from the menu. Dragging with Select in the window allows you to save the current image with a default filename (the same leafname as when the image was loaded, or ScrapFile for application-to-application transfer items). This makes it easy to work through a set of images and see if you want to keep them - save the "keepers" into another directory, say, and just drag from the window directly without having to use the menu and Save As dialogue box.

Title bar in "Save as" window change to "Save copy" to highlight that a copy of the image is being saved; SwiftJPEG still reloads the image from its original location and continues to load other images from the group / directory related to the original image location.

Doesn't reload the current image when going into Full Screen mode now.

Input focus is given to the Display window if it is reopened by clicking with Select on the icon bar. An iconiser is told that the window has closed too, as this seems to be the only way to get it to remove the icon if we re-open the window "under its feet". Broadcasting this Message_WindowClosed message seems to be the defined way to do it, but the window isn't closed at all. This is a broken part of the protocol.

The Close icon has been removed from the info windows; whilst I prefer to have it as the windows have no other buttons (unlike Save As), on RISC OS 4 it introduces an iconise icon which if accidentally clicked on can cause problems. Iconising with Shift+Click on the close icon also gives undesirable side effects so it's better to get rid of it.

For pictures with EXIF information - typically those taken by digital cameras - the camera make and model, and the picture's internal record of its creation/modification date, will be extracted where possible and shown in the picture information window. The date stamp for the actual physical file itself is also shown (the two usually differ because the picture has been copied out of the camera onto some filesystem some time after it was originally taken). In Full Screen mode, the information overlay now shows the picture's creation/modification date if it is available, else it shows the date stamp for the file.

Pressing "I" when the Display window has input focus now opens the picture information window (as a menu) next to the pointer.

Version 1.10-B5 (June 2004) Bugs fixed

Fade in/fade out when used with screen banking appeared to not work, as you had to wait for an image to plot whilst faded out. This was actually waiting for the *next cached bank* image to load, because the fade in was done after, rather than before, the cache call. Moving the point where the fading is done gives a much more pleasing result with the fade in and out appearing to move from one image to the next instantly and the hourglass shown when key presses are made during plotting is now visible rather than hidden by the gamma fadeout.

Information overlay in Full Screen mode didn't appear reliably when screen banking was used - if the relevant key was pressed during the plot of the off-screen image, it would fail because of a mistyped key number in the key handler.

Checking of mouse buttons could be overridden by OS_Byte values in a key scan loop in Full Screen mode; the right and middle mouse buttons would operate erratically as a result, depending on whether OS_Byte or a MOUSE call caught the button change. Now MOUSE overrides OS_Byte.

Since the split between ignoring errors in the desktop or in Full Screen mode was introduced, occasionally a load error from Full Screen mode might be seen even if the relevant 'ignore' setting was turned on. The related code has been tidied up and this bug fixed in passing.

When an image was loaded from a directory and the directory contents cached, subdirectories would be included. This could lead to "not found" errors if stepping through to such an item. Subdirectories are no longer stored when storing directory contents based on loading a single image.

If the dithering method was changed in the Choices window, the display window didn't redraw fully. It now does.

Caret handling was a little eratic; SwiftJPEG endeavours to restore one level of previous caret position, but for various reasons would not do this reliably. Behaviour has been considerably improved.

Bounding box of drag for saving items was set to the drag box, rather than the pointer, due to incorrect flags to Wimp_DragBox. Correct flags are now set.

Typing error meant application to application transfers would fail with a "Mistake at line..." report. One other instance of same class of error found and fixed which would only show up when a particular out of memory report should have been raised.

Close icon removed from save box and button sizes reduced to improve style guide compliance.

The icon bar clicking behaviour has changed, now that it is very easy to get into Full Screen mode with so many key press and clicking options available - Select re-opens the display window as before, if an image is loaded; Menu opens the main menu; and Adjust opens the Choices window.

Double-clicking on a JPEG image to load SwiftJPEG now does include the directory contents as well as just the image, which was the previous behaviour. I can't remember if this was intentional or an oversight, but it's easy to change if you want to - see around line 220 in the uncompressed !Runimage source.

If "Don't scale up" was set, then fixed zoom settings of > 1:1 would be ignored as well as the "fit to area" and "fill to area" scaling. It is now possible to set a >1:1 scaling when "don't go above 1:1" is set, as this flag is only intended to

apply to the automatically calculated zoom settings, not the manually specified values.

Version 1.10-B4 (31-Apr-2004) New features

Optional dual bank support, to load the next JPEG into the offscreen bank whilst you look at the current item (if you move forwards, it'll cache the next item; if you start going backwards, it'll cache the previous item; so when changing direction, the cached item has to be discarded and there will be a delay before the next image is shown). This is turned off by default - use the Choices window to change the setting.

F and keypad Enter exit Full Screen mode; now in the Desktop, if the Display window has focus, they enter Full Screen mode too (because of the Wimp's key numbers, Return and Enter both go into FS mode, but only Enter exits it (intentionally - Return is commonly used as a "next image" key). ESCcape closes the display window, if it has input focus. The Display window can be set to grab input focus on loading a picture (indeed, this is now the default); use the Choices window to change the setting.

Up and down arrows in the Desktop or Full Screen mode, for groups of pictures, advance in steps of 10 (by default; you can change this value in the Choices). Images not already marked bad will be included in the count even if they fail.

The space bar behaves the same way as "P" now (pausing in full screen if a timer is set, then when paused, moving to the next image; or if no timer is set, just going to the next image).

Overlays are briefly shown when zooming in Full Screen mode to indicate the zoom type. Two equal sized rectangles with an equals sign between them show 1:1; if the left or right rectangle is larger, the zoom mode is taken from the Desktop scaling and shows a scale up (right hand rectangle is larger) or down (left hand is larger); a rectangle between two horizontal bars depicts the picture scaled to fit within the boundaries of the screen, whilst a rectangle overlapping two horizontal bars depicts a picture scaled to fill the screen, cropping if necessary.

Adjust-click in Choices doesn't close the window now.

New "don't zoom above 1:1" option for people who want to scale to fill the screen only for images too *big* to fit, but don't want to zoom up and get a blocky image. When using the "Z" full-screen mode zoom key, this doesn't apply (i.e. scaling above 1:1 will be allowed if you explicitly ask for it).

Keys 0-9 on the top row or the keypad set zoom factors in Full Screen mode. 1 to 5 set 1:1 through 5:1; 6 to 9 set 1:2 through 1:5; 0 sets 1:1 again.

Information panel in full screen view ("I") is at least 16 characters wide normally, but if a leaf name for a picture is greater than this, it attempts to expand to accommodate at least the leafname. The width may be truncated below this, though, if the pixel width of the screen mode used requires it. The panel now shows the current image number and the total number of images, though if SwiftJPEG later reads thus far unread images and discovers they cannot be shown, the total may decrease.

Double-clicking on an image, or dragging a single image to SwiftJPEG, now doesn't add just that image to the file list. So long as Shift isn't held down (to append the item to an existing list, or to start a new list with just that one picture), the contents of the directory in which the image lies are added. This means you can use the next and previous functions to step through any other images in that directory. If you have "show images in Full Screen mode on loading" switched on, and "ignore errors in Full Screen mode" on too, then dragging a bad image, or a non-JPEG image to SwiftJPEG will result in it looking forwards in the directory and if another viewable item is found that will get shown instead. This is just a (sometimes useful, sometimes not) side effect of loading a directory contents rather than just a single image. This automatic directory loading only happens when you load individual images - if you drag a group to SwiftJPEG, then only the items in that group will be queued.

You can, in the Choices window, tell SwiftJPEG to load *directories* into Full Screen mode on loading, but not individual images. With this set, images are shown automatically in Full Screen mode when you drag a directory to SwiftJPEG, but not single images or image groups. The setting only affects the way in which images are shown when the directory is first dragged to SwiftJPEG.

Holding down Ctrl when loading a single image defeats directory loading (useful in particular if it's a big directory or a slow filesystem).

When loading directory contents for any reason, only files with the JPEG filetype already set are read.

In Full Screen mode with a slideshow timer set, using the left and right arrow keys changes the direction of slideshow

playback too. An on-screen indication of the change of direction is shown briefly on screen.

There is now a set of "ignore errors" and zoom options that apply to the Desktop, rather than only being available in Full Screen mode. A default fixed zoom (m:d) ratio can be set, the image can be made to fit the maximum available window visible area in the current screen mode without having to scroll to see part of the image, taking account of the "don't cover icon bar" CMOS setting, or fill the maximum window visible area, with scrolling as necessary according to the maximum visible area's aspect ratio versus the image's aspect ratio. The old Zoom window on the icon bar or Display window menus still works - it will override the Choices and set whatever zoom setting you put into it. Reload the image to reset the zoom to the Choices-specified value.

Version 1.10-B3 (27-Apr-2004) Bugs fixed

Only sets filetype of an image (1) if loading it was completely successful and (2) if it isn't already correctly set (previously, could sometimes try to set a filetype on a non-JPEG and would always re-set existing types).

Gamma fade in/out wouldn't quite reach full or minimum intensity, particularly at fast speeds (had forgotten to put final set-black or set-normal calls in all versions of SwiftJPEG prior to this).

When loading a directory, only queues images of filetype &C85 rather than accepting any filetype.

Version 1.10 B1 (25-Apr-2004)

Many internal changes and bug fixes, particularly for screen mode selection. Much more stable on RISC OS 3.1 if using JPEGMsgs / JPEGRender; uses the system sprite area for file lists and application space for the JPEG, so the 4MB picture size limit imposed by the sprite area maximum size is removed.

Rectangular pixel modes (e.g. wide pixels such as mode 2, or tall pixels such as mode 15) fully supported on all OS versions. "Ignore errors in Full Screen mode" option. Range of Full Screen mode zooms supported - 1:1 or same as Desktop, as before, along with "fit to screen", which scales to fit the current mode without cropping, and "fill screen" which fills the screen, cropping if necessary. Can cycle through these modes in full screen view by pressing "Z". Can now step forwards and backwards through picture lists with either the Main Menu options, or left/right keys in Full Screen mode or in the Desktop (click on the Display window to give it input focus).

Choices window rearranged to a more logical layout given the new options. Version taken to 1.10 from 1.02 to reflect the significant number of internal changes.

Version 1.02 (04-Dec-2002)

RISC OS 3.1 support (does not use dynamic areas internally). Source code indented for clarity. Minor bug fixes in mode selection routines. Avoids setting Wimp palette to fix "pinboard re-caches the backdrop sprite once for every viewed image" problem in Full Screen mode. Does not do strict "is this a JPEG we can handle?" check for RISC OS 4 upwards. On-screen displays in Full Screen mode work for 4 OS unit per pixel modes now (e.g. mode 13). Better behaviour in 16 colours or less. Supports JPEG Render or Sprite Extend modules for JPEG rendering. On- the-fly conversion of EXIF and unidentified images to JFIFs - digital camera output now viewable directly. Help file reformatted to fit in 78 characters wide - seemed like a good idea at the time...

Version 1.01 (24-Oct-2002)

Bug fix - SWI JPEG_Info was being called with R0:0 clear, which according to the PRMs means it should not return size information. In RISC OS versions prior to 4.32 it does, but in 4.32 or later it does not, resulting in a failure to display JPEGs properly. This was reported in newsgroup comp.sys.acorn.apps with André Timmermans finding the bug (thanks).

Changed date format within this !Help file to DD-MMM-YYYY.

Version 1.00 (25-Apr-1996)

As version 1.00ß (below), with additions of the new Dynamic Area for filenames of groups. This allows a 'Next image' menu item, where a group of pictures dragged on can be viewed one by one in the Desktop rather than ending up with the last picture. You can drop in and out of Full Screen views of a group or cycle round all the files repeatedly Full Screen. So revision of Full Screen procedure, error handling and loading mechanisms, plus addition and completion of HTML documentation and rebuilding of the text and Junior manuals was conducted.

As such this version shows some of the most radical internal changes in the version history so far. Unfortunately the WimpSlot had to increase as a result of the added functionality (mostly the new loading procedures - gamma fades have little memory overhead which is why they're still present). It's only gone up to 56K (from 52K), which is the size it was at in earlier versions anyway... There's also a reasonable amount of room left in the extra 4K for small future additions or fixes should they be required. Documentation (and perhaps sources) may well be released 'officially' in separate archives with filenames making the format of their contents obvious...

Version 1.00ß (Unreleased)

Radical revision to Choices arrangements, including support for hardware fade/flash in, and checking for presence of the Compo_ReadGamma SWI to maintain any previous Desktop gamma settings. Dithering and Full Screen options moved to menus, the latter with a specific mode selection added. Problems with reopening Display window solved as far as possible.

A few minor bugs cured and documentation is now in HTML form.

Minor improvements to polling system and way the return to the Desktop screen mode is both handled and prompted - PollIdle now restored (an unstable intermediate between version 0.09 and 1.00 which was not completed and given no version number had to have PollIdle replaced by Poll - less efficient).

Version 0.09 (21-Jan-1996)

Major changes are slideshow time delay and an 'always plot at 1:1' option. A 'Reload image' menu item has been added and there's been 'debouncing' on a few buttons. The Choices I/O mechanism had to be adapted to cope with the new stuff in the Choices window. Memory handling changed (see below). Save/App-app load (see below).

With multiple file loads and auto Full Screen, can now click Adjust to 'pause' the image when slideshow wait isn't zero - that is, the image will stay on screen until a key/button is pressed. A pause symbol is flashed 3 times in the bottom left of the screen to indicate this.

App-to-SwiftJPEG transfer supported through !Scrap; JPEG can be saved out. Allows e.g. UUDecodes from !TTFN to be dropped straight into SwiftJPEG and saved out if liked.

WimpSlot taken down from 56 to 52K, and JPEGs now placed in a Dynamic Area.

Changes to many accompanying texts/manuals, and any files where 'Hipposoft 1995' appears have been altered for '1995,1996'...! Noticed Info window had '1993-1995' in it, this now reads '1995-1996'. Remembered to spellcheck the manual and found quite a few errors... Now corrected.

Version 0.09i

Unreleased intermediary. A couple of additions made over version 0.08 which gave me new ideas for the 'proper' version 0.09. (The 'i' stands for 'internal').

Version 0.08 (12-Dec-1995)

Changed loader routine to be filetype-independent. JPEG (&C85) filetype now only needed for double-click loading. Also added proper error handler; various circumstances will force a quit but most of the time, there is a choice of OK or Cancel buttons. Or just OK, if it's only a message. These changes were suggested by Stephen Borrill.

Added 'Auto Filetype' option, to set the filetype of a non-JPEG filetype JPEG file if it was loaded. Noticed a minor bug in the Choices save routine connected with diagnostics, which was fixed (this would never have lead to incorrect preferences saves though).

Fixed minor bug in window centering routine (!) which manifested itself when I set CMOS RAM to allow windows off-screen in all directions. Single pixel errors when accounting for scroll and title bars also fixed.

Updated manuals and the odd minor change to above History text.

Version 0.07

The original text History file itself was created on the same day as Version 0. 07 was first saved to disc.

On request of Stuart Halliday, the title bar of the display window now holds all or part of the pathname of the JPEG file loaded.

A 'true' ellipsis was tried instead of '...' for display of pathnames to long to fit in either the Picture window or the title bar of the Display window. However, this doesn't look good for system font users, and only lets you see another 2 characters of the pathname. I decided to stick with three full stops.

The REMs at the start of the application's crunched !RunImage (which are inserted manually after the crunching process) have been made a little more informative.

Fixed a rather stunningly silly bug where Adjust-clicking on the close icon would open progressive parent directories, and Shift-Adjust- Clicking would result in a crash as soon as the JPEG tried to subsequently redraw...

Version 0.06

First 'Release' version, distributed to a few people over EMail by specific requests following a mention of the software in an Acorn- related newsgroup.

This was the first version that conformed to the original design specification drawn up prior to starting to build the application.

Versions 0.05 to 0.01

Development versions. A new version number was assigned after a complete, working copy of the software had been established (v0.01 was the version name used for all initial development). None of these versions were released. Fairly major additions were generally made between version numbers.

None of these versions fully implemented the original design specification for the software.

Disclaimer

If you do not agree with the following text, you may not use SwiftJPEG and must delete it immediately.

"The software" or "this software" refers to the !SwiftJPEG application and any related files distributed with it, including this manual in all of its various formats.

This software is supplied "as is"; Hipposoft makes no claims as to its reliability or suitability for any purpose, regardless of any comments made elsewhere. No responsibility can be taken for any failure of the software, or any loss or damage caused directly or indirectly, or not caused, by the software.

Contents of this manual are not guaranteed to be correct, and may change without notice.

This software is Freeware, and must be distributed intact, with all the files present and in the same form as received by the end user. If being distributed in any profit-making way, permission must be granted by Andrew Hodgkinson (Hipposoft) first.

This revision of the Disclaimer and Credits section was made on Tuesday the 7 th of August, 2012 and refers to SwiftJPEG v1.13 and the manual presented herein. It does not refer to versions of SwiftJPEG or the manual made prior to this date unless explicitly indicated.

Andrew Hodgkinson 2012
ahodgkin@rowing.org.uk