

ImageStudio

COLLABORATORS

	<i>TITLE :</i> ImageStudio		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		January 9, 2023	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	ImageStudio	1
1.1	ImageStudio.guide	1
1.2	ImageStudio.guide/Introduction	2
1.3	ImageStudio.guide/Copyright and Disclaimer	2
1.4	ImageStudio.guide/Machine requirements	3
1.5	ImageStudio.guide/Brief description	3
1.6	ImageStudio.guide/List of features	4
1.7	ImageStudio.guide/Shareware version	6
1.8	ImageStudio.guide/Starting ImageStudio	7
1.9	ImageStudio.guide/Upgrading from version 1.x.x	7
1.10	ImageStudio.guide/Configuring ImageStudio	8
1.11	ImageStudio.guide/Tutorial	9
1.12	ImageStudio.guide/Changing the image format	9
1.13	ImageStudio.guide/Changing the number of colours	10
1.14	ImageStudio.guide/Changing the colour balance	11
1.15	ImageStudio.guide/Applying an effect	12
1.16	ImageStudio.guide/Applying a convolution	13
1.17	ImageStudio.guide/Scaling the image	14
1.18	ImageStudio.guide/Menu options	15
1.19	ImageStudio.guide/Project	15
1.20	ImageStudio.guide/Open	16
1.21	ImageStudio.guide/Save	16
1.22	ImageStudio.guide/Screen_mode	17
1.23	ImageStudio.guide/Prefs	17
1.24	ImageStudio.guide/About	18
1.25	ImageStudio.guide/Info	18
1.26	ImageStudio.guide/Help	18
1.27	ImageStudio.guide/Create_keyfile	19
1.28	ImageStudio.guide/Iconify	20
1.29	ImageStudio.guide/Quit	20

1.30	ImageStudio.guide/Edit	20
1.31	ImageStudio.guide/Undo	21
1.32	ImageStudio.guide/Redo	21
1.33	ImageStudio.guide/Copy	21
1.34	ImageStudio.guide/Paste	22
1.35	ImageStudio.guide/Region_co-ords	22
1.36	ImageStudio.guide/Region_clear	22
1.37	ImageStudio.guide/Select_all	23
1.38	ImageStudio.guide/View	23
1.39	ImageStudio.guide/Full_image	23
1.40	ImageStudio.guide/Zoom_in	24
1.41	ImageStudio.guide/Zoom_out	24
1.42	ImageStudio.guide/Internal_viewer	24
1.43	ImageStudio.guide/View_screenmode	25
1.44	ImageStudio.guide/External_viewer	25
1.45	ImageStudio.guide/Process	26
1.46	ImageStudio.guide/Crop	26
1.47	ImageStudio.guide/Scale	26
1.48	ImageStudio.guide/Colours	27
1.49	ImageStudio.guide/Palette	28
1.50	ImageStudio.guide/Tools	29
1.51	ImageStudio.guide/Command_shell	30
1.52	ImageStudio.guide/Show_balance	30
1.53	ImageStudio.guide/Show_effects	32
1.54	ImageStudio.guide/Show_convolve	33
1.55	ImageStudio.guide/Show_scripts	34
1.56	ImageStudio.guide/ARexx	35
1.57	ImageStudio.guide/Introduction to ARexx	36
1.58	ImageStudio.guide/Writing scripts	37
1.59	ImageStudio.guide/Basic ARexx	38
1.60	ImageStudio.guide/Command templates	41
1.61	ImageStudio.guide/Return values	43
1.62	ImageStudio.guide/Error checking	45
1.63	ImageStudio.guide/Common ARexx problems	46
1.64	ImageStudio.guide/ARexx problem 1	46
1.65	ImageStudio.guide/ARexx problem 2	47
1.66	ImageStudio.guide/ARexx problem 3	48
1.67	ImageStudio.guide/ARexx problem 4	48
1.68	ImageStudio.guide/ARexx problem 5	49

1.69	ImageStudio.guide/ARexx tips	49
1.70	ImageStudio.guide/ARexx tip 1	49
1.71	ImageStudio.guide/ARexx tip 2	50
1.72	ImageStudio.guide/Example scripts	50
1.73	ImageStudio.guide/BalanceTest	51
1.74	ImageStudio.guide/BatchConvert	52
1.75	ImageStudio.guide/BatchProcess	52
1.76	ImageStudio.guide/BatchProcessNotify	53
1.77	ImageStudio.guide/ConvolveTest	54
1.78	ImageStudio.guide/Demo	55
1.79	ImageStudio.guide/ToIcon	55
1.80	ImageStudio.guide/ARexx commands	55
1.81	ImageStudio.guide/ARexx_BALANCE	57
1.82	ImageStudio.guide/ARexx_COLOURS	59
1.83	ImageStudio.guide/ARexx_CONVOLVE	60
1.84	ImageStudio.guide/ARexx_COPY	61
1.85	ImageStudio.guide/ARexx_CROP	62
1.86	ImageStudio.guide/ARexx_EFFECT	63
1.87	ImageStudio.guide/ARexx_FILES_MATCH	64
1.88	ImageStudio.guide/ARexx_FILE_JOIN	65
1.89	ImageStudio.guide/ARexx_FILE_RENAME	66
1.90	ImageStudio.guide/ARexx_FILE_SPLIT	67
1.91	ImageStudio.guide/ARexx_FULL_IMAGE	68
1.92	ImageStudio.guide/ARexx_GUI_BLOCK	69
1.93	ImageStudio.guide/ARexx_GUI_UNBLOCK	70
1.94	ImageStudio.guide/ARexx_HELP	71
1.95	ImageStudio.guide/ARexx_HSV_TO_RGB	72
1.96	ImageStudio.guide/ARexx_IMAGEINFO_GET	73
1.97	ImageStudio.guide/ARexx_IMAGEINFO_SET	74
1.98	ImageStudio.guide/ARexx_NOTIFY_DIR	75
1.99	ImageStudio.guide/ARexx_NOTIFY_FILE	77
1.100	ImageStudio.guide/ARexx_OPEN	78
1.101	ImageStudio.guide/ARexx_PALETTE_GET	79
1.102	ImageStudio.guide/ARexx_PALETTE_LOAD	80
1.103	ImageStudio.guide/ARexx_PALETTE_SAVE	81
1.104	ImageStudio.guide/ARexx_PALETTE_SET	82
1.105	ImageStudio.guide/ARexx_PALETTE_SORT	84
1.106	ImageStudio.guide/ARexx_PASTE	85
1.107	ImageStudio.guide/ARexx_PREF_GET	86

1.108	ImageStudio.guide/ARexx_PREF_SET	87
1.109	ImageStudio.guide/ARexx_QUIT	88
1.110	ImageStudio.guide/ARexx_REDO	89
1.111	ImageStudio.guide/ARexx_REDRAW	90
1.112	ImageStudio.guide/ARexx_REGION_CLEAR	91
1.113	ImageStudio.guide/ARexx_REGION_GET	91
1.114	ImageStudio.guide/ARexx_REGION_SET	93
1.115	ImageStudio.guide/ARexx_REQUEST_DIR	94
1.116	ImageStudio.guide/ARexx_REQUEST_FILE	95
1.117	ImageStudio.guide/ARexx_REQUEST_LIST	97
1.118	ImageStudio.guide/ARexx_REQUEST_MESSAGE	98
1.119	ImageStudio.guide/ARexx_REQUEST_MULTIFILE	100
1.120	ImageStudio.guide/ARexx_REQUEST_MULTIVALUE	101
1.121	ImageStudio.guide/ARexx_REQUEST_SCREENMODE	103
1.122	ImageStudio.guide/ARexx_REQUEST_STRING	106
1.123	ImageStudio.guide/ARexx_RGB_TO_HSV	108
1.124	ImageStudio.guide/ARexx_SAVE	109
1.125	ImageStudio.guide/ARexx_SCALE	110
1.126	ImageStudio.guide/ARexx_SCREEN_BACK	111
1.127	ImageStudio.guide/ARexx_SCREEN_FRONT	112
1.128	ImageStudio.guide/ARexx_SELECT_ALL	113
1.129	ImageStudio.guide/ARexx_UNDO	114
1.130	ImageStudio.guide/ARexx_VIEW	115
1.131	ImageStudio.guide/ARexx_ZOOM_IN	115
1.132	ImageStudio.guide/ARexx_ZOOM_OUT	116
1.133	ImageStudio.guide/Reference	117
1.134	ImageStudio.guide/File formats	117
1.135	ImageStudio.guide/IFF-ILBM	118
1.136	ImageStudio.guide/BMP	119
1.137	ImageStudio.guide/EPS	120
1.138	ImageStudio.guide/GIF	121
1.139	ImageStudio.guide/JPEG	122
1.140	ImageStudio.guide/PCX	123
1.141	ImageStudio.guide/Targa	124
1.142	ImageStudio.guide/Effects	124
1.143	ImageStudio.guide/Dynamic range	125
1.144	ImageStudio.guide/Flip X	125
1.145	ImageStudio.guide/Flip Y	126
1.146	ImageStudio.guide/Greyscale	126

1.147ImageStudio.guide/Highlight	126
1.148ImageStudio.guide/Negative	127
1.149ImageStudio.guide/Random	127
1.150ImageStudio.guide/Remove isolated pixels	127
1.151ImageStudio.guide/Roll X	128
1.152ImageStudio.guide/Roll Y	128
1.153ImageStudio.guide/Pixelise	129
1.154ImageStudio.guide/Shadow	129
1.155ImageStudio.guide/Image types	129
1.156ImageStudio.guide/Colour representations	130
1.157ImageStudio.guide/Tooltypes	131
1.158ImageStudio.guide/Tooltype_BALANCE	133
1.159ImageStudio.guide/Tooltype_BALANCELEFT	133
1.160ImageStudio.guide/Tooltype_BALANCETOP	133
1.161ImageStudio.guide/Tooltype_CONVOLVE	134
1.162ImageStudio.guide/Tooltype_CONVOLVELEFT	134
1.163ImageStudio.guide/Tooltype_CONVOLVETOP	134
1.164ImageStudio.guide/Tooltype_EFFECT	135
1.165ImageStudio.guide/Tooltype_EFFECTLEFT	135
1.166ImageStudio.guide/Tooltype_EFFECTTOP	135
1.167ImageStudio.guide/Tooltype_SCRIPTS	135
1.168ImageStudio.guide/Tooltype_SCRIPTSLEFT	136
1.169ImageStudio.guide/Tooltype_SCRIPTSTOP	136
1.170ImageStudio.guide/Tooltype_PREVIEWLEFT	136
1.171ImageStudio.guide/Tooltype_PREVIEWTOP	137
1.172ImageStudio.guide/Tooltype_PREVIEWWIDTH	137
1.173ImageStudio.guide/Tooltype_PREVIEWHEIGHT	137
1.174ImageStudio.guide/Tooltype_SCREENMODEID	138
1.175ImageStudio.guide/Tooltype_SCREENOVERSCAN	138
1.176ImageStudio.guide/Tooltype_SCREENDEPTH	138
1.177ImageStudio.guide/Tooltype_SCREENWIDTH	139
1.178ImageStudio.guide/Tooltype_SCREENHEIGHT	139
1.179ImageStudio.guide/Tooltype_BLANKSCRIPT	139
1.180ImageStudio.guide/Tooltype_CONVOLVEDIR	140
1.181ImageStudio.guide/Tooltype_EXTERNALVIEWER	140
1.182ImageStudio.guide/Tooltype_HELPFILE	141
1.183ImageStudio.guide/Tooltype_IMAGEDIR	141
1.184ImageStudio.guide/Tooltype_KEYFILE	142
1.185ImageStudio.guide/Tooltype_PALETTEIDIR	142

1.186ImageStudio.guide/Tooltype_PORTNAME	142
1.187ImageStudio.guide/Tooltype_REXXOUTPUT	143
1.188ImageStudio.guide/Tooltype_SCREENNAME	143
1.189ImageStudio.guide/Tooltype_SCRIPTSDIR	144
1.190ImageStudio.guide/Tooltype_SCRIPTSEXT	144
1.191ImageStudio.guide/Tooltype_TEMPDIR	144
1.192ImageStudio.guide/Tooltype_TEXTEDITOR	145
1.193ImageStudio.guide/Tooltype_CLIPUNIT	145
1.194ImageStudio.guide/Tooltype_PAGESIZE	146
1.195ImageStudio.guide/Tooltype_UNDOBUFFERS	146
1.196ImageStudio.guide/Tooltype_APPICON	147
1.197ImageStudio.guide/Tooltype_HELP	147
1.198ImageStudio.guide/Tooltype_PREVIEWDITHER	147
1.199ImageStudio.guide/Tooltype_PREVIEWREDRAW	148
1.200ImageStudio.guide/Tooltype_SPLASHWINDOW	148
1.201ImageStudio.guide/Tooltype_SCREENFRONT	148
1.202ImageStudio.guide/Known bugs	149
1.203ImageStudio.guide/Problems with datatypes	149
1.204ImageStudio.guide/Common questions	150
1.205ImageStudio.guide/Common question 1	151
1.206ImageStudio.guide/Common question 2	151
1.207ImageStudio.guide/Common question 3	151
1.208ImageStudio.guide/Common question 4	152
1.209ImageStudio.guide/Common question 5	152
1.210ImageStudio.guide/Common question 6	152
1.211ImageStudio.guide/Common question 7	153
1.212ImageStudio.guide/Common question 8	153
1.213ImageStudio.guide/Common question 9	153
1.214ImageStudio.guide/Common question 10	154
1.215ImageStudio.guide/The authors	155
1.216ImageStudio.guide/How to register	155
1.217ImageStudio.guide/Credits	157
1.218ImageStudio.guide/Future additions	158

Chapter 1

ImageStudio

1.1 ImageStudio.guide

Welcome to ImageStudio, shareware image processing and ↔
conversion
package for the Amiga.

This document applies to version 2.0.x, written on 13th March 1995,
Copyright (C) 1994,1995 Andy Dean, Graham Dean.

Introduction

Welcome to ImageStudio

Tutorial

Worked Examples

Menu options

Detailed description of program's functions

ARexx

Using ARexx to write scripts

Reference

Extra information on the program's workings

Common questions

Frequently asked questions

The authors

How to contact us, reporting problems

How to register

Receiving the full version

Credits

Thanks to...

Future additions

Extra planned upgrades

1.2 ImageStudio.guide/Introduction

Introduction

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

Copyright and Disclaimer

Machine requirements

Brief description

List of features

Shareware version

Upgrading from version 1.x.x

Starting ImageStudio

Configuring ImageStudio

1.3 ImageStudio.guide/Copyright and Disclaimer

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.

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 program you must use the original unmodified

distribution archive.

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

Credits
).

1.4 ImageStudio.guide/Machine requirements

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.

1.5 ImageStudio.guide/Brief description

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

Image types

.

ImageStudio comes with a fully featured ARexx port, which allows the writing of macro scripts to automate repetitive 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.

1.6 ImageStudio.guide/List of features

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.
- * 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 (external for third party 24-bit graphics cards).
- * Loading / saving / manipulating of AGA image formats (e.g. 256 colours, HAM8) on non-AGA machines.
- * Max image size of 32000 x 32000 (limited to 250 x 250 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.
- * Standard Workbench2 interface.
- * Public screen.
- * Requires no third party libraries or utilities.

Import:

- * IFF-ILBM formats (Standard palette based, HAM6, HAM8, extra halfbright, ILBM24)
- * BMP
- * GIF (conforming to GIF87a)
- * JPEG (conforming to JFIF standard)
- * PCX
- * Targa
- * Any installed Amiga datatype (with Workbench 2.1+)

Export:

- * IFF-ILBM formats (Standard palette based, HAM6, HAM8, extra halfbright, ILBM24)
- * BMP
- * EPS
- * JPEG (conforming to JFIF standard)
- * PCX
- * Targa

Colour Balance:

- * All operations are available to the R,G,B components separately.
-

* Brightness (upto $\pm 100\%$)

* 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 dithers available).

1.7 ImageStudio.guide/Shareware version

Shareware version

=====

To encourage users to register, the freely distributable version of ImageStudio is limited to loading in pictures upto 250x250 pixels. All other operations are available.

For details on how to register, See
How to register

.

1.8 ImageStudio.guide/Starting ImageStudio

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 preview dithering on. See [Tooltypes](#), for a full list of available tooltypes.

1.9 ImageStudio.guide/Upgrading from version 1.x.x

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 [Create keyfile](#), 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 lose 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.

1.10 ImageStudio.guide/Configuring ImageStudio

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

Prefs

, 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.

1.11 ImageStudio.guide/Tutorial

Tutorial

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

Changing the image format

Changing the number of colours

Changing the colour balance

Applying an effect

Applying a convolution

Scaling the image

1.12 ImageStudio.guide/Changing the image format

Changing the image format

=====

The simplest use of ImageStudio is just as a file format converter; See

File formats
, 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:

- * ImageStudio automatically recognises the filetype of the incoming image. It will use its internal loaders first, then trying datatypes if running on Workbench 2.1+.
- * All time consuming operations show their progress in the Infobar's fuelgauge and can be aborted by clicking on the 'Abort' gadget.

1.13 ImageStudio.guide/Changing the number of colours

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 colour 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.

1.14 ImageStudio.guide/Changing the colour balance

Changing the colour balance

=====

When receiving images from external sources (scanners, frame grabbers) it is often 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
 Show balance
 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

Show balance
, for more information on the balance floating

palette.

1.15 ImageStudio.guide/Applying an effect

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

Effects
, for a comprehensive description of all the available effects.

See

Show effects
, for more information on the effects floating palette.

1.16 ImageStudio.guide/Applying a convolution

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 colour-mapped image and convolution only works in 24-bit (see
Show convolves
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 is 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

Show convolves
, for more information on the convolves floating

palette.

1.17 ImageStudio.guide/Scaling the image

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.
-

1.18 ImageStudio.guide/Menu options

Menu options

This chapter describes ImageStudio's menu options in detail.

Selecting a main menu from the list below will give a list of sub-menu options.

Project

Edit

View

Process

Tools

1.19 ImageStudio.guide/Project

Project

=====

Select the submenu you wish to investigate.

Open

Save

Screen mode

Prefs

About

Info

Help

Create keyfile

Iconify

Quit

1.20 ImageStudio.guide/Open

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

Image types

.

1.21 ImageStudio.guide/Save

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...' and 'Screen...' buttons remain unghosted. Some formats (e.g. IFF-ILBM) have further options available by clicking on the 'Options...' button. The 'Screen...' button can be clicked on to change the screenmode of the saved image (only used with IFF-ILBM images).

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.

1.22 ImageStudio.guide/Screen_mode

Screen mode

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

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'.

1.23 ImageStudio.guide/Prefs

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

Tooltypes

) name is given on the right.

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

```
PREF_GET  
) and written
```

(see

```
PREF_SET  
) from ARexx.
```

1.24 ImageStudio.guide/About

About

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

1.25 ImageStudio.guide/Info

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.

1.26 ImageStudio.guide/Help

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
 Prefs
).
3. The full filename of the AmigaGuide helpfile must be given in the
 'HELPPFILE' preference (see
 HELPPFILE
).

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 wish to know more about, but instead of releasing the right mouse button to select the item, press 'Help' on the keyboard. 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.

1.27 ImageStudio.guide/Create_keyfile

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 registered 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

```
KEYFILE
) 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.

1.28 ImageStudio.guide/Iconify

```
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 reduce 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.

1.29 ImageStudio.guide/Quit

```
Quit
----
```

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

1.30 ImageStudio.guide/Edit

```
Edit
=====
```

Select the submenu you wish to investigate.

Undo
Redo
Copy
Paste
Region co-ords
Region clear
Select all

1.31 ImageStudio.guide/Undo

Undo

Keyboard shortcut - Amiga - U

Undos last operation.

The maximum number of undos is set in the preferences requester, See

Prefs

.

1.32 ImageStudio.guide/Redo

Redo

Keyboard shortcut - Amiga - R

Redos last undo.

The maximum number of redos is set in the preferences requester, See

Prefs

.

1.33 ImageStudio.guide/Copy

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.

1.34 ImageStudio.guide/Paste

Paste

Keyboard shortcut - Amiga - V

Reads in image from the clipboard.

1.35 ImageStudio.guide/Region_co-ords

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.

1.36 ImageStudio.guide/Region_clear

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.

1.37 ImageStudio.guide/Select_all

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
Zoom in
) ,

'Select all' will not select the whole image but just the displayed image in the preview window.

1.38 ImageStudio.guide/View

View

====

Select the submenu you wish to investigate.

Full image

Zoom in

Zoom out

View

View screenmode

External viewer

1.39 ImageStudio.guide/Full_image

Full image

Keyboard shortcut - Amiga - F

Displays the whole image in the preview window.

1.40 ImageStudio.guide/Zoom_in

Zoom in

Keyboard shortcut - Amiga - <

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

1.41 ImageStudio.guide/Zoom_out

Zoom out

Keyboard shortcut - Amiga - >

Zooms out by a factor of 3 times.

1.42 ImageStudio.guide/Internal_viewer

Internal viewer

Keyboard shortcut - Amiga - I

Views the current image with the internal viewer.

If the image is colour-mapped, the viewer will try and open a screen with the same number of colours as the image. Under the AGA graphics chipset this should always be possible (providing there is enough free CHIP RAM), however under the ECS chipset it is impossible to open up screens of greater than 5 bitplanes. Viewing a 256 colour image, for example, on a ECS machine is not possible directly.

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

View screenmode

.

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 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 increase the number of colours to 16 million (see Colours) and use the 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.

1.43 ImageStudio.guide/View_screenmode

View screenmode

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

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

1.44 ImageStudio.guide/External_viewer

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
Prefs
.

1.45 ImageStudio.guide/Process

Process

=====

Select the submenu you wish to investigate.

Crop

Scale

Colours

Palette

1.46 ImageStudio.guide/Crop

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.

1.47 ImageStudio.guide/Scale

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.

1.48 ImageStudio.guide/Colours

Colours

Keyboard shortcut - Amiga - K

Allows increasing / decreasing of the number of colours in the image, 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 become a 32 colour image.

Colour-mapped images can also be turned into 24-bit images by clicking 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 inevitable 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 reduce 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 colour 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.

1.49 ImageStudio.guide/Palette

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

Colours

. 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

Colour representations

).

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. Clicking on the first colour, then on 'Spread', then 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 increasing 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!.

----- Footnotes -----

(1) Except Brilliance, which seems to save all 384 colours of its palette.

1.50 ImageStudio.guide/Tools

Tools

=====

Select the submenu you wish to investigate.

Command shell

Show balance

Show effects

Show convolves

Show scripts

1.51 ImageStudio.guide/Command_shell

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 `Show scripts`). 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 necessary 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

Common ARexx problems

):

```
REQUEST_MESSAGE TEXT "A string with spaces in it"
```

Help on a particular command's template (see

`Command templates`

) can

be obtained by typing:

```
help <command>
```

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

1.52 ImageStudio.guide/Show_balance

Show balance

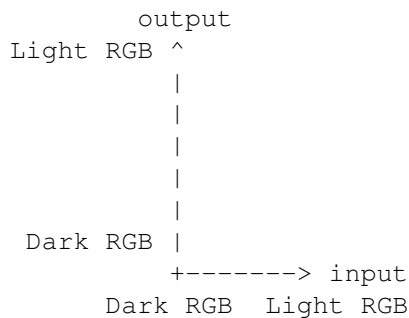
Keyboard shortcut - Amiga - 1

Opens / closes the balance floating palette.

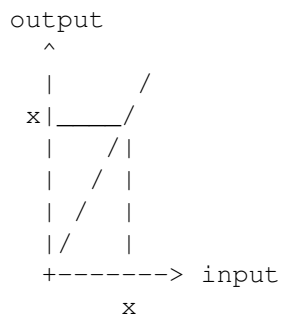
The balance floating palette is used to control the brightness, contrast and gamma of the current image. On 24-bit images, the colour balance can be altered on selected regions as well as the whole image whereas colour-mapped images only allow alterations to the whole image.

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' checkboxes 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.



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



Brightness

Altering the brightness is achieved by multiplying 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 Effects

.

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%).

1.53 ImageStudio.guide/Show_effects

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.

See

Effects

, for details of all available effects.

1.54 ImageStudio.guide/Show_convolves

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

Prefs

.

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

```

$((0 \times a) + (1 \times b) + (0 \times c) + (1 \times d) + (2 \times e) + (1 \times f) + (0 \times g) + (1 \times h) + (0 \times i)) / \text{Div} + \text{Bias}$

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.

1.55 ImageStudio.guide/Show_scripts

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'. 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

Writing scripts

, 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 necessary.

To edit the currently selected script, the 'Edit...' button is used. The text editor set in the preferences (see

Prefs

) 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.

1.56 ImageStudio.guide/ARexx

ARexx

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

Introduction to ARexx

Writing scripts

Basic ARexx

Command templates

Return values

Error checking

Common ARexx problems

ARexx tips

Example scripts

ARexx commands

1.57 ImageStudio.guide/Introduction to ARexx

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 because 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 directory and whenever a new file is generated to perform a set of operations 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 Example scripts), 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.

1.58 ImageStudio.guide/Writing scripts

Writing scripts

=====

We have tried to simplify the process of writing an ARexx script as much as 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

Prefs

). 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 Formual racing car. This could now be copied into the clipboard 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

```
ARexx commands
, for a full description of the ARexx commands
```

used in the above example.

Note, please don't remove the "/* BEGIN PROGRAM ****" and corresponding "END PROGRAM" lines in the blank macro template - these will be used to insert the recorded ARexx macros when they are implemented (see

```
Future additions
).
```

1.59 ImageStudio.guide/Basic ARexx

```
Basic ARexx
```

```
=====
```

This section is meant as a beginners guide to using ARexx with ImageStudio. 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

```
Show scripts
).
```

For further information on ARexx, we suggest reading Commodore's

ARexx user guide supplied with the A4000 or the Workbench2 and 3 upgrade 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 similarities. It is not similar to Assembler language, Lisp or Prolog. An ARexx program 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?'" ,
    BUTTONTTEXT "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

ARexx problem 1

, 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 [Return values](#)) 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

Example scripts
) 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

REXXOUTPUT
) . After the script has been executed, this file can be examined.

1.60 ImageStudio.guide/Command templates

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 omitted 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 an 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 overwritten:

```
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
SCALE
). 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 PERCENT 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

ARexx commands

, for more detailed descriptions of each of the individual ARexx commands.

1.61 ImageStudio.guide/Return values

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

```
FILE_JOIN
```

```
) 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 variable called "FULLNAME"

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

Some ARexx commands return multiple values, and these to can be returned in a single variable - each returned value in the variable is seperated with a space. The following returns information about the current image (see

```
IMAGEINFO_GET
):
```

```
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" directory:

```
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

ARexx tips

, for more information on how to turn array stem variables into string variables.

1.62 ImageStudio.guide/Error checking

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. mis-spelled 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 necessary, 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
```

1.63 ImageStudio.guide/Common ARexx problems

Common ARexx problems

=====

```
ARexx problem 1
    I can't use strings with spaces in them

ARexx problem 2
    I can't use AmigaDos commands in a script

ARexx problem 3
    I can't run scripts from the ram disk

ARexx problem 4
    I can't set the same variable twice with VAR

ARexx problem 5
    I can't get any ARexx script to run
```

1.64 ImageStudio.guide/ARexx problem 1

ARexx problem 1

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

Care must be taken when specifying string parameters 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.

1.65 ImageStudio.guide/ARexx problem 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.

1.66 ImageStudio.guide/ARexx problem 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:'.

1.67 ImageStudio.guide/ARexx problem 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'
```

1.68 ImageStudio.guide/ARexx problem 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.

1.69 ImageStudio.guide/ARexx tips

ARexx tips

=====

ARexx tip 1

How to turn stem arrays into strings

ARexx tip 2

Shortening command names

1.70 ImageStudio.guide/ARexx tip 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 necessary 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 OLDDPALETTE.

/* Convert the stem to a parameter list */

NEWPALETTE = ''

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

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

PALETTE_SET PALETTE NEWPALETTE REMAP
```

1.71 ImageStudio.guide/ARexx tip 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 Command shell) and not be used in ARexx scripts.
- * If the shortened command name is ambiguous, the first matching command will be executed. For example, if the shortened command 'REQUEST' is used, 'REQUEST_DIR' will be executed.

1.72 ImageStudio.guide/Example scripts

Example scripts

=====

BalanceTest	Alter the colour balance of an image
BatchConvert	Converts many files to a chosen file format
BatchProcess	Apply image processing to a series of images
BatchProcessNotify	Start image processing when given file changes
ConvolveTest	Applies chosen convolution filters to an image
Demo	Demonstration of some of ImageStudio's features
ToIcon	Converts image to the size / colours of an icon

1.73 ImageStudio.guide/BalanceTest

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.

1.74 ImageStudio.guide/BatchConvert

BatchConvert

Description

This script 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 subformat.
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.

1.75 ImageStudio.guide/BatchProcess

BatchProcess

Description

This script allows the processing of multiple images to be output as one image format. The script is based on 'BatchConvert' (see

BatchConvert
)

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 subformat.
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.

1.76 ImageStudio.guide/BatchProcessNotify

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 BatchProcess).

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 BatchProcess),

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' directory. 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 doing 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.

1.77 ImageStudio.guide/ConvolveTest

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 'Convolve' floating palette. This is because convolution filters with the CONVOLVE command are chosen by name, not filename (see

CONVOLVE
).

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.

1.78 ImageStudio.guide/Demo

Demo

Description

Demonstrates some of the features of the ImageStudio ARexx interface.

Simply follow the prompts to see the features being explained.

Known bugs

None.

1.79 ImageStudio.guide/Tolcon

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.

1.80 ImageStudio.guide/ARexx commands

ARexx commands

=====

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

BALANCE
COLOURS
CONVOLVE
COPY
CROP
EFFECT
FILES_MATCH
FILE_JOIN
FILE_RENAME
FILE_SPLIT
FULL_IMAGE
GUI_BLOCK
GUI_UNBLOCK
HELP
HSV_TO_RGB
IMAGEINFO_GET
IMAGEINFO_SET
NOTIFY_DIR
NOTIFY_FILE
OPEN
PALETTE_GET
PALETTE_LOAD
PALETTE_SAVE
PALETTE_SET
PALETTE_SORT
PASTE
PREF_GET
PREF_SET

QUIT
REDO
REDRAW
REGION_CLEAR
REGION_GET
REGION_SET
REQUEST_DIR
REQUEST_FILE
REQUEST_LIST
REQUEST_MESSAGE
REQUEST_MULTIFILE
REQUEST_MULTIVALUE
REQUEST_SCREENMODE
REQUEST_STRING
RGB_TO_HSV
SAVE
SCALE
SCREEN_BACK
SCREEN_FRONT
SELECT_ALL
UNDO
VIEW
ZOOM_IN
ZOOM_OUT

1.81 ImageStudio.guide/ARexx_BALANCE

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

 Show balance
 , 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 component of the image by -20:

BALANCE BRIGHTNESS -20 NORED NOBLUE

Known bugs
None.

1.82 ImageStudio.guide/ARexx_COLOURS

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

Colours
, 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.

1.83 ImageStudio.guide/ARexx_CONVOLVE

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

Show convolves
, 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.

1.84 ImageStudio.guide/ARexx_COPY

COPY

Command

COPY

Parameters template

None.

Return template

None.

Description

Copies the current image into the clipboard buffer.

See

Copy
, for a full description of copying images to the clipboard.

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.

1.85 ImageStudio.guide/ARexx_CROP

```
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

Crop
, 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.

1.86 ImageStudio.guide/ARexx_EFFECT

```
----- 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

```
Effects
, 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.

1.87 ImageStudio.guide/ARexx_FILES_MATCH

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 BUTTONTTEXT "More|Cancel" AUTOCANCEL,
  TEXT '""FILENAMES.FILEPARTS.l'""
end
```

Known bugs

None.

1.88 ImageStudio.guide/ARexx_FILE_JOIN

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 separate 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 separate path and fileparts (note how the '/' separator 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.

1.89 ImageStudio.guide/ARexx_FILE_RENAME

FILE_RENAME

Command

```
FILE_RENAME
```

Parameters template

```
FILE/A, FROM/A, TO/A
```

Return template

```
FILE
```

Description

Replaces the last occurrence 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 BatchConvert).

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.

1.90 ImageStudio.guide/ARexx_FILE_SPLIT

FILE_SPLIT

Command

```
FILE_SPLIT
```

Parameters template

```
FILE/A
```

Return template

```
PATHPART, FILEPART
```

Description

Splits the given filename into separate 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 separates the filename "Pics/HappyFace_240x250.bmp" into separate 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 separates the filename "T:TempImage.jpg" into separate 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.

1.91 ImageStudio.guide/ARexx_FULL_IMAGE

FULL_IMAGE

Command

FULL_IMAGE

Parameters template

None.

Return template

None.

Description

Displays the full image in the preview window.

See

Full image
, 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.

1.92 ImageStudio.guide/ARexx_GUI_BLOCK

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 information. 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.

1.93 ImageStudio.guide/ARexx_GUI_UNBLOCK

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 program 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.

1.94 ImageStudio.guide/ARexx_HELP

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 `Command shell`), 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.

1.95 ImageStudio.guide/ARexx_HSV_TO_RGB

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

Colour representations
, 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 component value of the colour.

G/N

The green component value of the colour.

B/N

The blue component 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.

1.96 ImageStudio.guide/ARexx_IMAGEINFO_GET

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

`REQUEST_SCREENMODE`

). 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.

1.97 ImageStudio.guide/ARexx_IMAGEINFO_SET

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.

1.98 ImageStudio.guide/ARexx_NOTIFY_DIR

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

FILE_JOIN

, for

information on how to add the path part of the filename to create a full filename.

ACTION

A string containing a description 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 standard OFS or FFS filesystems.

1.99 ImageStudio.guide/ARexx_NOTIFY_FILE

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

NOTIFY_DIR

), 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 and 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 standard OFS or FFS filesystems.

1.100 ImageStudio.guide/ARexx_OPEN

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

File formats

, 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

File formats

, 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

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

1.101 ImageStudio.guide/ARexx_PALETTE_GET

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.

1.102 ImageStudio.guide/ARexx_PALETTE_LOAD

```
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
COLOURS
) . 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.

1.103 ImageStudio.guide/ARexx_PALETTE_SAVE

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.

1.104 ImageStudio.guide/ARexx_PALETTE_SET

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 OLDDPALETTE.

/* Set the top 3 colours to red, white and blue */
OLDDPALETTE.PALETTE.87 = 255 /* Red */
OLDDPALETTE.PALETTE.88 = 0
OLDDPALETTE.PALETTE.89 = 0

OLDDPALETTE.PALETTE.90 = 255 /* White */
OLDDPALETTE.PALETTE.91 = 255
OLDDPALETTE.PALETTE.92 = 255

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

/* Convert the stem to a parameter list */
NEWPALETTE = ''

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

/* Force the new palette onto the image */
PALETTE_SET PALETTE NEWPALETTE REMAP
```

Known bugs

None.

1.105 ImageStudio.guide/ARexx_PALETTE_SORT

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 is 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.

1.106 ImageStudio.guide/ARexx_PASTE

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

assumes the file pasted OK. This will be fixed when the loaders / savers become external modules.

1.107 ImageStudio.guide/ARexx_PREF_GET

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

Prefs

, 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.

1.108 ImageStudio.guide/ARexx_PREF_SET

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

Prefs

, 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 virtual memory pagesize to be 200K:

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

Known bugs

None.

1.109 ImageStudio.guide/ARexx_QUIT

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.

1.110 ImageStudio.guide/ARexx_REDO

```
REDO
```

```
----
```

Command

```
REDO
```

Parameters template

```
None.
```

Return template

```
None.
```

Description

```
Re-does the last UNDO operation (see
UNDO
).
```

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.

1.111 ImageStudio.guide/ARexx_REDRAW

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.

1.112 ImageStudio.guide/ARexx_REGION_CLEAR

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.

1.113 ImageStudio.guide/ARexx_REGION_GET

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.

1.114 ImageStudio.guide/ARexx_REGION_SET

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 specified by the co-ordinates 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.

1.115 ImageStudio.guide/ARexx_REQUEST_DIR

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

```
Error checking
, 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.
```

1.116 ImageStudio.guide/ARexx_REQUEST_FILE

```
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

Error checking
, 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.

1.117 ImageStudio.guide/ARexx_REQUEST_LIST

```
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

```
Error checking
, 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.

1.118 ImageStudio.guide/ARexx_REQUEST_MESSAGE

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 button. 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

Error checking
, 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 separated 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 than 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.

1.119 ImageStudio.guide/ARexx_REQUEST_MULTIFILE

```
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 filenames.

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

Error checking

, 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 MULTIFILEINFO.
```

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

Known bugs

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

1.120 ImageStudio.guide/ARexx_REQUEST_MULTIVALUE

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

Error checking
, 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 cancels 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.STRING.1||'\n' ||,
    'Depth : ' ||MULTIVALUEINFO.STRING.3||'"

  /* Construct the new INSTRINGS */

  INSTRINGS = ''

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

end
```

Known bugs

None.

1.121 ImageStudio.guide/ARexx_REQUEST_SCREENMODE

```
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

Error checking
, 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 different.

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 screenmode 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 requester may be more compatible with older software which doesn't understand 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 screenmodes 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 endeavour to alter the internal 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 screenmode given - it may be an equivalent. Clicking on the screenmode ensures that it returns that mode's real ID.

1.122 ImageStudio.guide/ARexx_REQUEST_STRING

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

Error checking
, 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.

1.123 ImageStudio.guide/ARexx_RGB_TO_HSV

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

Colour representations
, for more details on RGB and HSV colour
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 component value of the colour.

S/N

The saturation component value of the colour.

V/N

The value component 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.

1.124 ImageStudio.guide/ARexx_SAVE

SAVE

Command

SAVE

Parameters template

FILE/A, FORMAT, ARGS, FORCE/S

Return template

None.

Description

Saves the current image out to disk.

See

Save
, 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
File formats

,
for more information 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.

1.125 ImageStudio.guide/ARexx_SCALE

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

Scale
, 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.

1.126 ImageStudio.guide/ARexx_SCREEN_BACK

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.

1.127 ImageStudio.guide/ARexx_SCREEN_FRONT**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.

1.128 ImageStudio.guide/ARexx_SELECT_ALL

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.

1.129 ImageStudio.guide/ARexx_UNDO

```
UNDO
```

Command

```
UNDO
```

Parameters template

None.

Return template

None.

Description

Un-does the last operation (see
REDO
).

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.

1.130 ImageStudio.guide/ARexx_VIEW

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 Prefs) 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.

1.131 ImageStudio.guide/ARexx_ZOOM_IN

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.

1.132 ImageStudio.guide/ARexx_ZOOM_OUT

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.

1.133 ImageStudio.guide/Reference

Reference

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

File formats

Effects

Image types

Colour representations

Tooltypes

Known bugs

Problems with datatypes

1.134 ImageStudio.guide/File formats

File formats

=====

Select the file format you wish to investigate.

IFF-ILBM

BMP

EPS

GIF

JPEG

PCX

Targa

1.135 ImageStudio.guide/IFF-ILBM

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

View screenmode
).

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 SUBFORMAT 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"'
```

1.136 ImageStudio.guide/BMP

BMP

Name

BMP

Load types

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.

1.137 ImageStudio.guide/EPS

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 printable size of 4inches by 2inch:

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

1.138 ImageStudio.guide/GIF

GIF

Name

GIF

Load types

Colour mapped ("GIF87a" and "GIF89a" formats).

Save types

None.

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 should load in both GIF87a and GIF89a although the latter is untested as we couldn't find any genuine GIF89a files.

Due to recent patent licensing arrangements, GIF save has been removed from ImageStudio. It may be added to later versions of the program when the legal situation becomes fully clear, but for the moment GIF saving is not possible within ImageStudio.

It should be pointed out that due to the restrictions imposed on the use of GIF files by CompuServe and the licensing fee imposed by Unisys, the GIF file format is likely to be soon superceded by a patent free format. We will endeavour to implement this new file format as soon as it becomes available.

ARexx OPEN command ARGS

None.

ARexx SAVE command ARGS

None.

Example

None.

1.139 ImageStudio.guide/JPEG

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 lose 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 Independent 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 degradation of image quality. For higher degree of compression choose a lower value(1). For a higher degree of quality, choose a higher value; values of 85 to 90 result in an almost unnoticeable 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"'
```

----- Footnotes -----

(1) Values less than 25 may cause problems with some JPEG readers

1.140 ImageStudio.guide/PCX

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.

1.141 ImageStudio.guide/Targa

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.

1.142 ImageStudio.guide/Effects

Effects

=====

Select the effect you wish to investigate.

Dynamic range

Flip X

Flip Y

Greyscale

Highlight

Negative

Random
Remove isolated pixels
Roll X
Roll Y
Pixelise
Shadow

1.143 ImageStudio.guide/Dynamic range

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.

1.144 ImageStudio.guide/Flip X

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.

1.145 ImageStudio.guide/Flip Y

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.

1.146 ImageStudio.guide/Greyscale

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.

1.147 ImageStudio.guide/Highlight

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.

1.148 ImageStudio.guide/Negative

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.

1.149 ImageStudio.guide/Random

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.

1.150 ImageStudio.guide/Remove isolated pixels

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.

1.151 ImageStudio.guide/Roll X

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.

1.152 ImageStudio.guide/Roll Y

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.

1.153 ImageStudio.guide/Pixelise

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.

1.154 ImageStudio.guide/Shadow

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.

1.155 ImageStudio.guide/Image types

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 colours 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).

1.156 ImageStudio.guide/Colour representations

Colour representations

=====

ImageStudio works internally with the R,G,B format of colour representation. This is the most common method of storing colour information 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.

1.157 ImageStudio.guide/Tooltypes

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 Workbench uses the "Prefs" requester of ImageStudio to configure it (see

Prefs
) , 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.

Window positions

BALANCE
BALANCELEFT
BALANCETOP
CONVOLVE
CONVOLVELEFT
CONVOLVETOP
EFFECT
EFFECTLEFT
EFFECTTOP
SCRIPTS
SCRIPTSLEFT
SCRIPTSTOP
PREVIEWLEFT
PREVIEWTOP
PREVIEWWIDTH
PREVIEWHEIGHT
ImageStudio screen

SCREENMODEID

SCREENOVERSCAN

SCREENDEPTH

SCREENWIDTH

SCREENHEIGHT

String preferences

BLANKSCRIPT

CONVOLVEDIR

EXTERNALVIEWER

HELPERFILE

IMAGEDIR

KEYFILE

PALETTERDIR

PORTNAME

REXXOUTPUT

SCREENNAME

SCRIPTSDIR

SCRIPTSEXT

TEMPDIR

TEXTEDITOR

Numeric preferences

CLIPUNIT

PAGESIZE

UNDOBUFFERS

Boolean preferences

APPICON

HELP

PREVIEWDITHER

PREVIEWREDRAW

SPLASHWINDOW

SCREENFRONT

1.158 ImageStudio.guide/Tooltype_BALANCE

BALANCE

Name

BALANCE

Type

Boolean

Description

Determines whether the balance floating palette should be open at startup.

1.159 ImageStudio.guide/Tooltype_BALANCELEFT

BALANCELEFT

Name

BALANCELEFT

Type

Numeric

Description

The top position to open the balance floating palette.

1.160 ImageStudio.guide/Tooltype_BALANCETOP

BALANCETOP

Name

BALANCETOP

Type

Numeric

Description

The left position to open the balance floating palette.

1.161 ImageStudio.guide/Tooltype_CONVOLVE

CONVOLVE

Name

CONVOLVE

Type

Boolean

Description

Determines whether the convolve floating palette should be open at startup.

1.162 ImageStudio.guide/Tooltype_CONVOLVELEFT

CONVOLVELEFT

Name

CONVOLVELEFT

Type

Numeric

Description

The top position to open the convolve floating palette.

1.163 ImageStudio.guide/Tooltype_CONVOLVETOP

CONVOLVETOP

Name

CONVOLVETOP

Type

Numeric

Description

The left position to open the convolve floating palette.

1.164 ImageStudio.guide/Tooltype_EFFECT

EFFECT

Name
 EFFECT

Type
 Boolean

Description
 Determines whether the effect floating palette should be open at startup.

1.165 ImageStudio.guide/Tooltype_EFFECTLEFT

EFFECTLEFT

Name
 EFFECTLEFT

Type
 Numeric

Description
 The top position to open the effect floating palette.

1.166 ImageStudio.guide/Tooltype_EFFECTTOP

EFFECTTOP

Name
 EFFECTTOP

Type
 Numeric

Description
 The left position to open the effect floating palette.

1.167 ImageStudio.guide/Tooltype_SCRIPTS

SCRIPTS

Name

SCRIPTS

Type

Boolean

Description

Determines whether the scripts floating palette should be open at startup.

1.168 ImageStudio.guide/Tooltype_SCRIPTSLEFT

SCRIPTSLEFT

Name

SCRIPTSLEFT

Type

Numeric

Description

The top position to open the scripts floating palette.

1.169 ImageStudio.guide/Tooltype_SCRIPTSTOP

SCRIPTSTOP

Name

SCRIPTSTOP

Type

Numeric

Description

The left position to open the scripts floating palette.

1.170 ImageStudio.guide/Tooltype_PREVIEWLEFT

PREVIEWLEFT

Name

PREVIEWLEFT

Type

Numeric

Description

The top position to open the preview window.

1.171 ImageStudio.guide/Tooltype_PREVIEWTOP

PREVIEWTOP

Name

PREVIEWTOP

Type

Numeric

Description

The left position to open the preview window.

1.172 ImageStudio.guide/Tooltype_PREVIEWWIDTH

PREVIEWWIDTH

Name

PREVIEWWIDTH

Type

Numeric

Description

The width to open the preview window.

1.173 ImageStudio.guide/Tooltype_PREVIEWHEIGHT

PREVIEWHEIGHT

Name

PREVIEWHEIGHT

Type

Numeric

Description

The height to open the preview window.

1.174 ImageStudio.guide/Tooltype_SCREENMODEID

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 REQUEST_SCREENMODE).

1.175 ImageStudio.guide/Tooltype_SCREENOVERSCAN

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.

1.176 ImageStudio.guide/Tooltype_SCREENDEPTH

SCREENDEPTH

Name

SCREENDEPTH

Type

Numeric

Description

The depth of the ImageStudio screen. Valid values are 1 to 8.

1.177 ImageStudio.guide/Tooltype_SCREENWIDTH

SCREENWIDTH

Name

SCREENWIDTH

Type

Numeric

Description

The width of the ImageStudio screen. Valid values are greater than 640.

1.178 ImageStudio.guide/Tooltype_SCREENHEIGHT

SCREENHEIGHT

Name

SCREENHEIGHT

Type

Numeric

Description

The height of the ImageStudio screen. Valid values are greater than 200.

1.179 ImageStudio.guide/Tooltype_BLANKSCRIPT

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.

1.180 ImageStudio.guide/Tooltype_CONVOLVEDIR

CONVOLVEDIR

Name

CONVOLVEDIR

Type

String

Description

The directory to scan on startup for convolution filters to include in the 'Convolve' floating palette.

1.181 ImageStudio.guide/Tooltype_EXTERNALVIEWER

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 "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.

1.182 ImageStudio.guide/Tooltype_HELPFILE

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 directory 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 referred to as:

PROGDIR:Docs/ImageStudio.guide

See

HELP

, for more information on AmigaGuide help.

1.183 ImageStudio.guide/Tooltype_IMAGEDIR

IMAGEDIR

Name

IMAGEDIR

Type

String

Description

The default directory to use for images when ImageStudio is loaded.

1.184 ImageStudio.guide/Tooltype_KEYFILE

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

How to register
).

1.185 ImageStudio.guide/Tooltype_PALETTE DIR

PALETTE DIR

Name

PALETTE DIR

Type

String

Description

The default directory to use for palettes when ImageStudio is loaded.

See

Palette
, for more details on loading palettes into ImageStudio.

1.186 ImageStudio.guide/Tooltype_PORTNAME

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.

1.187 ImageStudio.guide/Tooltype_REXXOUTPUT

REXXOUTPUT

Name

REXXOUTPUT

Type

String

Description

The name of the filename to use for ARexx scripts' output. See

Basic ARexx
, for more information on this file.

1.188 ImageStudio.guide/Tooltype_SCREENNAME

SCREENNAME

Name

SCREENNAME

Type

String

Description

The name of ImageStudio's public screen. This name must be unique, otherwise the screen won't open.

1.189 ImageStudio.guide/Tooltype_SCRIPTSDIR

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 SCRIPTSEXT) will be placed in the list.

1.190 ImageStudio.guide/Tooltype_SCRIPTSEXT

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.

1.191 ImageStudio.guide/Tooltype_TEMPDIR

TEMPDIR

Name

TEMPDIR

Type

String

Description

The directory in which ImageStudio can put its virtual memory temporary 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.

1.192 ImageStudio.guide/Tooltype_TEXTEDITOR

TEXTEDITOR

Name

TEXTEDITOR

Type

String

Description

The text editor used to edit ARexx scripts from the 'Scripts' floating 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.

1.193 ImageStudio.guide/Tooltype_CLIPUNIT

CLIPUNIT

Name

CLIPUNIT

Type

Numeric

Description

The system clipboard used to copy (see Copy) and paste (see Paste)

by the program. There is very little need to change this from the default value of 0.

1.194 ImageStudio.guide/Tooltype_PAGESIZE

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 TEMPDIR).

At most, two of these pages will be allocated in RAM at any one time.

1.195 ImageStudio.guide/Tooltype_UNDOBUFFERS

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).

1.196 ImageStudio.guide/Tooltype_APPICON

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.

1.197 ImageStudio.guide/Tooltype_HELP

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.

See

Known bugs
, for more information on a known problem with
AmigaGuide help.

1.198 ImageStudio.guide/Tooltype_PREVIEWDITHER

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.

1.199 ImageStudio.guide/Tooltype_PREVIEWREDRAW

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

PREF_SET
) to speed up some processing.

1.200 ImageStudio.guide/Tooltype_SPLASHWINDOW

SPLASHWINDOW

Name

SPLASHWINDOW

Type

Boolean

Description

Turns on or off the opening of the 'About' window when the program starts up.

1.201 ImageStudio.guide/Tooltype_SCREENFRONT

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.

1.202 ImageStudio.guide/Known bugs

Known bugs

=====

Known bugs:

- * There seems to be a bug in the AmigaGuide library which causes memory to be lost (typically a few hundred bytes) when an AmigaGuide document is opened. If this really bothers you, turn AmigaGuide help off (see
HELP
).
- * At the moment ImageStudio doesn't handle IFF-ILBM files with an interleaved stencil. Make sure when saving from an paint package that the stencil is turned off.
- * 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

Common ARexx problems

).

- * There is a bug in the AGA graphics hardware which causes some screens to be displayed corrupted. This either manifests itself as a fast flickering (usually with small screens) or not displaying the extreme edges of the screen (usually with large screens). This doesn't happen very often and we're powerless to do anything about it. It doesn't corrupt any data, it's simply annoying.
- * 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.

1.203 ImageStudio.guide/Problems with datatypes

Problems with datatypes

=====

If ImageStudio is running on a Workbench 2.1+ Amiga it will utilise the built in datatypes for image conversion; the user should be made aware of differences between the datatypes and the built-in ImageStudio loaders.

The datatypes are only used as a "last resort" if the ImageStudio loaders fail to recognise the incoming image. Datatypes have the following disadvantages:

- * They do not utilize virtual memory. Therefore when loading a large image, large amounts of RAM are required.
- * They can be slow. The datatype first has to convert the image to IFF-ILBM, which ImageStudio has to read in and convert to its own internal format.
- * Datatypes convert the image to a colour-mapped image. 24-bit image formats (e.g. JPEG) therefore are reduced in quality.
- * Some datatypes we have tried are fundamentally bugged - crashing with odd sized images or different file layouts. ImageStudio has no control over the quality of the installed datatypes.

Datatypes though can be useful to load in image formats not yet supported by ImageStudio, and are therefore made available to the user.

1.204 ImageStudio.guide/Common questions

Common questions

If you have any questions about ImageStudio, make sure that it hasn't already been answered below:

Common question 1

Why doesn't ImageStudio support TIFF?

Common question 2

Will you be adding SHAM, PCHG ...

Common question 3

Can I turn virtual memory off?

Common question 4

Why is there not a colour preview window?

Common question 5

What other programs have the authors written?

Common question 6

How much disk space to the swap files use?

Common question 7

What is the .dvi and .ps documentation?

Common question 8

Will you be adding support for animations?

Common question 9

Can you write a general virtual memory program?

Common question 10

Why don't you use MUI for your windows design?

1.205 ImageStudio.guide/Common question 1

Common question 1

=====

"Why doesn't ImageStudio support TIFF?"

TIFF is a powerful and flexible image format, but that is also its downfall. TIFF supports so many compression algorithms, for example, that it would require a very large amount of code to deal with even a modest range of TIFF files.

TIFF will probably get added later when the loader / savers become external modules.

1.206 ImageStudio.guide/Common question 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 when the loaders / savers become external modules.

1.207 ImageStudio.guide/Common question 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
TEMPDIR
).

The overhead of using virtual memory from RAM is negligible.

1.208 ImageStudio.guide/Common question 4

Common question 4

=====

"Why is there not a colour preview window?"

Even with a relatively small amount of greyscales, a good approximation of the image can be obtained (balance, contrast, etc...). With a small number of colours, a poor representation of the image is the result. A greyscale preview is also faster.

1.209 ImageStudio.guide/Common question 5

Common question 5

=====

"What other programs have the authors written?"

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.

1.210 ImageStudio.guide/Common question 6

Common question 6

=====

"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

1.211 ImageStudio.guide/Common question 7

Common question 7

=====

"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 by sending them directly to the printer.

1.212 ImageStudio.guide/Common question 8

Common question 8

=====

"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.

1.213 ImageStudio.guide/Common question 9

Common question 9

=====

"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 instructions 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.

1.214 ImageStudio.guide/Common question 10

Common question 10

=====

"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 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 operating system, ImageStudio would need to be re-written - not something we have any wish to do.

1.215 ImageStudio.guide/The authors

The authors

ImageStudio was written by Andy Dean and Graham Dean.

Queries and orders (see
How to register
) should be sent to Graham at:

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

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
Future additions
) let us
know!.
2. Other work. Graham is studying 'A' levels and Andy is doing a PhD and this work will take priority (sad, but true).

If you find a bug in ImageStudio that is not covered in the 'Known bugs' list (see

Known bugs
) , inform the authors at the above addresses.

Be sure to include as much information as possible, the version of ImageStudio being used, a description of the Amiga system you are running (model, amount of RAM, Workbench version, any expansion cards).

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.

1.216 ImageStudio.guide/How to register

How to register

To receive the full version of ImageStudio, send 10 pounds sterling (20 US dollars overseas) to:

Graham Dean,
14 Fielding Avenue,
Poynton,
Stockport,
Cheshire.
SK12 1YX
ENGLAND

We will accept the following methods of payment:

- * 10UK pounds cash.
- * A 10UK pounds cheque, drawn on a UK bank.
- * A 10UK pounds postal order, purchased in the UK.
- * 20US dollars cash.
- * International money order.

We don't accept any foreign cheques drawn on non-UK banks and we don't accept any foreign postal orders. We also cannot accept Eurocheques for any value (USDollars or UKpounds).

Note: Make sure that when sending cash, it is well wrapped in the envelope.

In return you will receive the latest version of ImageStudio, 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 reasonable 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 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 lots of registered users all wanting upgrades at the same time!).

The version number of ImageStudio (see
About
) 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).

1.217 ImageStudio.guide/Credits

Credits

The authors would like to thank:

- * Commodore-Amiga.
- * David Cusick, Don Cox and Julie Brandon for beta testing ImageStudio.
- * Our parents, for their support (especially our mum for also helping with the posting and packing!!!).
- * 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 design.
- * Michael Balzer, for 'ARexxBox' - used to implement the ARexx port.
- * 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.
- * The Independant JPEG Group, for their essential JPEG code and information.
- * All those involved with the excellent TeX and 'TeXinfo' packages.

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, 4MByte FAST RAM, Power PC1204 expansion 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.

1.218 ImageStudio.guide/Future additions

Future additions

The following features will probably be added to future versions of the packages (roughly in order):

- * Making the loader / savers external modules.
 - * "Clear unused buffers" option.
 - * Making the effects external modules.
 - * Write a separate program used to build ARexx batch processing scripts by simply clicking on the actions you wish to perform.
 - * ARexx macro record.
 - * Spare buffer.
 - * Alpha channel buffer.
 - * Halftoning operators.
 - * Image rotation.
 - * Aspect ratio correction in the preview window.
-