ImageStudio

Shareware image processing and conversion package for the Amiga.

by Andy Dean and Graham Dean

1 Introduction

This chapter gives a brief introduction into the features offered by the program.

1.1 Copyright and Disclaimer

No guarantee of any kind is given that the programs described in this document are 100% reliable. You are using this material at your own risk. The authors **can not** be made responsible for any damage which is caused by using these programs.

The unregistered package is freeware, but still copyright by Andy Dean and Graham Dean. This means that you can copy it freely as long as you don't ask for a more than nominal copying fee.

The registered version of the program and its associated keyfile **may not** be freely distributed.

The GIF loader/saver module (gif.isio) remains freely-distributable for both the registered and unregistered versions of the program. No charge is made for this module.

Permission is granted to include the unregistered package in Public-Domain collections, especially in the excellent Fred Fish Amiga Disk Library (including CD ROM versions of it). The distribution file may be uploaded to Bulletin Board Systems or FTP servers. If you want to distribute this package you must not remove or alter any of the supplied files, although the files may be re-compressed using any Amiga archiver program.

This program (or parts of it) may not be included or used in commercial programs unless by written permission from the authors.

Installer and Installer project icon (c) Copyright 1991-93 Commodore-Amiga, Inc. All Rights Reserved. Reproduced and distributed under license from Commodore.

INSTALLER SOFTWARE IS PROVIDED "AS-IS" AND SUBJECT TO CHANGE; NO WARRANTIES ARE MADE. ALL USE IS AT YOUR OWN RISK. NO LIABILITY OR RESPONSIBILITY IS ASSUMED.

amigaguide.library (c) Copyright 1991-93 Commodore-Amiga, Inc. All Rights Reserved. Reproduced and distributed under license from Commodore.

AMIGAGUIDE SOFTWARE IS PROVIDED "AS-IS" AND SUBJECT TO CHANGE; NO WARRANTIES ARE MADE. ALL USE IS AT YOUR OWN RISK. NO LIABILITY OR RESPONSIBILITY IS ASSUMED.

Jonathan Forbes, the author of LX, has granted permission to include the unmodified Version 1.03 LX executable by itself (i.e. without documentation) with any PUBLIC DOMAIN or SHAREWARE package, provided that a brief credit note is included in the program's documentation (see Chapter 9 [Credits], page 145).

1.2 Machine requirements

ImageStudio requires the following system to run:

- Workbench 2.04 or above.
- Around 1 megabyte of free memory.
- Several megabytes of free hard disk space.

If ImageStudio is run on an AGA machine (A1200 or A4000), it will use the new display modes and palette routines to improve the quality of the internal viewer images.

1.3 Brief description

ImageStudio is written for the casual graphics user who wishes to convert or manipulate various graphics formats on a modest Amiga system. There are several commercial offerings available, however the casual user is paying a lot of money for many facilities and options they would probably never use.

Bitmap graphics, by their nature, usually require large amounts of RAM. One of the main objectives of ImageStudio was to reduce this burden by utilising virtual memory - most users have more spare hard disk space than spare RAM.

ImageStudio works with several buffers at any one time (dependant on how many levels of undo are specified), each of these buffers can hold either colour-mapped or 24-bit images. For a detailed description of colour-mapped and 24-bit images, See Section 5.3 [Image types], page 122.

ImageStudio comes with a fully featured ARexx port, which allows the writing of macro scripts to automate repetative processes and also allows communication with other ARexx aware programs. Again, ARexx has the reputation of being “something pretty difficult”, so we’ve simplified the use of ARexx in ImageStudio by allowing the user to create, edit and run scripts entirely within ImageStudio - ImageStudio even provides a blank template as a starting point for each new script.

1.4 List of features

General:

- Full 24-bit image buffers, with optimizations for colour-mapped (palette based) images.
- Up to 100 levels of undo / redo.
- User configurable virtual memory.
- Fully featured, easy to use, ARexx interface.
- Fully font sensitive, style guide complient, user interface.
- Fully CyberGraphX compatibility for both the viewers and the preview window.
- Modular loaders and savers.
- User saveable preferences.
- Operations applicable to the whole image or a selected region.
- Up to 256 greyshade preview window (with optional dither).
- Zoom on preview window.
- Internal / external viewers.
- Loading / saving / manipulating of AGA image formats (e.g. 256 colours, HAM8) on non-AGA machines.
- Max image size of 32000 x 32000 (crops images to 512 x 512 in the unregistered version).
- Copy / paste to / from the system clipboard.
- Runs on all Workbench 2.04+ Amiga’s - utilises AGA chipset if available.
- Online AmigaGuide help, as well as ASCII, $\TeX$ ‘`.dvi`’ and PostScript documentation.
- Multi-level help error requesters.
- Public screen.
- Requires no third party libraries or utilities.

Import:

- IFF-ILBM formats (Standard palette based, HAM6, HAM8, extra half bright, ILBM24)

- BMP
- Datatypes (with Workbench3.0 and above)
- GIF (conforming to GIF87a or GIF89a)
- IFF-DEEP
- JPEG (conforming to JFIF standard)
- PCX
- PNM (PBM, PGM, PPM, ASCII and binary)
- QRT (DKB, POV)
- SGI
- Targa
- TIFF
- VMEM (internal virtual memory format)

Export:

- IFF-ILBM formats (Standard palette based, HAM6, HAM8, extra half bright, ILBM24)
- BMP
- EPS
- GIF (conforming to GIF87a or GIF89a)
- IFF-DEEP
- JPEG (conforming to JFIF standard)
- PCX
- PNM (PBM, PGM, PPM, ASCII and binary)
- QRT (DKB, POV)
- SGI
- Targa
- TIFF
- VMEM (internal virtual memory format)

Colour Balance:

- All operations are available to the R,G,B components separately.
- Brightness (upto 100%)
- Contrast (non to full)
- Gamma (+ and -)

Convolution:

- Many supplied convolution filters.
- User definable convolutions, load and save to disk.

Effects:

- Built in effects: Dynamic range expansion, FlipX, FlipY, RollX, RollY, Negative, Greyscale, Highlight, Shadow, Random, Pixelize, Remove isolated pixels.

Scale:

- Crop to selected region.
- Increase / decrease scale by percentage or absolute image size.
- Simple scale or colour averaged.

Colour reduction:

- Increase colour mapped images to 24-bit.
- Decrease number of colours in 24-bit or colour mapped images via Heckbert median cut algorithm.
- Dithers available for colour reduction: None, Floyd-Steinberg, Burkes, Stucki, Sierra, Jarvis, Stevenson-Arce.

Palette:

- Edit palette colours and ranges.
- Save current palette.
- Force palette onto current image, dithering if necessary (all dith ers available).

1.5 Shareware version

To encourage users to register, the freely distributable version of ImageStudio will crop images to 512x512 pixels. All other operations are available.

For details on how to register, See Chapter 8 [How to register], page 144.

1.6 Starting ImageStudio

ImageStudio can be started from either the Workbench or CLI. From the Workbench it is simply a case of double-clicking on the icon. ImageStudio supports shift-clicking on a file to start the program with (see the Workbench manual for more information).

To start ImageStudio from the CLI, simply type:

```
run ImageStudio [filename]
```

where '`filename`' is an optional file to load in at startup. The full range of tooltypes is supported, and can be appended to the CLI command. For example:

```
run ImageStudio "SCREENNAME=Image2" "PREVIEWDITHER=YES" [filename]
```

would start the program on a public screen named 'Image2' with pre view dithering on. See Section 5.5 [Tooltypes], page 123, for a full list of available tooltypes.

1.7 Upgrading from version 1.x.x

All versions of ImageStudio from version 2.0.0 require a *keyfile* to work fully.

Users who have registered the ImageStudio package after v2.0.0 will be provided with a unique keyfile that can be used immediately.

Users who have registered v1.x.x of the ImageStudio package will have to create their own keyfile using the 'Create keyfile' option under the 'Project' menu. See Section 3.1.8 [Create˙keyfile], page 17, for more details.

The idea of the keyfile is that once a keyfile has been created, it can be placed somewhere where all the programs in the ImageStudio package can access it. This is typically either the ImageStudio drawer or the 'S:' directory. The user should never need to create another keyfile, it should work with all future versions of the software; for this reason, it is suggested that the keyfile is backed up and kept somewhere safe, as we will be unwilling to give out replacements should you loose it.

It should also be noted that each keyfile is unique to each user, and your keyfile should not be given to others. If the number of registrations should drop due to people abusing the keyfile scheme and distributing pirate versions of the program, we shall be forced to do one or more of the following:

1. Stop providing free upgrades. All programs will be individually stamped and all upgrades must be paid for.
2. Remove online help and provide printed manuals instead. This will naturally raise the price substantially.
3. Sell the software to a software publisher for commercial distribution. This is likely to lead to a large price increase.
4. Stop developing ImageStudio and related products.

We see no reason why we should have to resort to any of these measures, but if you're using a pirated copy of ImageStudio now and you refuse to register, you **WILL** force this upon us.

If you are using a pirate copy of ImageStudio, so will hundreds of others. A hundred orders to us is the difference between us writing the next version of the program, and us abandoning it. It really is your choice.

1.8 Configuring ImageStudio

In order to benefit from ImageStudio's virtual memory, it is recommended that the default location used for the storage of the temporary files is changed. The default location for the storage of these files is '`T:`' which is usually in RAM - we want to move this out onto harddisk.

To do this, select 'Prefs' from the 'Project' menu and open up the prefs requester. Change the TEMPDIR preference to the desired location for the temporary files. See Section 3.1.4 [Prefs], page 14, for more information on changing preference variables.

It is suggested that a drawer be made on a harddisk partition with lots of space to store these files. For example, make a drawer in your '`Work:`' partition called '`tmp`', and change the TEMPDIR preference to read '`Work:tmp`'.

2 Tutorial

This chapter introduces ImageStudio by way of a few tutorials demonstrating commonly performed operations.

2.1 Changing the image format

The simplest use of ImageStudio is just as a file format converter; See Section 5.1 [File formats], page 104, for details about the supported file formats.

In this example we will change the image format of the ‘`FW14B_250x250.gif`’ from GIF to IFF-ILBM.

1. Load the file ‘`FW14B_250x250.gif`’ from the ‘`Pics`’ drawer. To do this, select ‘Open’ from the ‘Project’ menu. When the file requester opens, select the file and it will load into ImageStudio. The greyscale preview will show the image.
2. The Infobar at the bottom of the screen shows the current image size and number of colours, as well as a fuelgauge showing progress when applicable. The current co-ordinates of the pointer are also shown when the preview window is active.
3. Open the save requester. To do this, select ‘Save’ from the ‘Project’ menu. A requester will open, containing (amongst other things) possible save formats.
4. Select the file format to save. To do this, click on ‘IFF-ILBM’ in the listview.
5. Change the filename to avoid overwriting the original file. To do this, type the new filename - ‘`FW14B_250x250.ilbm`’ into the ‘Filename’ string gadget.
6. Save out the file by clicking on the ‘Save’ gadget.

The file will now be saved out as a 256 colour IFF-ILBM onto the disk.

Note:

- All time consuming operations show their progress in the Infobar’s fuelgauge and can be aborted by clicking on the ‘Abort’ gadget.

2.2 Changing the number of colours

Often it is necessary to reduce the number of colours in an image, either to reduce the file size or produce images compatible with non-AGA software.

In this example we will reduce the number of colours in the '`FW14B_250x250.gif`' image from 256 colours to 16 colours.

1. Load in the '`FW14B_250x250.gif`' from the '`Pics`' drawer, if it not already loaded.
2. Open the colours requester by selecting 'Colours...' from the 'Process' menu.
3. The gadgets in the requester will show that the image is a 256 col our colour-mapped image. Change the number of colours to 16 by clicking on the cycle gadget or sliding the 'No. of colours' slider.
4. Leave the 'Colour choice' and 'Dithering' gadgets for now.
5. Click on 'OK' to perform the operation.
6. When the operation is complete, view the image with the internal viewer by selecting 'View' from the 'View' menu. A 16 colour Lores image will be displayed.
7. Remove the internal viewer by clicking the right mouse button.

The colour reduced image should contain all the main colours used in the original image (blue, yellow, red and grey), but should contain less shades of the colours. To give the impression of more colours, dithering can be used to mix pixels of the chosen colours. To perform the last operation with dithering:

1. Undo the colour reduction operation to return to the original 256 colour image. To do this, select 'Undo' from the 'Edit' menu.
2. Bring up the colours requester as before and select 16 colours. Also change the 'Dithering' gadget from 'None' to 'Floyd-Steinberg'.
3. Click on 'OK' to perform the operation.
4. View as before.

The image will now perform more gradual changes to colour changes.

2.3 Changing the colour balance

When receiving images from external sources (scanners, frame grab bers) it is ofter necessary to change to "colour balance" of the image. Frame grabbers, for example, may have too much 'red' in the image.

In this example we shall see the effects on the '`ColourFace_200x250RED.ham6`' image of altering the colour balance.

1. Load in the '`ColourFace_200x250RED.ham6`' image from the '`Pics`' drawer. As the file is in HAM6 format, it is turned into 24-bit internally.
2. View the image with the internal viewer. As the Amiga doesn't have true 24-bit screenmodes, the internal viewer will approximate the 24-bit image with a HAM preview screen (HAM6 on ECS machines, HAM8 on AGA machines). It should be obvious from the viewer that the image is too red.
3. Open the balance floating palette if it is not already open. Do this by selecting 'Show balance' from the 'Tools' menu.
4. To remove some of the red component, make sure that we are only dealing with the red component. To do this, make sure that only the 'Red' checkbox at the bottom of the floating palette is checked.
5. Reduce the brightness slider, by say 20%. The graph on the right of the floating palette will reflect the change (see Section 3.5.2 [Show˙balance], page 26 for more details on the graph).
6. View the image again, this time the image should have lost much of its unnatural red tint.

Brightness and contrast work in very much the same way as a TV set, but gamma may need some explanation.

When printing an image out, it is usual for light colours to be resolved well and dark colours to be reduced to a dark mush. It is therefore preferable to have some way boost the brightness of the mid-dark colours whilst still leaving the very light colours light and the very dark colours dark. Gamma is the operator to perform this change. By applying a small amount of positive gamma, a much better balanced image can be produced for printing out.

See Section 3.5.2 [Show˙balance], page 26, for more information on the balance floating palette.

2.4 Applying an effect

ImageStudio has many built in effects for performing commonly used operations.

This example will remove some noise from a region of '`HappyFace_240x250.bmp`', a simulated scanned image.

1. Load in the '`HappyFace_240x250.bmp`' image from the '`Pics`' drawer.
2. The image represents what may happen if you hand scan an image into the computer - lots of "noise".
3. Open the effects floating palette if it is not already open. Do this by selecting 'Show effects' from the 'Tools' menu.
4. Select 'Remove isolated pixels' in the listview of the floating palette. If you clicked on the 'Apply' button now, the effect would be applied to the whole image. To compare the image before and after the effect, we'll only apply the effect to the left hand side of the image.
5. Open the 'Region co-ords' requester by selecting 'Region co-ords...' from the 'Edit' menu. To select the left hand side of the image, set the following values in the gadgets: Min x = 0, Min y = 0, Width = 120, Height = 250. Make sure that the radio button on the left of the requester shows that the Width / Height are being used, not the Max values; click on 'Ok'.
6. A region of "crawling ants" will show the selected region.
7. Click on 'Apply' of the effects floating palette. The 'Remove isolated pixels' effect will be applied to the selected region.
8. The left of the image will have had a lot of the noise automatically removed. Clear the selected region by clicking in the preview window.

Note:

- Not all effects can be applied to regions and whole images. See Section 5.2 [Effects], page 117, for a comprehensive description of all the available effects.

See Section 3.5.3 [Show˙effects], page 28, for more information on the ef fects floating palette.

2.5 Applying a convolution

Convolution is a powerful image processing tool, ImageStudio allows the user to define their own convolution filters.

This example will apply a 'Texture' filter to the '`CheetahFace_250x200.ilbm`' image.

1. Load in the '`CheetahFace_250x200.ilbm`' image from the '`Pics`' drawer.
2. The '`CheetahFace_250x200.ilbm`' image is a 32 colour col our-mapped image and convolution only works in 24-bit (see Section 3.5.4 [Show˙convolves], page 29 for information on how convolves actually work). We therefore need to turn the image into a 24-bit.
3. Open the colours requester and click on the '16 million colours' radio button on the left. Click on 'OK'. The image in converted into 24-bit.
4. Open the convolves floating palette if it is not already open. Do this by selecting 'Show convolves' from the 'Tools' menu.
5. There should be many convolution filters in the list, click on 'Texture'.
6. Apply the convolution filter to the image by clicking on 'Apply' at the bottom of the floating palette.
7. After the filter has been applied, you could view the result with the internal viewer in 24-bit, but for a clearer image we'll convert it back to 32 colours.
8. Open the colours requester, select 'No. colours' = 32 and 'Dither' = 'None'. Click on 'OK'.
9. Now view the 32 colour image with the internal viewer. The image now has a rough paper(?) texture applied to it.

Note:

- There are many commonly used convolution filters available, it is up to the user to build a collection suitable filters for their own use.

See Section 3.5.4 [Show˙convolves], page 29, for more information on the convolves floating palette.

2.6 Scaling the image

In this example we will scale the '`CheetahFace_250x200.ilbm`' image from 250 x 200 pixels down to 80 x 40 (icon size).

1. Load in the '`CheetahFace_250x200.ilbm`' image from the '`Pics`' drawer.
2. Open the scale requester by selecting 'Scale...' in the 'Process' menu.
3. Set the 'Width' = 80 and 'Height' = 40. Click on 'OK'.
4. The image is re-scaled to that of an icon.

The finish the creation of the icon, we can load it into Commodore's '`IconEdit`' program. Both ImageStudio and IconEdit support the clipboard, so we can copy the image into the clipboard from ImageStudio and paste it into IconEdit.

1. Copy the image to the clipboard by selected 'Copy' in the 'Edit' menu of ImageStudio.
2. Run IconEdit from the '`Tools`' drawer of your system partition.
3. Select 'Paste' from IconEdit's 'Edit' menu. The image will be copied into IconEdit for final editing.

3 Menu options

This chapter describes ImageStudio's menu options in detail.

3.1 Project

3.1.1 Open

Keyboard shortcut - `Amiga - O`

This is how the user loads in an image into the program.

A file requester will appear, through which the user can select a file to open. Upon selecting a file, ImageStudio will test the file against its known file formats - loading the file if the image type is recognised. If the image format is not recognised, an error will be shown.

In most cases the image will load directly into ImageStudio; however in the case of HAM6 and HAM8 formats the image is converted into 24-bit data as it is loaded in, as ImageStudio cannot work directly on HAM images. For a detailed description of colour-mapped and 24-bit images, See Section 5.3 [Image types], page 122.

3.1.2 Save

Keyboard shortcut - `Amiga - S`

The save requester allows the user to choose the filename for the saved image as well as the image's format.

To change the filename, either click in the string gadget and edit the filename directly or click on the 'Choose...' gadget to select the filename with a file requester.

To change the image format of the file to be saved, click in the listview on the appropriate format. Depending on the format selected, depends on whether the 'Options...' button is available. Some formats (e.g. IFF-ILBM) have further options available by clicking on the 'Options...' button.

When the user has selected the filename and the image format, the file can be saved by clicking on 'OK' or no action can be performed by clicking on 'Cancel'. If the currently selected filename already exists, the user will be warned that they are about to overwrite it.

3.1.3 Screen

The user may select the current screen's resolution, number of colours and font.

Screen mode

> The screenmode requester allows the user to change the properties of the current screen. Click on the desired screenmode as well as the size, number of colours and overscan settings. To bring the changes into effect, click on 'OK' else to perform no action click on 'Cancel'.

Screen font

> The screenfont requester allows the user to choose the font to be used in all the ImageStudio windows and menus. Any Amiga font can be used, although if a font which is too large to allow some of the windows to fit on the screen is chosen, the program will default back to using topaz 8.
>
> Under Workbench2.04, not all gadgets will scale properly with dif ferent sized fonts. Checkbox gadgets and mutual exclude (radio but ton) gadgets will remain a constant size, which may cause some windows to look slightly strange with very large or very small font sizes.

3.1.4 Prefs

The user may configure the program to their own needs with the prefs requester. The requester consists of a scrolling list with all the available preferences, several gadgets that show the current tooltype value whilst allowing the user to change it, and a couple of checkboxes for saving the current screenmode and window positions. When the user is happy with the current values, they may save them as defaults by clicking 'Save', use them for the current session by clicking 'Use' or discard the changes by clicking 'Cancel'.

The preferences in the list are separated into string values (e.g. filenames, ARexx port name etc...), numeric values (e.g. virtual memory pagesize, number of undo buffers etc...) and boolean, or ON / OFF values (e.g. Online AmigaGuide help, splash window on startup etc...). A short description of the preference is shown on the left of the list, the tooltype (see Section 5.5 [Tooltypes], page 123) name is shown below.

To change a preference value, click on its name in the list. The current value will be copied to the text or numeric value below; the exception to this is a boolean preference, see below. The value is changed by changing the displayed value in the gadget, it will be stored as this value if another preference is selected or the user presses 'Save' or 'Use'. If a string preference is selected that requires a filename to be chosen, the 'Choose...' gadget can be used to select the desired filename from a file requester.

Boolean preferences are changed by double-clicking on their entries in the list. The value will toggle between the two states.

If the user wishes to save the current screenmode or window positions to be the defaults next time the program is started, they may click on the 'Save screenmode' and / or the 'Save window positions' checkboxes.

When changing any of the preferences that require the use of an external program (e.g. the external text editor), the string must be formatted to contain a `%s` placed where the filename should be placed.

The default string of

```
run sys:Tools/Memacs <NIL: >NIL: "%s"
```

for the `TEXTEDITOR` preference would run Commodore's Micro-emacs text editor to edit files. It is recommended that the "run" is added at the start of the string to run the program in the background (i.e. ImageStudio doesn't have to wait for the text editor to finish) and that the double-quotes are placed around the `%s` to allow for filenames containing spaces.

The preferences values may also be read (see Section 4.10.27 [ARexx`PREF`GET], page 76) and written (see Section 4.10.28 [ARexx`PREF`SET], page 77) from ARexx.

3.1.5 About

This brings up a small requester containing information about the program version number and the user name (only in registered version).

3.1.6 Info

This brings up an information requester containing memory and file usage information.

The 'Memory' figures are the amount of RAM used by the buffers, the 'VMem' figures are the amount of disk space used by the virtual memory.

At the bottom of the requester the amount of free RAM and ARexx port name is shown.

3.1.7 Help

Keyboard shortcut - `Amiga - H`

This brings up the main page of the AmigaGuide document.

In order for AmigaGuide help to work, the following must be satisfied:

1. Commodore's AmigaGuide help has to be installed on your system. This comes with Workbench3.0+ and can be freely obtained for earlier versions of Workbench.
2. The preference 'HELP' must be 'ON' (see Section 3.1.4 [Prefs], page 14).
3. The **full** filename of the AmigaGuide helpfile must be given in the 'HELPFILE' preference (see Section 5.5.27 [Tooltype`HELPFILE], page 129).

It is recommended that help be turned off if you are short of memory.

Once help is running, it can either be accessed by selecting the 'Help' submenu in the 'Project' menu or by performing a standard "help menu pick" on a menu item. To do this, highlight the menu item you item you wish to know more about, but instead of releasing the right mouse button to select the item, press 'Help' on the key board. To clarify, perform the following steps:

1. Press right mouse button.
2. Highlight menu item you wish to find out help on.
3. Press 'Help' on the keyboard.
4. Release right mouse button.

3.1.8 Create keyfile

This allows registered users with v1.x.x of ImageStudio to create a keyfile to 'unlock' the freely distributable versions of ImageStudio.

A file requester will open asking the user to select their regis tered version of ImageStudio v1.x.x. If ImageStudio recognises this as a registered copy, a keyfile with the name given by the 'KEYFILE' preference (see Section 5.5.30 [Tooltype˙KEYFILE], page 130) will created.

If all is successful, when ImageStudio is next started it will be fully operational.

A plea:

> Please do not give your keyfile to anyone else. Each keyfile is individual to each user, so should your keyfile get into distribution it can be traced back to you. Don't try altering your keyfile, it won't work.
>
> Many thanks for your co-operation in this matter.

3.1.9 Iconify

Keyboard shortcut - `Amiga - !`

This reduces the program to a small icon on the Workbench screen. To reopen the program, simply double-click on the icon. In order to re duce the amount of memory used by the program when it is iconified, some of the working memory used by the program is purged; this could lead to a freeing of several hundred K of memory, depending on the virtual memory page size and data in internal caches.

3.1.10 Quit

This quits the program. If any changes remain unsaved, the user is warned before the program quits.

3.2 Edit

3.2.1 Undo

Keyboard shortcut - `Amiga - U`

Undos last operation.

The maximum number of undos is set in the preferences requester, See Section 3.1.4 [Prefs], page 14.

3.2.2 Redo

Keyboard shortcut - `Amiga - R`

Redos last undo.

The maximum number of redos is set in the preferences requester, See Section 3.1.4 [Prefs], page 14.

3.2.3 Copy

Keyboard shortcut - `Amiga - C`

Copies the current image to the clipboard.

Once the image has been copied to the clipboard it can be used by any other program that supports the Amiga clipboard.

3.2.4 Paste

Keyboard shortcut - `Amiga - V`

Reads in image from the clipboard.

3.2.5 Flush buffers

Flushing the buffers will free any memory and diskspace used by undo / redo buffers. The user is warned that once the buffers have been freed, the data cannot be recovered. The current buffer (i.e. the image in the preview window) is not affected.

3.2.6 Region co-ords

Keyboard shortcut - `Amiga - D`

Allows the user to select a region by typing the co-ordinates.

The region co-ords requester allow the user to specify the selected region by either typing in the co-ordinates of the minimum and maximum corners of the rectangle or the minimum co-ordinates and the rectangle's width and height. A radio button on the left of the requester shows whether the maximum co-ords or the width and height are to be used to select the region.

If there is already a selected region, these values are copied into the requester when it is opened.

3.2.7 Region clear

Removes any selected region from the preview window.

As well as this menu item, the region can be cleared by simply clicking in the preview window.

3.2.8 Select all

Keyboard shortcut - `Amiga - A`

Makes the selected region the whole of the displayed image in the preview window.

Note:

If the user has zoomed in on a region of the image (see Section 3.3.2 [Zoom˙in], page 20), ‘Select all’ will not select the whole image but just the displayed image in the preview window.

3.3 View

3.3.1 Full image

Keyboard shortcut - `Amiga - F`

Displays the whole image in the preview window.

3.3.2 Zoom in

Keyboard shortcut - `Amiga - <`

Zooms in to make the currently selected region fill the preview window.

3.3.3 Zoom out

Keyboard shortcut - `Amiga - >`

Zooms out by a factor of 3 times.

3.3.4 Internal viewer

Keyboard shortcut - `Amiga - I`

Views the current image with the internal viewer.

The user may choose the viewer module to be used via the INTERNALVIEWER preference (see Section 5.5.29 [Tooltype`INTERNALVIEWER], page 130) to adapt the view er's capablities based on the Amiga on which it is being used.

In general, if the image is colour-mapped, the viewer will try and open a screen with the same number of colours as the image. Viewing 24-bit images is done by using the HAM screenmodes (HAM6 under ECS, HAM8 under AGA) to approximate the 24-bit image. In order to keep the viewer reasonably fast, the HAM image sometimes suffers from 'colour fringing' as the approximation is relatively crude (especially under ECS).

If the CyberGraphX retargetable system is being run on the Amiga with a compatible graphics card, the image can be viewed using any of the supplied viewers directly on a 15/16 or 24bit screen.

If the image originated as an IFF-ILBM, the viewer screen will try and open up in the same screen mode as the image. If this is not possible, the user may change to a more suitable screenmode with the 'View screenmode' menu option, See Section 3.3.5 [View`screenmode], page 22.

If the internal viewer won't display the current image, check the following:

1. If the image was loaded in as an IFF-ILBM, the saved screenmode may not be supported by your machine. For example, the screenmode may be 'Productivity' and your machine doesn't have a multiscan monitor. Simply change the screenmode to one your machine does support - e.g. 'Hires Laced'
2. A colour-mapped image contains more colours than it is possible to show on a ECS machine. Either reduce the number of colours in the image to a number that can be displayed or use the 'amigaplus.isio' internal viewer to display an approximation in a HAM screen.
3. Make sure you have enough CHIP RAM free to open the screen. Large 256 colour and 16 million colours images take lots of CHIP RAM.

To stop the viewer at any time, press the right mouse button or click on 'Abort' in the infobar.

The following internal viewer modules causes slight differences in the behaviour in the way the image is handled in certain cases:

amiga.isio This module will work with both ECS and AGA Amigas. Colour-mapped images are displayed directly (upto 16 or 32 colours on ECS machines, 256 colours on AGA

Amigas) and 24 bit images are shown on HAM6 screens. If the viewer cannot open the desired colour mapped screen, it fails.

amiga-aga.isio

This module will only work with AGA Amigas. It functions exactly the same as the 'amiga.isio' module, only it displays 24 bit images in HAM8.

amigaplus.isio

This module is aimed at the ECS Amiga, essentially being the 'amiga.isio' module but with the automatic capability of being able to display colour mapped images as HAM6 if the colour mapped image has too many image for the ECS chipset to display directly. This is useful for viewing the many 256 colour images on an ECS Amiga.

3.3.5 View screenmode

This allows the user to set the screenmode of the image, and there fore of the internal viewer.

Click on the desired screenmode for the image in the screenmode requester. The current screenmode is highlighted in the listview.

3.3.6 External viewer

Uses an external viewer program to view the image.

This calls up the external viewer program to view the current image. If a third party 24-bit graphics card is installed, a viewer can be used to view the image on that.

To specify the external viewer to use, See Section 3.1.4 [Prefs], page 14.

3.4 Process

3.4.1 Crop

Keyboard shortcut - `Amiga - W`

Crops the current image to the selected region.

This reduces the image to only that which is in the selected region. A region must be selected in order for this operation to work.

3.4.2 Scale

Keyboard shortcut - `Amiga - L`

Reduces / increases the size of the image.

The scale requester allows the user to change the image's width / height by either selecting the absolute size of the new image or the percentage by which to scale. A radio gadget to the left shows which operation will be performed.

The percentage value may also be changed by sliding the width and height sliders to achieve the desired final size; divide and multiply by 2 buttons are provided to quickly scale the image by common amounts.

Finally, two methods of scaling are supported: fast and colour average. Fast scaling works with both colour-mapped and 24-bit images and produces results adequate for most needs. If the image is to be scaled up by a large amount the image may become very 'blocky' and if the image is scaled down a large amount, information in the image may be lost. To reduce this, colour average scaling is available on 24-bit images which reduces blockiness when increasing the scale and reduces information loss when reducing the scale. Colour average rescale can take significantly longer than a fast rescale.

3.4.3 Colours

Keyboard shortcut - `Amiga - K`

Allows increasing / decreasing of the number of colours in the im age, with various dithers.

The colours requester allows the user control over the number of colours in the image. A radio button on the left hand side shows whether the current image is colour-mapped or 24-bit.

To increase the number of colours in a colour-mapped image, simply select the new number of colours with the top cycle gadget or the 'No. colours' slider. Although the number of colours need not be a power of 2 (2, 4, 8, 16, 32, 64, 128 or 256), internally the number will be rounded up to the nearest power of 2. If, for example, a 16 colour picture was increased to 20 colours then the image would be come a 32 colour image.

Colour-mapped images can also be turned into 24-bit images by click ing on the '16 million colours' radio button on the left hand side of the requester. This is useful if the user wishes to perform an operation on a colour-mapped image that can only be performed on a 24-bit image. The resultant 24-bit image can then be turned back into a colour-mapped image after the operation is complete.

To reduce the number of colours in an image, the same process is followed as above with a few differences. Whereas increasing the number of colours in an image does not lose any image information, it is enevitable that reducing the number of colours must lose some of the colour information. In order to help reduce the effect of this, two other aids are used: dithering and palette choice.

The result of colour reduction is always a colour-mapped image. The user may select the number of colours in the final image with the top cycle gadget or the 'No. colours' slider. Again, although the number of colours need not be a power of 2 (2, 4, 8, 16, 32, 64, 128 or 256), internally the number will be rounded up to the nearest power of 2. This though can be useful, as the user may want to re duce a 24-bit image down to 30 colours - leaving 2 spare for his / her own use.

In order to give the impression of more colours in the reduced col our image, dithering can be employed to smoothly distribute colours over areas of high colour change. 'Floyd-Steinberg' is the most common method and works well in most cases. For larger images, better contrast can be obtained by using a more computationally intensive dither ('Burkes', 'Stucki', 'Sierra', 'Jarvis') and for the user with large images and lots of time to spare, 'Stevenson-Arce'. Again, there is no hard and fast rule which method of dithering is best; if you're not happy with the result, try a different method.

3.4.4 Palette

Allows the saving of the current palette and loading of new palette onto the current image.

Palettes can either be loaded, saved and edited in ImageStudio:

Palette load

This is used to force a palette onto an image. The requester allows the user to choose the palette to load and any dithering to be applied to the image, See Section 3.4.3 [Colours], page 23. Various sample palettes are given with the distribution to map the image to the Workbench colours or a general purpose palette. New palettes can be generated with any popular paint package[1].

Palette save

Saves the current palette out to the filename chosen by the user in the requester. The palette file is compatible with the popular paint packages. This option has no relevance for 24-bit images, as they have no palette.

Palette edit

This brings up the palette edit requester. Here the user may edit each colour individually or move the colours around with the 'Copy' and 'Swap' operations. Colour ranges can be created and ranges may be sorted into order of increasing or decreasing luminosity.

To edit a colour, simply click on the colour in the scrolling viewer and edit the R,G,B or H,S,V colour values (see Section 5.4 [Colour representations], page 122).

To copy or swap two colours, click on the first one in the scrolling viewer then click on either 'Copy' or 'Swap'. The pointer will change to a "To" pointer to allow you to click on another colour to swap or copy to.

To create a colour spread, alter one colour to be the start of the spread and alter another colour to be the end of the spread. Click ing on the first colour, then on 'Spread', the on the end colour will cause a smooth transition of colours between the start and end colours.

Sorting the colours is like a colour spread, only no colour values are actually changed. Simply click on the start of the sort, followed by either of the 'Sort' buttons and finally on the end colour. The colours between these values will be sorted into either an in creasing or decreasing order of luminosity.

Finally, to apply the new colour palette to the image click on 'OK'. To remap the image to the new palette (swap the old colours with their nearest match in the new palette), click on 'Remap'.

Whilst changing the colours in the scrolling viewer, the colours in the preview window will change. **This is not a representation of what is happening to the image!**.

3.5 Tools

---

[1] Except Brilliance, which seems to save all 384 colours of its palette.

3.5.1 Command shell

Keyboard shortcut - `Amiga - 0`

Opens the ARexx command shell.

The command shell allows the user to enter ARexx commands directly, without having to write a script file (see Section 3.5.5 [Show˙scripts], page 31). This means that the effect of the ARexx command may be seen directly, allowing the commands to be experimented with before adding them to a script.

The values returned by a command are displayed in a readable form in the command shell.

It should be noted that the command shell differs from executing an actual ARexx script in the following ways:

1. Each entered command is passed directly to ImageStudio without going through ARexx first. This means that no ARexx statements can be used (e.g. loops) and no variables may be defined.
2. The string entered in the command shell is *exactly* what is seen by the ImageStudio command parser - nothing is evaluated. It is not neccessary therefore to enclose strings in both single and double quotes to stop them from being evaluated. For example, the following line is valid in the command shell, but invalid in an ARexx script as the string would have its quotes removed (see Section 4.7 [Common ARexx problems], page 43):
   ```
   REQUEST_MESSAGE TEXT "A string with spaces in it"
   ```

Help on a particular command's template (see Section 4.4 [Command templates], page 37) can be obtained by typing:

```
help <command>
```

in the command shell. The command's template for the input and return values will be show.

3.5.2 Show balance

Keyboard shortcut - `Amiga - 1`

Opens / closes the balance floating palette.

The balance floating palette is used to control the brightness, contrast and gamma of the current image. On 24-bit images, the colour balance can be altered on selected regions as well as the whole image whereas colour-mapped images only allow alterations to the whole image.

The colour balance effects are usually applied to all the red / green / blue components simultaneously, but each component can be altered individually by checking the 'Red', 'Green' or 'Blue' check boxes at the bottom of the floating palette.

The effect of changing either the brightness, contrast or gamma can be seen in the graph on the right hand side of the floating palette. The graph shows the input RGB component along the X-axis and the output RGB component along the Y-axis.

```
output
Light RGB ^
  |
  |
  |
  |
  |
 Dark RGB |
  +-------> input
      Dark RGB  Light RGB
```

No operation is shown therefore with a straight diagonal line - the input value is mapped to the same output value.

```
 output
   ^
   |     /
  x|____/
   |   /|
   |  / |
   | /  |
   |/   |
   +-------> input
x
```

Brightness Altering the brightness is achieved by mutliplying up / down the RGB components by the specified amount. The range of the slider is from -100% (everything becomes black) to +100% (everything is twice as bright).

Contrast Altering the contrast forces dark colours darker and light colours lighter. The range of the slider is from -100% (everything becomes mid grey) to +100% (RGB components are either on/off).

Note: 100% contrast on a colour image doesn't produce a black and white image as may be expected. As each RGB component is treated individually, it leaves you with an 8 colour image - the 8 colours being composed of combinations of the RGB components as below:

Black	0% Red, 0% Green, 0% Blue
Red	100% Red, 0% Green, 0% Blue
Green	0% Red, 100% Green, 0% Blue
Blue	0% Red, 0% Green, 100% Blue
Yellow	100% Red, 100% Green, 0% Blue
Magenta	100% Red, 0% Green, 100% Blue
Cyan	0% Red, 100% Green, 100% Blue
White	100% Red, 100% Green, 100% Blue

If you wish to turn a colour image into 2 colour black and white, greyscale the image first with the greyscale effect, See Section 5.2 [Effects], page 117.

Gamma Adjusting the gamma of an image has the effect of lightening some of the mid-dark colours, whilst leaving the dark colours dark. This can often enhance the eye's perception of the image, as the eye is more responsive to light colours. Gamma correction can also be useful when printing an image out, as mid-dark colours tend to get printed too dark.

Only small alterations are usually needed with this operator (-20% to +20%).

3.5.3 Show effects

Keyboard shortcut - `Amiga - 2`

Opens / closes the effects floating palette.

The effects floating palette contains a list of ImageStudio's built in effects. Not all types of effect can be applied to all types of buffer, the details are given below. Any numerical values required by the individual effects are set using the slider on the effect floating palette.

An effect can be applied by clicking on the ‘Apply’ button or double-clicking on the list entry.

See Section 5.2 [Effects], page 117, for details of all available effects.

3.5.4 Show convolves

Keyboard shortcut - `Amiga - 3`

Opens / closes the convolves floating palette.

The convolves floating palette allows the user to apply their own convolution effects to a 24-bit image; convolution will not work on colour-mapped images.

To create a new convolution filter, select ‘New’ and then ‘Edit’ from the floating palette. The convolve grid requester contains the convolution filter’s name at the top as well as gadgets for the filter, divisor and bias values. When the user has set the filter values, click on ‘OK’ to return to the convolve floating palette.

To apply a filter, select it in the listview and click on ‘Apply’. To delete a filter from the list, click on ‘Del’. This will not delete the file from the disk, this has to be done from the Workbench.

To scan a new drawer for convolution filters, click on ‘Load’ and select a directory to scan. To save the current list’s convolution filters, click on ‘Save’ and select a drawer to save to.

The default drawer to scan at startup is set in the preferences, See Section 3.1.4 [Prefs], page 14.

A convolve can be applied by clicking on the ‘Apply’ button or double-clicking on the list entry.

What follows is a quick description of convolution, it is not necessary to understand this to use the filters.

It is convenient think of the convolution filter as an array of numbers that "slides" over the image a pixel at a time. To find the new colour value of the pixel at the centre of the filter, multiply the filter values by the values of the colours under the array then divide the result by the ‘Div’ value, then add the ‘Bias’ value.

If we take the example of 3 x 3 ‘blur low’ filter being applied to the pixels below:

```
filter                 pixels

0 1 0                  a b c
1 2 1  convolved with  d e f   gives:
0 1 0                  g h i
```

((0 x a) + (1 x b) + (0 x c) + (1 x d) + (2 x e) + (1 x f) + (0 x g) + (1 x h) + (0 x i)) / Div + Bias

which would be applied to the new pixel in the position of the 'e' pixel.

Although the pixels shown above are shown as 'a', 'b' etc... they are actually the 3 red, green and blue values that comprise the colour.

Examples:

1.

```
0 0 0
0 1 0
0 0 0
Div = 3, Bias = 0
```

would make each pixel one third of its original brightness.

2.

```
0 0 0
0 0 0
0 1 0
Div = 1, Bias = 0
```

would move each pixel up by one.

3.

```
0 0 0
0 1 0
0 0 0
Div = 1, Bias = 50
```

would add 50 onto each of the red, green, blue components of the centre pixel.

Note:

- The red, green, blue components of a pixel can have values in the range 0 to 255. If a convolution value is greater than 255 it is made equal to 255. Similarly if a convolution value is less than 0 it is made equal to 0.
- ImageStudio has optimized routines for 1x1, 3x3 and 5x5 filters. If the program detects that only values in a 3x3 filter are being used, only calculations for a 3x3 filter are performed.

3.5.5 Show scripts

Keyboard shortcut - `Amiga - 4`

Opens / closes the scripts floating palette.

The scripts floating palette gives a list of easily available ARexx scripts for the user to apply.

To execute a script in the list, select the desired script and click on 'Apply' or simply double-click on the list entry. The script can be stopped by either clicking on 'Abort' during any operation or choosing 'Cancel' on any given requester (if the script is written correctly, that is). See Section 4.2 [Writing scripts], page 33, for more information on creating ARexx scripts.

New scripts can be created by clicking on 'New...' button. The user is requested to name the new script and it is placed in the listview. By default, the script will already contain some ARexx commands to help the user - the user simply adds what is neccessary.

To edit the currently selected script, the 'Edit...' button is used. The text editor set in the preferences (see Section 3.1.4 [Prefs], page 14) is used to edit the file.

The 'Load...' button allows the user to scan another directory for ARexx script files. 'Other...' will let the user select another ARexx script file to run, without adding it to the list.

3.5.6 Show coords

Keyboard shortcut - `Amiga - 5`

Opens / closes co-ords floating palette.

The co-ords floating palette shows the current position in the image of the mouse pointer. When dragging box to select a region, the current region width and height are shown.

4 ARexx

This chapter gives information about the program's interface to the ARexx programming language.

4.1 Introduction to ARexx

ARexx is the script language that is distributed with all Amigas sporting Workbench 2.04 and above. It is used on the Amiga for two main tasks:

1. Providing an easy and consistent method of adding macro facilities to programs.
2. To allow ARexx aware programs to communicate with each other.

Most users are dissuaded from using ARexx with their programs be cause of the learning curve involved in (i) learning ARexx and (ii) using the functions provided with each program. With ImageStudio, we have tried to simplify the process of creating an ARexx script by:

1. Providing an easy interface for creating and running the scripts.
2. Providing a ready-made script template which the user can just "fill in the blanks" to produce a fully working program.
3. Providing many commands to perform commonly performed operations. This means the user needs to write less code in ARexx and doesn't need to rely on external utilities and libraries to perform the operations.

Typical uses for ARexx in ImageStudio include:

- Batch processing. ImageStudio can now be told to repeatedly perform the same operation on many images. The user could, for example, convert all PCX files in a given directory into IFF-ILBM files.
- Background processing. ImageStudio can be told to watch a given di rectory and whenever a new file is generated to perform a set of op erations on it. This is useful for producing ray-traced animations, where each frame of the animation could be converted from 24bits to HAM6 (say) as each frame was generated by the ray tracer.
- Adding additional image manipulation abilities to other programs. ImageStudio could be passed an image from another program, process it, then return it back to the original program. By using a desktop publishing package that supports ARexx, an image could be saved from the DTP program, gamma corrected by ImageStudio, then reloaded back into the DTP program all automatically.

Many example files are given with ImageStudio (see Section 4.9 [Example scripts], page 47), which can either be used directly or modified to perform the desired operation.

In order for ARexx to be available to ImageStudio, you must start ARexx at startup time by including the line:

```
rexxmast >NIL:
```

in your 'User-Startup' file in the 'S:' directory. Normally this line should be present in your User-Startup, but if you find no scripts run from ImageStudio you must add this line manually.

4.2 Writing scripts

We have tried to simplify the process of writing an ARexx script as much a possible to provide access to the power of ARexx scripts without (too much(!)) pain.

An ARexx script can be written and run without ever having to leave ImageStudio. The only extra tool needed is a text editor; ImageStudio can be configured to use your favourite text editor by changing the program's preferences (see Section 3.1.4 [Prefs], page 14). The basic process of creating an ARexx script is as follows:

1. Open the 'Scripts' floating palette.
2. Create and name the new script by clicking on 'New...'.
3. Load the blank script template into your editor by clicking on 'Edit...'.
4. Add the desired commands to the template in the space provided.
5. Save the file and exit the text editor.
6. Run the script by clicking on the 'Apply' button on the scripts floating palette.

Any errors the script may generate will be displayed in a requester on the ImageStudio screen.

To try this process out, try the following:

1. Open the 'Scripts' floating palette.
2. Click on 'New...' and call the script "FirstScript".
3. Edit the new script by clicking on 'Edit...'.
4. In the space provided in the script, type the following lines:
   ```
   OPEN "Pics/FW14B_250x250.gif"
   SCALE 80 40
   PALETTE_LOAD "Palettes/Workbench4.palette" DITHER FS
   VIEW
   ```

5. Save the file and exit the text editor.
6. Run the script by clicking on the 'Apply' button on the scripts floating palette.

The result of the above script should be a 4 colour icon sized image of a Formula1 racing car. This could now be copied into the clip board by selecting 'Copy' in the 'Edit' menu and pasted into Commodore's IconEdit program for final conversion into an icon. Alternatively, the command to copy the image into the clipboard could be added to the end of the script.

See Section 4.10 [ARexx commands], page 52, for a full description of the ARexx commands used in the above example.

Note, please don't remove the "/* BEGIN PROGRAM ****" and corre sponding "END PROGRAM" lines in the blank macro template - these will be used to insert the recorded ARexx macros when they are implemented (see Chapter 10 [Future additions], page 146).

4.3 Basic ARexx

This section is meant as a beginners guide to using ARexx with Imag eStudio. We cannot hope to teach you the ARexx language, although it is only neccessary to the know the very basics to start using ARexx scripts with ImageStudio. It is assumed that the user is editing and running their scripts from within ImageStudio (see Section 3.5.5 [Show˘scripts], page 31).

For further information on ARexx, we suggest reading Commodore's ARexx user guide supplied with the A4000 or the Workbench2 and 3 up grade packs. For A600 and A1200 users who don't get this manual, we recommend the "ARexxGuide" AmigaGuide document by Robin Evans which is a shareware document containing extensive information on the ARexx language. The guide can be obtained from all good PD houses.

The ARexx programming language is similar to many other programming languages in its structure. Users who have BASIC, C, FORTRAN, Pascal, Modula2 or Oberon experience will notice many similarites. It is not similar to Assember language, Lisp or Prolog. An ARexx pro gram is, in its simplest form, a list of instructions for ImageStudio to perform. Here is a simple ARexx program:

```
/* A simple ARexx program */

REQUEST_MESSAGE TEXT '"Hello world!"'
```

```
exit
```

This shows some important things about an ARexx program:

1. All ARexx programs **must** start with a comment line. A comment line is a line which starts with the '/*' sequence of characters and ends with the '*/' characters. Anything between these characters is ignored by ARexx.
2. For clarity, all of ImageStudio's commands are shown CAPITALISED, ARexx commands are kept in lower case. REQUEST_MESSAGE is therefore an ImageStudio command that should be performed.
3. The REQUEST_MESSAGE has some 'arguments' or 'parameters' following it. These tell the REQUEST_MESSAGE command how to behave, in this case they tell the command to pop up greeting message.
4. To stop an ARexx program, use the command 'exit'.

OK, lets enhance our program a little:

```
/* A better simple ARexx program */

REQUEST_MESSAGE TEXT '"What do you think of\n'||,
   'the show so far?"',
   BUTTONTEXT "Great|Mediocre|Rubbish"

exit
```

From this example we learn:

1. To separate a long command line, place a comma ',' as the last character on the line. This tells ARexx to treat the next line as a continuation of the previous. Two line breaks are used in the above example.
2. ARexx loves to evaluate things. If we want to stop ARexx evaluating variables, the variable should be enclosed in single quotes ' '.

See Section 4.7.1 [ARexx problem 1], page 43, if little explanation is needed as to the many double and single quotes used above. If we now tell you that the '\n' characters are used represent a newline and the '`||`' characters glue string together, we should see that:

```
'"What do you think of\n'||'the show so far?"'
```

would be evaluated to:

```
"What do you think of*the show so far?"
```

where '*' represents a newline. The lesson to be learnt here is that whenever you use a string (with or without spaces) it is best to enclose the whole thing in single quotes outside the double quotes to keep the whole thing together.

On with the examples. The previous script isn't much use if we can't test for which button the user pressed, so:

```
/* A better simple ARexx program */

options results

REQUEST_MESSAGE TEXT '"What do you think of\n'||,
   'the show so far?"',
   BUTTONTEXT "Great|Mediocre|Rubbish"

if RESULT == 0 then
   REQUEST_MESSAGE TEXT '"Sorry, I was trying very hard."'
else if RESULT == 2 then
   REQUEST_MESSAGE TEXT '"It gets better."'
else do
   REQUEST_MESSAGE TEXT '"We like happy users."'
   REQUEST_MESSAGE TEXT '"Treat yourself to a coffee."'
   end

exit
```

This shows:

1. Normally ARexx ignores the values returned by commands. To allow commands to return values, use "options results"; this is done for you in the blank ARexx script.
2. Unless otherwise specified (see Section 4.5 [Return values], page 39) commands return the results of their operation in a variable called "RESULT". The command REQUEST_MESSAGE returns the value of the button that the user pressed. It is this value that we can test for.
3. The 'if' tests are shown above. Note that if you only want to perform one operation as part of the 'if', you can just place it after the 'then'. If you wish to perform more operations, they must be placed in a 'do / end' set.

OK, that's about it for the introduction to ARexx. We really suggest now that you look at the example scripts provided with ImageStudio (see Section 4.9 [Example scripts], page 47) to learn more examples. Have fun!

Note, if you use any ARexx command which prints text out (e.g. "say"), this text is printed to the file given by the tooltype REXXOUTPUT (see Section 5.5.34 [Tooltype'REXXOUTPUT], page 131). After the script has been executed, this file can be examined.

4.4 Command templates

The parameters passed to the ARexx commands closely follow Commodore's style guidelines. The parsing of the arguments follows the standard template format described below.

Commands are always of the form:

```
command [options]
```

The command may be something like 'OPEN' or 'SCALE' and the options may be filenames, numbers etc... A typical command template may look like:

```
OPEN FILE/A,FORMAT,ARGS,FORCE/S
```

The commands and options are not case sensitive, therefore 'OPEN', 'Open' or 'open' can be used to open a file. The options after the command name are separated by commas, and are named (e.g. FILE or FORCE are option names). After the name, follows an optional modifier (e.g. /A or /S are modifiers) which describes what type of information the option specifies.

When using the command, the option names may be ommitted if the parameters for the command are given in the same order as the options in the template, but for clarity it is recommended that the option names be used.

The following modifiers are used:

No modifier

If the option has no modifier, the option is expecting a string. Strings are lines of text with no spaces; to use a string with a space, place the string in double-quotes (").

Multiple strings (/M)

Many strings can be specified if an option uses this modifier.

Numeric (/N)

Numeric options allow both positive and negative integers. Floating point numbers are not yet used by ImageStudio.

Boolean (/S)

Some options can be specified to "switch" that option on. By leaving the option out, the option is switched off.

Keyword (/K)

A keyword option shows that the option name must be used to set this option.

Always (/A)

This option must always be included in this command.

In practice, it soon becomes very easy to interpret command templates - some examples with explanations are given below:

```
OPEN FILE/A,FORMAT,ARGS,FORCE/S
```

The command 'OPEN' is used to open a file and load it into ImageStudio. OPEN requires a filename (FILE/A is a string, and is always required), an optional FORMAT string, an optional ARGS string and and optional FORCE switch. The following are valid OPEN commands:

The following would load a file called 'Pics/CheetahFace250x200.ilbm', forcing the old project to be over written:

```
OPEN FILE "Pics/CheetahFace250x200.ilbm" FORCE
```

The following would load a file called 'Ram Disk:Tulip.jpg', asking first if the current project has changed:

```
OPEN '"Ram Disk:Tulip.jpg"'
```

The following is an error, if the filename has spaces in it, it should be enclosed in single and double-quotes:

```
OPEN "Ram Disk:Tulip.jpg"
```

```
SCALE X/N,Y/N,PERCENT/S,METHOD
```

'SCALE' is used to change the size of the image (see Section 4.10.45 [ARexx`SCALE], page 98). SCALE expects two numerical values (X/N,Y/N are numbers but neither are required), can be used to scale either to an absolute size or by a given percentage (adding the PER CENT switch specifies percentage scaling) and can use a variety of methods (METHOD is used to specify a description string of the method to be used - again, if this is not specified a default method is used). The following are valid SCALE commands:

The following scales the image to 640x480 pixels:

```
SCALE X 640 Y 480
```

The following makes the image 50 percent of its original height:

```
SCALE Y 50 PERCENT
```

The following scales the image to 800x600 pixels using the colour averaging method:

```
SCALE 800 600 METHOD AVERAGE
```

The following is an error, no X or Y values given:

```
SCALE X PERCENT
```

The following is an error, floating point values are not allowed:

```
SCALE Y 127.5
```

See Section 4.10 [ARexx commands], page 52, for more detailed descriptions of each of the individual ARexx commands.

4.5 Return values

The return values for the ARexx commands are specified in the same notation as the input parameters, although the types of returned values is more limited than the input parameter types. In order for results to be returned from ARexx commands, it is essential that the line:

```
options results
```

be placed near the start of the ARexx script.

Commands may return either strings, numbers or arrays of either. By default, all ARexx commands return their values in a variable called "RESULT". This is fine if the command returns a single number or string. For example, the following call to the FILE_JOIN command (see Section 4.10.8 [ARexx`FILE`JOIN], page 58) would return the string "T:Image.ilbm" in the RESULT variable:

```
FILE_JOIN PATHPART "T:" FILEPART "Image.ilbm"
```

If the user wishes to return the result in another variable other than RESULT, they may specify the VAR keyword. For example, the following would perform the same action as above, only putting the result in the varible called "FULLNAME"

```
FILE_JOIN PATHPART "T:" FILEPART "Image.ilbm" VAR FULLNAME
```

Some ARexx commands return multiple values, and these to can be re turned in a single variable - each returned value in the variable is seperated with a space. The following returns information about the current image (see Section 4.10.16 [ARexx`IMAGEINFO`GET], page 65):

```
IMAGEINFO_GET
```

and the RESULT variable might look something like:

```
640 400 8 Pics/Zebra.ilbm 32772
```

It is possible then to extract the desired information using ARexx's built in parsing routines. A neater way to return multiple values though is through a “stem” variable. Here, a base name for a variable is given and the returned values' names get added to it. It is clearer with an example:

```
IMAGEINFO_GET STEM IMAGEINFO.
```

would return the same information as previously, only it would create the following variables:

```
IMAGEINFO.WIDTH = 640
IMAGEINFO.HEIGHT = 400
IMAGEINFO.DEPTH = 8
IMAGEINFO.FILE = Pics/Zebra.ilbm
```

```
IMAGEINFO.MODEID = 32772
```

Now you can refer easily to the returned values.

If an ARexx function returns an array of results, they are named as follows:

```
STEMNAME.RESULTNAME.NUMBER
```

with the variable STEMNAME.RESULTNAME.COUNT holding the number of returned results. Again, an example being the following which gets all the ".ilbm" images in the "Pics" dirctory:

```
FILES_MATCH PATHPART "Pics" PATTERN "#?.ilbm" STEM MATCHED.
```

which might return the following:

```
MATCHED.FILEPARTS.COUNT = 4
MATCHED.FILEPARTS.0 = Zebra.ilbm
MATCHED.FILEPARTS.1 = WilliamsFW14B.ilbm
MATCHED.FILEPARTS.2 = Spitfire.ilbm
MATCHED.FILEPARTS.3 = Brightside.ilbm
```

See Section 4.8 [ARexx tips], page 46, for more information on how to turn array stem variables into string variables.

4.6 Error checking

ImageStudio uses the standard ARexx method of returning errors, with a further extension.

Whenever a command is executed, a variable called "RC" has its value set by ARexx. If the command executed normally, RC is set to zero. If any failure happened, RC is set to either 5 (warning), 10 (failure) or 20 (serious failure).

ImageStudio also sets the value of a further variable called "RC2", which either contains a text description of the reason for failure or a standard AmigaDos error code.

A description string is returned in RC2 if a failure occurs within the execution of a command. RC2 will be an AmigaDos error number if there is an error with the command syntax (e.g. misspelled command name or missing quotes).

If, for example the user was to try to use the scale command when there was no image in the buffer, RC and RC2 would be set to the following:

```
RC = 10
RC2 = "SCALE, No image"
```

If the scale operation were to be performed with the command:

```
SCLAE 80 40
```

the following values would be set:

```
RC = 10
RC2 = 236
```

where AmigaDos error 236 represents 'not implemented', i.e. unknown command. The default blank script template will convert the most common likely AmigaDos error codes into description strings (see Commodore's AmigaDos manual for a full description of AmigaDos errors).

By default, the blank script template turns on automatic error checking. The line:

```
signal on error
```

tells ImageStudio to jump to the ERROR: label whenever a command fails. The blank script then puts up a requester showing the error.

The user may wish to turn off the automatic error checking to perform error checking themselves. This is neccessary, for example, if the user wishes to trap the user pressing 'Cancel' on a requester (this returns an error). The following checks when the user cancels the file requester:

```
/* Turn off automatic error checking */

signal off error

/* Open the requester */

REQUEST_FILE
```

```
/* Check for the error condition */

if RC ~= 0 then do
   REQUEST_MESSAGE TEXT '"An error occurred (user\n'||,
      'probably pressed Cancel)"'
   end
else do
   REQUEST_MESSAGE TEXT '"You chose: '||RESULT||'"'
   end
```

4.7 Common ARexx problems

4.7.1 ARexx problem 1

"I can't use strings with spaces in them."

Care must be taken when specifying string paramters when the string contains space characters. Single quotes must be used around double quotes to stop the string from being seen as many different strings.

Consider the following example:

```
REQUEST_MESSAGE TEXT "Hello"
```

ARexx would evaluate the string "Hello" and give ImageStudio the following command to execute:

```
REQUEST_MESSAGE TEXT Hello
```

i.e. without the double quotes. In this example, REQUEST_MESSAGE would do as expected. The problems start when strings have spaces in them; consider the following:

```
REQUEST_MESSAGE TEXT "Hello world"
```

ARexx would evaluate the string "Hello world" and give ImageStudio the following command to execute:

```
REQUEST_MESSAGE TEXT Hello world
```

which is not what is desired. The Hello becomes the TEXT value and the world becomes the value of the next parameter (BUTTONTEXT in this case). The result would be a requester with the text of "Hello" and a button called "world". Now we must use the single quotes to stop ARexx from evaluating the string:

```
REQUEST_MESSAGE TEXT '"Hello world"'
```

would send ImageStudio the following command:

```
REQUEST_MESSAGE TEXT "Hello world"
```

which shows that the whole string "Hello world" belongs to the TEXT parameter.

4.7.2 ARexx problem 2

"I can't use AmigaDos commands in a script."

There is a bug which causes ImageStudio to sometimes crash the machine if the output from AmigaDos command is not properly re-directed. The stdin and stdout for all external CLI commands should be redirected to the NIL: device.

For example:

```
address command 'rename ram:foo ram:bar'
```

could crash the machine if the rename fails (e.g. the file "ram:foo" doesn't exist). To avoid this, use:

```
address command 'rename <NIL: >NIL: ram:foo ram:bar'
```

The failure will still be trapped by ARexx and sets RC to 10.

4.7.3 ARexx problem 3

> “I can’t run scripts from the ram disk.”

This is due to ARexx scripts being treated by ARexx as external commands. Command names may not contain spaces, and scripts in the ram disk will be called something like ‘Ram Disk:MyScript.isrx’, which is not allowed.

To work around this, either move the script to a location without a space in its filename or specify the ram disk as ‘ram:’ rather than ‘Ram Disk:’.

4.7.4 ARexx problem 4

> “I can’t set the same variable twice with VAR”

If you are able to return a value from a command into a given variable name once in a program, but unable to do it again it’s probably due to ARexx evaluating your variable the second time it is used.

For example, the following won’t work:

```
FILE_JOIN FILEPART '"Work:"' '"MyFile"' VAR fullname

FILE_JOIN FILEPART '"Work:"' '"MyOtherFile"' VAR fullname
```

because ARexx will evaluate ‘fullname’ in the second FILE_JOIN, i.e. ARexx will see the second FILE_JOIN as:

```
FILE_JOIN FILEPART "Work:" "MyFile" VAR Work:MyFile
```

The solution is to enclose the variable name in single quotes to stop it from being evaluated, i.e. our second FILE_JOIN is written as:

```
FILE_JOIN FILEPART '"Work:"' '"MyOtherFile"' VAR 'fullname'
```

4.7.5 ARexx problem 5

"I can't get any ARexx script to run."

In order for ARexx to be available to ImageStudio, you must start ARexx at startup time by including the line:

```
rexxmast >NIL:
```

in your 'User-Startup' file in the 'S:' directory. Normally this line should be present in your User-Startup, but if you find no scripts run from ImageStudio you must add this line manually.

4.8 ARexx tips

4.8.1 ARexx tip 1

"How to turn stem arrays into strings."

It usually desirable for commands that return arrays to return the values in a stem, making the return values easier to deal with. In some cases it is neccessary to pass these values back to ImageStudio after reading or altering them. As ImageStudio commands can't accept stems directly, these stems have to be converted back into strings.

We suggest the following method, using the PALETTE_GET and PALETTE_SET commands as examples of getting and setting an array of values:

```
/* Get the current palette */

PALETTE_GET STEM OLDPALETTE.

/* Convert the stem to a parameter list */

NEWPALETTE = ''

do l = 0 to (OLDPALETTE.PALETTE.COUNT - 1)
   NEWPALETTE = NEWPALETTE||' '||OLDPALETTE.PALETTE.l
   end
```

```
/* Force the new palette back onto the image */

PALETTE_SET PALETTE NEWPALETTE REMAP
```

4.8.2 ARexx tip 2

“Shortening command names”

Using the current ARexx command interpreter within ImageStudio, it is possible to specify a shorter version of each ARexx command. For example, ‘OP’ could be used as a synonym for ‘OPEN’ and ‘RG’ is a synonym for ‘RGB_TO_HSV’. The following should be noted however:

> This behaviour may be removed in a future version of ImageStudio. Therefore we recommend that this feature only be used to reduce typing in the command shell (see Section 3.5.1 [Command˙shell], page 26) and not be used in ARexx scripts.
>
> If the shortened command name is ambiguous, the first matching com mand will be executed. For example, if the shortened command ‘REQUEST’ is used, ‘REQUEST_DIR’ will be executed.

4.9 Example scripts

4.9.1 BalanceTest

Description

This script allows the user to see the result of changing the brightness, contrast and gamma values over their full ranges.

An image is loaded in if one is not present already. The image is then divided into 3 strips - the top, middle and bottom representing changes to the balance, contrast and gamma respectively. Depending on the number of divisions the user chooses, each of the 3 strips is divided further horizontally and each of the brightness, contrast and gamma values are applied starting from -100 on the left to +100 on the right. An odd number of horizontal divisions are used to leave a central, vertical area of the image which remains unchanged.

Known bugs

None.

4.9.2 BatchConvert

Description

This scripts allows the conversion of multiple images to be output as one image format.

The script allows the following:

1. Saving of the converted image into a different location as the source.
2. Choose any of the available image formats to save, with controls over their sub-format.
3. Automatic file renaming.
4. Automatic deleting of source images if different from the destination image.

Known bugs

If the source and destination files are the same, but have different filenames (e.g. ‘T:Bud2.gif’ and ‘Ram:T/Bud2.gif’) the script will delete the source file (which will be the destination file). To avoid this, make sure both filenames are both specified in the same manner.

4.9.3 BatchProcess

Description

This scripts allows the processing of multiple images to be output as one image format. The script is based on ‘BatchConvert’ (see Section 4.9.2 [BatchConvert], page 48).

The commands to control the processing should be typed in to the appropriate requester as though they were ARexx commands to be executed in a script. For example:

```
SCALE 640 480
```

would scale each image to 640x480 pixels before saving out. Multiple commands can be separated with a semi-colon ‘;’, for example:

```
SCALE 800 600;COLOURS 256 DITHER FS
```

would scale the image to 800x600 and then reduce to 256 colours with Floyd-Steinberg dithering before saving out. Commands are executed from left to right.

The script allows the following:

1. Saving of the processed image into a different location as the source.
2. Choose any of the available image formats to save, with controls over their sub-format.
3. Automatic file renaming.
4. Application of multiple commands to process the image before saving.
5. Automatic deleting of source images if different from the destination image.

Known bugs

If the source and destination files are the same, but have different filenames (e.g. 'T:Bud2.gif' and 'Ram:T/Bud2.gif') the script will delete the source file (which will be the destination file). To avoid this, make sure both filenames are both specified in the same manner.

4.9.4 BatchProcessNotify

Description

This script starts a batch lot of processing to be performed when a given file is changed or created; the script is based on 'BatchConvert' (see Section 4.9.3 [BatchProcess], page 48).

This is useful if you wish to convert the output from a program that has generated multiple frames (e.g. a raytracer or landscape renderer) into a format that can be compiled into an animation (e.g. ILBM24 to HAM6).

The first thing the user must select is the filename of the final file in the sequence. When this file has been created, ImageStudio will start the processing of the images. This file may not of course have been created yet, so the user will have to type the name into the file requester.

After the user has specified the output format (see Section 4.9.3 [BatchProcess], page 48), the script will wait for the specified file be created before proceeding with the processing on all the files in the chosen directory with the same basename as the selected final filename.

Known bugs

If the source and destination files are the same, but have different filenames (e.g. 'T:Bud2.gif' and 'Ram:T/Bud2.gif') the script will delete the source file (which will be the destination file). To avoid this, make sure both filenames are both specified in the same manner.

Example

The user wishes to convert the 24bit output files created by a raytracer into HAM6 format, ready for compiling into an animation. The animation is 200 frames and the files are numbered 'pic.0001', 'pic.0002' etc... and are located in the 'Work:Render' direc tory. No extra processing is to be performed on the images.

Upon running the BatchProcessNotify script, the user enters 'Work:Render/pic.0200' as the final frame in the sequence and sets the output file format to be IFF-ILBM, HAM6.

The script now waits for the final file to be created, and upon do ing so, matches all files in the 'Work:Render' directory that start with the basename 'pic.'. 200 files should be found. Each of these files are loaded and saved as HAM6, before the script ends.

4.9.5 ConvolveTest

Description

Applies a chosen set of convolution filters to an image.

A number of convolution filters are chosen and the image is tiled depending on the number of filters chosen, each filter is applied to each of the tiles.

Known bugs

1. The convolution filters chosen to apply **must** be taken from the directory currently shown in the 'Convolves' floating palette. This is because convolution filters with the CONVOLVE command are chosen by name, not filename (see Section 4.10.3 [ARexx`CONVOLVE], page 54).
2. The tiling algorithm used isn't very smart, the number of tiles vertically is the same as the number horizontally. This can lead to almost half of the image being unaffected if the number of chosen filters is just above the nearest square number.
3. The maximum number of filters to be applied to the image is 100.

4.9.6 Crop512x512

Description

Removes blank area from unregistered images.

When a large image in loaded into the unregistered version of ImageStudio, the actual image is only contained in the top-left 512x512 pixels of the buffer. This script crops away the blank buffer to leave a buffer of only 512x512 pixels in size.

Known bugs

None.

4.9.7 Demo

Description

Demonstrates some of the features of the ImageStudio ARexx inter face.

Simply follow the prompts to see the features being explained.

Known bugs

None.

4.9.8 RegionRecall

Description

Sets a previously remembered region.

This script allows the region that was set with RegionStore (see Section 4.9.9 [RegionStore], page 51) to be recalled as the active region for the current image. The user is warned if there is already a currently selected region which will be lost if the remembered region is recalled.

Known bugs

None.

4.9.9 RegionStore

Description

Remembers a region for future recall.

The region maybe recalled by using the RegionRecall (see Section 4.9.8 [RegionRecall], page 51) script.

Known bugs

None.

4.9.10 ToIcon

Description

Resizes and remaps the image to that suitable as an icon.

The script allows the following:

1. Choosing of an alternative palette other than the default 4 colour Workbench palette.
2. Copying of the image into the clipboard, ready to be pasted into IconEdit.

Known bugs

None.

4.10 ARexx commands

More detailed information on each of the individual ARexx commands can be found below.

4.10.1 BALANCE

Command BALANCE

Parameters template

BRIGHTNESS/N, CONTRAST/N, GAMMA/N, NORED/S, NOGREEN/S, NOBLUE/S

Return template

None.

Description

This command allows the user to change the colour balance of the image. The user may select specify one of the BRIGHTNESS, CONTRAST or GAMMA values to adjust - specifying more than one will only result in the first operation being acted upon.

By default, the operation is applied to all the red, green and blue values of the image. The user may stop any of the RGB channels being affected by selecting any of the NORED, NOGREEN or NOBLUE switches. Multiple switches may be used, but not all three together.

See Section 3.5.2 [Show'balance], page 26, for a full description of altering the image's colour balance.

Parameters

BRIGHTNESS

This adjusts the brightness / darkness of the image. Valid values are between -100 (make everything black) to 100 (make everything twice as bright).

CONTRAST

This adjusts the relative difference between dark and light colours. Valid values are -100 (everything to mid-grey) to 100 (everything to extreme contrast).

GAMMA This adjusts the gamma response of the image. Valid values are -100 to 100.

Returns Nothing.

Errors rc = 0 if the operation was successful.

rc = 10 if the operation failed for any reason, rc2 will contain a string describing the problem.

Example

The following adjusts the gamma of the image by 10:

```
BALANCE GAMMA 10
```

The following darkens the green componant of the image by -20:

```
BALANCE BRIGHTNESS -20 NORED NOBLUE
```

Known bugs

None.

4.10.2 COLOURS

Command COLOURS

Parameters template

NUMCOLOURS/N, SIXTEENMILLION/S, COLOURCHOICE, DITHER

Return template

None.

Description

Allows the user to change the number of colours of the current image. The image can be changed to either 2-256 colours or 16 million colours (24bit). Methods of colour choice and dithering are available when reducing the number of colours in the image.

See Section 3.4.3 [Colours], page 23, for a full description of changing the number of colours in the image.

Parameters

NUMCOLOURS/N

The number of colours desired for the image. Valid values are 2 to 256, the result will always be a colour-mapped image.

SIXTEENMILLION/S

In order to increase the number of possible colours in the image to the maximum possible (16 million), the user should specify this switch. The user may not specify this switch as well as the NUMCOLOURS option.

COLOURCHOICE

This is a string which determines which method of colour choice should be used to reduce the number of colours in an image. At the present time, HECKBERT is the only available option and will be used by default if no COLOURCHOICE is specified.

DITHER

This string determines which method of dithering should be used when reducing the number of colours in an image. Valid strings are: NONE, FLOYD-STEINBERG or FS, STUCKI, JARVIS, BURKES, SIERRA and STEVENSON-ARCE. By default, no dithering is used.

Returns Nothing.

Errors rc = 0 if the operation was successful.

rc = 10 if the operation failed for any reason, rc2 will contain a string describing the problem.

Example

The following reduces the image to 16 colours with Floyd-Steinberg dithering:

```
COLOURS NUMCOLOURS 16 DITHER "FS"
```

The following increases the image to 24bits, allowing 16 million possible colours:

```
COLOURS SIXTEENMILLION
```

Known bugs

None.

4.10.3 CONVOLVE

Command CONVOLVE

Parameters template

NAME/A

Return template

None.

Description

Applies the named convolution to the 24bit image. Convolution can only be applied to 16 million colour (24bit) images.

See Section 3.5.4 [Show˙convolves], page 29, for a full description of convolution filters.

Parameters

NAME/A	The name of the convolution filter as it appears in the “Show convolves” floating palette.

Returns Nothing.

Errors rc = 0 if the operation was successful.

rc = 10 if the operation failed for any reason, rc2 will contain a string describing the problem.

Example

The following applies the "Blur high" convolution filter to the current image:

```
CONVOLVE NAME '"Blur high"'
```

The following increases the image to 24bits if necessary, then applies the "Edge detect" convolution filter:

```
IMAGEINFO_GET STEM IMAGEINFO.

if IMAGEINFO.DEPTH ~= 24 then do
   COLOURS SIXTEENMILLION
   end

CONVOLVE NAME '"Edge detect"'
```

Known bugs

None.

4.10.4 COPY

Command COPY

Parameters template

None.

Return template

None.

Description

Copies the current image into the clipboard buffer.

See Section 3.2.3 [Copy], page 18, for a full description of copying images to the clip board.

Parameters

None.

Returns Nothing.

Errors rc = 0 if the operation was successful.

rc = 10 if the operation failed for any reason, rc2 will contain a string describing the problem.

Example

The following copies the current image to the clipboard:

```
COPY
```

Known bugs

None.

4.10.5 CROP

Command CROP

Parameters template

None.

Return template

None.

Description

Crops the image to the currently selected region. A region must be selected for this command to work.

See Section 3.4.1 [Crop], page 22, for a full description of cropping images.

Parameters

None.

Returns Nothing.

Errors rc = 0 if the operation was successful.

rc = 10 if the operation failed for any reason, rc2 will contain a string describing the problem.

Example

The following crops the current image to the currently selected region:

```
CROP
```

The following crops the current image to the currently selected region only if a region exists:

```
REGION_GET STEM REGIONINFO.

if REGIONINFO.MINX ~= -1 then do
   CROP
   end
```

Known bugs

None.

4.10.6 EFFECT

Command EFFECT

Parameters template

NAME/A, ARGS

Return template

None.

Description

Applies the named effect to the image. Optional arguments that the effect may require can be passed via the ARGS parameter.

See Section 5.2 [Effects], page 117, for a full description of the available effects.

Parameters

NAME/A	The name of the effect to apply, as it appears in the "Effects" floating palette.
ARGS	Any optional arguments that the desired effect may require.

Returns Nothing.

Errors rc = 0 if the operation was successful.

rc = 10 if the operation failed for any reason, rc2 will contain a string describing the problem.

Example

The following applies the "Negative" effect to the image:

```
EFFECT NAME "Negative"
```

The following pixelizes the image to a size of 4 pixels:

```
EFFECT NAME "Pixelize" ARGS '"PIXELSIZE 4"'
```

Known bugs

None.

4.10.7 FILES_MATCH

Command FILES_MATCH

Parameters template

PATHPART/A, PATTERN

Return template

FILEPARTS/M

Description

Returns a list of files in a directory matching an optional pattern.

Parameters

PATHPART/A	The path (directory) name from which the filenames should be taken.
PATTERN	Optional file matching pattern, to allow the inclusion of only specific filenames. By default, all the files in a directory are returned.

Returns

FILEPARTS/M	An array of strings containing the matching filenames in the given PATH.

Errors rc = 0 if the operation was successful.

rc = 10 if the operation failed for any reason, rc2 will contain a string describing the problem.

Example

The following gets all the filenames from the "Pics" directory and returns them in the stem FILENAMES.:

```
FILES_MATCH PATHPART "Pics" STEM FILENAMES.
```

The following gets all the filenames in the current directory with a ".ilbm" or ".iff" filename extension:

```
FILES_MATCH PATHPART ’""’ PATTERN "#?.(ilbm|iff)" STEM FILENAMES.
```

The following gets all the filenames in the "S:" directory that start with an "S" and puts them in pop up requesters:

```
FILES_MATCH PATHPART "S:" PATTERN "S#?" STEM FILENAMES.

do l = 0 to (FILENAMES.FILEPARTS.COUNT - 1)
   REQUEST_MESSAGE BUTTONTEXT "More|Cancel" AUTOCANCEL,
      TEXT ’"’FILENAMES.FILEPARTS.l’"’
   end
```

Known bugs

None.

4.10.8 FILE_JOIN

Command FILE_JOIN

Parameters template

PATHPART/A, FILEPART/A

Return template

FILE

Description

Joins the path part of a filename to the file part of a filename, returning the full filename. Adds ‘/’ and ‘:’ where appropriate to create a full filename.

Parameters

PATHPART/A

The path (directory) part of the filename to be created.

FILEPART/A

The file part of the filename to be created.

Returns

FILE	The full filename created from the path and file parts.

Errors rc = 0 if the operation was successful.

rc = 10 if the operation failed for any reason, rc2 will contain a string describing the problem.

Example

The following creates the filename "Pics/HappyFace_240x250.bmp" from the seperate path and fileparts - the result is put in a pop up requester:

```
FILE_JOIN PATHPART "Pics" FILEPART "HappyFace_240x250.bmp"

REQUEST_MESSAGE TEXT '"'RESULT'"'
```

The following creates the filename "T:TempImage.jpg" from the seperate path and fileparts (note how the '/' seperater is not needed) - the result is put in a pop up requester:

```
FILE_JOIN PATHPART "T:" FILEPART "TempImage.jpg"

REQUEST_MESSAGE TEXT '"'RESULT'"'
```

Known bugs

None.

4.10.9 FILE_RENAME

Command FILE_RENAME

Parameters template

FILE/A, FROM/A, TO/A

Return template

FILE

Description

Replaces the last occurrance of a given string in a filename with another string. Useful for renaming filename extensions.

Note: **This command doesn't actually rename the file**, it simply returns what the new filename should be.

Hint: To rename *any* filename extension to a chosen extension, you can set "FROM ." and "TO .newextension". This removes any old extension and replaces it with the given new extension. This can be useful if you are converting a large number of different format files to one format (see Section 4.9.2 [BatchConvert], page 48).

Parameters

FILE/A	The original filename to be renamed.
FROM/A	The string to remove from the old filename.
TO/A	The string to replace the FROM string in the filename.

Returns

FILE	The renamed filename. If no FROM string was found in the original filename, the original filename is returned with the new TO string appended.

Errors rc = 0 if the operation was successful.

rc = 10 if the operation failed for any reason, rc2 will contain a string describing the problem.

Example

The following renames the filename "Zebra_250x250.pcx" to "Zebra_250x250.ilbm", the final filename is placed in RESULT:

```
FILE_RENAME FILE "Zebra_250x250.pcx" FROM ".pcx" TO ".ilbm"
```

The following appends ".out" to the filename "pic.0001":

```
FILE_RENAME FILE "pic.0001" FROM "XXX" TO ".out"
```

Known bugs

None.

4.10.10 FILE_SPLIT

Command FILE_SPLIT

Parameters template

FILE/A

Return template

PATHPART, FILEPART

Description

Splits the given filename into seperate path and file parts.

Parameters

FILE/A	The full filename to be split.

Returns

PATHPART	The path (directory) part of the filename.
FILEPART	The file part of the filename.

Errors rc = 0 if the operation was successful.

rc = 10 if the operation failed for any reason, rc2 will contain a string describing the problem.

Example

The following seperates the filename "Pics/HappyFace_240x250.bmp" into seperate path and fileparts - the result is put in a pop up requester:

```
FILE_SPLIT FILE "Pics/HappyFace_240x250.bmp" STEM FILENAME.

REQUEST_MESSAGE TEXT '"Path:'FILENAME.PATHPART,
   'File:'FILENAME.FILEPART'"'
```

The following seperates the filename "T:TempImage.jpg" into seperate path and fileparts - the result is put into the default settings of a file requester:

```
FILE_SPLIT FILE "T:TempImage.jpg" STEM FILENAME.

REQUEST_FILE PATHPART '"'FILENAME.PATHPART'"',
   FILE '"'FILENAME.PATHPART'"'
```

Known bugs

None.

4.10.11 FULL_IMAGE

Command FULL_IMAGE

Parameters template

None.

Return template

None.

Description

Displays the full image in the preview window.

See Section 3.3.1 [Full˜image], page 20, for a full description of this command.

Parameters

None.

Returns Nothing.

Errors rc = 0 if the operation was successful.

rc = 10 if the operation failed for any reason, rc2 will contain a string describing the problem.

Example

The following displays the full image in the preview window:

```
FULL_IMAGE
```

Known bugs

None.

4.10.12 GUI_BLOCK

Command GUI_BLOCK

Parameters template

None.

Return template

None.

Description

Blocks all input to any open ImageStudio windows. This command is used to stop the user from entering any more input into the ImageStudio windows whilst an ARexx script is running. If the script has been started from ImageStudio (i.e. from the "Scripts" floating palette), all the GUI blocking / unblocking is handled automatically - the GUI is blocked when the script starts and unblocked when it finishes.

If the script is started externally (i.e. from another ARexx program or from the CLI using 'rx'), the user should block the GUI if they think the ARexx is going to spend a long time processing some in formation. The GUI is still automatically blocked when a requester is opened however.

Parameters

None.

Returns Nothing.

Errors rc = 0 if the operation was successful.

rc = 10 if the operation failed for any reason, rc2 will contain a string describing the problem.

Example

The following blocks all input to the ImageStudio GUI:

```
GUI_BLOCK
```

Known bugs

None.

4.10.13 GUI_UNBLOCK

Command GUI_UNBLOCK

Parameters template

None.

Return template

None.

Description

Unblocks all input to any open ImageStudio windows after a GUI_BLOCK command. If the script has been started from ImageStudio (i.e. from the "Scripts" floating palette), all the GUI blocking / unblocking is handled automatically - the GUI is blocked when the script starts and unblocked when it finishes.

If the scripts is started externally (i.e. from another ARexx pro gram or from the CLI using 'rx'), the user should unblock the GUI after a GUI_BLOCK command has been issued. The GUI is still automatically unblocked after a requester has been satisfied however.

Parameters

None.

Returns Nothing.

Errors rc = 0 if the operation was successful.

rc = 10 if the operation failed for any reason, rc2 will contain a string describing the problem.

Example

The following unblocks all input to the ImageStudio GUI:

```
GUI_UNBLOCK
```

Known bugs

None.

4.10.14 HELP

Command HELP

Parameters template

COMMAND

Return template

COMMANDDESC, COMMANDLIST/M

Description

Returns help on a given ARexx command. This command is meant mainly for use with the command shell (see Section 3.5.1 [Command shell], page 26), as it is of very little use within a script. Both the command's parameter and return templates are returned.

Parameters

COMMAND

The ARexx command to obtain help on.

Returns

COMMANDDESC

The parameter template.

COMMANDLIST/M

The result template.

Errors rc = 0 if the operation was successful.

rc = 10 if the operation failed for any reason, rc2 will contain a string describing the problem.

Example

The following gets help on the ARexx OPEN command:

```
HELP OPEN
```

The following gets help on the HELP command:

```
HELP HELP
```

Known bugs

None.

4.10.15 HSV_TO_RGB

Command HSV_TO_RGB

Parameters template

H/N/A, S/N/A, V/N/A

Return template

R/N, G/N, B/N

Description

Converts a HSV colour value into a RGB colour value.

See Section 5.4 [Colour representations], page 122, for more details on RGB and HSV colour representations.

Parameters

H/N/A	The hue value of the colour to convert. Valid values are 0 to 360.
S/N/A	The saturation value of the colour to convert. Valid values are 0 to 100.
V/N/A	The value of the colour to convert. Valid values are 0 to 100.

Returns

R/N	The red componant value of the colour.
G/N	The green componant value of the colour.
B/N	The blue componant value of the colour.

Errors rc = 0 if the operation was successful.

rc = 10 if the operation failed for any reason, rc2 will contain a string describing the problem.

Example The following converts yellow from HSV to RGB representation, putting the result in RESULT:

```
HSV_TO_RGB 60 100 100
```

The following converts mid-grey from HSV to RGB representation, putting the result in the stem COLOUR.:

```
HSV_TO_RGB 0 0 49 STEM COLOUR.
```

Known bugs

None.

4.10.16 IMAGEINFO_GET

Command IMAGEINFO_GET

Parameters template

None.

Return template

WIDTH/N, HEIGHT/N, DEPTH/N, FILE, MODEID/N, CHANGED/N

Description

Returns information about the current image. If no image is currently loaded, -1 is returned in all the numeric fields.

Parameters

None.

Returns

WIDTH/N

The width of the image in pixels, -1 if no image is loaded.

HEIGHT/N

The height of the image in pixels, -1 if no image is loaded.

DEPTH/N

The colour depth of the image, -1 if no image is loaded. Returns 1 to 8 for 2 to 256 colour images, 24 for 16 million colour images.

FILE

The full filename of the current image.

MODEID/N

The current screenmode of the image. This number is not meant to be interpreted directly, but can be used to be passed to the screenmode requester (see Section 4.10.41 [ARexx`REQUEST`SCREENMODE], page 91). When the image loaded is a non IFF-ILBM image, this screenmode value is "guessed" at by ImageStudio to be the closest Amiga equivalent based on the image's dimensions.

CHANGED/N

A numeric value, taking the value 1 to represent a change in the current project or 0 for no change.

Errors rc = 0 if the operation was successful.

rc = 10 if the operation failed for any reason, rc2 will contain a string describing the problem.

Example

The following gets the current image's information and returns it in the stem IMAGE.:

```
IMAGEINFO_GET STEM IMAGE.
```

The following gets the current image's info and opens a screenmode requester with the current screenmode if an image is loaded:

```
IMAGEINFO_GET STEM IMAGE.

if IMAGE.WIDTH ~= -1 then do
   REQUEST_SCREENMODE MODEID IMAGE.MODEID
   end
```

Known bugs

None.

4.10.17 IMAGEINFO_SET

Command IMAGEINFO_SET

Parameters template

MODEID/N

Return template

None.

Description

Sets information about the current image. Currently, only the image's screenmode can be set.

Parameters

MODEID/N

The current screenmode ID of the image.

Returns Nothing.

Errors rc = 0 if the operation was successful.

rc = 10 if the operation failed for any reason, rc2 will contain a string describing the problem.

Example

The following sets the current image's screenmode to LoRes.

```
IMAGEINFO_SET MODEID 0
```

The following opens up a screenmode requester and allows the user to choose the screenmode of the current image:

```
REQUEST_SCREENMODE STEM SCREENMODEINFO.

IMAGEINFO_SET MODEID SCREENMODEINFO.MODEID
```

Known bugs

None.

4.10.18 NOTIFY_DIR

Command NOTIFY_DIR

Parameters template

PATHPART/A

Return template

FILEPART, ACTION

Description

Monitors the specified directory and returns when a file is either updated or added to the directory. The affected filename is returned as well as the action that had been performed (either updated or added).

Whilst the command is waiting for any change in the specified directory, the fuelgauge will flash and the user may press the 'Abort' button on the infobar to cancel the operation.

Parameters

PATHPART

The path (directory) to be monitored.

Returns

FILEPART

The filename of the file that has been changed; the filename returned is without the full pathname. See Section 4.10.8 [ARexx`FILE`JOIN], page 58, for information on how to add the path part of the filename to create a full filename.

ACTION A string containing a descripion of the action performed on FILE, either "ADDED" if the file is new to the directory or "UPDATED" if the file has been updated since the notify started.

Errors rc = 0 if the operation was successful.

rc = 10 if the operation failed for any reason, rc2 will contain a string describing the problem.

Example

The following monitors the ram disk for any change, returning any change in the NOTIFYINFO. stem:

```
NOTIFY_DIR PATHPART "ram:" STEM NOTIFYINFO.
```

The following monitors the "Pics" directory for any change and pops up a requester informing the user of what has happened:

```
NOTIFY_DIR PATHPART "Pics" STEM NOTIFYINFO.

REQUEST_MESSAGE TEXT '"'NOTIFYINFO.FILEPART||' has been '||,
   NOTIFYINFO.ACTION'"'
```

Known bugs

1. AmigaDos won't notify us if a file is deleted from the directory.
2. If many files are added / updated in the directory quickly, NOTIFY_DIR may not necessarily return the first changed file. The same is true if the a file is created with an icon, NOTIFY_DIR may return the name of the ".info" file.
3. File notification is not implemented on all filesystems (notably some network filesystems). No problems occur with either the stand ard OFS or FFS filesystems.

4.10.19 NOTIFY_FILE

Command NOTIFY_FILE

Parameters template

FILE/A

Return template

None.

Description

Waits for a change in the specified file. The function will return if either a new file by the given name is created, or if the file is updated. Unlike NOTIFY_DIR (see Section 4.10.18 [ARexx'NOTIFY'DIR], page 67), NOTIFY_FILE also returns if the specified file is deleted.

Whilst the command is waiting for any change in the specified file, the fuelgauge will flash and the user may press the 'Abort' button on the infobar to cancel the operation.

This function can be used to trigger ImageStudio to perform a given set of operations when the specified file has been created. For example, if 50 frames of an animation were being rendered by a ray-tracer then ImageStudio could be told to wait for the last frame to be created an then convert them all to HAM format.

Parameters

FILE	The file to be monitored.

Returns Nothing.

Errors rc = 0 if the operation was successful.

rc = 10 if the operation failed for any reason, rc2 will contain a string describing the problem.

Example

The following monitors the file "Pics/CheetahFace250x200.ilbm" for any change:

```
NOTIFY_FILE "Pics/CheetahFace250x200.ilbm"
```

The following waits for the 50th frame in the sequence "Render." to be created in the "Work:RayTrace":

```
NOTIFY_FILE "Work:RayTrace/Render.050"
```

Known bugs

File notification is not implemented on all filesystems (notably some network filesystems). No problems occur with either the stand ard OFS or FFS filesystems.

4.10.20 OPEN

Command OPEN

Parameters template

FILE/A, FORMAT, ARGS, FORCE/S

Return template

None.

Description

Loads the specified file into ImageStudio. Most file formats are automatically recognised by the program, but it is possible to specify extra information with the FORMAT and ARGS parameters.

Parameters

FILE	The filename of the file to be loaded.
FORMAT	Most file formats are automatically recognised by ImageStudio, but some raw formats need to be specified. If the file to be loaded is known to be a raw format, this parameter should be used to specify the file format. See Section 5.1 [File formats], page 104, for more information on raw file formats.
ARGS	Some file formats require extra information to be specified a load time, this parameter should be used to specify more information. See Section 5.1 [File formats], page 104, for more information on extra arguments allowed by the loaders.
FORCE/S	By default the user will be warned if they are about to overwrite the current project. By specifying FORCE, the user is not warned.

Returns Nothing.

Errors rc = 0 if the operation was successful.

rc = 10 if the operation failed for any reason, rc2 will contain a string describing the problem.

Example

The following opens the file "Pics/CheetahFace250x200.ilbm":

```
OPEN "Pics/CheetahFace250x200.ilbm"
```

Known bugs

None.

4.10.21 PALETTE_GET

Command PALETTE_GET

Parameters template

None.

Return template

PALETTE/N/M

Description

Gets the palette information from the current image.

Parameters

None.

Returns

PALETTE/N/M

An array of the colours in the palette, ordered red, green then blue. Check PALETTE.COUNT for the number of entries in the array, divide this value by 3 to get the number of colours in the palette.

Errors rc = 0 if the operation was successful.

rc = 10 if the operation failed for any reason, rc2 will contain a string describing the problem.

Example

The following gets the palette from the current image and returns it in the PALETTE. stem:

```
PALETTE_GET STEM PALETTE.
```

The following gets the palette from the current image (if possible) and puts the first colour value in a requester:

```
IMAGEINFO_GET STEM IMAGEINFO.

if IMAGEINFO.DEPTH ~= 24 then do
   PALETTE_GET STEM PALETTE.
```

```
            numcolours = PALETTE.PALETTE.COUNT / 3

            REQUEST_MESSAGE TEXT '"'numcolours||' colours, colour 0 = '||,
               PALETTE.PALETTE.0||','||PALETTE.PALETTE.1||','||,
               PALETTE.PALETTE.2||'"'
            end
         else do
            REQUEST_MESSAGE TEXT '"Image has no palette."'
            end
```

Known bugs

None.

4.10.22 PALETTE_LOAD

Command PALETTE_LOAD

Parameters template

FILE/A, DITHER

Return template

None.

Description

Loads and remaps a palette onto the current image. Dithering is also allowed to get a better approximation with the new palette.

Parameters

FILE/A	The filename of the palette file to load.
DITHER	A string containing the name of the dither to apply when applying the new palette. The same dither names as the COLOURS command are used (see Section 4.10.2 [ARexx'COLOURS], page 53). By default, no dithering is applied.

Returns Nothing.

Errors rc = 0 if the operation was successful.

rc = 10 if the operation failed for any reason, rc2 will contain a string describing the problem.

Example

The following loads a general 256 colour palette onto the image:

```
PALETTE_LOAD FILE "Palettes/General256.palette"
```

The following loads a general 16 colour palette with Floyd-Steinberg dithering onto the image:

```
PALETTE_LOAD FILE "Palettes/General16.palette" DITHER "FS"
```

Known bugs

None.

4.10.23 PALETTE_SAVE

Command PALETTE_SAVE

Parameters template

FILE/A

Return template

None.

Description

Saves the palette of the current image out to disk. The image must be colour-mapped for this operation to work.

Parameters

FILE/A The filename of the palette file to save.

Returns Nothing.

Errors rc = 0 if the operation was successful.

rc = 10 if the operation failed for any reason, rc2 will contain a string describing the problem.

Example

The following saves the current image's palette to the ram disk:

```
PALETTE_SAVE FILE "ram:Image.palette"
```

The following only saves out the palette of the current image if the current image is colour-mapped:

```
IMAGEINFO_GET STEM IMAGEINFO.

if IMAGEINFO.DEPTH ~= 24 then do
   PALETTE_SAVE FILE "Image.palette"
   end
```

Known bugs

None.

4.10.24 PALETTE_SET

Command PALETTE_SET

Parameters template

PALETTE/N/M/A, REMAP/S

Return template

None.

Description

Forces the array of numbers as the current palette for the image. The depth of the resultant image is taken from the number of entries in the array. This is useful for adding colours into the current image's palette.

Parameters

PALETTE/N/M/A

The array of numbers that will build the palette. The total number of elements in the array determines the number of palette entries and the depth of the resultant images. The entries in the array are arranged colour0_red, colour0_green, colour0_blue, colour1_red etc... The values of the red, green and blue values are 0 to 255.

REMAP

By default, the given palette is forced up on the current image. By specifying the REMAP switch, the image can be remapped to best fit the new palette.

Returns Nothing.

Errors rc = 0 if the operation was successful.

rc = 10 if the operation failed for any reason, rc2 will contain a string describing the problem.

Example

The following sets the current image to black and white, remapping as best as possible:

```
PALETTE_SET PALETTE "0 0 0 255 255 255" REMAP
```

The following reduces the number of colours in the image to 29, then sets the top 3 colours to be red, white and blue. This is an example of how the returned stem value can be turned into a list of parameters for another command:

```
/* Reduce the number of colours */

COLOURS NUMCOLOURS 29 DITHER "FS"

/* Get the current palette */

PALETTE_GET STEM OLDPALETTE.

/* Set the top 3 colours to red, white and blue */

OLDPALETTE.PALETTE.87 = 255  /* Red */
OLDPALETTE.PALETTE.88 = 0
OLDPALETTE.PALETTE.89 = 0
```

```
OLDPALETTE.PALETTE.90 = 255  /* White */
OLDPALETTE.PALETTE.91 = 255
OLDPALETTE.PALETTE.92 = 255

OLDPALETTE.PALETTE.93 = 0    /* Blue */
OLDPALETTE.PALETTE.94 = 0
OLDPALETTE.PALETTE.95 = 255

/* Convert the stem to a parameter list */

NEWPALETTE = ''

do l = 0 to (OLDPALETTE.PALETTE.COUNT - 1)
   NEWPALETTE = NEWPALETTE||' '||OLDPALETTE.PALETTE.l
   end

/* Force the new palette onto the image */

PALETTE_SET PALETTE NEWPALETTE REMAP
```

Known bugs

None.

4.10.25 PALETTE_SORT

Command PALETTE_SORT

Parameters template

FROM/N, TO/N, LIGHTTODARK/S

Return template

None.

Description

Sorts the colours in the palette into ascending / descending order of brightness. The whole palette can be sorted or a selected range.

The colours in the palette are numbered from zero, so a 32 colour image would have palette entries 0 to 31 inclusive.

The image in automatically remapped to the new palette after the operation.

Parameters

FROM/N	The first colour in the palette to sort from. By default this is zero.
TO/N	The last colour in the palette to sort to. By default this is the last colour in the image's palette.
LIGHTTODARK/S	By default the palette is sorted from dark to light. This option allows the palette to be sorted light to dark.

Returns Nothing.

Errors rc = 0 if the operation was successful.

rc = 10 if the operation failed for any reason, rc2 will contain a string describing the problem.

Example

The following sorts the colours in the image's palette dark through to light:

```
PALETTE_SORT
```

The following sorts the lower 32 colours in a 64 colour image from light to dark:

```
PALETTE_SORT FROM 0 TO 31 LIGHTTODARK
```

Known bugs

None.

4.10.26 PASTE

Command PASTE

Parameters template

FORCE/S

Return template

None.

Description

Pastes the image in the clipboard into the program.

Parameters

FORCE/S	By default the user is warned if the the current project is unsaved and they are about to overwrite it. This parameter will not warn the user and overwrite the project regardless.

Returns Nothing.

Errors rc = 0 if the operation was successful.

rc = 10 if the operation failed for any reason, rc2 will contain a string describing the problem.

Example

The following pastes the image in the clipboard, warning the user if the current project isn't saved:

```
PASTE
```

The following pastes the image in the clipboard with no warning to the user if the current project isn't saved:

```
PASTE FORCE
```

Known bugs

With v2.0.x, if the user presses 'Abort' when the file is being pasted, an error message is not returned and so the ARexx script as sumes the file pasted OK. This will be fixed when the loaders / savers become external modules.

4.10.27 PREF_GET

Command PREF_GET

Parameters template

NAME/A

Return template

VALUE

Description

Allows the user to read any of the preferences values currently in use by the program.

See Section 3.1.4 [Prefs], page 14, for a full description of the available preference values.

Parameters

NAME/A	The name of the preference whose value should be returned. The tooltype name is given here, so to read the virtual memory pagesize for example, NAME would be PAGESIZE. If the preference name is not found, an error is returned.

Returns

VALUE	The value of the preference. If the preference is a string, VALUE is the string value, if the preference is numeric, VALUE is the number value and if the preference is boolean, VALUE is either the value 1 for a positive setting ("YES" or "ON") or 0 for a negative setting ("NO" or "OFF").

Errors rc = 0 if the operation was successful.

rc = 10 if the operation failed for any reason, rc2 will contain a string describing the problem.

Example

The following finds the current virtual memory pagesize:

```
PREF_GET NAME "PAGESIZE"

say 'The pagesize is 'RESULT * 1024' bytes'
```

The following detects whether the preview dithering is being used:

```
PREF_GET NAME "PREVIEWDITHER"

if RESULT == 1 then do
   say 'Preview dithering is ON'
   end
else do
   say 'Preview dithering is OFF'
   end
```

Known bugs

None.

4.10.28 PREF_SET

Command PREF_SET

Parameters template

NAME/A, VALUE/A

Return template

None.

Description

Allows the user to set any of the preferences values currently in use by the program. Changing some preference variables may have no effect until the next time the program is run.

See Section 3.1.4 [Prefs], page 14, for a full description of the available preference values.

Parameters

NAME/A The name of the preference whose value should be changed. The tooltype name is given here, so to set the virtual memory pagesize for example, NAME would be PAGESIZE.

If the preference name is not found, an error is returned.

VALUE/A

The desired value of the preference. If the preference is a string, VALUE should be a string value, if the preference is numeric, VALUE should be a number value. If the preference is boolean, VALUE can be either the strings "YES" or "ON" to set a positive value, "NO" or "OFF" to set a negative value.

Returns Nothing.

Errors rc = 0 if the operation was successful.

rc = 10 if the operation failed for any reason, rc2 will contain a string describing the problem.

Example

The following turns the preview redraw off:

```
PREF_SET NAME "PREVIEWREDRAW" VALUE "OFF"
```

The following sets the virtal memory pagesize to be 200K:

```
PREF_SET NAME "PAGESIZE" VALUE 200
```

Known bugs

None.

4.10.29 QUIT

Command QUIT

Parameters template

FORCE/S

Return template

None.

Description

Quits the program. By using the FORCE option, the program can be forced to quit without warning the user.

The program cannot be quit by issuing the QUIT command from the command shell.

Parameters

FORCE/S By default the user is warned if the program is about to quit and the current project remains unsaved. Specifying this parameter will force the program to quit regardless.

Returns Absolutely nothing.

Errors rc = 0 if the operation was successful.

rc = 10 if the operation failed for any reason, rc2 will contain a string describing the problem.

Example

The following quits the program, warning the user if the current project is unsaved:

```
QUIT
```

The following quits the program regardless of the status of the current project:

```
QUIT FORCE
```

Known bugs

None.

4.10.30 REDO

Command REDO

Parameters template
None.

Return template
None.

Description
Re-does the last UNDO operation (see Section 4.10.49 [ARexx'UNDO], page 101).

Parameters
None.

Returns Nothing.

Errors rc = 0 if the operation was successful.

rc = 10 if the operation failed for any reason, rc2 will contain a string describing the problem.

Example

The following re-does the last UNDO operation:

```
REDO
```

Known bugs
None.

4.10.31 REDRAW

Command REDRAW

Parameters template
None.

Return template
None.

Description
Forces a redraw of the image in the preview window. This is not normally needed, as all the redrawing is done automatically however it could be used if the PREVIEWREDRAW preference is changed within a script.

Parameters
None.

Returns Nothing.

Errors rc = 0 if the operation was successful.

rc = 10 if the operation failed for any reason, rc2 will contain a string describing the problem.

Example

The following forces a redraw of the image in the preview window:

```
REDRAW
```

The following forces a redraw of the image after the preview redraw has been turned off with the PREVIEWREDRAW preference:

```
PREF_SET NAME "PREVIEWREDRAW" VALUE "OFF"

REDRAW
```

Known bugs

None.

4.10.32 REGION_CLEAR

Command REGION_CLEAR

Parameters template

None.

Return template

None.

Description

Removes the currently selected region, if one exists. No error is given if a region doesn't exist.

Parameters

None.

Returns Nothing.

Errors rc = 0 if the operation was successful.

rc = 10 if the operation failed for any reason, rc2 will contain a string describing the problem.

Example

The following clears the currently selected region:

```
REGION_CLEAR
```

The following checks that a region is selected before trying to clear it:

```
REGION_GET STEM REGIONINFO.
```

```
          if REGIONINFO.MINX ~= -1 then do
             REGION_CLEAR
             end
```

Known bugs

None.

4.10.33 REGION_GET

Command REGION_GET

Parameters template

None.

Return template

MINX/N, MINY/N, MAXX/N, MAXY/N, WIDTH/N, HEIGHT/N

Description

Gets the current region dimensions from the image. If no region is selected, -1 is returned in all the fields.

The values returned are the values of all the pixels inside the selected region. For example, if the top left pixel only of the image was selected the following values would be returned:

```
MINX = 0
MINY = 0
MAXX = 0
MAXY = 0
WIDTH = 1
HEIGHT = 1
```

Parameters

None.

Returns

MINX/N	The left-most pixel included in the selected region, -1 if no region is selected.
MINY/N	The top-most pixel included in the selected region, -1 if no region is selected.
MAXX/N	The right-most pixel included in the selected region, -1 if no region is selected.
MAXY/N	The bottom-most pixel included in the selected region, -1 if no region is selected.
WIDTH/N	The width of the selected region, -1 if no region is selected.
HEIGHT/N	The height of the selected region, -1 if no region is selected.

Errors rc = 0 if the operation was successful.

rc = 10 if the operation failed for any reason, rc2 will contain a string describing the problem.

Example

The following gets the currently selected region and returns the value in the REGIONINFO. stem:

```
REGION_GET STEM REGIONINFO.
```

The following checks that a region is selected, popping up an message requester:

```
REGION_GET STEM REGIONINFO.

if REGIONINFO.MINX ~= -1 then do
   REQUEST_MESSAGE TEXT '"Width,Height = '||REGIONINFO.WIDTH||,
      ','||REGIONINFO.HEIGHT||'"'
   end
else do
   REQUEST_MESSAGE TEXT '"No region selected"'
   end
```

Known bugs

None.

4.10.34 REGION_SET

Command REGION_SET

Parameters template

None.

Return template

XSTART/N, YSTART/N, TO/S, XEND/N, YEND/N

Description

Sets the selected region of the image. The region can either be specifed by the coordinates of its corners or by its width, height and position.

Parameters

XSTART/N
The left-most co-ordinate included in the region.

YSTART/N
The top-most co-ordinate included in the region.

TO/S
By default the region is specified by the co-ordinates of its top-left corners and its width and height. By using the TO parameter, the region can be specified with the lower-bottom co-ordinate of the region.

XEND/N
The width of the region. If the TO parameter is used, this value is used to specify the right-most pixel included by the region.

YEND/n
The height of the region. If the TO parameter is used, this value is used to specify the bottom-most pixel included by the region.

Returns Nothing.

Errors rc = 0 if the operation was successful.

rc = 10 if the operation failed for any reason, rc2 will contain a string describing the problem.

Example

The following sets the selected region to have it's top-left corner at (10,20) with a width of 30 and height of 40:

```
REGION_SET 10 20 30 40
```

The following sets the selected region to have it's top-left corner at (50,60) and its bottom-right corner to include (70,80):

```
REGION_SET 50 60 TO 70 80
```

Known bugs

None.

4.10.35 REQUEST_DIR

Command REQUEST_DIR

Parameters template

PATHPART, TITLE

Return template

PATHPART

Description

Opens a directory requester, allowing the user to choose a directory name.

The other ImageStudio windows are automatically blocked when the requester is opened and unblocked when the requester is closed.

In common with all ImageStudio requesters, if the user presses 'Cancel', an error message is returned. For the script to trap this error, global error checking must be turned off. See Section 4.6 [Error checking], page 41, for more information.

Parameters

PATHPART
The default path name to put in the requester.

TITLE The text for the title bar of the requester.

Returns

PATHPART
The selected path from the requester.

Errors rc = 0 if the operation was successful.

rc = 10 if the operation failed for any reason or the user cancelled requester, rc2 will contain a string describing the problem.

Example

The following puts up a directory requester, with the results being put in the DIRINFO. stem:

```
REQUEST_DIR STEM DIRINFO.
```

The following puts up a directory requester with a default directory of "T:", the result being printed in a message requester:

```
REQUEST_DIR PATHPART "T:" STEM DIRINFO.

REQUEST_MESSAGE TEXT '"You chose '||DIRINFO.PATHPART||'"'
```

Known bugs

None.

4.10.36 REQUEST_FILE

Command REQUEST_FILE

Parameters template

PATHPART, FILEPART, PATTERN, TITLE

Return template

FILE

Description

Opens a file requester, allowing the user to choose a filename.

The other ImageStudio windows are automatically blocked when the requester is opened and unblocked when the requester is closed.

In common with all ImageStudio requesters, if the user presses 'Cancel', an error message is returned. For the script to trap this error, global error checking must be turned off. See Section 4.6 [Error checking], page 41, for more information.

Parameters

PATHPART
: The default path name to put in the requester.

FILEPART
: The default filename to put in the requester.

PATTERN
: An AmigaDos pattern matching pattern, will only show files in the requester which match the given pattern. By default, all files are shown.

TITLE
: The text for the title bar of the requester.

Returns

FILE
: The selected filename from the requester, the filename consists of both the FILEPART and PATHPART parts.

Errors

rc = 0 if the operation was successful.

rc = 10 if the operation failed for any reason or the user cancelled requester, rc2 will contain a string describing the problem.

Example

The following puts up a file requester, with the results being put in the FILEINFO. stem:

```
REQUEST_FILE STEM FILEINFO.
```

The following puts up a file requester with the result being printed in a message requester. The default file is "Pics/HappyFace240x250.bmp":

```
REQUEST_FILE PATHPART "Pics" FILEPART "HappyFace240x250.bmp",
   STEM FILEINFO.

REQUEST_MESSAGE TEXT '"You chose '||FILEINFO.FILE||'"'
```

The following will only show files with a ".ilbm" file extension:

```
REQUEST_FILE PATTERN "#?.ilbm"
```

Known bugs

None.

4.10.37 REQUEST_LIST

Command REQUEST_LIST

Parameters template

STRINGS/M/A, TITLE

Return template

NUMBER/N, STRING

Description

Opens a requester containing a list of options for the user to choose.

The other ImageStudio windows are automatically blocked when the requester is opened and unblocked when the requester is closed.

In common with all ImageStudio requesters, if the user presses 'Cancel', an error message is returned. For the script to trap this error, global error checking must be turned off. See Section 4.6 [Error checking], page 41, for more information.

Parameters

STRINGS/M/A
The a list of string options for the user to choose.

TITLE The text for the title bar of the requester.

Returns

NUMBER/N
The number in the list of the selected string. The strings are numbered from zero, so selecting the first choice in the list would set NUMBER to 0.

STRING The selected string.

Errors rc = 0 if the operation was successful.

rc = 10 if the operation failed for any reason, or the user cancelled requester, or no choice was made. rc2 will contain a string describing the problem.

Example

The following puts up a list requester, with the results being put in the LISTINFO. stem:

```
REQUEST_LIST STRINGS "IFF-ILBM" "PCX" "BMP" STEM LISTINFO.
```

The following puts up a list requester, the result being printed in a message requester:

```
REQUEST_LIST STRINGS "First" "Second" "Third" STEM LISTINFO.

REQUEST_MESSAGE TEXT '"You chose '||LISTINFO.STRING||','||,
   ' option '||LISTINFO.NUMBER'"'
```

Known bugs

None.

4.10.38 REQUEST_MESSAGE

Command REQUEST_MESSAGE

Parameters template

TEXT/A, BUTTONTEXT, AUTOCANCEL/S, TITLE

Return template

NUMBER/N

Description

Opens a general purpose message requester. Simple messages can be presented to the user for them to "OK" them. OK / Cancel requesters can be built with this requester, as well a complex multiple choice requesters.

When designing requesters, it is worth remembering the following rules:

1. The “Negative” response should be placed on the far right-hand button. For example, the ‘Cancel’ button should be placed here.
2. The “Positive” response should be placed on the far left-hand but ton. For example, the ‘OK’ button should be placed here.
3. Try to word your requesters to keep the positive and negative text as "OK" and "Cancel". Using options like "Go to it" and "Stop right here" doesn’t make for a very intuitive interface.
4. Keep the request text short. The user shouldn’t have to read a screen full of text to find out what to do next.
5. You should **NEVER** swap the "OK" and "Cancel" buttons around.
6. The last point is **VERY** important.

The other ImageStudio windows are automatically blocked when the requester is opened and unblocked when the requester is closed.

If the AUTOCANCEL option is used and the user presses ‘Cancel’, an error message is returned. For the script to trap this error, global error checking must be turned off. See Section 4.6 [Error checking], page 41, for more information.

Parameters

TEXT/A The text to put into the requester. The text may contain multiple lines by including the ‘\n’ characters in the string (see examples below).

BUTTONTEXT

The text for the buttons of the requester. The different buttons are seperated with a ‘|’ character (i.e. BUTTONTEXT "OK|Cancel"). By default, only an "OK" button is placed in the requester.

AUTOCANCEL/S

By default REQUEST_MESSAGE simply returns the number of the button that the user selected. If the requester is of the OK / Cancel variety, specifying the AUTOCANCEL switch allows the requester to stop the script should the user press ‘Cancel’.

TITLE The text for the title bar of the requester.

Returns

NUMBER The number of the selected button. If the requester has one button, NUMBER is set to 0. For more that one button, the right-most button sets NUMBER to 0, with the buttons being numbered from 1 upwards working left to right. For example, with a BUTTONTEXT of "OK|Save first|Cancel", "OK" would return 1, "Save first" would return 2 and "Cancel" would return 0.

Errors rc = 0 if the operation was successful.

rc = 10 if the operation failed for any reason, rc2 will contain a string describing the problem. rc will also be set to 10 if the AUTOCANCEL option is used and the user selects 'Cancel'.

Example

The following puts up a message requester:

```
REQUEST_MESSAGE TEXT '"Operation finished"'
```

The following puts up a OK / Cancel requester, stopping the script if the user selects 'Cancel':

```
REQUEST_MESSAGE TEXT '"Continue ?"' BUTTONTEXT "OK|Cancel",
   AUTOCANCEL
```

The following shows a multiple choice requester, followed by a requester showing which option was chosen:

```
REQUEST_MESSAGE TEXT '"Choose an option..."',
   BUTTONTEXT "First|Second|Third"

REQUEST_MESSAGE TEXT '"You chose option '||RESULT||'"'
```

The following shows a message requester with multiple lines of text using the '\n' characters:

```
REQUEST_MESSAGE TEXT '"Top line\nMiddle line\nBottom line"'
```

Known bugs

None.

4.10.39 REQUEST_MULTIFILE

Command REQUEST_MULTIFILE

Parameters template

PATHPART, FILEPART, PATTERN, TITLE

Return template

FILES/M

Description

Opens a file requester, allowing the user to choose multiple filena mes.

The other ImageStudio windows are automatically blocked when the requester is opened and unblocked when the requester is closed.

In common with all ImageStudio requesters, if the user presses 'Cancel', an error message is returned. For the script to trap this error, global error checking must be turned off. See Section 4.6 [Error checking], page 41, for more information.

Parameters

PATHPART	The default path name to put in the requester.
FILEPART	The default filename to put in the requester.
PATTERN	An AmigaDos pattern matching pattern, will only show files in the requester which match the given pattern. By default, all files are shown.
TITLE	The text for the title bar of the requester.

Returns

FILES/M	The selected filenames from the requester, the filenames consists of both the FILEPART and PATHPART parts.

Errors rc = 0 if the operation was successful.

rc = 10 if the operation failed for any reason or the user cancelled requester, rc2 will contain a string describing the problem. rc will also be set to 10 if no files are chosen.

Example

The following puts up a multifile requester, with the results being put in the MULTIFILEINFO. stem:

```
REQUEST_MULTIFILE STEM MULTIFILEINFO.
```

The following puts up a multifile requester, with a default path of "Pics" and loops through all the selected files by putting them in message requesters:

```
REQUEST_MULTIFILE PATHPART "Pics" STEM MULTIFILENFO.

do l = 0 to (MULTIFILENFO.FILES.COUNT - 1)
   REQUEST_MESSAGE TEXT '"'MULTIFILENFO.FILES.l'"',
      BUTTONTEXT '"More...|Cancel"' AUTOCANCEL
   end
```

Known bugs

If no file is chosen, the command returns a "user cancelled" error. This is normal.

4.10.40 REQUEST_MULTIVALUE

Command REQUEST_MULTIVALUE

Parameters template

STRINGS/M, TITLE

Return template

STRINGS/M

Description

Opens a multivalue requester, allowing the user to change any of the listed values.

The other ImageStudio windows are automatically blocked when the requester is opened and unblocked when the requester is closed.

In common with all ImageStudio requesters, if the user presses 'Cancel', an error message is returned. For the script to trap this error, global error checking must be turned off. See Section 4.6 [Error checking], page 41, for more information.

Parameters

STRINGS/M

A list of strings with the values to display in the requester. The strings are grouped in pairs, the first string of the pair being the text for the requester, the second being the default value for that string.

For example, the following would place width and height values in the requester, with default values of 640 and 480 respectively:

```
STRINGS "Width" "640" "Height" "480"
```

If there isn't an even number of strings, an error is returned.

TITLE The text for the title bar of the requester.

Returns

STRINGS/M

A list of strings with the values to display in the requester. The strings are grouped in pairs, the first string of the pair being the text for the requester, the second being the returned value for that string.

Errors rc = 0 if the operation was successful.

rc = 10 if the operation failed for any reason or the user cancelled requester, rc2 will contain a string describing the problem. rc will also be set to 10 if no files are chosen.

Example

The following puts up a multivalue requester, allowing the user to change the width, height and depth of an image. The result is put in the MULTIVALUEINFO. stem:

```
REQUEST_MULTIVALUE "Width" "640" "Height" "480",
   STEM MULTIVALUEINFO.
```

The following puts up a multivalue requester, allowing the user to change the name and depth of an image. The result is put displayed in a message requester and the values recycled until the user can cels the requesters:

```
INSTRINGS = 'Name ImageName.bmp Depth 4'

do forever
   REQUEST_MULTIVALUE STRINGS INSTRINGS STEM MULTIVALUEINFO.

   REQUEST_MESSAGE TEXT BUTTONTEXT "OK|Cancel" AUTOCANCEL,
      TEXT '"Name: '||MULTIVALUEINFO.STRINGS.1||'\n'||,
      'Depth : '||MULTIVALUEINFO.STRINGS.3||'"'

   /* Construct the new INSTRINGS */

   INSTRINGS = ''

   do l = 0 to (MULTIVALUEINFO.STRINGS.COUNT - 1)
      INSTRINGS = INSTRINGS||' '||MULTIVALUEINFO.STRINGS.l
      end

   end
```

Known bugs

None.

4.10.41 REQUEST_SCREENMODE

Command REQUEST_SCREENMODE

Parameters template

MODEID/N, WIDTH/N, HEIGHT/N, OVERSCAN/N, DEPTH/N,
GADS_ENABLED/S, MAXDEPTH/N, MINWIDTH/N, MINHEIGHT/N

Return template

MODEID/N, WIDTH/N, HEIGHT/N, OVERSCAN/N, DEPTH/N, TEXT

Description

Opens a screenmode requester, allowing the user to select an Amiga screenmode. By specifying the GADS_ENABLED, the user may also select the width, height, depth and overscan values for the screenmode.

The other ImageStudio windows are automatically blocked when the requester is opened and unblocked when the requester is closed.

In common with all ImageStudio requesters, if the user presses 'Cancel', an error message is returned. For the script to trap this error, global error checking must be turned off. See Section 4.6 [Error checking], page 41, for more information.

Parameters

MODEID/N

The default screenmode ID for the requester which will be selected in the requester. By default, LORES is selected.

WIDTH/N

The default screen width value to put in the requester. By default, the default width of the selected screenmode is used.

HEIGHT/N

The default screen height value to put in the requester. By default, the default height of the selected screenmode is used.

OVERSCAN/N

The default overscan value to put in the requester. The following values are valid: 1 for text size, 2 for graphics size, 3 for extreme and 4 for maximum overscan. By default, text size is used.

DEPTH/N

The default depth value to put in the requester. By default, a depth of 2 is used.

GADSENABLED/S

By default the screenmode requester only allows the selecting of the screenmode. By using this switch, the user may also set the width, height, overscan and depth of the screenmode.

MAXDEPTH/N

The maximum depth allowed by the depth gadget. This value is only relevant to the internal screenmode requester.

MINWIDTH/N

The minimum allowable width of the screenmode.

MINHEIGHT/N

The minimum allowable height of the screenmode.

Returns

MODEID/N
The chosen screenmode ID.

WIDTH/N
The screen width chosen in the requester. This value is only set if the GADSENABLED switch is used.

HEIGHT/N
The screen height chosen in the requester. This value is only set if the GADSENABLED switch is used.

OVERSCAN/N
The overscan value chosen in the requester. The following values are returned: 1 for text size, 2 for graphics size, 3 for extreme and 4 for maximum overscan. This value is only set if the GADSENABLED switch is used.

DEPTH/N
The depth value chosen in the requester. This value is only set if the GADSENABLED switch is used.

TEXT A text description of the screenmode chosen.

Errors rc = 0 if the operation was successful.

rc = 10 if the operation failed for any reason or the user cancelled requester, rc2 will contain a string describing the problem.

Example

The following puts up a screenmode requester, the result is put in the SCREENMODEINFO. stem:

```
REQUEST_SCREENMODE STEM SCREENMODEINFO.
```

The following puts up a screenmode requester, complete with the width, height, depth and overscan gadgets. The chosen screenmode text is put in a message requester:

```
REQUEST_SCREENMODE GADSENABLED STEM SCREENMODEINFO.

REQUEST_MESSAGE TEXT '"'SCREENMODEINFO.TEXT'"'
```

Known bugs

1. There are two screenmode requesters that can be used by ImageStudio. If ImageStudio is running on a Workbench2.1+ Amiga the system ASL screenmode requester is used, if the Workbench2.04 is running then ImageStudio will use its internal screenmode requester (Workbench2.04 doesn't have an ASL screenmode requester). Because of the two different requesters and the complex way in which the Amiga deals with screenmodes, the operation of the two can be subtly dif ferent.

   The most noticeable difference is in the handling of the Amiga's built in screenmodes. A built in screenmode is something like "LoRes" or "HighRes Interlaced" as opposed to a disk based screen mode like "MULTISCAN:Productivity" or "SUPER72:High Res".

If you choose a built in screenmode in the ASL screenmode requester, the requester will return the screenmode ID something like "PAL:LoRes". This means that the requester specifies that the screenmode is "LoRes" and the monitor to be used is "PAL" (or "NTSC" in America). If you choose a built in screenmode in the internal screenmode requester, the requester will return the screenmode ID something like "LoRes", i.e. **it doesn't specify the monitor to be used**.

This is not usually a problem, but we feel that our screenmode re quester may be more compatible with older software which doesn't un derstand the system of specifying the monitor in the screenmode.

The matter is further complicated if you are using a monitor with mode promotion. Here, the internal screenmode requester's screen modes are promoted to the new double scanning modes (e.g. "HighRes" gets promoted to "DBLPAL:HighRes"). The ASL screenmodes *aren't* promoted, as they already contain the desired monitor information in the screenmode. This feature is either "desirable" or "undesirable" depending on your point of view. If you've ever wondered why some screenmodes don't promote, this is why - they have been told to be specifically "PAL" or "NTSC" in their screenmode.

If major problems are found in the differences between the internal and ASL screenmode requesters, we will endevour to alter the inter nal screenmode requester, but we think this is unlikely to cause any real problems.

2. To be safe in selecting a mode, you should always click on it in the requester. When you pass a default screenmode, the mode highlighted in the requester may not be exactly the same as the default screen mode given - it may be an equivalent. Clicking on the screenmode en sures that it returns that mode's real ID.

4.10.42 REQUEST_STRING

Command REQUEST_STRING

Parameters template

TEXT1, TEXT2, TEXT3, STRING, TITLE

Return template

STRING

Description

Opens a string requester, allowing the user to type in one line of text.

If a filename or directory name is required, it is suggested that either a special file or directory requester is used instead.

The other ImageStudio windows are automatically blocked when the requester is opened and unblocked when the requester is closed.

In common with all ImageStudio requesters, if the user presses 'Cancel', an error message is returned. For the script to trap this error, global error checking must be turned off. See Section 4.6 [Error checking], page 41, for more information.

Parameters

TEXT1	The top line of description text in the requester. The text will be left justified.
TEXT2	The middle line of description text in the requester. The text will be left justified.
TEXT3	The bottom line of description text in the requester. The text will be left justified.
STRING	The default string to be used in the requester.
TITLE	The text for the title bar of the requester.

Returns

STRING	The string in the requester.

Errors rc = 0 if the operation was successful.

rc = 10 if the operation failed for any reason or the user cancelled requester, rc2 will contain a string describing the problem.

Example

The following puts up a string requester, allowing the user to type in some text. The returned string is put in RESULT:

```
REQUEST_STRING TEXT2 '"Enter some text"'
```

The following puts up a string requester with a default string of "A raytraced image". The description text is displayed over 3 lines:

```
REQUEST_STRING TEXT1 '"Enter the"' TEXT2 '"desired name"',
   TEXT3 '"for the image"' STRING '"A raytraced image"'
```

Known bugs

None.

4.10.43 RGB_TO_HSV

Command RGB_TO_HSV

Parameters template

R/N/A, G/N/A, B/N/A

Return template

H/N, S/N, V/N

Description

Converts a RGB colour value into a HSV colour value.

See Section 5.4 [Colour representations], page 122, for more details on RGB and HSV col our representations.

Parameters

R/N/A	The red value of the colour to convert. Valid values are 0 to 255.
G/N/A	The green value of the colour to convert. Valid values are 0 to 255.
B/N/A	The red of the colour to convert. Valid values are 0 to 255.

Returns

H/N	The hue componant value of the colour.
S/N	The saturation componant value of the colour.
V/N	The value componant of the colour.

Errors rc = 0 if the operation was successful.

rc = 10 if the operation failed for any reason, rc2 will contain a string describing the problem.

Example The following converts yellow from RGB to HSV representation, putting the result in RESULT:

```
RGB_TO_HSV 255 255 0
```

The following converts mid-grey from RGB to HSV representation, putting the result in the stem COLOUR.:

```
RGB_TO_HSV 127 127 127 STEM COLOUR.
```

Known bugs

None.

4.10.44 SAVE

Command SAVE

Parameters template

FILE/A, FORMAT, ARGS, FORCE/S

Return template

None.

Description

Saves the current image out to disk.

See Section 3.1.2 [Save], page 13, for more details on saving images.

Parameters

FILE/A	The filename of the file to save.
FORMAT	The string containing the format of the file to save. By default, images are saved out as IFF-ILBM. See Section 5.1 [File formats], page 104, for more in formation on the available file formats.
ARGS	Any extra arguments that should be passed to the saver.
FORCE/S	By default the user will be warned if they are about to overwrite a file on the disk. Specifying this switch will stop such warnings.

Returns Nothing.

Errors rc = 0 if the operation was successful.

rc = 10 if the operation failed for any reason, rc2 will contain a string describing the problem.

Example The following saves the current image out as an IFF-ILBM:

```
SAVE FILE "ram:Image.ilbm"
```

The following saves the current image out as an IFF-ILBM, HAM6:

```
SAVE FILE "ram:Image.ham6" ARGS '"SUBFORMAT HAM6"'
```

The following saves out the current image as a JPEG, with a quality setting of 90:

```
SAVE FILE "ram:Image.jpg" FORMAT "JPEG" ARGS '"QUALITY 90"'
```

Known bugs

None.

4.10.45 SCALE

Command SCALE

Parameters template

X/N, Y/N, PERCENT/S, METHOD

Return template

None.

Description

Scales the current image either up or down using different methods optimised for speed or quality.

The user need not specify both X and Y scales, so scaling an image to only alter its width or height is possible.

See Section 3.4.2 [Scale], page 23, for more details on scaling images.

Parameters

X/N	Amount to scale the image in X (width) direction. By default this value is an absolute value in pixels, but by specifying the PERCENT option this value can be read as a percentage of the image's current width.
Y/N	Amount to scale the image in Y (height) direction. By default this value is an absolute value in pixels, but by specifying the PERCENT option this value can be read as a percentage of the image's current height.
PERCENT	Reads the X and Y values as percentages of the image's current width and height.
METHOD	A string describing the method of scaling to use. By default, FAST is used but AVERAGE can provide higher quality scales on 24bit images at the cost of computing time.

Returns Nothing.

Errors rc = 0 if the operation was successful.

rc = 10 if the operation failed for any reason, rc2 will contain a string describing the problem.

Example

The following scales the image to 800 by 600 pixels:

```
SCALE 800 600
```

The following doubles the size of the image:

```
SCALE 200 200 PERCENT
```

The following halves the height of the image, using colour average scaling:

```
SCALE Y 50 PERCENT METHOD "AVERAGE"
```

Known bugs

None.

4.10.46 SCREEN_BACK

Command SCREEN_BACK

Parameters template

None.

Return template

None.

Description

Moves the ImageStudio screen behind all other open screens.

Parameters

None.

Returns Nothing.

Errors rc = 0 if the operation was successful.

rc = 10 if the operation failed for any reason, rc2 will contain a string describing the problem.

Example

The following moves ImageStudio's screen behind all other open screens:

```
SCREEN_BACK
```

Known bugs

None.

4.10.47 SCREEN_FRONT

Command SCREEN_FRONT

Parameters template

None.

Return template

None.

Description

Moves the ImageStudio screen in front of all other open screens.

Parameters

None.

Returns Nothing.

Errors rc = 0 if the operation was successful.

rc = 10 if the operation failed for any reason, rc2 will contain a string describing the problem.

Example

The following moves ImageStudio's screen in front of all other open screens:

```
SCREEN_FRONT
```

Known bugs

None.

4.10.48 SELECT_ALL

Command SELECT_ALL

Parameters template

None.

Return template

None.

Description

Sets the current region to the whole of the image being displayed in the preview window.

Note that this doesn't always select the whole image; if the user has zoomed in on a region, only this region will be selected. If you want to be sure of selecting the whole image, issue a FULL_IMAGE command first.

Parameters

None.

Returns Nothing.

Errors rc = 0 if the operation was successful.

rc = 10 if the operation failed for any reason, rc2 will contain a string describing the problem.

Example

The following selects the whole of the currently viewed image:

```
SELECT_ALL
```

Known bugs

None.

4.10.49 UNDO

Command UNDO

Parameters template

None.

Return template

None.

Description

Un-does the last operation (see Section 4.10.30 [ARexx'REDO], page 79).

Parameters

None.

Returns Nothing.

Errors rc = 0 if the operation was successful.

rc = 10 if the operation failed for any reason, rc2 will contain a string describing the problem.

Example

The following un-does the last operation:

```
UNDO
```

Known bugs

None.

4.10.50 VIEW

Command VIEW

Parameters template

EXTERNAL/S

Return template

None.

Description

Views the current image using either the internal or external viewer.

Parameters

EXTERNAL/S

By default the image is shown using the internal viewer, however by including this parameter the external viewer as defined in the preferences (see Section 3.1.4 [Prefs], page 14) will be used.

Returns Nothing.

Errors rc = 0 if the operation was successful.

rc = 10 if the operation failed for any reason, rc2 will contain a string describing the problem.

Example The following views the current image using the internal viewer:

```
VIEW
```

The following views the current image using the external viewer:

```
VIEW EXTERNAL
```

Known bugs

None.

4.10.51 ZOOM_IN

Command ZOOM_IN

Parameters template

None.

Return template

None.

Description

Zooms in to the currently selected region in the preview window.

A region has to be selected in order for this command to work.

Parameters

None.

Returns Nothing.

Errors rc = 0 if the operation was successful.

rc = 10 if the operation failed for any reason, rc2 will contain a string describing the problem.

Example The following zooms in to the currently selected region:

```
ZOOM_IN
```

Known bugs

None.

4.10.52 ZOOM_OUT

Command ZOOM_OUT

Parameters template

None.

Return template

None.

Description

Zooms out by a factor of 3 times in the preview window.

Parameters

None.

Returns Nothing.

Errors rc = 0 if the operation was successful.

rc = 10 if the operation failed for any reason, rc2 will contain a string describing the problem.

Example

The following zooms out by a factor of 3 times in the preview window:

```
ZOOM_OUT
```

Known bugs

None.

5 Reference

This chapter gives detailed explanations about various aspects of the program.

5.1 File formats

5.1.1 IFF-ILBM

Name IFF-ILBM

Load types

Colour mapped, 24 bit, HAM6, HAM8, Extra halfbright.

Compressed and uncompressed.

Save types

Colour mapped, 24 bit, HAM6, HAM8, Extra halfbright.

Compressed and uncompressed.

Description

IFF-ILBM is the Amiga's native bitmap graphic file format.

IFF-ILBM files are usually compressed using simple run-length compression, but they can be uncompressed for simplicity and speed.

IFF-ILBM is ImageStudio's default save format.

ImageStudio will load and save AGA images on a non-AGA machine.

The original image's screenmode will be preserved, unless changed by the user (see Section 3.3.5 [View'screenmode], page 22).

ARexx OPEN command ARGS

None.

ARexx SAVE command ARGS

SUBFORMAT, DITHER, NOCOMPRESSION/S

SUBFORMAT

By default the image is saved out as the buffer, i.e. either a 2 to 256 colour colour-mapped image or as 24bit. By specifying the SUB FORMAT parameter, the user can select HAM6, HAM8 or EHB as extra save formats. The image will be converted into the chosen subformat before saving. The image in ImageStudio remains unchanged.

DITHER

If the SUBFORMAT option is used, DITHER can be set to FS to give Floyd-Steinberg dithering of the HAM6, HAM8 or EHB image.

NOCOMPRESSION/S

By default the IFF-ILBM image is compressed using simple run-length compression. Use this switch to save uncompressed data.

Example

The following saves out the current image in the same format as the buffer:

```
SAVE FILE "Image.ilbm"
```

The following saves out the image as HAM6:

```
SAVE FILE "Image.ilbm" ARGS '"SUBFORMAT HAM6"'
```

The following saves out the image as Floyd-Steinberg dithered Extra halfbright:

```
SAVE FILE "Image.ilbm" ARGS '"SUBFORMAT EHB DITHER FS"'
```

5.1.2 BMP

Name BMP

Load types (“Windows” and “OS/2” format).
Colour mapped, 24 bit.

Save types
Colour mapped, 24 bit (“Windows” format).

Description
BMP files are commonly found on PCs running Microsoft Windows.

BMP images are usually uncompressed and come in 2 flavours - ‘Windows’ and ‘OS/2’.

ARexx OPEN command ARGS
None.

ARexx SAVE command ARGS
None.

Example None.

5.1.3 Datatype

Name Datatype

Load types
Colour mapped.

Save types
None.

Description
The datatype IO module will use any datatypes installed on the system to try and load the image. The datatype loader will only work with Workbench3.0 and above.

Datatypes are a poor way of loading images into ImageStudio, for the following reasons:

1. Datatypes are slow. First the datatype has to convert the image into its internal format, then ImageStudio has to convert this to ImageStudio’s internal format.
2. Datatypes cannot handle more than 256 colours. Any 24bit image for mats are converted by the datatype to 256 colours (or less) which will cause a loss in quality.
3. Datatypes cannot use the progress bar whilst converting the incomming image. It is therefore not possible to stop a datatype loading an image during this first phase.

Because of these problems, ImageStudio will only try the datatype loader as a “last resort” - trying its own loaders before attempt ing to the use the datatype module.

During testing we have found a few datatypes that cause problems with odd subformats of certain formats. Obviously, ImageStudio has no control over the quality of the datatypes installed on the system.

ARexx OPEN command ARGS

None.

ARexx SAVE command ARGS

None.

Known bugs

The datatype loader currently doesn't handle datatypes which return HAM (HAM6 or HAM8) images.

Example None.

5.1.4 EPS

Name EPS

Load types

None.

Save types

Greyscale, 24 bit.

Description

EPS files are ASCII text files written in the PostScript language. They can be printed out directly to a PostScript printer or imported into word processing or DTP packages.

EPS files are an ineffecient method of storing files, as they are uncompressed and are stored as ASCII text as opposed to binary data. Unless colour is specifically required it is recommended that EPS files be saved in the greyscale format, as they are one third of the size of a colour EPS file.

ARexx OPEN command ARGS

None.

ARexx SAVE command ARGS

DPI/N, COLOUR

DPI/N	The resolution of the image in dots per inch. By default, 300 dpi is used.
COLOUR	By default the output image is 256 greyscale. Specifying this option outputs the image as 24bit colour.

Example

The following saves out a COLOUR EPS file:

```
SAVE FILE "Image.eps" FORMAT "EPS" ARGS "COLOUR"
```

The following saves out a 600 by 300 file at 150 dpi, given a print able size of 4inches by 2inch:

```
SAVE FILE "Image.eps" FORMAT "EPS" ARGS '"DPI 150"'
```

5.1.5 GIF

Name GIF

Load types

Colour mapped (“GIF87a” and “GIF89a” formats).

Save types

Colour mapped (“GIF87a” and “GIF89a” formats).

Description

GIF is a common format for images upto 256 colours.

GIF is a trademark of Compuserve Incorporated.

GIF images are normally smaller than their equivalent IFF-ILBM counterparts due to GIF’s LZW compression algorithm. GIF files are always compressed.

GIF comes in 2 flavours - ‘GIF87a’ and ‘GIF89a’. GIF87a is the most popular format; ImageStudio will load in both GIF87a and GIF89a.

ARexx OPEN command ARGS

None.

ARexx SAVE command ARGS

SUBFORMAT,INTERLACED/S,TRANSPARENCY/S,TRANSPCOLOUR/N

SUBFORMAT

By default GIF files are saved as the popular GIF87a format. By selecting a subformat of GIF89A, the user may include extra information in the file (e.g. transparent colour information). Specifying a subformat of GIF89a alone provides no extra functionality.

INTERLACED/S

By default the information in a GIF file is saved in a sequential fashion (e.g. line 1,line 2,line 3...). By specifying this switch the image in saved in a series of interlaced passes which can give faster redraws on some viewers (e.g. World Wide Web browsers). Both GIF87A and GIF89A support interlacing.

Note: This interlace has nothing to do with the Amiga’s interlaced screens.

TRANSPARENCY/S

By specifying this switch, one colour out of the image’s pallete can be made ‘transparent’, causing the background colour to show through. This again can be useful with World Wide Web browsers. By default, the transparent colour is colour 0, but this can be changed (see below). This switch is only used in the GIF89A subformat.

TRANSPCOLOUR/N

This is the palette entry number that is made transparent if the TRANSPARENT switch is used. This value is only used in the GIF89A subformat.

Example The following saves out a standard GIF87a format file:

```
SAVE FILE "Image.gif" FORMAT "GIF"
```

The following saves out a GIF87a file with interlace:

```
SAVE FILE "Image.gif" FORMAT "GIF" ARGS "INTERLACED"
```

The following saves out a GIF89a file with a transparent colour of 10:

```
SAVE FILE "Image.gif" FORMAT "GIF",
   ARGS "SUBFORMAT GIF89A TRANSPARENCY TRANSPCOLOUR 10"
```

5.1.6 IFF-DEEP

Name IFF-DEEP

Load types

24 bit uncompressed.

Save types

24 bit uncompressed.

Description

The IFF-DEEP format a fast way of storing 24bit data in a IFF format file.

IFF-DEEP files can be compressed, but ImageStudio currently only supports the loading and saving of the common uncompressed format. Saving a colour mapped image as IFF-DEEP will cause the image to be promoted to 24bits and the colour map information to be lost.

ARexx OPEN command ARGS

None.

ARexx SAVE command ARGS

None.

Example

The following saves out an IFF-DEEP file:

```
SAVE FILE "Image.deep" FORMAT "IFF-DEEP"
```

5.1.7 JPEG

Name JPEG

Load types

Greyscale, 24 bit ("JFIF" format).

Save types

Greyscale, 24 bit ("JFIF" format).

Description

JPEG allows the storage of 24-bit images as very small files due to its lossy compression algorithm. Whereas the compression algorithms used by other file formats loose none of the image information, JPEG trades off a little loss in image quality for a high degree of compression.

As JPEG is a relatively new format, an exact format of the JPEG file was only agreed on recently. This format is called 'JFIF' and these are the most commonly used JPEG format files - and the format that ImageStudio loads and saves. It is highly unlikely that any old JPEG files are still being circulated, but should you find one it is uncertain whether ImageStudio would accept it.

A high degree of compatibility is obtained with our JPEG loader / saver routines, as they are based in part on the work of the Independant JPEG group's routines.

ARexx OPEN command ARGS

None.

ARexx SAVE command ARGS

QUALITY/N, GREYSCALE/S

QUALITY/N

The quality of the jpeg output file. A quality value of 75 is given by default, resulting in an acceptable degredation of image quality. For higher degree of compression choose a lower value[2]. For a higher degree of quality, choose a higher value; values of 85 to 90 result in an almost unnoticable loss of quality.

GREYSCALE/S

By default the JPEG output is 24bit colour. By using this option, the image can also be saved in a greyscale format, where the colour information is lost but the output file size is correspondingly smaller.

Example

The following saves out the current image as JPEG:

```
SAVE FILE "Image.jpg" FORMAT "JPEG"
```

The following saves out the current image as JPEG, with a high degree of compression:

```
SAVE FILE "Image.jpg" FORMAT "JPEG" ARGS '"QUALITY 50"'
```

The following saves out the current image as greyscale JPEG:

```
SAVE FILE "Image.jpg" FORMAT "JPEG" ARGS '"GREYSCALE"'
```

---

[2] Values less than 25 may cause problems with some JPEG readers

5.1.8 PCX

Name PCX

Load types

Colour mapped (2 to 16, 256 colours), 24 bit.

Save types

Colour mapped (256 colours), 24 bit.

Description

PCX files are commonly found on PCs running Microsoft Windows.

PCX files are always compressed using a very inefficient run-length encoding algorithm. This algorithm can, in some cases, lead to an increase in file size over an uncompressed image. PCX is included in ImageStudio for compatibility with other platforms, but we do not recommend the general storing of images in this format.

ImageStudio only saves 256 colour PCX files as these are the most compatible accross programs. The specification is a little vague as to how to handle 2 to 16 colour PCX files.

ARexx OPEN command ARGS

None.

ARexx SAVE command ARGS

None.

Example None.

5.1.9 PNG

Name PNG

Load types

Colour mapped, 24, 32 and 48bits.

Save types

Colour mapped, 24bit.

Description

PNG is the license free replacement for the GIF file format.

After the controversy with the LZW compression algorithm used in the GIF file format, it was decided that a new standard should be created which uses the license free LZ compression algorithm along with many improvments over the GIF file format (e.g. support for 24bit images). PNG is that format, and supports many advanced features to obtain the best lossless compression for any type of bitmap image. PNG also supports features that will find themselves useful in World Wide Web browsers, etc...

A high degree of compatibility is obtained with our PNG loader/saver, as it is based in part on the PNGlib code by Group42.

ARexx OPEN command ARGS

None.

ARexx SAVE command ARGS

INTERLACE,TRANSPARENCY/S,TRANSPCOLOUR/N

INTERLACE

By default PNG files are not interlaced, however by specifying an INTERLACE value of ADAM7 they can be saved out with 7 levels of in terlacing. This gives progressive redraw for World Wide Web browsers.

Note: This interlace has nothing to do with the Amiga's interlaced screens.

TRANSPARENCY/S

By specifying this switch, one colour out of the image's pallete can be made 'transparent', causing the background colour to show through. This again can be useful with World Wide Web browsers. By default, the transparent colour is colour 0, but this can be changed (see below).

TRANSPCOLOUR/N

This is the palette entry number that is made transparent if the TRANSPARENT switch is used.

Example The following saves out a standard PNG format file:

```
SAVE FILE "Image.png" FORMAT "PNG"
```

The following saves out a PNG file with interlace:

```
SAVE FILE "Image.png" FORMAT "PNG" ARGS "INTERLACE=ADAM7"
```

The following saves out a PNG file with a transparent colour of 10:

```
SAVE FILE "Image.png" FORMAT "PNG",
   ARGS "TRANSPARENCY TRANSPCOLOUR 10"
```

5.1.10 PNM

Name PNM

Load types

Black and white, greyscale, 24bit colour. ASCII and binary.

Save types

Black and white, greyscale, 24bit colour. ASCII and binary.

Description

PNM is an amalgamation of the PBM, PGM and PPM file formats, commonly found on UNIX machines.

The PNM (Portable aNyMap) family of formats are simple way to store images as either binary or ASCII format. The formats are uncompressed, so tend to produce very large files. This module should handle all PNM formats.

When loading a PNM file, the file **must** have either a '.pnm', '.pbm', '.pgm' or '.ppm' filename extension.

When saving out a black and white PBM file, the current image is converted simply into black and white without dithering. If you require a dithered black and white output you have to reduce the number of colours first within ImageStudio.

ARexx OPEN command ARGS

None.

ARexx SAVE command ARGS

SUBFORMAT,ASCII/S

SUBFORMAT

SUBFORMAT can be set to either PBM, PGM, PPM depending on whether the output should be black and white, greyscale or 24bit colour.

ASCII/S By default PNM files are saved in their more compact binary format. By specifying this switch, the output can be pure ASCII text characters. This format produces files whice are substantially larger than the binary format, but which can be directly transferred via text only media (e.g. e-mail).

Example The following saves out a 24bit PNM format file:

```
SAVE FILE "Image.ppm" FORMAT "PNM" ARGS "SUBFORMAT=PPM"
```

The following saves a black and white ASCII PBM file:

```
SAVE FILE "Image.pbm" FORMAT "PNM" ARGS "SUBFORMAT=PBM ASCII"
```

5.1.11 QRT

Name QRT

Load types

24bit colour.

Save types

24bit colour.

Description

QRT is the output file format for the QuickRayTracer program, as well as the DKB and POV raytracers.

QRT is a simple uncompressed 24bit colour file format.

When loading a QRT file, the file **must** have either a '.qrt', '.dkb' or '.pov' filename extension.

ARexx OPEN command ARGS

None.

ARexx SAVE command ARGS

None.

Example The following saves out a 24bit QTR format file:

```
SAVE FILE "Image.qrt" FORMAT "QRT"
```

5.1.12 SGI

Name SGI

Load types

Greyscale and 24bit colour. Compressed and uncompressed.

Save types

Greyscale and 24bit colour. Compressed and uncompressed.

Description

SGI-RGB is the file format used by Silicon Graphics workstations.

SGI-RGB files can be compressed using run length encoding. As the data is 24bits, this usually has little effect.

ARexx OPEN command ARGS

None.

ARexx SAVE command ARGS

GREYSCALE/S,COMPRESSED/S

GREYSCALE/S

By default the image is saved in full 24bit colour. By specifying this switch, the output image can be saved as 256 greyscales.

COMPRESSED/S

By specifying this switch the output file may be compressed using run length encoding.

Example The following saves out a 24bit SGI format file:

```
SAVE FILE "Image.sgi" FORMAT "SGI"
```

The following saves a compressed greyscale image:

```
SAVE FILE "Image.sgi" FORMAT "SGI" ARGS "GREYSCALE COMPRESSED"
```

5.1.13 Targa

Name TARGA

Load types

Colour mapped (2 and 256 colours), 15, 16, 24 and 32bits.

Compressed and uncompressed.

Save types

Colour mapped (256 colours), 24 bit.

Uncompressed.

Description

Targa is usually used for storing 24-bit images, although it can also handle colour-mapped images as well. The data is usually stored as simple uncompressed data, however it can also be run-length encoded to allow compression.

ARexx OPEN command ARGS

None.

ARexx SAVE command ARGS

None.

Example None.

5.1.14 TIFF

Name TIFF

Load types

Baseline TIFF v6 without CCITT Huffman compression.

Save types

Baseline TIFF v6.

Description

TIFF is a very powerful and flexible file format.

TIFF is probably the most complex and powerful image file format in common use. TIFF allows the TIFF saver many options in order to optimize the output file for any given use. This is also the problem with TIFF; in order for a program to load TIFF images it must have the ablility to understand all the aspects of the file format. This leads to great incompatibilties as programs create TIFF files which cannot be read in by other programs claiming TIFF loading ability.

In order to overcome this problem, the TIFF Advisory Committee pro duced a 'Baseline' TIFF specification which it was intended that all TIFF loaders/savers could handle this minimum level of file format.

The TIFF module provided with ImageStudio is based on this baseline specification with some differences. The main difference is lack of compression type 2 (CCITT Huffman) in either the loader or saver. The loader should handle most other TIFF files and has been tested with Macintosh and UNIX image processing programs. Images produced by the module differ from the recommended strip size as dictated by the TIFF specifications. The TIFF specs recommend that the image is broken down into strips and saved out as multiple strips; ImageStudio saves the image out as one large strip to try and maintain compatibility with programs that cannot handle multiple strip images.

Due to TIFF's complexity, it is very likely that ImageStudio will encounter invalid TIFF files that will not load into the program. Similarly, TIFF files produced by ImageStudio may not load into other programs (e.g. ones which rely on the strip size being small). This is a fact of TIFF life, we're afraid. The ImageStudio TIFF module was written to compatible, not clever.

ARexx OPEN command ARGS

None.

ARexx SAVE command ARGS

COMPRESSION,DPI/N

COMPRESSION
Currently no compression is supported, so COMPRESSION=NONE.

DPI/N The user may specify the resolution of the output image in dots-per-inch. Images are saved at a default DPI of 300.

Example The following saves out a TIFF format file:

```
SAVE FILE "Image.tiff" FORMAT "TIFF"
```

The following saves a TIFF image at 72 DPI:

```
SAVE FILE "Image.tiff" FORMAT "TIFF" ARGS "DPI=72"
```

5.1.15 VMEM

Name VMEM

Load types

Colour mapped and 24bit.

Save types

Colour mapped and 24bit.

Description

VMEM is the data from ImageStudio's virtual memory swap files, plus header information.

VMEM files can be loaded and saved by ImageStudio, but their main use is with the 'UnCrash' CLI utility given with ImageStudio for resurrecting the virtual memory swap files after a crash.

The file format is given below, as this is probably the easiest way of creating and reading images for ImageStudio for use in other programs.

File Format

The file format for the VMEM files is given below should you wish to use the format in your own programs. The file consists of the of an information header (descriped below) followed by the image data. The image header is always a fixed size and the image data is uncom pressed.

Image header:

'C' notation is used, all numbers are stored in Motorola byte order,

```
     struct {
          UBYTE magic1,magic2;       // Values 0xDE and 0xAD
          UBYTE magic[4];            // Values 'V','M','E','M'

          WORD width,height;         // Image dimensions

          BYTE bytes_per_pixel;      // 1 for colour map, 3 for RGB

          UBYTE cmap[256][3];        // The colour map (empty for RGB)
          };
```

The structure is not byte padded, i.e. it's 779 bytes in size.

Image data:

The image data is either the colour map values if the image was col our mapped, or RGB data ordered R,G,B. The length of the image data should be the width times height times bytes_per_pixel.

ARexx OPEN command ARGS

None.

ARexx SAVE command ARGS

None.

Example None.

5.2 Effects

5.2.1 Dynamic range

Works with

- Full colour-mapped images.
- Full and regions of 24bit images.

Description

Expands the dynamic range of the image to the maximum possible, without altering the colour balance. This is useful for automatically increasing the contrast of poor contrast images, e.g. the output from a scanner.

ARexx EFFECT command ARGS

None.

5.2.2 Flip X

Works with

- Full and regions of colour-mapped images.
- Full and regions of 24bit images.

Description

Flips the whole image or selected region horizontally.

ARexx EFFECT command ARGS

None.

5.2.3 Flip Y

Works with

- Full and regions of colour-mapped images.
- Full and regions of 24bit images.

Description

Flips the whole image or selected region vertically.

ARexx EFFECT command ARGS

None.

5.2.4 Greyscale

Works with

- Full colour-mapped images.
- Full and regions of 24bit images.

Description

Reduces a colour image to a greyscale image. The actual greyscale values (or more correctly, luminosity) value is calculated as 30% of the red component + 59% of the green component + 11% of the blue component.

ARexx EFFECT command ARGS

None.

5.2.5 Highlight

Works with

- Full colour-mapped images.
- Full and regions of 24bit images.

Description

Turns all colours with greater than the given luminance value to white.

ARexx EFFECT command ARGS

LUMINANCE/N/A

LUMINANCE/N/A

The luminance value above which pixels should be turned white.

5.2.6 Negative

Works with

- Full colour-mapped images.
- Full and regions of 24bit images.

Description

Negates the colour values of the image.

ARexx EFFECT command ARGS

None.

5.2.7 Random

Works with

- Full and regions of 24bit images.

Description

Adds random noise to the image. The greater the random value, the greater the noise.

ARexx EFFECT command ARGS

RANDOMNESS/N/A

RANDOMNESS/N/A

The amount of randomness to apply to the image. Values range between 1 and 255.

5.2.8 Remove isolated pixels

Works with

- Full and regions of colour-mapped images.
- Full and regions of 24bit images.

Description

Removes any single pixels that are a different colour to their neighbours. Useful in removing some of the noise in black and white scans.

ARexx EFFECT command ARGS

None.

5.2.9 Roll X

Works with

- Full and regions of colour-mapped images.
- Full and regions of 24bit images.

Description

Rolls the whole image or selected region horizontally, wrapping the image around.

ARexx EFFECT command ARGS

DISTANCE/N/A

DISTANCE/N/A

The distance, in pixels, to move the image. A positive value moves the image to the right, a negative value moves the image to the left.

5.2.10 Roll Y

Works with

- Full and regions of colour-mapped images.
- Full and regions of 24bit images.

Description

Rolls the whole image or selected region vertically, wrapping the image around.

ARexx EFFECT command ARGS

DISTANCE/N/A

DISTANCE/N/A

The distance, in pixels, to move the image. A positive value moves the image down, a negative value moves the image up.

5.2.11 Pixelise

Works with

- Full and regions of colour-mapped images.
- Full and regions of 24bit images.

Description

Replaces all pixels in the whole image or selected region with larger pixels.

ARexx EFFECT command ARGS

PIXELSIZE/N/A

PIXELSIZE/N/A

The size, in pixels, of the larger pixel blocks.

5.2.12 Shadow

Works with

- Full colour-mapped images.
- Full and regions of 24bit images.

Description

Turns all colours with less than the given luminance value to black.

ARexx EFFECT command ARGS

LUMINANCE/N/A

LUMINANCE/N/A

The luminance value below which pixels should be turned black.

5.3 Image types

ImageStudio works internally with either "colour-mapped" or "24-bit" images. A description of the workings of both methods follows.

Colour-mapped images

Colour-mapped (palette based) images are used by the standard (non-HAM) screenmodes on the Amiga. A set of colours (palette) is chosen for the image and every pixel in the image can have one of these colours.

Colour-mapped images have the advantage of being a fairly compact way of storing image information and with a large palette (greater than 64 colours) high quality images can be produced. They have the disadvantage that the colours in the image are limited to the col ours in the palette - with a small palette it becomes a complex task choosing the correct colours to best portray the image.

24-bit images

24-bit images allow every pixel in the image to be an individual colour - this is essential for high quality images.

24-bit images have the disadvantage that they are typically at least 3-times larger than colour-mapped images and require sophisticated display hardware to show them in their true glory.

When should I use each type of image?

In general, try to leave the image in the format in which it came. If, for example, you load in a colour-mapped image try and perform all your operations directly to the colour-mapped image; only change to a 24-bit image if absolutely necessary (e.g. to apply a convolution filter).

5.4 Colour representations

ImageStudio works internally with the R,G,B format of colour representation. This is the most common method of storing colour in formation on computers, as it represents the amounts of intensities applied to the 3 colour guns of a computer monitor.

H,S,V stands for Hue, Saturation and Value. The hue is the basic colour (e.g. red, yellow, green, purple etc...), saturation is the amount of that colour (e.g. weak red, strong red etc...) and the value is the brightness of the colour.

5.5 Tooltypes

ImageStudio supports the configuring of the program via tooltypes from either the Workbench or CLI.

It is recommended that the user who starts the program from Work bench uses the "Prefs" requester of ImageStudio to configure it (see Section 3.1.4 [Prefs], page 14), whereas the CLI user should be aware of the tooltype options.

Boolean tooltypes can have values 'ON' or 'YES' for a positive value, 'OFF' or 'NO' for a negative values. Numeric tooltypes are positive and negative integers; floating point values are not allowed.

5.5.1 BALANCE

Name BALANCE

Type Boolean

Description

Determines whether the balance floating palette should be open at startup.

5.5.2 BALANCELEFT

Name BALANCELEFT

Type Numeric

Description

The top position to open the balance floating palette.

5.5.3 BALANCETOP

Name BALANCETOP

Type Numeric

Description

The left position to open the balance floating palette.

5.5.4 CONVOLVE

Name CONVOLVE

Type Boolean

Description

Determines whether the convolve floating palette should be open at startup.

5.5.5 CONVOLVELEFT

Name CONVOLVELEFT

Type Numeric

Description

The top position to open the convolve floating palette.

5.5.6 CONVOLVETOP

Name CONVOLVETOP

Type Numeric

Description

The left position to open the convolve floating palette.

5.5.7 EFFECT

Name EFFECT

Type Boolean

Description

Determines whether the effect floating palette should be open at startup.

5.5.8 EFFECTLEFT

Name EFFECTLEFT

Type Numeric

Description

The top position to open the effect floating palette.

5.5.9 EFFECTTOP

Name **Type** Numeric

Description
The left position to open the effect floating palette.

5.5.10 SCRIPTS

Name SCRIPTS
Type Boolean
Description
Determines whether the scripts floating palette should be open at startup.

5.5.11 SCRIPTSLEFT

Name SCRIPTSLEFT
Type Numeric
Description
The top position to open the scripts floating palette.

5.5.12 SCRIPTSTOP

Name SCRIPTSTOP
Type Numeric
Description
The left position to open the scripts floating palette.

5.5.13 PREVIEWLEFT

Name PREVIEWLEFT
Type Numeric
Description
The top position to open the preview window.

5.5.14 PREVIEWTOP

Name PREVIEWTOP

Type Numeric

Description

The left position to open the preview window.

5.5.15 PREVIEWWIDTH

Name PREVIEWWIDTH

Type Numeric

Description

The width to open the preview window.

5.5.16 PREVIEWHEIGHT

Name PREVIEWHEIGHT

Type Numeric

Description

The height to open the preview window.

5.5.17 FONTNAME

Name FONTNAME

Type String

Description

The name of the font to use to layout the ImageStudio windows and menus.

When specifying the font name, the font name must be have a '.font' appended to its name. For example, to set the default to be helvetica, the tooltype should be specified as:

```
FONTNAME=helvetica.font
```

The fontname is not case sensitive.

5.5.18 FONTSIZE

Name FONTSIZE

Type Numeric

Description

The size of the font to use to layout the ImageStudio windows and menus.

If a large font is specified which would cause some of the windows to become too large to fit on the ImageStudio screen, the font will default to topaz 8.

5.5.19 SCREENMODEID

Name SCREENMODEID

Type Numeric

Description

The screenmode to open the ImageStudio screen. This value is the value returned by the REQUEST_SCREENMODE ARexx command in the MODEID value (see Section 4.10.41 [ARexx`REQUEST`SCREENMODE], page 91).

5.5.20 SCREENOVERSCAN

Name SCREENOVERSCAN

Type Numeric

Description

The overscan to use with the ImageStudio screen. Valid values are 1 for text overscan, 2 for graphics size, 3 for extreme and 4 for maximum.

Note, this tooltype has changed from v1.x.x, where it was a string type.

5.5.21 SCREENDEPTH

Name SCREENDEPTH

Type Numeric

Description

The depth of the ImageStudio screen. Valid values are 1 to 8.

5.5.22 SCREENWIDTH

Name SCREENWIDTH

Type Numeric

Description

The width of the ImageStudio screen. Valid values are greater than 640.

5.5.23 SCREENHEIGHT

Name SCREENHEIGHT

Type Numeric

Description

The height of the ImageStudio screen. Valid values are greater than 200.

5.5.24 BLANKSCRIPT

Name BLANKSCRIPT

Type String

Description

The filename of the Blank ARexx script to use a template when a new script is created in the 'Scripts' floating palette.

5.5.25 CONVOLVEDIR

Name CONVOLVEDIR

Type String

Description

The directory to scan on startup for convolution filters to include in the 'Convolves' floating palette.

5.5.26 EXTERNALVIEWER

Name EXTERNALVIEWER

Type String

Description

The external program to run to view the image. The string is in the format that would be typed from the command line, with a '%s' where the image filename should be placed.

It is recommended that the string be prefixed with a "run" to allow the external viewer to run in the background, otherwise ImageStudio has to wait for the program to finish. For example:

```
sys:Utilities/VT <NIL: >NIL: %s
```

would leave ImageStudio waiting until ViewTek had finished, whereas

```
run sys:Utilities/VT <NIL: >NIL: %s
```

would run ViewTek in the background. The "<NIL:" and ">NIL:" are used to stop error messages being printed to the console.

5.5.27 HELPFILE

Name HELPFILE

Type String

Description

The full filename of the AmigaGuide help file to provide online help.

Note, the **full** filename must be provided, a filename taken from the current directory will not work. For example:

```
Docs/ImageStudio.guide
```

will not work, whereas

```
Work:Graphics/ImageStudio/Docs/ImageStudio.guide
```

will work fine.

If you do wish to specify the current directory, you may use the 'PROGDIR:' volume assignment. PROGDIR: is set to be the current di rectory of the current program, each program having its own PROGDIR: value that can only be used within the program. Therefore, the above AmigaGuide helpfile can be refered to as:

```
PROGDIR:Docs/ImageStudio.guide
```

See Section 5.5.45 [Tooltype`HELP], page 134, for more information on AmigaGuide help.

5.5.28 IMAGEDIR

Name IMAGEDIR

Type String

Description

The default directory to use for images when ImageStudio is loaded.

5.5.29 INTERNALVIEWER

Name INTERNALVIEWER

Type String

Description

The filename of module used for the internal viewer. See See Section 3.3.4 [Internal`viewer], page 20, for more information on the different capablities of the different viewer modules.

5.5.30 KEYFILE

Name KEYFILE

Type String

Description

The filename of the keyfile to use to unlock ImageStudio to use full sized images. Keyfiles are obtaining by registering (see Chapter 8 [How to register], page 144).

5.5.31 MODULEDIR

Name MODULEDIR

Type String

Description

The default directory to use for external modules.

Inside the module directory are other directories containing modules specific to file loading/saving, viewers, effects etc... These directories are scanned when ImageStudio starts up.

5.5.32 PALETTEDIR

Name PALETTEDIR

Type String

Description

The default directory to use for palettes when ImageStudio is loaded.

See Section 3.4.4 [Palette], page 24, for more details on loading palettes into ImageStudio.

5.5.33 PORTNAME

Name PORTNAME

Type String

Description

The name of the ARexx portname used by the program.

If a portname by that name already exists, the name is incremented until a free portname is found. For example, if ‘IMAGESTUDIO’ was already in use, the following sequence of names would be tried: ‘IMAGESTUDIO.1’, ‘IMAGESTUDIO.2’, ‘IMAGESTUDIO.3’ . . .

When choosing an ARexx portname, try to keep it fairly short.

5.5.34 REXXOUTPUT

Name REXXOUTPUT

Type String

Description

The name of the filename to use for ARexx scripts’ output. See Section 4.3 [Basic ARexx], page 34, for more information on this file.

5.5.35 SCREENNAME

Name SCREENNAME

Type String

Description

The name of ImageStudio's public screen. This name must be unique, otherwise the screen won't open.

5.5.36 SCRIPTSDIR

Name SCRIPTSDIR

Type String

Description

The default directory that will be scanned for ARexx script files to put in the 'Scripts' floating palette list. Any file with the chosen ARexx script extension (see Section 5.5.37 [Tooltype˙SCRIPTSEXT], page 132) will be placed in the list.

5.5.37 SCRIPTSEXT

Name SCRIPTSEXT

Type String

Description

The default filename extension for the ImageStudio ARexx scripts. Only scripts with this extension will be added to the list in the 'Scripts' floating palette.

5.5.38 TEMPDIR

Name TEMPDIR

Type String

Description

The directory in which ImageStudio can put its virtual memory tempo rary swap files. This should be some location on your hard disk, as the data will be accessed a lot during some operations. If you have lots of RAM, the temporary directory can be placed in the ram disk for maximum speed.

5.5.39 TEXTEDITOR

Name TEXTEDITOR

Type String

Description

The text editor used to edit ARexx scripts from the 'Scripts' float ing palette. The string is in the format that would be typed from the command line, with a '%s' where the image filename should be placed.

It is recommended that the string be prefixed with a "run" to allow the text editor to run in the background, otherwise ImageStudio has to wait for the program to finish. For example:

```
sys:Tools/Memacs <NIL: >NIL: %s
```

would leave ImageStudio waiting until Microemacs had finished, whereas

```
run sys:Tools/Memacs <NIL: >NIL: %s
```

would run Microemacs in the background. The "<NIL:" and ">NIL:" are used to stop error messages being printed to the console.

5.5.40 CLIPUNIT

Name CLIPUNIT

Type Numeric

Description

The system clipboard used to copy (see Section 3.2.3 [Copy], page 18) and paste (see Section 3.2.4 [Paste], page 19) by the program. There is very little need to change this from the default value of 0.

5.5.41 PAGESIZE

Name PAGESIZE

Type Numeric

Description

The size, in K, of the virtual memory pages. The larger this number, the more real RAM is used but the less frequent the accesses to the swap files (see Section 5.5.38 [Tooltype TEMPDIR], page 132).

At most, two of these pages will be allocated in RAM at any one time.

5.5.42 UNDOBUFFERS

Name UNDOBUFFERS

Type Numeric

Description

The number of levels of undo / redo available. The larger this number, the more disk space is needed for the virtual memory swap files (this doesn't change the amount of RAM required).

5.5.43 APPICON

Name APPICON

Type Boolean

Description

Turns the AppIcon on the Workbench on or off. By having an AppIcon, users may double-click on the icon to bring the ImageStudio screen to the front or drop image files on it to load them.

5.5.44 COLOURPREVIEW

Name COLOURPREVIEW

Type Boolean

Description

Switches between a greyscale and colour preview display. A colour preview can only be displayed when ImageStudio is run on screens with 16 colours or more. The colour preview will always use an ordered dither to enhance the display of the image, whereas the dithering can be turned off when using a greyscale preview (see Section 5.5.44 [Tooltype˙COLOURPREVIEW], page 134).

The value of the PREVIEWDITHER preference is ignored when using a colour preview - the colour preview is always dithered.

5.5.45 HELP

Name HELP

Type Boolean

Description

Turns AmigaGuide help on or off. Turning on AmigaGuide help uses more RAM as some pages are kept in memory.

5.5.46 MODULECACHE

Name MODULECACHE

Type Boolean

Description

Allows modules to be kept resident in RAM to speed up access times.

Turning this option off forces all modules to be flushed from memory after they are used; this results in the program requiring very little memory but involves reading the modules off disk everytime they are required.

Module caching on will leave the modules resident in RAM, making for faster access. If the Amiga's operating system starts to get really low on RAM, the modules are still flushed.

All modules are flushed from memory when the program quits.

5.5.47 PREVIEWDITHER

Name PREVIEWDITHER

Type Boolean

Description

Turns ordered dithering on or off in the preview window. Turning preview dithering off uses slightly less memory and is slightly faster than the ordered dither.

5.5.48 PREVIEWREDRAW

Name PREVIEWREDRAW

Type Boolean

Description

Turns the redrawing of the image on or off in the preview window. Normally this is kept on, but ARexx scripts may wish to turn this off (see Section 4.10.28 [ARexx˙PREF˙SET], page 77) to speed up some processing.

5.5.49 SPLASHWINDOW

Name SPLASHWINDOW

Type Boolean

Description

Turns on or off the opening of the 'About' window when the program starts up.

5.5.50 SCREENFRONT

Name SCREENFRONT

Type Boolean

Description

ImageStudio starts up by keeping its screen behind all others. By turning this tooltype on, the screen is brought to the front after the startup initialisations have taken place.

5.6 Known bugs

Known bugs:

- Using the freely distributable PNG IO module with ImageStudio can cause side effects with the Scripts/Edit and View/External options. This is being investigated.

  If this causes you problems, remove the PNG IO module from the ImageStudio modules directory and use the freely distributable PNG datatype to load PNG files (no save) until we fix the problem.
- Calling AmigaDos commands from an ARexx script running from the 'Scripts' floating palette sometimes causes the machine to freeze if the command fails and tries to print some text. To fix this, pipe the input and output of all commands to 'NIL:' (see Section 4.7 [Common ARexx problems], page 43).

- At the moment it is possible to run more than one ARexx script from outside ImageStudio and this will lead to the ARexx commands from the multiple scripts being mixed together. We don't know if it's possible to only run one script at a time externally. If you run all your scripts from inside ImageStudio, this will never happen.
- Under Workbench2.x the checkbox and mutual exclude (radio button) gadgets don't scale with font size. This may lead to slightly strange looking windows with very small or very large fonts. Workbench3.x scales these gadgets correctly.
- ImageStudio has problems when running with the MagicFileRequester (MFR) utility. It seems that ImageStudio uses some features of the ASL file requesters that are unsupported by MFR and this causes MFR to crash. **This is not our problem, we are not doing anything wrong**. We've tried to contact the author about this, but to no avail. We suggest that when using ImageStudio, MFR is not used.

If you think you have found a bug not covered here, please contact the authors to report it (see Chapter 7 [The authors], page 142).

5.7 UnCrash

ImageStudio comes with a small CLI utility called 'UnCrash' which can be useful in trying to recover image data from the virtual memory swap files left after a crash. It is worth mentioning that there should be **no way** in which ImageStudio should crash the computer; UnCrash is for use when ImageStudio is running and an other program causes a crash or there is a power cut, etc...

UnCrash should be run immediately after the crash; running ImageStudio again before running UnCrash may cause the original virtual memory swap files to be overwritten.

UnCrash needs the following information to recover a swap file:

1. The filename of the ImageStudio virtual memory swap file. This will be located in the directory given by the TEMPDIR preference, and be called something like 'IMAGESTUDIO.1'. Normally these files are deleted when ImageStudio quits, but if the machine crashes there is no way these can be automatically deleted.
2. The filename of the output image file to be created. The output file is in the 'VMEM' file format (see Section 5.1.15 [VMEM], page 116) which only ImageStudio will load.
3. The width and height of the original image. ImageStudio will only swap out the raw image data when dealing with virtual memory; details like the image's number of colours, palette and size are not saved in the swap file so they must be provided by the user.

UnCrash will always try its best to recover the swap file and create a useable VMEM image file. Should you specify incorrect dimensions for the image or the swap file has become damaged in some way, UnCrash may produce warnings and ImageStudio may warn about the quality of the VMEM file created by UnCrash.

The following example shows how UnCrash could be used to recover a 640x480 virtual memory swap file after a crash:

```
UnCrash Work:tmp/IMAGESTUDIO.3 ram:outimage.vmem 640 480
```

6 Common questions

If you have any questions about ImageStudio, make sure that it hasn't already been answered below:

6.1 Common question 1

“Why doesn't ImageStudio support TIFF fully?”

The TIFF module supplied with ImageStudio closely follows the version 6 Baseline TIFF specifications, so it should handle most of the common TIFF files. We will probably be improving the TIFF module in the future, but to fully support the TIFF standard would be an enormous amount of programming.

As a side note, we originally were going to use the TIFFlib from Silicon Graphics' Sam Leffler. After compiling the code, we had a library (minus module code) of around 250K. We decided that this was just too much code to include as a module (of similar size to the ImageStudio executable itself!), so we wrote our own TIFF code. Bear in mind that even the TIFFlib doesn't handle all of the TIFF v6 specification, although it does handle most cases.

6.2 Common question 2

“Will you be adding SHAM, PCHG ...”

At the moment we have no plans to add SHAM or CTBL formats due to their hardware dependence. PCHG will probably be added later.

6.3 Common question 3

> "Can I turn virtual memory off?"

ImageStudio will always work with virtual memory. If you are lucky enough to have lots of RAM and you wish to use that instead of your hard disk, simply put the temp. files in the ram disk (see Section 5.5.38 [Tooltype'TEMPDIR], page 132). The overhead of using virtual memory from RAM is negligible.

6.4 Common question 4

> "What other programs have the authors written?"

Andy and Graham have written "TextureStudio", a shareware Imagine3 format texture renderer. See Chapter 11 [TextureStudio], page 146, for more information.

Andy has written "StickIt" - an Amiga equivalent of the 'PostIt' note; useful for reminding you of things to do.

Graham has written "MultiSample" and "MooseDrive". Multisample is a utility for converting to and from common Amiga / PC / Atari ST sound sample formats. MooseDrive is a frantic "viewed from the top" car racing game with multiple large scrolling tracks and the ability to upgrade your car as you win races.

All the above programs are available from PD libraries as well as the Internet's 'Aminet' servers.

6.5 Common question 5

> "How much disk space do the swap files use?"

If the image is colour-mapped, 1 byte per pixel is used. Therefore, the total number of bytes used is calculated as:

bytes used = image_width x image_height

If the image is 16 million colours, 3 bytes per pixel are used. Therefore, the total number of bytes used is calculated as:

bytes used = image_width x image_height x 3

6.6 Common question 6

“What is the .dvi and .ps documentation?”

".dvi" files are created with TEX program and can be viewed or printed with appropriate utilities. This allows any user with the TEX system installed to print the documentation out on any type of printer.

".ps" files are raw PostScript and can be printed on any PostScript printer simply be sending them directly to the printer.

6.7 Common question 7

“Will you be adding support for animations?”

Adding animation support is possible, but very very low on our list of priorities. If you require image processing of animations with ImageStudio, we recommend that you obtain a copy of Marcus Moenig’s ‘MainActor’ program. With a bit of ARexx glue, MainActor could split the animation, pass the frames to ImageStudio for processing and then join them back into an animation afterwards.

MainActor is a shareware program available from all good PD houses.

6.8 Common question 8

"Can you write a general virtual memory program?"

We have been asked many times about writing a general purpose virtual memory program that can be used with *all* Amigas (i.e. doesn't require an MMU). To the best of our knowledge, this is not possible. Our virtual memory routines don't require a MMU because they are internal to ImageStudio and ImageStudio knows which parts of memory it is currently using and which are safe to swap out to disk.

A general purpose virtual memory program has to trap processor in structions and decide whether that particular part of memory needs to be loaded in. It also has to make smart decisions as to which part of least used memory can be swapped out. In order to trap these processor memory accesses, a MMU is vital.

6.9 Common question 9

"Why don't you use a MUI for your windows design"

'MUI', or 'Magic User Interface', is a tool written by Stefan Stuntz for designing font sensitive GUIs for Amiga programs.

We do not use MUI for the same reason that we don't use any other third party GUI software that requires runtime libraries. There are 2 main problems in using third party utilities:

1. The user of ImageStudio is then expected to install extra software on their system just to get ImageStudio to run. In the case of MUI's installation, this can be be several hundred kilobytes of extra files.
2. If at a later date a new version of the Amiga's operating system causes these third party products to fail, ImageStudio will also fail. We have been very careful in writing ImageStudio to adhere to Commodore's programming guidelines and therefore we see no reason why ImageStudio should fail to work with further operating system versions. Relying on third party products does not give us that confidence.

In the very worst case that the author of third party product was unable or unwilling to update his / her software for the new operat ing system, ImageStudio would need to be re-written - not something we have any wish to do.

6.10 Common question 10

"Will you be releasing the module format"

When the format of the modules is fixed we will release development documentation describing how users may write thier own modules. Currently the module format is liable to change, so it would be unwise to release such information now.

7 The authors

ImageStudio is written by Andy Dean and Graham Dean.

Queries should be sent to:

```
Andy and Graham Dean,
14 Fielding Avenue,
Poynton,
Stockport,
Cheshire.
SK12 1YX
ENGLAND
```

Andy can be reached for queries (no orders) via Internet Email at:

```
adean@eleceng.ucl.ac.uk
```

Note: This Email address may change after September 1995; the postal address given above will remain valid.

Graham can be reached for queries (no orders) via Internet Email at:

```
ELA95GND@sheffield.ac.uk
```

The rate at which ImageStudio progresses depends on a few things:

1. You. If you like and use the program, please register it. If you like the program but think it is missing something that isn't already in our future additions list (see Chapter 10 [Future additions], page 146) **let us know!**.
2. Other work. Graham is studying for a degree and Andy is finishing a PhD and this work will take priority (sad, but true).

If you find a bug in ImageStudio that is not convered in the 'Known bugs' list (see Section 5.6 [Known bugs], page 136), please inform the authors at the above addresses; don't assume that someone else will, we don't notice all the problems ourselves.

Please include as much information about the problem as possible, including:

Version of ImageStudio and whether it is the 68000 or 68020 version.

Amiga model.

Version of Workbench/Kickstart.

Amount of RAM (fast and chip), hard disk size and type, processor.

Any extra hardware boards you have connected to your Amiga.

Any utilities you are running at the same time as ImageStudio.

Whether the problem is repeatable or just occasional.

If you have Enforcer or Mungwall, do these report anything?

We treat all bug reports with the utmost importance but it is difficult for us to investigate problems with very little detail on how they happen, so please, send us as much information as possible.

If you are having problems loading a particular file into ImageStudio, test whether it will load into any another package and if possible whether other files created by the same program also give problems. We cannot really test every faulty file, but if files created by one particular program only give problems on ImageStudio then we'll look into that.

ImageStudio has been tested on:

- A500, Workbench 2.04, 1Mbyte CHIP RAM, 2Mbyte FAST RAM, A590 85Mbyte SCSI hard drive, Microbotics VXL*30 accelerator (no 32-bit RAM).

- A1200, Workbench 3.0, 2Mbyte CHIP RAM, 8MByte FAST RAM, Power ViperI 68030 accelerator card, 68882 FPU, 270Mbyte IDE hard drive.
- A4000/EC030, Workbench 3.0, 2Mbyte CHIP RAM, 8MByte FAST RAM, 68882 FPU, 130Mbyte + 420Mbyte IDE hard drives, CyberVision64 (2MB) graphics card.

ImageStudio shows no problems with either the Enforcer or Mungwall debugging tools.

8 How to register

To receive the full version of ImageStudio, send 15 pounds sterling (30 US dollars overseas) to:

`LH Publishing`
`13 Gairlock Avenue`
`Bletchley`
`MK2 3DH`
`ENGLAND`

`larry@em.powernet.co.uk`

LH Publishing accept most types of payment, but if you have any questions please don't hesitate to contact them first.

Note: If you are sending cash, please make sure that it is well wrapped in the envelope.

In return you will receive the latest version of ImageStudio complete with printed manual, along with a personal keyfile to unlock the package. Each keyfile is unique to the registered user, please do not distribute the keyfile to others as it can be traced back to you. Allow a resonable time to allow cheque clearance, the processing of the order, etc...

Upgrades will be offered to registered users free of charge. As we are now operating a keyfile concept, upgrades can be obtained by getting the latest version from the Internet, Aminet, BBS's, PD houses etc... If your local provider doesn't have the latest version, pester them until they get it!

Upgrades and orders will not now be given by contacting the authors directly, unless there is a very good reason for it (we're sorry, but we don't have the resources to deal with all the orders and upgrades directly!).

The version number of ImageStudio (see Section 3.1.5 [About], page 16) is to be interpreted as:

```
version.revision.subrevision
```

The ‘version’ shows the main version of the program, ‘revision’ will be increased as small additions and improvements are made to the program. The ‘subrevision’ value is incremented with bug fixes. All the values are simple decimal, not floating point, so version 1.9.0 would be followed by version 1.10.0.

New versions will be distributed with every change in revision number, bug fixes are likely to be distributed as “patches” (more details to follow).

9 Credits

The authors would like to thank:

- Escom / Amiga Technologies.
- Commodore-Amiga.
- Our beta testers.
- Our parents, for their support (especially our mum for also helping with the posting and packing!!!).
- Larry Hickmott, for his help and support and the time he's put into creating the printed manual.
- Matt Dillon, for the ‘Dice’ C compiler.
- SAS Institute, for the ‘SAS/C’ C compiler.
- Ian OConner, for ‘The Designer’ - used to do all the GUI windows de sign.
- Michael Balzer, for ‘ARexxBox’ - used to implement the ARexx port.
- All those associated with the CyberGraphX retargetable graphics system.
- Jonathan Forbes, for ‘LX’ - used to decompress the .lha files in the distribution.
- All the public domain / freeware / shareware authors, for loads of great software.
- To those on the comp.sys.amiga.programmer newsgroup who've responded to our programming problems.
- The Independant JPEG Group, for their essential JPEG code and in formation.
- All those involved with the excellent $\TeX$ and ‘TeXinfo’ packages.

10 Future additions

The following features will probably be added to future versions of the packages (roughly in order):

- Making the effects external modules.
- Spare buffer.
- Alpha channel buffer.
- Aspect ratio correction in the preview window.
- ARexx macro record.
- Halftoning operators.
- Image rotation.
- Write a separate program used to build ARexx batch processing scripts by simply clicking on the actions you wish to perform.
- Toolbar.

11 TextureStudio

Brief description:

TextureStudio supports the loading of texture modules in Imagine3 format. The parameters of the texture can quickly and easily be adjusted by means of slider gadgets or by typing in the numbers. The texture can then be mapped onto a plane, cylinder or sphere and rendered to a preview screen and/or as a 24-bit image to disk.

Many aspects of the texture and render can be altered including axis position/alignment/size, lighting settings, object colours, object size etc.

TextureStudio allows the user to quickly render the texture and explore the effects of changing various parameters without the need to ray-trace a new image each time something altered.

TextureStudio can render images to disk in IFF-ILBM24, JPEG or Targa format. This allows high quality images to be rendered and loaded into other programs. 24-bit images of any size can be rendered, regardless of memory available.

Some example textures are included in the distribution but Imagine3 is required to be able to use it's textures.

List of feature:

Supports Imagine3 texture format.
Control of features via ARexx port.
Render unlimited number of textures simultaneously.
Render to HAM screen and 24-bit images on all Amigas.
Render 24-bit images of any size onto disk in IFF-ILBM24, JPEG or Targa format, regardless of memory available.
Saving of HAM preview screen to disk.
Support for colour, filter and bump type textures.
Easy adjustment of parameters by means of slider gadgets.
Map textures onto a plane, cylinder or sphere.
Control of light colour, distance, position, backlighting and intensity.
Adjustment of axis alignment, size and position.
Control of object size, visible width and image aspect ratio.
Multiple pass render to allow quick preview of image whilst it renders.
5 levels of anti-aliasing available.
Preview colours with colourbox window.
Alter render screens width, height and screenmode.
Control of all main functions from floating windows.
Optimised code for 68881 and 68882 FPU's for maximum speed.
Render plane, cylinder or sphere without any texture to quickly set up lighting etc.
Loading and saving of textures settings, parameters and axis positions.
Render preferences to alters speed and accuracy of render.
Configure window positions, screenmode, default settings and then save preferences to disk.
Runs on all Amigas with Workbench2.04 or above and an FPU (floating point unit).
Standard Workbench2 interface.
Uses public screen.

Shareware version:

The shareware version is limited in that only the first 8 parameters of the texture can be adjusted. No other features have been removed.

The full version allows all 16 parameters to be altered.

See Chapter 8 [How to register], page 144, for details on how to register. Registration is 15UK pounds or 30US dollars.

Table of Contents