Image Engineer

Contents

Chapter 1

Image Engineer

1.1 Image Engineer Documentation

```
                    Image Engineer V1.1

                            by

                       Simon Edwards

                 - Freely Distributable -

             Copyright © 1995 by Simon Edwards
                    All rights reserved

             This program uses reqtools.library
                    by Nico François
```

1.2 intro

```
Introduction
~~~~~~~~~~~~

Image Engineer is brand new freely distributalbe image processing
application.  Image Engineer can for tasks varying from converting images
between different fileformats, rendering 24 bits down to standard Amiga
screen modes and enhancing baddly scanned images.  What you can use it for
is basicly limited by what you can think of.

Features
~~~~~~~~

* Coded in 100% 68020 Assembler
* Multiple image editing
* All processing is done in 8 bit grey or 24 bit colour
* Full ARexx support, including macros.
* Uses superview.library for loading and saving. superview.library currently
  supports IFF ILBM, IFF ACBM, PCX, SVO, GIF, BMP, FBM, C64 (Koala, Doodle),
  IMG, TIFF, Targa, WPG, SunRaster, Pictor, MacPaint, JPEG.
* Can display images using superview.library. superview.library currently
  supports EGS and OpalVision (more in the works).
* Supports AGA where available.
* Can render and dither image to the standard Amiga screen modes, as well as
  HAM6 and HAM8.
* Image Composition
* User Convolves
* Image balance control, brightness, contrast, gamma.
* Extensive filtering control.
* It’s Free! (The money you save on software you can spend on hardware ;-)
```

1.3 copyright

```
Copyrights
~~~~~~~~~~

Image Engineer is © 1995 Simon Edwards. All rights reserved.

ReqTools is © Nico François. All rights reserved.

superview.library is © 1993-94 by Andreas R. Kleinert.  All rights reserved.
```

1.4 disclaimer

```
Disclaimer
~~~~~~~~~~

The author neither assumes nor accepts any responsibility for the use or
misuse of this software.  He also will not be held liable for damages or any
compensation due to loss of profit or any other damages arising out of the
use, or inability to use this software.
```

Have a nice day. :)

1.5 distribution

Distribution
~~~~~~~~~~~~~

Image Engineer V1.1 is freely distributable. It is *not* public domain, as
all copyright remains with the author. This means that you may copy and
distribute Image Engineer provided the following conditions are met.

* All parts of the distribution are included in an unmodified form.

* No profit is made beyond minimal copy and posting fees.

* Image Engineer may be included in public domain software libraries like
  the excellent Fred Fish collection. (Fred, if that's you reading this,
  please, please, please, include Image Engineer in your collection.)

* Image Engineer may *not* be commericially distributed without the author's
  written permission.

By using or distributing this software you automatically agree to the above
terms.

1.6 motivation

Motivation (or "Why did I bother?")
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

About half way through last year I read the book "Digital Image Processing"
by Gonzalez & Woods. After that I wanted to try out some of the techniques
and algorithms (I was also looking for some sort of programming project to
do) so I sat down with the intention of writing a small program to do some
basic image processing on 8 bit grey scale images. I also had the idea of
releasing something as freely distributable planted in my mind. So once I
had my little program up and running (even though it couldn't do much), my
summer break (December-Febuary, for the benefit of those in the northern
hemisphere) was coming up. It was about now that I decided to write a full
blown 24 bit colour image processing program, I figured that would keep me
busy over the long months stuck at home in the middle of nowhere. The rest
is pretty well history.

1.7 system

```
System Requirements
~~~~~~~~~~~~~~~~~~~

* 68020 or higher processor.
* OS V2.04+, Image Engineer was developed under OS V3.0 and is therefore
  fully compatable.
* 2Mb of RAM bare minimum, at least 4Mb recommended, infinity is best.
* Reqtools.library to be installed.
  (available on aminet:/util/libs/ReqTools22_us.lha)
* superview.library to be installed.
  (available on aminet:/gfx/show/svlib???U.lha, where ??? is the name of
  the current version, I can't keep these docs up to date with what the
  current version is ;-)
```

1.8 starting

```
Installation & Starting Up
~~~~~~~~~~~~~~~~~~~~~~~~~~

The file IE.config (found in the same directory as IE) should be copied to
your S: directory.  Simply click on the icon "Click to Copy IE.config" and
it will be copied to S:.  (If you're upgrading from V1.0, your old IE.config
will be destroyed and your will have set up your Prefs again, sorry!)

Image Engineer (IE) can be started by clicking on its icon from the
workbench or entering "IE" from the Shell.  IE currently takes no arguments
from the Shell.  When starting up IE will try and load it's Prefs from
S:IE.config.  It will then put up a screen mode requester allowing you to
choose the screen mode and dimensions that you want IE to operate in.  (IE
can be made to open its screen without asking, see Prefs.  Once it has
finished opening it's screen you'll be left looking at a very unexciting
screen.  The Project menu will the only menu currently available.

If you're not familiar with IE I suggest that you work through (or at least
skim through) the tutorials section to acquaint yourself with how IE does
things.  Once you've gone through the tutorials, to learn more I recommend
that you simply play and experiment.  Don't be intimidated by some of the
technical sounding menu items. ;)
```

1.9 tutorials

```
Tutorials
~~~~~~~~~

#1 Loading, Rendering and Saving an Image
#2 Scaling and Locking an image to a palette
#3 Filtering and Removing Noise from an Image
#4 Applying a Vigette effect using an Alpha channel
```

1.10 tutorial1

```
Loading, Rendering and Saving an Image
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Loading, rendering and saving are the three operations that you'll be doing
with Image Engineer more than anything else.

1. IE destingushes between two types of images, grey (8 bit) and colour
   (24 bit) images.  Images loaded as 8 bit grey are automaticly converted
   and stored in memory using 8 bits per pixel.  Images loaded as 24 bit
   colour are automaticly converted and stored in memory using 24 bits per
   pixel, 8 bits for each colour component (Red, green, blue).  It pays to
   make sure that you load grey images as 8 bit, as they will take up less
   memory and all operations on them will be faster.

   What images are treated as when loading is set on the Open submenu on
   the Project menu.

  +----------------+
  |File...         |
  |Clipboard       |
  +----------------+
  |   8 bit Grey   | <-- These two options control whether images are
  |   24 bit Colour| <-- loaded as 8 bit grey or 24 bit colour.
  +----------------+

   The image we're going to use is in colour, so now is the time to set
   IE to load images as 24 bit colour.  Just select "24 bit colour" so
   that it now has a tick beside it.

2. Select the "File..." menu item on the Open submenu.  The file requester
   will now appear.  Select the file molecules.iff which should be in
   the Pics directory.  IE will now identify, load, and convert the image
   to 24 bit colour.  A window will now be opened on IE's screen and a grey
   preview image will be drawn.

   This window is referred to as the project window, and the image is
   referred to as a project.  The title bar of the project window
   shows the name of the project (each project has a unique name), followed
   by the x and y co-ordinates that the mouse pointer is currently over
   followed by the grey level at that point for grey images, or the Red Green
   Blue (RGB) value at that point for colour images.

   Note: The co-ords are only shown if that window is the active one. In
   IE when a project window is opened it is *not* made the active one,
   unlike most programs.  The active window's title bar is generally darker
   than an inactive window's.

3. Each project has its own set of information associated with it that
   describes how it should be rendered.

   Select the "Render Control..." menu item from the Screen menu.  The
   render control requester will now appear.  From here you can everything
   about how this project should be rendered.  What screenmode to use, how
   many colours to use, how to choose the palette, what dithering to use
```

etc. For this tutorial we want to render it down to 32 colours Low-Res using Jarvis dithering. To do this, click on the "Device:" gadget till it displays "Amiga", now click on the "Choose..." button, a screen mode requester will come up, select a Low Res screenmode, (like "NTSC:Low Res" for example) then click on "Ok" to return to the render control requester. Move the "Colours" slider so that it says 32. Set the "Colours to use:" slider to 32 also. And finally, click on the "Quantize:" gadget so that it says "Median Cut" and the "Dither" gadget so that says "Jarvis". When you're finished hit the "Ok" button.

Now all of the render options have been set up. Thankfully you can set what default values should be used for new grey and colour projects, so that you don't have set up the render options to something intelligent everytime you load in a new image.

4. Select "Render" from the screen menu. This tells IE to actually render the image. An indicator entitled "Rendering..." will appear. When it says that it's half way through rendering, a new screen will open and IE will draw the image.

   This screen is referred to as the project's render screen.

   If you press the right mouse button, IE's screen will reappear. The project's render screen has not been closed, just moved behind all of the other screens (like the WorkBench screen and IE's screen). To bring it to the front of the display again, select "Show Render" from the Screen menu.

5. Once a project has been rendered it can them be saved to disk.

   Select "Render" from the save submenu on the project menu. A file requester will appear, allowing you to enter the file name to save the rendered image as, use molecules32col.iff. Next is the Save Format Requester, this is where you select what save format you would like the image saved as. Select "ILBM CmpByteRun1", this the standard Amiga image format, used by DPaint, Brilliance, and every other Amiga graphics program. Click on "Ok" once you've made your selection. IE will now save the image.

Well, that's that. If you don't want to render the image down to a displayable screen mode, but instead wanted to save the 8 bit grey or 24 bit colour image data. You would use the "8 bit grey/24 bit colour" on the Save submenu. This basicly works the same way as the save render item, except that there is no need to render the image first.

To close the render screen use "Close Render" on the Screen menu.

1.11 tutorial2

Scaling and Locking an image to a palette
~~~~~~

In this tutorial will be how to scale an image down to the size of an icon and then render it using the WorkBench palette.
~~~~~~

1. Open up the picture molecules.iff as 24 bit colour.

2. Make the molecules.iff project window the active one. (Just click on the window's title, it's the active one if title has a darker background.) Now go to the Edit menu and select the Scale item. The scale requester will now appear. Click on the "Lock Aspect" check box (so that it shows a tick). The two sliders control the percentage scale for each dimension. Now drag the slider beneath the "Width:" gadget all the way to the left. The slider towards the bottom of the requester will also move to stay at the same level as the one above, in order to preserve the aspect ratio. Click on the "Method:" gadget once, it should now say "Colour Average". This means that the Colour Average method will be used to scale the image, this is slower than the fast method (also known as "Nearest Neighbour"), but produces better results in most cases. Now click on "Ok" to let it go to work.

3. Go to the "Render Control" requester for the small image and set the Device to "Amiga" and select a High Res screen mode. Set the "Colours" and "Colours to use" to 4. Set "Quantize" to "Lock Palette Best", this forces to IE to use this project's current palette when rendering. Specify that you want "Floyd-Steinberg" dithering. Just click on the dither gadget until it says "Floyd-Steinberg" (should be one click after "None"). Hit "Ok" when finished.

4. Before we can render the image we need to load in the palette that we want it locked to. Choose the menu item "Load..." on the palette submenu under the screen menu. A file requester will let you choose the palette file you want to load. Load the file WB.palette which should be in the palettes directory.

5. Render the image.

   Save the rendered image as molecules.icon.iff.

The saved image could now be imported into an icon editor (like IconEdit supplied with Workbench).

1.12 tutorial3

Filtering and Removing Noise from an Image
~~~~~~

In this due I'll demonstrate how Image Engineer's filtering functions can be used to remove noise from a corrupt image.

1. Open up the picture NoiseU2.iff from the Pics directory as 8 bit grey.

2. As you can see the image has been corrupted such that it looks like U2 are in a snow storm. Go to the "Filter" menu and select "3x3" from the "Lowpass" submenu. This blurs the image. The resulting image now looks worse than what we had at the beginning. There is a better way.

3. Close the blurry project (just click on the project window's close gadget). Go to the "Filter" menu and select "3x3" from the "Median"
~~~~~~

submenu. This applies a 3x3 sized median filter to the image. Notice how the filter has removed about all of the noise instead of just blurring it.

4. Make the filtered active and then select "View Histogram" from the process menu. This shows the histogram of the image. Notice how grey levels are clustered towards the dark left end of the graph and that it doesn’t use the full range. Click on the Histogram window’s close gadget.

5. Select "Contrast Stretch" from the "Process" menu. This will increase the contrast of the image. Now go back to the histogram. The histogram will now be stretched to cover the whole range.

Now you could render and save the enhanced image to disk.
When trying to enhance, remove noise and bring out detail in an image, it pays to try several approaches and then seeing which one gave the best result, as some images respond well to some filters, while others are simply degraded more. There are no hard rules.

1.13 tutorial4

Applying a Vigette effect using an Alpha channel
~~~~~~

Have you ever noticed that effect that photographers sometimes use that makes images appear blurred around the edges. This tutorial shows you how the same effect can be created using IE.

1. Open up the picture face.iff as 24 bit colour.

2. Apply a 7x7 lowpass (blurring) filter to the image. To do this just select "7x7" on the Lowpass submenu under the Filter menu. After a delay the new image will be displayed.

3. Open the picture Spherical2.alpha which should be in the Alpha directory, as 8 bit grey.

4. Make the blurred image active and select info from the Edit menu. Now take note of the width and height of the blurred image.

5. Go to the Spherical image and scale it to the same size as the blurred image.

6. Now it’s time to composite the images. We’re going to use the original face image as the primary image, the blurred image as the secondary image, and the scaled grey image as the alpha channel. First make the original image active and select "Primary" from the "Composite" menu. There should be a ’P’ just before the image’s title on the project window. Now go to the blurred image and select "Secondary" from the "Composite" menu. This time there should be a ’S’ before it’s title. Finally, go to the scaled grey image and select "Alpha" from the "Composite" menu. There will be a ’A’ in front of it’s title. By doing all of this you will have specified which images are going to be used for what when we go to create the composite image.
~~~~~~

7. Select "Composite..." from the "Composite" menu. The "Composition Control" requester should now pop up. We want to combine the images using the Alpha channel, so click on the radio button to the left of the words "Alpha Channel". Now click on "Ok". It will go to work and produce the composite image.

8. You might want to render the image now to get a good look at it. Notice how the image is blurred around the edges, since the alpha channel was dark around the edges it meant that more of the blurred image should be used in those regions. While at the parts of the image that correspond to the light parts in the alpha channel were mostly like the orignal image.

By using alpha channels to combine images, it’s possible to create 1000’s of interseting effects that would otherwise not be possible. (Instead of blurring the image, try converting it to grey, false colour etc.)

1.14 menus

```
Menus
~~~~~

Project Edit Screen Process Filter Composite ARexx
```

1.15 projectmenu

```
Project
About...
-------------------
Open             »
Save             »
---------------
Prefs            »
Screen Mode...<A>M
-------------------
Quit          <A>Q
```

1.16 about

```
About
~~~~~
```

This justs brings up a small requester giving some information like the name of the program, it’s version, who the author is etc.

1.17 open

```
Open Submenu
~~~~~~~~~~~~

+---------------------+
|File...          <A>O|
|Clipboard        <A>V|
+---------------------+
|_/ 8 bit Grey        |         "_/" is meant to be a checkmark. BTW.
|   24 bit Colour     |
+---------------------+
```

The "File..." item lets you open an image from disk. A file requester will appear, letting you choose the file to open. Image Engineer will automatically identify the image format, load it and convert it to either 8 bit grey or 24 bit colour.

The "Clipboard" item opens an image from the system clipboard if it currently contains graphics.

"8 bit Grey" and "24 bit Colour" determine whether images will be loaded, converted and stored internally as either 8 bit grey or 24 bit full colour images. The check mark indicates what images will currently be loaded as.

Note: When IE is loading an image it may appear that the loading indicator is broken. This is normal. Unfortunately, due to technical constrains it's currently impossible to make it work "properly". Sorry.

ARexx Equivalent: OPEN, OPEN_CLIPBOARD, TYPE

1.18 save

```
Save Submenu
~~~~~~~~~~~~

+-------------------------+
|Render...            <A>S|
|8 bit/24 bit data...     |
+-------------------------+
```

If the current project has been rendered, "Render..." with let you save it to disk. You'll be presented with a file requester from where you can enter the file name to save the project as. Next a small requester entitled "Select Save File Type" will let you select the file format that you want the project saved as. (If you're unsure as to what file format you should use, use "ILBM CmpByteRun1", the standard Amiga image format).

Note: If the rendered image is in HAM or HAM8, only use ILBM. As most other formats do not support the saving of Amiga HAM modes, even if they do appear to work.

The "8 bit/24 bit data..." item let's you save a project's image data directly to disk with out having to render it first. The procedure is the same as saving normal rendered images.

Note: Not all formats will support the writing of 8 bit or 24 bit data. (ie the C64 format Doodle doesn’t support 24 bit images, (surprise, surprise) :)

1.19 prefs

```
Preferences submenu
~~~~~~~~~~~~~~~~~~~

Prefs... <A>?
ARexx...
```

1.20 prefs2

```
Preferences
~~~~~~~~~~~

This lets you setup defaults for Image Engineer.  The options are as
follows.

Default Grey Render Prefs:-
        Clicking on the "Choose..." button on the right will bring up a
        "Render Control" requester where you set what the default render
        options to use for 8 bit grey images.  When IE opens an image
        as 8 bit grey, these defaults will be used.  This basicly saves
        you the hassle of having to set up the render options for every
        grey image that you open.

Default Colour Render Prefs:-
        This is the same as above except for 24 bit colour images.

Select Screen Mode at start up:-
        If this option is checked then IE will put up a screen mode
        requester at start up for you to choose the what screen mode you
        would like it to operate in.  If this option is not checked then
        IE will open its screen using the current screen mode.

Default Load Type:-
        This sets which of "8 bit Grey" or "24 bit Colour" on the
        Project/Open menu will be set initially after start up.

Convolves:-
        This sets the default directory for the file requester will start in
        on the Convolve requester.

Palettes:-
        This sets the default directory for the Palette Load/Save requester
        on the Screen/Palette menu.

Images:-
```

```
        This sets the default directory for the Open/Save image requester.
```

Clicking on "Ok" will use the new Prefs without saving them to disk.
Clicking on "Save" will save the Prefs to disk (in file "S:DIP.config") and use them. "Cancel" will close the Preferences requester without changing the Prefs.

1.21 arexxprefs

ARexx Preferences
~~~~~~~~~~~~~~~~~

This lets you set up the ARexx user menu. The requester looks vaguely like this:-

```
+---------------------------------------------------------+
|ARexx Preferences                                      []|
+---------------------------------------------------------+
|           +--------------------------------++---------+ |
|  Scripts: |                                ||Choose...| |
|           +--------------------------------++---------+ |
| +---------------------------+-+ +---------+ +---------+ |
| |Macro1                  F1 |#| |   Up    | |  Down   | |
| |Macro2                  F2 |#| +---------+ +---------+ |
| |~~~~~~~~~~~~~~~~~~~~~~~ F3 | |                         |
| |<Blank>                 F4 | | [] Blank    [] Bar      |
| |<Blank>                 F5 | |                         |
| |<Blank>                 F6 | |       +---------------+ |
| |<Blank>                 F7 +-+ Name: |Macro1         | |
| |<Blank>                 F8 +-+       +---------------+ |
| +---------------------------+-+                         |
|                                                         |
|           +--------------------------------++---------+ |
|  Command: |                                ||Choose...| |
|           +--------------------------------++---------+ |
|                                                         |
| +---------+          +---------+            +---------+ |
| |   Ok    |          |  Save   |            | Cancel  | |
| +---------+          +---------+            +---------+ |
+---------------------------------------------------------+
```

The Scripts gadget at the top of the requester lets you enter the directory where your ARexx scripts for Image Engineer are stored. By clicking on the "Choose..." gadget lets you select the Scripts directory via a standard directory requester. The Scripts directory is used as the initial directory for the fle requester used by the Execute ARexx Script command.

The scrolling list to the left side of the requester contains 30 'slots' that can be used for a ARexx script or a bar separator. Each slot has a function key short cut (sometimes in combination with the shift or alt key). Blank slots don't show up on the user menu.

To modify the contents of a given slot, select it in scrolling list. The gadgets will be updated to show the contents of the selected slot. The "Blank" check box, toggles the slot between being Blank and in use. A slot
~~~~~~~~~~~~~~~~~

that's not blank can be toggled between a bar separator and an ARexx script, by clicking on the "Bar" checkbox.

The "Name" gadget lets you edit the text that will appear on the menu. The "Command" gadget lets you enter the path and filename of the ARexx script that should be executed when this menu item is selected. The "Choose..." gadget to the right of the "Command" gadget lets you select the ARexx script to use via a file requester.

The "Up" and "Down" buttons simply let you move the current menu item up and down in the list.

Clicking the "Save" button will the save the changes to disk, while the "Ok" will use the changes without saving to disk. Click on "Cancel" to cancel all changes.

1.22 screenmode

Screen Mode
~~~~~~~~~~~

This lets you change the screen mode and dimensions of IE's screen. IE's screen will be closed and a screen mode requester will appear on the Workbench screen. If you change your mind and select Cancel, IE will reopen it's screen using the previous screen mode.

1.23 quit

Quit
~~~~

This closes all render screens and projects and exits the program completely. It does *not* ask if you're sure, and does *not* check to see if you have unsaved work. Make sure that you've saved all of your work before using this menu item. You will not get a 2nd chance. So make sure you mean it.

ARexx Equivalent: QUIT

1.24 editmenu

```
Edit
Info...         <A>I
History...      <A>H
Project Name...
Close Project
---------------------
Enter co-ords...<A>E
Cut             <A>X
Autocrop        <A>A
---------------------
```

```
Reflect X
Reflect Y
---------------------
Resize...
Scale...
```

1.25 projectinfo

```
Project Info
~~~~~~~~~~~~

This displays a requester detailing the Width and Height of the image, the
project’s name and the name of the file that the image was loaded from
originally.
```

1.26 projecthistory

```
Project History
~~~~~~~~~~~~~~~

Each project carries with it information about which file (or clipboard) it
came from and what operations have been carried out on it.  This is useful
if like me you have a 10 second memory and can’t remember what your up to
(or if your just having a play around, discover some great effect that you
can’t remember how you did it).
```

1.27 projectname

```
Project Name
~~~~~~~~~~~~

This lets you edit the name of a project to something more meaningful.
Project names mustn’t contain spaces, if it does the spaces will be changed
to underscores. Project names must also be unique, if it’s not unique a
number will be prepended to it.
```

1.28 closeproject

```
Close Project
~~~~~~~~~~~~~
This closes a project and removes it from memory completely.  Like the
"Quit" menu item is doesn’t check to see if the project has been saved
or not.  Does the same as thing as clicking on a project’s window’s close
gadget.

ARexx Equivalent: CLOSE
```

1.29 entercoords

```
Enter Co-ordinates for the Crop Box
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

This lets you enter co-ords for a project's crop box, instead of having to
dragging one out in the project's window.  The X and Y gadgets specify
where the top left corner of the box is.  The Width and Height gadgets let
you enter the width and height of the box (surprise, surprise).  Initial
values for the X, Y, width and height are taken from where the current
crop box is.  "Ok" accepts the new values, and "Cancel" returns you to
where you were before you opened for requester.
```

1.30 cut

```
Cut or Crop
~~~~~~~~~~~

This cuts out the boxed part of the current project to create a new
project.  Before you can use this menu item, the current project needs to
have a crop box marked out.  You can do this by just holding down the
left mouse button and dragging out a box in the project's window, or by
entering co-ords by hand.  The new project will have the same render
options as the project it was cut from.

ARexx Equivalent: CROP
```

1.31 autocrop

```
Autocrop
~~~~~~~~
This crops out all of the background surrounding an image.

                 ___________
                |           |                      _____
                |    ****   |                     | ****|
                |   *       |                     |*    |
       ie.      |    ***    | would become...     | *** |
                |       *   |                     |    *|
                |       *   |                     |    *|
                |   ****    |                     |**** |
                |           |                      ~~~~~
                 ~~~~~~~~~~~

ARexx Equivalent: AUTOCROP
```

1.32 reflectx

```
Reflect X
~~~~~~~~~

This effectively "flips" an image along the y axis (left-right).

ARexx Equivalent: REFLECT_X
```

1.33 reflecty

```
Reflect Y
~~~~~~~~~

This effectively "flips" an image along the x axis (up-down).

ARexx Equivalent: REFLECT_Y
```

1.34 resize

```
Resize
~~~~~~

This lets you change the size of an image by padding it out with black (ie
by *not* scaling it).  The resize requester will appear allowing you to
set the new size of the image and where the current image should be
positioned in the new one.  The "Width" and "Height" gadget and sliders
allow you to specify the dimensions of the new image.  The "X Offset" and
"Y Offset" gadget and slider control where the top left hand corner of the
current image will be positioned with respect to the new image.  The
diagram in the recessed box to the left of the sliders, shows
diagramatically the two images relative to each other with the current
image in black the new image size as an outline.  The "Centre" button
centres the current image over the new image.  The "Even X" and "Even Y"
buttons round the Width and Height of the new image up to the next
multiple of 8.

Resize is particularly useful if you’re trying to import an image into a
program that’s very picky about what size images it will take.

ARexx Equivalent: RESIZE
```

1.35 scale

```
Scale
~~~~~

This lets you scale an image to a new size.  The "Width" and "% Width"
gadgets let you enter the new width or new percentage width.  The slider
below the width gadgets lets you change the percentage width from 25% to
200% (although greater values can entered into the "% Width" gadget.
```

```
Below is the "Height" and "% Height" gadgets and corresponding slider.
The "Lock Aspect" checkbox forces the Width and Height to be scaled by
the same amount.  This preserves the aspect ratio and stops images from
becoming stretched or shrinked too much in any direction.  The "Method"
gadget controls whether the image should be scaled using a "Fast"
algorithm or the better (but slower) "Colour Average" algorithm.  The
"Colour Average" algorithm helps prevent images from becoming too
"blocky" when scaled up.

ARexx Equivalent: SCALE
```

1.36 screenmenu

```
Screen
Render           <A>R
Render Control...<A>P
Palette             »
----------------------
Show Render      <A>F
Close Render     <A>C
Save to Clipboard
```

1.37 render

```
Render
~~~~~~

This renders the project using the settings defined on the Render Control
requester.

ARexx Equivalent: RENDER
```

1.38 rendercontrol

```
Render Control
~~~~~~~~~~~~~~
This allows you to set up how you want the given project rendered.
Choosing this menu item brings up the "Render Control" requester.

+-------------------------------------------------+
|Render Control                                 []|
+-------------------------------------------------+
|         +-+--------------------------------+ |
| Device:  |@|            Amiga               | |
|         +-+--------------------------------+ |
|                  Screen Mode                    |
| +------------------------------------------+ |
| |DBLPAL:High Res No Flicker                | |
| +------------------------------------------+ |
|               __________________  +----------+ |
```

```
| Colours: 256 [________________**] |Choose...| |
|                                   +---------+ |
| [] Autoscroll                                 |
|           +-+--------------------------------+ |
| Overscan: |@|          Text Size             | |
|           +-+--------------------------------+ |
|                                               |
|                      _______________________  |
| Colours to Use: 256 [_____________________**] |
|           +-+--------------------------------+ |
| Quantize: |@|          Median Cut            | |
|           +-+--------------------------------+ |
|           +-+--------------------------------+ |
|   Dither: |@|            None                | |
|           +-+--------------------------------+ |
|                                               |
| +--------+                       +--------+ |
| |   Ok   |                       | Cancel | |
| +--------+                       +--------+ |
+-----------------------------------------------+
```

```
Device:-
        The "Device" cycle gadget at the top of the requester, determines
        how the image should be rendered.  It may be one of the following.
            Amiga    - Renders the image using the standard Amiga colour
                       mapped screen modes.
            SVDriver - Renders the image using the default SVDriver.  This
                       allows you to render images using 24bit graphics
                       boards provided you have a SVDriver that supports
                       it.  (See the superview.library documentation for
                       more inforamtion about SVDrivers).
            HAM8     - Renders the image using Amiga HAM8 mode (where
                       available).  This is only available for 24 bit
                       colour images.
            HAM6     - Renders the image using Amiga HAM6 mode.

Screen Mode:-
        Below the "Device" gadget is a gadget showing the name of the
        currently selected render screen mode.  To select a new screen
        mode click on the "Choose" button, this will bring up a screen
        mode requester from where you can choose a new mode.

Colours:-
        The "Colours" sliders determines how many colours the render
        screen should have (not to be confused with the number of colours
        that should be used).  This slider is only relevent when using the
        Amiga as the display device

Autoscroll:-
        The Autoscroll check box determines whether the render screen
        should scroll when the mouse reaches the edge of the screen if it
        bigger than the display.

Colours to Use:-
        The "Colours to Use" slider determines how many colours should be
        used to render the image.
```

```
Quantize:-
        The "Quantize" cycle gadget specifies what palette should be used.
        For 24 bit colour images, the Quantize gadget may be one of the
        following.
            Median Cut        - The Median Cut algorithm is used to choose
                                a palette for the image.
            Lock Palette Fast - The image's palette is used.  This is
                                option renders the image quickly, but is a
                                bit inaccurate (5 bits per colour
                                component as opposed to 8 bits per colour
                                component.  This is only noticable if your
                                palette consists on a many similar
                                shades).
            Lock Palette Best - The image's palette is used.  This is
                                slower than the fast lock to palette but
                                is 100% accurate.
        For grey images you have a choice of two.
            Lock Palette - Locks to current palette.
            Best Palette - Chooses the best palette.

Dither:-
        The "Dither" cycle gadget lets you specify a dithering algorithm
        to be used when rendering the image.  The dithering algorithms
        available in order of complexity are Floyd-Steinberg, Burkes,
        Stucki, Sierra, Jarvis and Stevenson-Arce.  Floyd-Steinberg is
        generally good for most things, the effectiveness of each is a
        subjective thing, I recommend that you try all of them and see
        what you think of each.

The "Ok" button accepts the changes, while the "Cancel" button forgets the
whole thing.

ARexx Equivalent: GET_RENDER,RENDER_AUTOSCROLL, RENDER_COLOURS,
 RENDER_DEPTH, RENDER_DEVICE, RENDER_DITHER,
 RENDER_QUANTIZE, RENDER_SCREENMODE, SET_RENDER
```

1.39 palette

```
Palette Submenu
~~~~~~~~~~~~~~~

Edit...
Load...
Save...
```

1.40 editpalette

```
Edit Palette
~~~~~~~~~~~~

The "Edit..." menu options opens up a palette requester.  (Unfortunately
it is only available on AGA Amigas, as it needs to open a 256 colour
```

screen). The palette requester shows the palette at the top of the
screen, below this are the following gadgets.

```
+----------------------------------------+
| +-----------+    ____________________  |
| |           | R [_______________*____] |
| +------+----+    ____________________  |
| Colour |  0 | G [_______________*____] |
| +-+----+----+    ____________________  |
| |@| RGB     | B [_______________*____] |
| +-+---------+                          |
|                                        |
| +----------+ +----------+ +----------+ |
| |   Copy   | |   Swap   | |  Spread  | |
| +----------+ +----------+ +----------+ |
| +----------+ +----------+ +----------+ |
| | Restore  | |  Load... | |  Save... | |
| +----------+ +----------+ +----------+ |
| +----------+              +----------+ |
| |    Ok    |              |  Cancel  | |
| +----------+              +----------+ |
+----------------------------------------+
```

In the top left hand corner the current colour is shown. Below that is
it's pen number. The cycle gadget below that lets you choose which colour
model you would like the sliders to be, out of a choice of standard Red,
Green, Blue (RGB) colour model, the Cyan, Magenta, Yellow (CMY) model, and
Hue, Saturation, Brightness (HSB) model. The rest of the buttons work in
the same way as most palette requesters. The "Copy" button allows you to
copy the current colour. "Swap" lets you exchange two colours. "Spread"
will create a smooth colour range. "Restore" restores the palette to the
state that it was in before any changes were made. "Load.." lets you load
in a palette from disk. (BTW, you can also load in a palette from any IFF
ILBM picture, not just palette files). "Save..." saves the curent palette
to file on disk.

ARexx Equivalent: LOAD_PALETTE, SAVE_PALETTE

1.41 loadpalette

Load Palette
~~~~~~~~~~~~

This lets you load a palette in from disk. (Same as the "Load..." button on
the palette requester).

ARexx Equivalent: LOAD_PALETTE

1.42 savepalette

Save Palette
~~~~~~~~~~~~

```
This lets you save the project’s palette to disk.  (Same as the "Save..."
button on the palette requester).

ARexx Equivalent: SAVE_PALETTE
```

1.43 showrender

```
Show Render
~~~~~~~~~~~
This merely brings a project’s render screen to the front of the display.

ARexx Equivalent: RENDER_TO_FRONT
```

1.44 closerender

```
Close Render
~~~~~~~~~~~~
This closes a project’s render screen.  To open the screen again it has to
be rendered again.

ARexx Equivalent: CLOSE_RENDER
```

1.45 savetoclip

```
Save to Clipboard
~~~~~~~~~~~~~~~~~

This saves the project’s render screen to the clipboard.

ARexx Equivalent: SAVE_CLIP
```

1.46 processmenu

```
Process
View Histogram         »
Brightness...
Contrast...
Gamma...
-------------------------
Negative
Contrast Stretch
Histogram Equalization
Threshold...
Transform...
-------------------------
Convert to Colour       /Convert to Grey
```

False Colour
Local Contrast Stretch »

1.47 viewhistogram

View Histogram
~~~~~~~~~~~~~~

This displays the image’s histogram, along with the number of unique values in the histogram, the mode value (ie the most common value), what the lowest value is and what the highest value is. For colour projects you can view the intensity histogram for the image or the histogram for each of the three colour components (Red, Green, Blue).

1.48 histograms

What’s a Histogram anyway?
~~~~~~~~~~~~~~~~~~~~~~~~~~

A histogram is just a graph showing the relative frequency of each grey level in an image. It typically looks something like this.

```
 |      *
 |      **
 |     ***
 |    ****
 |   ******
 |   ********               *
 | *************           ***
 |****************      ********   ***
 +--------------------------------------
 0       64       128      196        255
```

The grey levels are in the range of 0 to 255 inclusive. The axis along the botton is the grey value axis. The y axis shows the relative frequency. The higher the column graph for a given grey level, the greater frequency it has in the image relative to the other grey levels (In the graph above we can see the most frequent grey level is approximately 60, meaning that the grey level of 60 occurs more often than any other in the image).

In colour images each colour is converted to a grey value and that value is used to form the graph. It’s also possible to create the histogram using the values of a single component (ie one of red, green or blue).

What’s the good in it?
~~~~~~~~~~~~~~~~~~~~~~

Although it says nothing about the content of an image, the histogram does give important information about the global characteristics of an image. Information from the histogram comes in very useful when adjusting the brightness and contrast of an image, and when affecting the overall
~~~~~~~~~~~~~~~~~~~~~~

balance of an image.

```
 |   *
 |   **
 |   **
 |   ***
 |  *****
 | *******
 |**********
 |*************
 +--------------------------------------
 0       64       128      196        255
```

The histogram above shows that the grey values in this image are clustered
towards the lower end. This corresponds to a dark image. While the
histogram below corresponds to a bright image.

```
 |                                   *
 |                                  **
 |                                  **
 |                                 ***
 |                                *****
 |                              *******
 |                            **********
 |                         *************
 +--------------------------------------
 0       64       128      196        255
```

Since the grey levels are clustered together in the histogram below, it
therefore corresponds to an image with low contrast. This image would
appear a murky grey.

```
 |                 *
 |                 *
 |                **
 |                ***
 |                ****
 |               *****
 |               ******
 |              ********
 +--------------------------------------
 0       64       128      196        255
```

The histogram below corresponds to an iamge with high contrast, as the grey
levels are well spread.

```
 |   *                        *
 |   *              *        **
 |  **           ** *     *  **     *
 |  ***  *      *** **   *** ***   ***
 |***** **      ******* ********  *****
 |*********   *************************
 |*********  ***************************
 |**************************************
 +--------------------------------------
 0       64       128      196        255
```

```
The histogram below corresponds to an image of a white object with a black
background.  In this case the histogram would be useful in choosing a
threshold to use to turn the image into a real black and white image.
(BTW, you would choose the threshold to be at about 150).

 |     *
 |     *                          *
 |    ***                        **
 |   ****                        **
 |   *****                      ***
 |   *******                    ****
 |  *********                  ******
 | ************              **********
 +-------------------------------------
 0       64       128      196        255
```

1.49 function

```
Function Graphs
~~~~~~~~~~~~~~~
These graphs show how an input value is "mapped" on to an output value.

 O 255|       /
 u    |      /
 t    |     /
 p    |    /
 u    |   /
 t    |  /
      | /
      |/
   0  +-------
      0     255
        Input

The input values are along the x axis while the output values are along
the y axis.  In the graph above, each input value  maps onto the same
output value (ie it has no effect).

Below is a graph showing a function for reducing the brightness of an image.

 255|
    |
    |      /
    |     /
    |    /
    |   /
    |__/
   0+-------
    0      255

To find out what an input value of 200 would become you would read it of the
graph like so.

       255|
          |
          |      /
```

```
Output 128|______/
          |     /|
          |    / |
          |__/   |
       0  +------+-
          0      |255
                 |
            Input 200
```

Here an input value of 200 would map to an output 128.

1.50 brightness

Brightness
~~~~~~~~~~

This lets you alter the brightness of an image. The brightness requester looks something like this (if you use your imagination a bit).

```
+-------------------------------------------------+
|Change Brightness                              []|
+-------------------------------------------------+
|+-------------------------++-------------------+|
||                         ||Unique Values: 250||
||   *                     ||Mode          :  25||
||   *                     ||Lowest Value :   0||
||   *[New Histogram]      ||Highest Value: 250||
||   *                     |+-------------------+|
|| ***                     |      +--------+     |
|| ****            *       |      |       /|     |
||*****      **    **      |      |      / |     |
||*****     ***** ***      |      |     /  |     |
||******   ***********     |      |[Graph]|     |
||******* **************   |      |   /    |     |
||***********************  |      |  /     |     |
||***********************  |      |/       |     |
|+-------------------------+      +--------+     |
|0     64    128    196   255                    |
|                  ______________________________ |
| Brightness:  0 [______________*_______________]|
|                                                 |
| [] Intensity       [] Red   [] Green [] Blue   |
|                                                 |
|+---------+                       +---------+|
||   Ok    |                       | Cancel  ||
|+---------+                       +---------+|
+-------------------------------------------------+
```

The Histogram shows what the new histogram will look like, along with some qualitative information about it to the top right. The slider controls the number of grey levels that the brightness should be changed by. The graph shows the function that is used to produce the new image.
Click on "Ok" to make the change, "Cancel" cancels the whole requester.

The Intensity, Red, Green and Blue checkboxes control whether the
~~~~~~~~~~

brightness of the intensity component should be affected (what you would normally use), or if the brightness of the RGB components should be affected individually. This ability is useful in situations where you have to correct an image (perhaps a scan) that has one (or more) of its RGB comonents too bright with respect to the others components.

ARexx Equivalent: BRIGHTNESS

1.51 contrast

Contrast
~~~~~~~~

This lets you alter the contrast of an image. The contrast requester looks a bit like this.

```
+------------------------------------------------+
|Change Contrast                               []|
+------------------------------------------------+
|+-------------------------++-------------------+|
||                         ||Unique Values: 250||
||  *                      ||Mode         :  25||
||  *                      ||Lowest Value :   0||
||  *[New Histogram]       ||Highest Value: 250||
||  *                      |+-------------------+|
|| ***                     |      +-------+      |
|| ****             *      |      |      /|      |
||*****      **     **     |      |     / |      |
||*****     ***** ***      |      |    /  |      |
||******   ***********     |      |[Graph]|      |
||******* **************   |      |  /    |      |
||**********************  |      | /     |      |
||**********************  |      |/      |      |
|+-------------------------+      +-------+      |
|0     64    128    196   255                    |
|                  ______________________________ |
| Contrast:    0 [______________*______________]|
|                                                |
| [] Intensity       [] Red   [] Green [] Blue  |
|                                                |
|+---------+                      +---------+|
||   Ok    |                      | Cancel  ||
|+---------+                      +---------+|
+------------------------------------------------+
```

The Histogram shows what the new histogram will look like, along with some qualitative information about it to the top right. The slider controls how much the contrast should be increased on decreased. The graph shows the function that will currently be used to produce the new image.
Click on "Ok" to make the change, "Cancel" cancels the requester.

The Intensity, Red, Green and Blue checkboxes control whether the contrast of the intensity component should be affected (what you would normally use), or if the contrast of the RGB components should be affected individually.
~~~~~~~~

ARexx Equivalent: CONTRAST

1.52 gamma

Gamma
~~~~~

This lets you alter the gamma of an image. The gamma requester looks a
like this.

```
+------------------------------------------------+
|Change Gamma                                  []|
+------------------------------------------------+
|+-------------------------++------------------+|
||                         ||Unique Values: 250||
||   *                     ||Mode          :  25||
||   *                     ||Lowest Value :   0||
||   *[New Histogram]      ||Highest Value: 250||
||   *                     |+------------------+|
|| ***                     |      +-------+     |
|| ****             *      |      |      /|     |
||*****      **     **     |      |     / |     |
||*****     ***** ***      |      |    /  |     |
||******   ***********     |      |[Graph]|     |
||******* *************    |      |  /    |     |
||**********************   |      | /     |     |
||**********************   |      |/      |     |
|+-------------------------+      +-------+     |
|0     64    128    196   255                   |
|                  _____________________________ |
| Contrast:    0 [______________*______________]|
|                                               |
|                    [] Red   [] Green [] Blue  |
|                                               |
|+---------+                       +---------+|
||   Ok    |                       | Cancel  ||
|+---------+                       +---------+|
+------------------------------------------------+
```

The Histogram shows what the new histogram will look like, along with some
qualitative information about it to the top right. The slider controls
how the gamma content should be affected. The graph shows the function
that will currently be applied to create the new image. Click on "Ok" to
make the change, "Cancel" cancels the requester.

Increasing the gamma content of an image has the effect of darkening the
mid-grey values. While decreasing the gamma does the opposite. The main
use for changing the gamma (or Gamma Correction as it is known) is to
correct effects caused by the non-linear way in which monitors respond to
inputs.

The best way to see this is open the image Gradient.alpha which is in
the Alpha directory. Now render it using as many shades of grey as
possible. Notice how the perceived intensity decreases slowly from left
~~~~~

to right until it gets to the right where it drops of quickly to black. Now apply gamma correction of +47. Render the new image. Notice how the intensity decreases more uniformly across the image.

ARexx Equivalent: GAMMA

1.53 negative

Negative
~~~~~~~~

This merely takes the negative of an image.

ARexx Equivalent: NEGATIVE

1.54 contraststretch

Contrast Stretch
~~~~~~~~~~~~~~~~

This increases an image’s contrast so that is uses the full dynamic range available. This generally improves the appearence of an image and helps bring out detail.

ARexx Equivalent: CONTRAST_STRETCH

1.55 histogramequalization

Histogram Equalization
~~~~~~~~~~~~~~~~~~~~~~

This tries to change an image so that each grey level is represents an equal number of pixels in the image. Or to put it in other words, it tries to make the histogram of the image flat. Due the discrete nature of the whole operation the resulting histogram will not be flat, but only an approximation. This has the effect of increasing the contrast of an image and brings out detail.

ARexx Equivalent: HISTOGRAM_EQUALIZATION

1.56 threshold

Threshold
~~~~~~~~~

This allows you to turn a grey project into black and white by applying a threshold. The threshold requester looks like the histogram requester, by clicking on the histogram graph you can a position a vertical (usually white) line which determines the threshold. The exact position is given on the right hand side as "Threshold:". Clicking on "Ok" applies the threshold, everything that's below will be come black, else it'll become white.

ARexx Equivalent: THRESHOLD

1.57 transform

Transform
~~~~~~~~~

This allows you to apply a function to the intensity of an image. The transform requester looks like this.

```
+-------------------------------+
|Transform                    []|
+-------------------------------+
|255 +------------+ +---------+ |
|    |           /| | Initial | |
|    |          / | +---------+ |
|    |         /  | | Negate  | |
|    |        /   | +---------+ |
|    |       /    | |Monotonic| |
|    |      /     | +---------+ |
|128 |     /      | | Smooth  | |
|    |    /       | +---------+ |
|    |   /        | | + Bright| |
|    |  /         | +---------+ |
|    | /          | | - Bright| |
|    |/           | +---------+ |
|  0 +------------+             |
|    0    128    255            |
|                               |
| +------+             +------+ |
| |  Ok  |             |Cancel| |
| +------+             +------+ |
+-------------------------------+
```

The graph shows the function that will be used, with the input values along the bottom and the out put values along the left. The shape of the function can be editted by using the left mouse button to draw straight onto the graph.

The buttons do the following.

Initial:-
        Restores the function to the initial straight line.

Negate:-
        Negates the function (turns it upside down, in other words).

Monotonic:-
        This fixes the function so that each value is greater than or equal
~~~~~~~~~

```
        to the one to its left.

Smooth:-
        This smoothes the function.  (Useful if you drew part of the function
        in freehand, and you want to smooth some of your mistakes a bit).

+ Bright:-
        This increases the brightness (effectively moves the graph up a bit).

- Bright:-
        This decreases the brightness (effectively moves the graph down a
        bit).

As usual, "Ok" applies the function and "Cancel" forgets the whole thing.

One of the main uses for this feature is for creating a black and white image
using only range of greys.
```

1.58 converttocolour

```
Convert to Colour
~~~~~~~~~~~~~~~~~

This converts a grey image to a colour image.  Each pixel will be replaced
with the corresponding colour from the project's palette. (ie a pixel with
grey level 25 will be replaced with the colour number 25 in the project's
palette).

ARexx Equivalent: CONVERT_TO_COLOUR
```

1.59 converttogrey

```
Convert to Grey
~~~~~~~~~~~~~~~

This mearly converts an image to grey.

ARexx Equivalent: CONVERT_TO_GREY
```

1.60 falsecolour

```
False Colour
~~~~~~~~~~~~

This applies a False Colour effect to an image.  If the image is grey then
the resulting project will be colour.

ARexx Equivalent: FALSE_COLOUR
```

1.61 localcontraststretch

```
Local Contrast Stretch
~~~~~~~~~~~~~~~~~~~~~~

This applies a Local Contrast Stretch to an image.  This is like a normal
contrast stretch except that instead of using the whole image to calculate
what the new pixel value should be, only a small neighbourhood around the
pixel is used.  This effectively brings out local detail.

ARexx Equivalent: LOCAL_CONTRAST_STRETCH
```

1.62 filtermenu

```
Filter
Convolve...
------------
Lowpass   »
Highpass  »
Highboost »
Sharpen   »
------------
Median    »
```

1.63 convolve

```
Convolve
~~~~~~~~

The convolve requester looks like this.

+-------------------------------------------------------------+
|Convolve                                                   []|
+-------------------------------------------------------------+
|      +---------------------------------------------------+ |
| Name:|Untitled                                           | |
|      +---------------------------------------------------+ |
|                                                            |
|             Convolve Matrix:                               |
|                                                            |
|  +------+ +------+ +------+ +------+ +------+    +------------+  |
|  |     0| |     0| |     0| |     0| |     0|    |    New     |  |
|  +------+ +------+ +------+ +------+ +------+    +------------+  |
|  +------+ +------+ +------+ +------+ +------+    +------------+  |
|  |     0| |     0| |     0| |     0| |     0|    |  Load...   |  |
|  +------+ +------+ +------+ +------+ +------+    +------------+  |
|  +------+ +------+ +------+ +------+ +------+    +------------+  |
|  |     0| |     0| |     0| |     0| |     0|    |  Save...   |  |
|  +------+ +------+ +------+ +------+ +------+    +------------+  |
|  +------+ +------+ +------+ +------+ +------+           +------+ |
|  |     0| |     0| |     0| |     0| |     0| Divisor: |     1| |
|  +------+ +------+ +------+ +------+ +------+           + ----+ |
```

```
|  +-----+ +-----+ +-----+ +-----+ +-----+          +-----+ |
|  |    0| |    0| |    0| |    0| |    0|    Bias: |    0| |
|  +-----+ +-----+ +-----+ +-----+ +-----+          + ----+ |
|                                                           |
| +-----------+                               +-----------+ |
| |    Ok     |                               |  Cancel   | |
| +-----------+                               +-----------+ |
+-----------------------------------------------------------+
```

The name gadget holds the name given to this convolve. The number gadgets below the "Convolve Matrix:" hold the values of the matrix elements. (BTW, to get to the next one you don't have to click in the next one with the mouse, you can just press tab). The divisor and bias gadgets hold the values for divisor an bias respectively (funny that).

The "New" button will clear the matrix and the divisor and bias values. The "Load..." button opens a filerequester, allowing you load in a convolve from disk. While the "Save..." button will let you save the current convolve to disk. Finally "Ok" applies the convolve, while "Cancel" cancels the entire operation.

ARexx Equivalent: CONVOLVE

1.64 convolvehelp

What the smeg is a convolve?
~~~~~~~~~~~~~~~~~~~~~~~~~~~~

A convolve is just a matrix of elements, a number for the divisor, and one for the bias.

That's great, what do you do with it?
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

A convolve works like this. To calculate the new value of each pixel, the matrix is centered over the source pixel and each matrix element is multiplied by the value beneath it, the values summed then divided by the divisor and finally the bias is added.

For example.

If the convolve matrix looks like this.

```
  0 -1  0
 -1  4 -1
  0 -1  0
```

Divisor = 8
Bias = 0

Applying it to pixel e in the diagram below.

```
 a b c
 d e f
 g h i
```

The new value for pixel e would be.

```
(0xa + -1xb + 0xc + -1xd + 4xe + -1xf + 0xg + -1xh + 0xi)/Divisor +Bias
```

Simple, eh? Convolves can be used to create 100s of different effects.

Examples of convolves
~~~~~~~~~~~~~~~~~~~~~

A 3x3 lowpass filter mearly averages the pixel values in a 3x3
neighbourhood, therefore it’s convolve matrix looks like this.

```
 1 1 1
 1 1 1
 1 1 1
```

Divisor = 9
Bias = 0

A blur that’s not as heavy as a 3x3 lowpass filter would be done using
a convolve like this.

```
 0 1 0
 1 4 1
 0 1 0
```

Divisor = 8
Bias = 0

1.65 lowpass

LowPass (aka Blur)
~~~~~~~~~~~~~~~~~~

This applies a lowpass filter to an image. A lowpass filter removes the
high frequencies (edges, sharp intensity transitions) in an image, thus
effectively blurring it. This very useful for removing noise from an
image.

ARexx Equivalent: LOWPASS

1.66 lowpasshelp

How exactly does it work?
~~~~~~~~~~~~~~~~~~~~~~~~~

The values for each new pixel is calculated by averaging the values of its
surrounding neighbours. So for a 3x3 lowpass filter the new value of each
pixel is just the average of the nine surrounding neighbours.

```
 a b c
```
~~~~~~~~~~~~~~~~~~~~~~~~~

```
 d e f
 g h i
```

In this diagram the new value of pixel e will be (a+b+c+d+e+f+g+h+i)/9.

The larger the filter the "heavier" and more blurred it is.

Also, by using a long thin filter, you can sort of fake a motion blur.

1.67 highpass

```
HighPass
~~~~~~~~
```

This applies a highpass filter to an image. The highpass filter removes the low frequencies in an image, leaving the high ones that correspond to edges and sharp intensity transitions etc.

Since we can have negative transitions in an image the result from the highpass filter will be in the range of [-255,255] and therefore has to be converted back into the range [0,255]. There are four different ways that can be used to "normalize" it.

```
Absolute      - The absolute value is used.  In the result the edges will
                be surrounded by a double white line.
Scale         - It's scaled into the range [0,255].  The result doesn't
                look entirely unlike some sort emboss effect.
Clip Negative - All negative values simply become zero.
Clip Positive - All positive values become zero and the negative values
                are negated.
```

ARexx Equivalent: HIGHPASS

1.68 highboost

```
HighBoost
~~~~~~~~~
```

This is effectively the same as a highpass filter except that a fraction of the original image is added back in to the highpass result.

This can be useful in that the resulting image looks more like the original except that the high frequencies have been highlighted.

The percentage refers to how much of the original to add back in.

ARexx Equivalent: HIGHBOOST

1.69 sharpen

```
Sharpen
~~~~~~~

This sharpens a blurred image.  This is extremely useful for bring out
detail in a bad blurry image.  The larger the percentage used, the
greater the effect.  (BTW, the sharpened image is calculated by taking
the original and adding a fraction of the highpass result to it.  The
percentage actually refers to the fraction of the highpass result to
add in.)

I find that this used in conjuction with a lowpass filter is very good at
removing high frequency noise, by first applying a small (3x3) sized
lowpass filter to the image and then sharpening it to bring out the detail
again.

ARexx Equivalent: SHARPEN
```

1.70 median

```
Median Filter
~~~~~~~~~~~~~

This filter is extremely good at removing noise and preserving edges in
an image.  It works by replacing each pixel in an image with the median
value of the its neighbouring pixels.

ARexx Equivalent: MEDIAN
```

1.71 compositemenu

```
Composite
Composite...
-------------
  Primary
  Secondary
  Alpha
-------------
  LMB Drag
```

1.72 composite

```
Composite
~~~~~~~~~

This simply allows you to create composite images.  Before this menu item
can be used you need to mark an image for use the the primary image and
one for the secondary image.  The Composite requester looks vaguely like
this.
```

```
+-----------------------------------------------------------------------+
|Composition Control                                                  []|
+-----------------------------------------------------------------------+
|+------------------+                     +-----+           +-----+     |
||                  |    Primary   Width: |  320|   Height: |  256|     |
||  +--------+      |                     +-----+           +-----+     |
||  |        |      |                     +-----+           +-----+     |
||  |  ***********  |    Secondary Width: |  200|   Height: |  200|     |
||  |  ***********  |                     +-----+           +-----+     |
||  |  ***********  |            +------+  ___________________________  |
||  |  ***********  |  X Offset: |    50| [_________________*_________] |
||  +--***********  |            +------+                               |
||     ***********  |            +------+  ___________________________  |
||     ***********  |  Y Offset: |    50| [_________________*_________] |
||     ***********  |            +------+                               |
||                  |               +---+  ___________________________  |
||                  |  [@] Mix %    | 50| [_____________*_____________] |
||                  |               +---+                               |
||                  |               [] Black is transparent             |
||                  |                                                   |
|+------------------+  [ ] Alpha Channel                                |
|                                                                       |
| +------------+                                        +------------+ |
| |    Ok      |                                        |  Cancel    | |
| +------------+                                        +------------+ |
+-----------------------------------------------------------------------+
```

The "Primary Width/Height" and "Secondary Width/Height" boxes to the top right simply show the dimensions of the Primary and Secondary images. The diagram to the left of the requester, shows graphically how the two images are positioned relative to each other, with the black rectangle being the primary image and the outline being the secondary image. The "X Offset" and "Y Offset" gadget and sliders allow you to adjust where the primary image is placed with respect to the secondary image.

There are two ways in which images may be combined, one is simply to mix (or blend) them together using a given percentage of the primary image. Below the mix slider is a checkbox marked as "Black is transparent". This allows you have all blacks in the primary image to be treated as transparent (in much the same way as a genlock). This is very useful if you're trying to place a title over a background image.

The other method is to combine them using an image as a alpha channel. An alpha channel is simply an image that is used to determine how the primary and secondary image should be combined for each pixel. If a pixel in the alpha channel is 100% white, then it means that the pixel that it corresponds to should be 100% of the primary image and 0% of the secondary. If the pixel in the alpha channel was black (ie 0% white) then it means that 0% of the primary and 100% of the secondary should be used. While a value of 50% white in the alpha channel means that the corresponding pixel should be made up of 50% primary and 50% secondary.

Which of the two different methods that is used, is determined by which radio button is selected.

As always, hitting the "Ok" button will set it into operation, while cancel will forget the whole thing.

ARexx Equivalent: COMPOSITE

1.73 markas

```
Mark as Primary, Secondary or Alpha
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Before the "Composite..." function can be used, you need to specify which
images are to be used as the secondary, primary and sometimes alpha.  To
mark an image for use as the primary, secondary or alpha, you simply check
these menu items.  A marked image will have anyone of the letters P, S & A
in front of its title depending on what it has been marked as.  The
letters simply indicate that the image has been marked as Primary,
Secondary or Alpha.

ARexx Equivalent: MARK
```

1.74 lmbdrag

```
Left Mouse Button Drag
~~~~~~~~~~~~~~~~~~~~~~

This allows you to use the LMB to drag an image from it's window and place
it on the image it should be composited with, in much the same way that
you manipulate icons on the Workbench.  Just drag the image you want to use
as the primary image, onto the image you want to use as the secondary image
and release the LMB, making sure that the pointer is over the secondary
image.  The dragged image will then be marked as the Primary, image that
you released the LMB over will be marked as the Secondary image and
"Composition Control" requester will be invoked.  This facility is very
useful when you want to carefully position an image when compositing.
```

1.75 arexxmenu

```
ARexx
Execute...
-------------
Macro1  F1
Macro2  F2
...     F3
```

1.76 execute

```
Execute ARexx script
~~~~~~~~~~~~~~~~~~~~
```

This menu item lets you select an ARexx script from disk to execute. A filerequester will appear letting you select a script.

1.77 arexxuser

ARexx User Menu
~~~~~~~~~~~~~~~

The rest of the ARexx menu contains configurable menu items that allow you to execute ARexx scripts by selecting a menu item or by pressing the corresponding key combination. These menu items can be configured using the ARexx preferences requester, which is on menu Project/Prefs/ARexx.

1.78 arexx

ARexx
~~~~~

Image Engineer's has an ARexx port called IMAGEENGINEER. Most menu items have a corresponding ARexx command. You'll find that function and input descritions are usually quite breif, read the corresponding menu description for a more indepth view of how a particular operation works and what the inputs are.

Script Writing Tips <- Please Read
Supplied Scripts
Command List

1.79 scripttips

Script Writing Tips
~~~~~~~~~~~~~~~~~~~

* Options Results should be turned on to get results from commands. (Use the line "Options Results" at the start of your scripts).

* Image Engineer's commands return 0 in RC on success, 5 when the user aborts or cancels a command (by clicking on the "Abort" button or selecting cancel if it's a requester). If the commands fails due to bad parameters, out of memory etc 10 is returned. Use the command LAST_ERROR to get a string describing the error.

* ARexx scripts can be invoked from inside by using the "Execute..." menu item on the ARexx menu, or by choosing a user defined menu item from the ARexx menu. Scripts started this way automatically have thier command host set to IMAGEENGINEER. They're also called with argument one containing the name of the currently active project, or nothing if the script was invoked with no project active (ie before the user has even opened an image).
~~~~~~~~~~~~~~~~~~~

```
  It’s worth checking to see if your script has been passed a project name
  to operate on, with a line like this:-

  if arg()==0 then
    do
      /* If we need a project to operate on we could now */
      /* open a file requester and let the user choose an */
      /* image from disk, or put up a message saying that we */
      /* need a project, or we could just quietly exit now. */
    end

* Be aware of how ARexx interprets quotes in command clauses.
  For example.

        CLOSE ’bono.iff’ & ’CLOSE bono.iff’

  will send the string

        CLOSE bono.iff

  to IE, as ARexx interpretes the outside quotes.  This is important to
  keep in mind when you’re trying to send a command with a file name that
  contains a space.

  For example.

        ’OPEN "sys:pics/bono pic"’

  which would send the string

        OPEN "sys:pics/bono pic"

  to IE.

  If you’ve got a file name stored in a string (that you may have requested
  from the user) and you want to use it with a command, you should make
  sure that the name is enclosed in quotes when it’s send to IE, as you
  can’t be sure that is contains no spaces.
  For example.

  MyFile=’sys:pics/bono pic’

  ’OPEN "’||MyFile||’"’

  this would send

  OPEN "sys:pics/bono pic"

  to IE.  Doing it like this

  ’OPEN ’||MyFile

  would not work if the file name contains a space.

* IE also has several special commands for use in ARexx scripts.

  Screen Commands:-
  IE_TO_FRONT
  WB_TO_FRONT

  Error Command:-
  LAST_ERROR

  User Input Commands:-
  GET_DIR
  GET_FILE
  GET_FILES
  GET_FILE_TYPE
  GET_NUMBER
  GET_PERCENT
  GET_STRING
  REQUEST

  Task Priority Commands:-
  GET_PRI
  SET_PRI
```

1.80 suppliedscripts

```
Supplied Scripts & Macros
~~~~~~~~~~~~~~~~~~~~~~~~~

Note: For the example scripts to find the files they need to work on, IE:
needs to be assigned to the directory containing IE and the Convolves and
Alpha directories.  Add the following line to your user-startup.

assign IE: <path to IE directory goes here>

To start any of the scripts RexxMast needs to be running as well as Image
Engineer.

The following scripts and macros can be found in the IE's ARexx directory
and can be invoked by using the Execute menu item on the ARexx menu.

batch_convert.rexx
  This takes a list of files, loads each file, renders it to a given
  screenmode and then saves the rendered image in a given file format.
  By telling it to render in the background, it's possible to go do
  other work while it loads, renders and saves the images at reduced
  priority.

FadeInAnim.rexx
  This script takes an image of a title over a black background and
  composites it over another image, fading it in over a given number
  of frames.  By compiling the resulting frames you can create an
  animation of your title fading in.

FitAlpha.rexx
  This loads and scales the image Spherical.alpha to the same
  dimensions as the current image, then marks it for use as an
  alpha channel.  The new alpha channel image can be used to
  seemlessly composite the orignal onto another image.
```

```
FitAlpha2.rexx
  Same as above, except it uses Spherical2.alpha.

FitSelectAlpha.rexx
  Sames as above except it lets you select the image to scale for use
  as an alpha channel.

MakeIcon.rexx
  This takes the current image, scales it, renders it down to 4 Colours
  using the Workbench palette then saves the result to the Clipboard,
  ready to paste into a program like IconEdit.

MakeIconMWB.rexx
  Same as above except it renders to 8 colours using the Magic workbench
  palette.

Scale50.rexx{ub}
  Scales an image by 50%.

Scale200.rexx
  Scales an image by 200%.

Vigette.rexx
  This applies a blurred Vigette effect to an image, making it appear
  blurred around the edges.

VigetteGrey.rexx
  This is the same as above except the image appears in grey around
  the edges.

WetInk.rexx
  This takes a black on white image, and makes it appears that the
  black ink has run, by using a convolve.
```

1.81 commandlist

```
ARexx Commands
~~~~~~~~~~~~~~

AUTOCROP
BRIGHTNESS
CLOSE
CLOSE_RENDER
COMPOSITE
CONTRAST
CONTRAST_STRETCH
CONVERT_TO_COLOUR
CONVERT_TO_GREY
CONVOLVE
CROP
FALSE_COLOUR
GAMMA
GET_DIR
GET_FILE
```

```
GET_FILES
GET_FILE_TYPE
GET_NUMBER
GET_PERCENT
GET_PRI
GET_RENDER
GET_STRING
HIGHBOOST
HIGHPASS
HIGHPASS_NORMALIZE
HISTOGRAM_EQUALIZATION
IE_TO_FRONT
LAST_ERROR
LOAD_PALETTE
LOCAL_CONTRAST_STRETCH
LOWPASS
MARK
MEDIAN
NEGATIVE
OPEN
OPEN_CLIPBOARD
PROJECT_INFO
QUIT
RENDER
RENDER_AUTOSCROLL
RENDER_COLOURS
RENDER_DEPTH
RENDER_DEVICE
RENDER_DITHER
RENDER_QUANTIZE
RENDER_SCREENMODE
RENDER_TO_FRONT
REFLECT_X
REFLECT_Y
REQUEST
RESIZE
SAVE
SAVE_CLIP
SAVE_DATA
SAVE_PALETTE
SCALE
SET_PRI
SET_RENDER
SHARPEN
THRESHOLD
TYPE
WB_TO_FRONT
```

1.82 autocroprexx

```
Autocrop
~~~~~~~~

Synopsis:
  AUTOCROP <ProjectName>
```

```
Function:
  Crop out all of the background surrounding an image.

Inputs:
  ProjectName - Name of the project to autocrop.

Result:
  Returns the name of the newly created project in RESULT.

Example:
  options results
  ...
  MyProject='bono.iff'
  ...
  AUTOCROP MyProject    /* Autocrop MyProject */
  say 'Autocropped project name is' RESULT

Menu equivalent:
  Edit/Autocrop


See also:
```

1.83 brightnessrexx

```
Brightness
~~~~~~~~~~

Synopsis:
  BRIGHTNESS <ProjectName> <Value> [INTENSITY | [RED] [GREEN] [BLUE] ]

Function:
  Change the brightness of an image.

Inputs:
  ProjectName - Name of the project.
        Value   - Value to change brightness by.
  INTENSITY - Specifies that the intensity component should be
        affected (default).
  RED   - Specifies that the red colour component should be
        affected.
  GREEN   - Specifies that the green colour component should be
        affected.
  BLUE    - Specifies that the blue colour component should be
        affected.

Result:
  Returns the name of the newly created project in RESULT.

Example:

  BRIGHTNESS MyProject 50   /* Increase brightness by 50 */
  BRIGHTNESS MyProject 50 INTENSITY /* as above */
  ...
```

```
  BRIGHTNESS MyProject -50 GREEN  /* Reduce the brightness of the */
          /* green component by 50 */
  ...
  BRIGHTNESS MyProject 10 RED BLUE  /* Slightly increase the */
      /* brightness of the red and blue components */


Menu equivalent:
  Process/Brightness

See also:
```

1.84 closerexx

```
Close
~~~~~

Synopsis:
  CLOSE <ProjectName>

Function:
  Close a project.  Same as clicking on the project's close box.

Inputs:
  ProjectName - Name of the project.

Result:
  None.

Menu equivalent:
  Edit/Close Project

See also:
```

1.85 closerenderrexx

```
Close Render
~~~~~~~~~~~~

Synopsis:
  CLOSE_RENDER <ProjectName>

Function:
  Close a project's render screen.

Inputs:
  ProjectName - Name of the project.

Result:
  None.

Menu equivalent:
```

```
  Edit/Close Render

See also:
```

1.86 compositerexx

```
Composite
~~~~~~~~~

Synopsis:
  COMPOSITE <X Offset> <Y Offset> <<MIX <%> [GENLOCK]>|ALPHA>

Function:
  Create a composite image from the marked Primary and Secondary images
  by mixing the images or using an alpha to combine them.

Inputs:
        ProjectName - Name of the project.
        X Offset    - X Offset of the Primary image with respect to the
                Secondary image.
        Y Offset    - Y Offset of the Primary image with respect to the
                      Secondary image.
        MIX         - This keyword specifies that the images should be mixed
                together at <%> percent.
        %           - Percentage of the Primary image to use when mixing.
        GENLOCK     - This keyword specifies that all black in the primary
                image should be made transparent.
        ALPHA       - This keyword specifies that the two images should be
                      combined using an alpha channel.

Result:
  Returns the name of the newly created project in RESULT.

Example:
        /* Mix the BackgroungProject with the LogoProject using 50% mix */
        /* with black being transparent */
        MARK BackgroundProject SECONDARY
        MARK LogoProject PRIMARY
        COMPOSITE 0 0 MIX 50 GENLOCK
        ...

        /* Combine the BackgroundProject with LogoProject using */
        /* LogoAlphaProject as an alpha channel */
        MARK BackgroundProject SECONDARY
        MARK LogoProject PRIMARY
  MARK LogoAlphaProject ALPHA
        COMPOSITE 0 0 ALPHA
        ...

Menu equivalent:
  Composite/Composite

See also:
  MARK
```

1.87 contrastrexx

```
Contrast
~~~~~~~~

Synopsis:
  CONTRAST <ProjectName> <Value> [INTENSITY | [RED] [GREEN] [BLUE] ]

Function:
  Change the contrast of an image.

Inputs:
  ProjectName - Name of the project.
        Value   - Value to change contrast by.
  INTENSITY - Specifies that the intensity component should be
        affected (default).
  RED   - Specifies that the red colour component should be
        affected.
  GREEN   - Specifies that the green colour component should be
        affected.
  BLUE    - Specifies that the blue colour component should be
        affected.

Result:
  Returns the name of the newly created project in RESULT.

Examples:

  CONTRAST MyProject 50   /* Increase contrast */
  CONTRAST MyProject 50 INTENSITY /* same as above */
  ...

  CONTRAST MyProject -50 GREEN  /* Reduce the contrast of the */
  ...       /* green compent by 50 */

  CONTRAST MyProject 10 RED BLUE  /* Slightly increase the */
      /* contrast of the red and blue components */

Menu equivalent:
  Process/Contrast

See also:
```

1.88 contraststretchrexx

```
Contrast Stretch
~~~~~~~~~~~~~~~~

Synopsis:
  CONTRAST_STRETCH <ProjectName>

Function:
  Contrast stretch an image.
```

```
Inputs:
  ProjectName - Name of the project.

Result:
  Returns the name of the newly created project in RESULT.

Menu equivalent:
  Process/Contrast Stretch

See also:
```

1.89 converttocolourrexx

```
Convert to Colour
~~~~~~~~~~~~~~~~~

Synopsis:
  CONVERT_TO_COLOUR <ProjectName>

Function:
  Convert a 8 bit grey image to 24 bit colour.

Inputs:
  ProjectName - Name of the project.

Result:
  Returns the name of the newly created project in RESULT.


Menu equivalent:
  Process/Convert to Colour

See also:
```

1.90 converttogreyrexx

```
Convert to Grey
~~~~~~~~~~~~~~~

Synopsis:
  CONVERT_TO_GREY <ProjectName>

Function:
        Convert a 24 bit colour image to 8 bit grey

Inputs:
  ProjectName - Name of the project.

Result:
  Returns the name of the newly created project in RESULT.
```

```
Menu equivalent:
  Process/Convert to Grey

See also:
```

1.91 convolverexx

```
Convolve
~~~~~~~~
Synopsis:
        CONVOLVE <ProjectName> <ConvolveFilename>

Function:
        Apply a convolve to an image.

Inputs:
        ProjectName      - Name of the project.
        ConvolveFileName - Path and filename of the convolve to use.

Result:
  Returns the name of the newly created project in RESULT.

Example:
  options results
  ...
  MyProject='bono.iff'
  ...
  CONVOLVE MyProject "Convolve/Raise_Low" /* Convolve MyProject*/
              /* with Raise_Low    */
  say 'Convoled project name is' RESULT

Menu equivalent:
  Filter/Convolve

See also:
```

1.92 croprexx

```
Crop
~~~~

Synopsis:
  CROP <ProjectName> <x1> <y1> <x2> <y2>

Function:
        Crop the given rectangle out of the given project

Inputs:
        ProjectName - Name of the project to crop.
        x1    - X co-ord of the top left corner of the crop
```

```
        rectangle.
        y1    - Y co-ord of the top left corner of the crop
        rectangle.
        x2    - X co-ord of the bottom right corner of the crop
        rectangle.
        y2    - Y co-ord of the bottom right corner of the crop
        rectangle.

Result:
        Returns the name of the newly created project in RESULT.

Example:
  /* Cut out the rectangle from (10,10) to (20,20) from the image */
  /* MyProject */
  ...
  CROP MyProject 10 10 20 20
  ...

Menu equivalent:
  Edit/Cut

See also:
```

1.93 falsecolourrexx

```
False Colour
~~~~~~~~~~~~

Synopsis:
  FALSE_COLOUR <ProjectName>

Function:
        Apply a false colour effect to an image.

Inputs:
  ProjectName - Name of the project.

Result:
  Returns the name of the newly created project in RESULT.

Menu equivalent:
  Process/False colour

See also:
```

1.94 gammarexx

```
Gamma
~~~~~

Synopsis:
```

```
  GAMMA <ProjectName> <Value>

Function:
  Change the Gamma content of an image.

Inputs:
  ProjectName - Name of the project.
        Value       - Value to change gamma by.

Result:
  Returns the name of the newly created project in RESULT.

Menu equivalent:
  Process/Gamma

See also:
```

1.95 getdirrexx

```
Get Dir
~~~~~~~

Synopsis:
  GET_DIR <Title> [<Ok Text> [<Initial Dir>] ]

Function:
  Get a directory from the user.

Inputs:
  Title       - Title of the directory requester.
        Ok Text     - Optional text to use for the Ok button.
        Initial Dir - Optional directory that the requester should start in.

Result:
  Returns name of the choosen directory in RESULT.
  If the user cancels the requester, 5 will be returned in RC.

Example:

  /* This example shows how to get a directory from the user and  */
  /* how to fix it to make it ready to append filename to.  */
  /* ie 'sys:t' needs to become 'sys:t/' before we can use it to  */
  /* build complete pathnames.          */

  Options Results     /* We want to receive results */

  'GET_DIR "Select Destination Dir" "Go!!!"'  /* Get destination */
  if RC=5 then exit   /* Exit if we were cancelled */
  destdir=RESULT

  endpart=right(destdir,1) /* Fix it so that it ends in ':' or '/' */
  if endpart~=":" & endpart~="/" then destdir=destdir||"/"
      /* Now destdir is ready to append file names to it */
  ...
```

```
Menu equivalent:
  None.

See also:
  GET_FILE, GET_FILES
```

1.96 getfilerexx

```
Get File
~~~~~~~~

Synopsis:
  GET_FILE <Title> [<Ok Text> [<Initial Dir>] ]

Function:
  Get a path and filename from the user.

Inputs:
  Title       - Title of the file requester.
        Ok Text     - Optional text to use for the Ok button.
        Initial Dir - Optional directory that the requester should start in.

Result:
  Returns the complete path and file name in RESULT.
  If the user cancels the requester, 5 will be returned in RC.

Example:

  'GET_FILE "Select an Image to process" "Go!!!"' /* Get a file */
  if RC=5 then exit   /* Exit if we were cancelled */
  MyFile=RESULT
  ...

Menu equivalent:
  None.

See also:
  GET_DIR, GET_FILES
```

1.97 getfilesrexx

```
Get Files
~~~~~~~~~

Synopsis:
  GET_FILES <Title> [<Ok Text> [<Initial Dir>] ]

Function:
  Get multiple path and filenames from the user.

Inputs:
        Title       - Title of the file requester.
```

```
        Ok Text     - Optional text to use for the Ok button.
        Initial Dir - Optional directory that the requester should start in.

Result:
  Returns a list of the path and file names separated by a ';' in
  RESULT.  If the user cancels the requester, 5 will be returned in RC.

Example:

  'GET_FILES "Select Images to process" "Go!!!"' /* Get a files */
  if RC=5 then exit   /* Exit if we were cancelled */
  MyFileList=RESULT /* MyFileList now contains the list of files */
        /* in the form "sys:pics/File1;sys:pics/File2" */

  do while MyFileList~="" /* Keep going while we still have files left */
  parse var MyFileList FileName ';' MyFileList

  ... /* Do something with the image in file FileName */

  end

  /* See the script batch_convert.rexx for a full example */

Menu equivalent:
  None.

See also:
  GET_DIR, GET_FILE
```

1.98 getfiletyperexx

```
Get File Type
~~~~~~~~~~~~~

Synopsis:
  GET_FILE_TYPE [<Title>]

Function:
  Get a save file format name from the user.

Inputs:
  Title   - Optional title of the requester.

Result:
  Returns the name of a valid save file format in RESULT.
  If the user cancels the requester, 5 will be returned in RC.

Example:

  'GET_FILE_TYPE "Select File Format"'  /* Get a save file format */
  SaveFormat=RESULT

      /* Save MyProject using SaveFormat */
  'SAVE' MyProjectName '"'||SaveFileName||'" "'||SaveFormat||'"'
```

```
Menu equivalent:
  None.

See also:
  GET_FILE, GET_FILES, SAVE
```

1.99 getnumberrexx

```
Get Number
~~~~~~~~~~

Synopsis:
  GET_NUMBER <Title> <Min> <Max> [<Ok Text>] [<Initial>]

Function:
  Get a number in the range of Min to Max from the user.

Inputs:
  Title   - Title of the requester.
  Min - Smallest number to accept.
  Max - Greatest numebr to accept.
  Ok Text - Optional text to use for the buttons.  The text for the ok
      and cancel button should be separated by a | . ie
      "Ok|Cancel"  If only text for one button is supplied, it
      will be used as the cancel button text, and there will be
      no Ok button.
  Initial - Optional number to initailly place in the requester.

Result:
  Returns the entered number in RESULT.
  If the user cancels the requester, 5 will be returned in RC.

Example:

    /* Ask the user how many frames they want generated */
  'GET_NUMBER "Enter the number of Frames" 1 100 "Ok|Cancel" 10'
  if RC=5 then exit
  NumberOfFrames=RESULT

  say 'Number to frames to generate:' NumberOfFrames
  ...

Menu equivalent:
  None.

See also:
  GET_PERCENT, GET_STRING
```

1.100 getpercentrexx

```
Get Percent
~~~~~~~~~~~
```

```
Synopsis:
  GET_PERCENT <Title>

Function:
  Get a percentage in the range of 0% and 100% from the user.

Inputs:
  Title   - Title of the requester.

Result:
  Returns the percentage in RESULT.
  If the user cancels the requester, 5 will be returned in RC.

Example:

    /* Ask the user how many frames they want generated */
  'GET_PERCENT "Enter the % to mix by"'
  if RC=5 then exit
  MixPercent=RESULT

  say 'Mix Percentage :' MixPercent
  ...

Menu equivalent:
  None.

See also:
  GET_NUMBER, GET_STRING
```

1.101 getprirexx

```
Get Task Priority
~~~~~~~~~~~~~~~~~

Synopsis:
  GET_PRI

Function:
  Get Image Engineer's task priority.

Inputs:
  None.

Result:
  Returns IE's task priority in RESULT.

Example:

    /* Ask the user how many frames they want generated */
  'GET_PRI'
  TaskPri=RESULT

  say 'Image Engineers task priority currently is ' TaskPri
  ...
```

```
Menu equivalent:
  None.

See also:
  SET_PRI
```

1.102 getrenderrexx

```
Get Render Options
~~~~~~~~~~~~~~~~~~

Synopsis:
  GET_RENDER <GREY/8/8BIT | COLOR/COLOUR/24/24BIT> <title> [<Render Options>]

Function:
  Get render settings from the usr.

Inputs:
  GREY/8/8BIT   - Specifies that the render options should be
        - for an 8 bit grey image.
  COLOR/COLOUR/24/24BIT - Specifies that the render options should be
        - for a 24 bit colour image.
  Title     - Title to use for the requester.
  Render Options    - Optional string describing the initial
        - render settings.

Result:
  Returns a string of numbers describing render setting to use,
  suitable for passing to SET_RENDER.  If the user cancels the
  requester, 5 will be returned in RC.

Example:

  'TYPE COLOUR' /* We're working in colour */

    /* Find out how the user would like us to render the image */
  'GET_RENDER COLOUR "How would like it rendered?"'
  RenderOptions=RESULT

  'OPEN "bono.iff"' /* Open the image */
  BonoProject=RESULT

  'SET_RENDER' BonoProject RenderOptions
  'RENDER' BonoProject

Menu equivalent:
  Screen/Render Control

See also:
  SET_RENDER
```

1.103 getstringrexx

```
Get String
~~~~~~~~~~

Synopsis:
  GET_STRING <Title> [<Ok Text>] [<Initial>]

Function:
  Get a string from the user.

Inputs:
  Title   - Title of the requester.
  Ok Text - Optional text to use for the buttons.  The text for the ok
      and cancel button should be separated by a | . ie
      "Ok|Cancel"  If only text for one button is supplied, it
      will be used as the cancel button text, and there will be
      no ok button.
  Initial - Optional string to initially place in the requester.

Result:
  Returns the string in RESULT.
  If the user cancels the requester, 5 will be returned in RC.

Example:

    /* Ask the user what file extension should be added */
  'GET_STRING "Enter file extension to add" "Ok|Cancel" ".new"'
  if RC=5 then exit
  FileExt=RESULT

  say 'File extension set to' FileExt
  ...

Menu equivalent:
  None.

See also:
  GET_NUMBER, GET_STRING
```

1.104 highboostrexx

```
HighBoost
~~~~~~~~~

Synopsis:
  HIGHBOOST <Project> <%>

Function:
  Apply highboost to an image.

Inputs:
  Project - Name of the project.
  % - Percentage to the original to add to the highpass result.
```

```
Result:
  Returns the name of the newly created project in RESULT.

Menu equivalent:
  Filter/Highboost

See also:
```

1.105 highpassrexx

```
HighPass
~~~~~~~~

Synopsis:
  HIGHPASS <Project> <Width> <Height>

Function:
  Apply a highpass filter to an image.

Inputs:
  Project - Name of the project.
  Width - Width of the mask to use.  Currently must be an odd number.
    - (ie 1,3,5...) An even number will be rounded down to an odd
      number.
  Height  - Height of the mask to use.  Currently must be an odd number.
      (ie 1,3,5...) An even number will be rounded down to an odd
      number.

Result:
  Returns the name of the newly created project in RESULT.

Menu equivalent:
  Filter/Highpass

See also:
  HIGHPASS_NORMALIZE
```

1.106 highpassnormalizerexx

```
HighPass Normalize
~~~~~~~~~~~~~~~~~~

Synopsis:
  HIGHPASS_NORMALIZE <ABSOLUTE | SCALE | CLIPNEG | CLIPPOS>

Function:
  Set how the highpass filter should normalize values.

Inputs:
  ABSOLUTE  - Take the absolute value.
  SCALE   - Scale the values into range.
```

```
  CLIPNEG   - Clip negative values.
  CLIPPOS   - Clip positive values.

Result:
  None.

Menu equivalent:
  Filter/Highpass

See also:
  HIGHPASS
```

1.107 histogram_equalizationrexx

```
Histogram Equalizaion
~~~~~~~~~~~~~~~~~~~~~

Synopsis:
  HISTOGRAM_EQUALIZATION <Project>

Function:
  Apply histogram equalization to an image.

Inputs:
  Project - Name of the project.

Result:
  Returns the name of the newly created project in RESULT.

Menu equivalent:
  Process/Histogram Equalization

See also:
```

1.108 ietofrontrexx

```
Image Engineer to Front
~~~~~~~~~~~~~~~~~~~~~~~

Synopsis:
  IE_TO_FRONT

Function:
        Move IE's screen to the front of the display.

Inputs:
        None.

Result:
  None.

Menu equivalent:
```

```
  None.

See also:
        WB_TO_FRONT
```

1.109 lasterrorrexx

```
Last Error
~~~~~~~~~~

Synopsis:
        LAST_ERROR

Function:
        Get a string describing why the last error occurred.  This is useful
        in error handler routines.

Inputs:
        None.

Result:
        Returns an error string in RESULT.

Example:

        /* Example showing use of LAST_ERROR */
        /* I recommend that you use this code in your scripts */

        Options Results
        Signal On Error   /* Tell Rexx that we have an error routine */

        ...     /* Do work here... */

        exit

        /**********************************************************/
        /* This is our error handling routine.  When an error occurs */
  /* program execution goes here.  This routine gets the error */
  /* string from IE and displays a message saying that an      */
  /* error has occured, what the error message is and on what  */
  /* line of the script it occurred on.                        */
        /**********************************************************/
        Error:

        IE_TO_FRONT /* Move IE to the front of the display */
        LAST_ERROR  /* Get the error string */
                        /* Display an error message */
        'REQUEST "Error detected!!!'||D2C(10)||'Error message is as follows'||D2C ↩
           (10)||result||D2C(10)||'Script failed on line '||SIGL||'"' 'Doh!'
        Exit

Menu equivalent:
        None.
```

```
See also:
```

1.110 loadpaletterexx

```
Load Palette
~~~~~~~~~~~~

Synopsis:
  LOAD_PALETTE <Project> <FileName>

Function:
  Load a palette into a project.

Inputs:
  Project   - Name of the project.
  FileName  - File name of the palette to load.

Result:
  None.

Example:

    /* Ask the user for a new palette */
  'GET_FILE "Select New palette"'
  NewPalette=RESULT

    /* Load the new palette for MyProject */
  'LOAD_PALETTE' MyProject '"'||NewPalette||'"'

  ...

Menu equivalent:
  Screen/Palette

See also:
```

1.111 localcontraststretchrexx

```
Local Contrast Stretch
~~~~~~~~~~~~~~~~~~~~~~

Synopsis:
  LOCAL_CONTRAST_STRETCH <Project> <Width> <Height>

Function:
  Apply a local contrast stretch to an image.

Inputs:
  Project - Name of the project.
  Width - Width of the mask to use.  Currently must be an odd number.
      (ie 1,3,5...) An even number will be rounded down to an odd
      number.
```

```
  Height  - Height of the mask to use.  Currently must be an odd
            number. (ie 1,3,5...) An even number will be rounded down
            to an odd number.

Result:
  Returns the name of the newly created project in RESULT.

Menu equivalent:
  Process/Local COntrast Stretch

See also:
```

1.112 lowpassrexx

```
LowPass
~~~~~~~

Synopsis:
  LOWPASS <Project> <Width> <Height>

Function:
  Apply a lowpass filter to an image.

Inputs:
  Project - Name of the project.
  Width - Width of the mask to use.  Currently must be an odd
            number. (ie 1,3,5...) An even number will be rounded down
            to an odd number.
  Height  - Height of the mask to use.  Currently must be an odd
            number. (ie 1,3,5...) An even number will be rounded down
            to an odd number.

Result:
  Returns the name of the newly created project in RESULT.

Menu equivalent:
  Filter/Lowpass

See also:
```

1.113 markrexx

```
Mark As
~~~~~~~

Synopsis:
  MARK <ProjectName> <PRIMARY|SECONDARY|ALPHA>

Function:
  This marks a project for use with the Composite command.

Inputs:
```

```
        ProjectName - Name of the project.
        PRIMARY     - This keyword specifies that this project should be
                      used as the Primary image when compositing.
        SECONDARY   - This keyword specifies that this project should be
                      used as the Secondary image when compositing.
        ALPHA       - This keyword specifies that this project should be
                      used as an Alpha channel when compositing.

Result:
  None.

Example:

        /* Combine the BackgroundProject with LogoProject using */
        /* LogoAlphaProject as an alpha channel */
        MARK BackgroundProject SECONDARY
        MARK LogoProject PRIMARY
  MARK LogoAlphaProject ALPHA
        COMPOSITE 0 0 ALPHA
        ...

Menu equivalent:
  Composite/Primary,Secondary,Alpha

See also:
  COMPOSITE
```

1.114 medianrexx

```
Median Filter
~~~~~~~~~~~~~

Synopsis:
  MEDIAN <Project> <Width> <Height>

Function:
  Apply a median filter to an image.

Inputs:
        Project - Name of the project.
        Width   - Width of the mask to use.  Currently must be an odd
                  number. (ie 1,3,5...) An even number will be rounded down
                  to an odd number.
        Height  - Height of the mask to use.  Currently must be an odd
                  number.  (ie 1,3,5...) An even number will be rounded down
                  to an odd number.

Result:
        Returns the name of the newly created project in RESULT.

Menu equivalent:
  Filter/Median

See also:
```

```
  HIGHPASS_NORMALIZE
```

1.115 negativerexx

```
Negative
~~~~~~~~

Synopsis:
  NEGATIVE <Project>

Function:
        Negate an image.

Inputs:
  Project - Name of the project.

Result:
  Returns the name of the newly created project in RESULT.

Menu equivalent:
  Process/Negative

See also:
```

1.116 openrexx

```
Open
~~~~

Synopsis:
  OPEN <FileName>

Function:
  Open an image from disk, and convert to 8 bit grey or 24 bit colour.

Inputs:
  FileName  - Complete path and filename of the image to load.

Result:
  Returns the name of the newly created project in RESULT.

Example:

    /* Ask the user what image they want to process */
  'GET_FILE "Select an Image to process" "Go!!!"' /* Get a file */
  if RC=5 then exit   /* Exit if we were cancelled */
  MyFile=RESULT

  'OPEN "'||MyFile||'"'
  MyProject=RESULT
```

```
Menu equivalent:
  Project/Open

See also:
  TYPE
```

1.117 openclipboardrexx

```
Open Clipboard
~~~~~~~~~~~~~~

Synopsis:
  OPEN_CLIPBOARD

Function:
  Open an image from the clipboard, and convert to 8 bit grey or 24
  bit colour.

Inputs:
        FileName - Complete path and filename of the image to load.

Result:
  Returns the name of the newly created project in RESULT.

Menu equivalent:
  Project/Open

See also:
  TYPE
```

1.118 projectinforexx

```
Get Project Info
~~~~~~~~~~~~~~~~

Synopsis:
  PROJECT_INFO  <Project> <W/WIDTH | H/HEIGHT | TYPE>

Function:
  Get information about a project.

Inputs:
        Project  - Name of the project.
  W/WIDTH  - Specifies that the project's width should be returned.
        H/HEIGHT - Specifies that the project's height should be returned.
        TYPE     - Specifies that the project's type should be returned.

Result:
        Returns the project's width/height or type.  If type was specified,
        the string "GREY" or "COLOUR" will be returned.
```

```
Example:

        ...
        /* Get MyProjects width */
        'PROJECT_INFO' MyProject 'WIDTH'
        ProjectWidth=RESULT

        /* Get MyProjects height */
        'PROJECT_INFO' MyProject 'HEIGHT'
        ProjectHeight=RESULT
        ...

Menu equivalent:
  Edit/Info

See also:
```

1.119 quitrexx

```
Quit
~~~~

Synopsis:
        QUIT

Function:
        This closes all projects and render screens and quits Image
        Engineer completely.

Inputs:
        None.

Result:
        None.

Example:

        ...
        QUIT /* Finished work, exit IE */
        say 'See you later.'

Menu equivalent:
        Project/Quit

See also:
```

1.120 renderrexx

```
Render
~~~~~~

Synopsis:
```

```
  RENDER <Project> [QUIET]

Function:
  Render a project.

Inputs:
  Project - Name of the project to render.
  QUIET - This keyword forces IE to open the screen at the back of
      the display and not to interrupt the user by stealing the
      input focus.

Result:
  None.

Example:

               /* Find out how the user would like us to render the image */
        'GET_RENDER COLOUR "How would like it rendered?"'
        RenderOptions=RESULT

        ...

        /* Set up the project's render options */
        'SET_RENDER' MyProject RenderOptions

        /* Render the image quietly */
        'RENDER' MyProject 'QUIET'


Menu equivalent:
        Screen/Render

See also:
        CLOSE_RENDER
```

1.121 renderautoscrollrexx

```
Render Autoscroll
~~~~~~~~~~~~~~~~~

Synopsis:
        RENDER_AUTOSCROLL <Project> <YES/TRUE | NO/FALSE>

Function:
        Set a whether a project's render screen should autoscroll.

Inputs:
        Project  - Name of the project.
        YES/TRUE - Turn autoscroll on.
        NO/FALSE - Turn autoscroll off.

Result:
        None.

Menu equivalent:
```

```
  Screen/Render Control

See also:
  RENDER_COLOURS, RENDER_DEPTH, RENDER_DEVICE, RENDER_DITHER,
  RENDER_QUANTIZE, RENDER_SCREENMODE
```

1.122 rendercoloursrexx

```
Render Colours
~~~~~~~~~~~~~~

Synopsis:
        RENDER_COLOURS <Project> <Colours>

Function:
        Set the number of colours a project’s render screen should use.

Inputs:
        Project - Name of the project.
        Colours - The number of colours to use.

Result:
        None.

Menu equivalent:
        Screen/Render Control

See also:
        RENDER_AUTOSCROLL, RENDER_DEPTH, RENDER_DEVICE, RENDER_DITHER,
        RENDER_QUANTIZE, RENDER_SCREENMODE
```

1.123 renderdepthrexx

```
Render Depth
~~~~~~~~~~~~

Synopsis:
        RENDER_DEPTH <Project> <Depth>

Function:
        Set a whether the number of colours a project’s render screen should
        use.

Inputs:
        Project - Name of the project.
  Depth   - The maximum number of colour the screen should have.
                ie 2, 4, 8, ... 2^n, ...256.

Result:
        None.

Menu equivalent:
```

```
        Screen/Render Control

See also:
        RENDER_AUTOSCROLL, RENDER_COLOURS, RENDER_DEVICE, RENDER_DITHER,
        RENDER_QUANTIZE, RENDER_SCREENMODE
```

1.124 renderdevicerexx

```
Render Device
~~~~~~~~~~~~~

Synopsis:
        RENDER_DEVICE <Project> <AMIGA|SVDRIVER|HAM8|HAM6>

Function:
        Set a what device should be used to render a project.

Inputs:
        Project  - Name of the project.
        AMIGA    - Use the standard Amiga register based display.
        SVDRIVER - Use the current SVDriver.
        HAM8     - Use Amiga HAM8 (AGA only of course!).
        HAM6     - Use Amiga HAM6.

Result:
        None.

Menu equivalent:
        Screen/Render Control

See also:
        RENDER_AUTOSCROLL, RENDER_COLOURS, RENDER_DEPTH, RENDER_DITHER,
        RENDER_QUANTIZE, RENDER_SCREENMODE
```

1.125 renderditherrexx

```
Render Dither
~~~~~~~~~~~~~

Synopsis:
        RENDER_DITHER <Project> <NONE|FLOYD|BURKES|STUCKI|SIERRA|JARVIS|STEVENSON>

Function:
        Set a what dithering should be used to render a project.

Inputs:
        Project   - Name of the project.
        NONE      - none.
        FLOYD     - Use Floyd-Steinberg dithering.
        BURKES    - Use Burkes dithering.
        STUCKI    - Use Stucki dithering.
        SIERRA    - Use Sierra dithering.
```

```
        JARVIS    - Use Jarvis dithering.
        STEVENSON - Use Stevenson-Arce dithering.

Result:
        None.

Menu equivalent:
        Screen/Render Control

See also:
        RENDER_AUTOSCROLL, RENDER_COLOURS, RENDER_DEPTH, RENDER_DEVICE,
        RENDER_QUANTIZE, RENDER_SCREENMODE
```

1.126 renderquantizerexx

```
Render Quantize
~~~~~~~~~~~~~~~

Synopsis:
        RENDER_QUANTIZE <Project> <LOCK|BEST|LOCK_FAST|LOCK_BEST|MEDIAN_CUT>

Function:
        Set a how a project should be quantized when rendering.

Inputs:
        Project    - Name of the project.
        LOCK       - Lock to palette. 8 bit Grey only.
        BEST       - Choose Best palette. 8 bit grey only.
        LOCK_BEST  - Lock to palette using the best algorithm.
        LOCK_FAST  - Lock to palette using the fast algorithm.
        MEDIAN_CUT - Choose palette using the Median Cut algorithm.

Result:
        None.

Menu equivalent:
        Screen/Render Control

See also:
        RENDER_AUTOSCROLL, RENDER_COLOURS, RENDER_DEPTH, RENDER_DEVICE
        RENDER_DITHER, RENDER_SCREENMODE
```

1.127 renderscreenmoderexx

```
Render Screen Mode
~~~~~~~~~~~~~~~~~~

Synopsis:
        RENDER_SCREENMODE <Project> <ModeID>

Function:
        Set a how a project should be quantized when rendering.
```

```
Inputs:
        Project - Name of the project.
        ModeID  - Screen mode ID.

Result:
        None.

Menu equivalent:
        Screen/Render Control

See also:
        RENDER_AUTOSCROLL, RENDER_COLOURS, RENDER_DEPTH, RENDER_DEVICE,
        RENDER_DITHER, RENDER_QUANTIZE
```

1.128 rendertofrontrexx

```
Render to Front
~~~~~~~~~~~~~~~

Synopsis:
        RENDER_TO_FRONT <Project>

Function:
        Move a project's render screen to the front of the display.

Inputs:
        Project - Name of the project.

Result:
        None.

Menu equivalent:
        Screen/Show Render

See also:
```

1.129 reflectxrexx

```
Reflect X
~~~~~~~~~

Synopsis:
        REFLECT_X <ProjectName>

Function:
        Reflect a project on the x axis (ie left-right).

Inputs:
        ProjectName - Name of the project.

Result:
```

```
        Returns the name of the newly created project in RESULT.

Menu equivalent:
        Edit/Reflect X

See also:
```

1.130 reflectyrexx

```
Reflect Y
~~~~~~~~~

Synopsis:
        REFLECT_Y <Project>

Function:
        Reflect a project on the y axis (ie up-down).

Inputs:
        Project - Name of the project.

Result:
        Returns the name of the newly created project in RESULT.

Menu equivalent:
        Edit/Reflect Y

See also:
```

1.131 requestrexx

```
Request
~~~~~~~

Synopsis:
        REQUEST <Message> [<Buttons>]

Function:
        Put up a requester to the user and get a responce.

Inputs:
        Message - Message to put in the requester. The message may contain
                  newline characters (ASCII 10).
  Buttons - String of button labels separated by a |.

Result:
        Returns 1 for the left most button and 0 for the rightmost.  Buttons
        in the middle will return one more than the button to it's left.
        (ie Buttons: Brilliance, Ok, Average, Crap.
        return       1,          2,  3,       0.)
```

```
Example:
                  /* Tell the user that we've finished */
        'REQUEST "All done"'
        ...


                 /* Tell the user what this script does */
        'REQUEST "This shows how to put up' D2C(10),
                 'requesters that may be' D2C(10),
                 'split over several lines." "I understand"'

        ...      /* Find out what the user thinks */
        'REQUEST "What do think about this script" "Great|Ok|Crap"
        reply=RESULT
        if reply=1 then 'REQUEST "I am glad you like it!"'
        if reply=2 then 'REQUEST "That is good to hear."'
        if reply=0 then 'REQUEST "oh well, yous gets what yous pays for."'
        ...

Menu equivalent:
        None.

See also:
```

1.132 resizerexx

```
Resize
~~~~~~

Synopsis:
        RESIZE <Project> <Width> <Height> <X Offset> <Y Offset>

Function:
        Resize an image.

Inputs:
        Project  - Name of the project to resize.
  Width    - New width of image.
  Height   - New height of image.
  X Offset - X offset of the image into the new size.
  Y Offset - Y offset of the image into the new size.

Result:
        Returns the name of the newly created project in RESULT.

Menu equivalent:
        Edit/Resize

See also:
```

1.133 saverexx

```
Save
~~~~

Synopsis:
        SAVE <Project> <FileName> <FileType>

Function:
        Save a project's render screen to disk.

Inputs:
        Project  - Name of the project.
        FileName - Complete path and file name to save as.
        FileType - The name of the file format to save as.

Result:
        None.

Example:

        ...
                /* The image has been render, now */
                /* Ask the user for a save file name */
        'GET_FILE "What do you want to save it as?"'
        DestName=RESULT

                /* Ask the user for what format they want to save as */
        'GET_FILE_TYPE "What format do you want?"'
        FileType=RESULT

                /* Save it! */
        'SAVE' MyProject '"'||DestName||'" "'||FileType||'"'
        ...

Menu equivalent:
        Project/Save

See also:
        GET_FILE_TYPE, SAVE_DATA
```

1.134 savecliprexx

```
Save to Clipboard
~~~~~~~~~~~~~~~~~

Synopsis:
        SAVE_CLIP <Project>

Function:
        Save a project's render to the clipboards.

Inputs:
        Project - Name of the project.
```

```
Result:
  None.

Menu equivalent:
        Screen/Save to Clipboard

See also:
        OPEN_CLIPBOARD
```

1.135 savedatarexx

```
Save Data
~~~~~~~~~

Synopsis:
        SAVE_DATA <Project> <FileName> <FileType>

Function:
        Save a project’s image data to disk as 8 bit grey or 24 bit colour.

Inputs:
        Project  - Name of the project.
        FileName - Complete path and file name to save as.
        FileType - The name of the file format to save as.

Result:
        None.

Example:

        ...
                /* The image has been render, now */
                /* Ask the user for a save file name */
  'GET_FILE "What do you want to save the data as?"'
  DestName=RESULT

                /* Ask the user for what format they want to save as */
        'GET_FILE_TYPE "What format do you want?"'
        FileType=RESULT

                /* Save it! */
        'SAVE' MyProject '"'||DestName||'" "'||FileType||'"'
        ...

Menu equivalent:
        Project/Save

See also:
        GET_FILE_TYPE, SAVE
```

1.136 savepaletterexx

```
Save Palette
~~~~~~~~~~~~

Synopsis:
        SAVE_PALETTE <Project> <FileName>

Function:
        Save project’s palette to disk.

Inputs:
        Project  - Name of the project.
        FileName - File name to save palette as.

Result:
        None.

Example:

                /* Ask the user what they want the palette saved as */
        ’GET_FILE "Select File to save palette as"’
        PaletteName=RESULT

                /* Save MyProject’s palette */
        ’SAVE_PALETTE’ MyProject ’"’||PaletteName||’"’
        ...

Menu equivalent:
        Screen/Palette

See also:
```

1.137 scalerexx

```
Scale
~~~~~

Synopsis:
        SCALE <Project> <Width> <Height> <FAST | BEST/COLOUR_AVERAGE>

Function:
        Scale an image to a new size.

Inputs:
        Project             - Name of the project to scale.
        Width               - New width to scale to.
        Height              - New height to scale to.
        FAST                - Use fast nearest neighbour algorithm.
        BEST/COLOUR_AVERAGE - Use colour average algorithm.

Result:
        Returns the name of the newly created project in RESULT.

Example:
                /* Scale image down to postage stamp size */
```

```
        'SCALE' MyProject '80 50 FAST'
        PostageStamp=RESULT

Menu equivalent:
        Edit/Scale

See also:
```

1.138 setprirexx

```
Set Task Priority
~~~~~~~~~~~~~~~~~

Synopsis:
        SET_PRI <Priority>

Function:
        Set Image Engineer's task priority.

Inputs:
        Priority - IE's new task priority.  In the range of -20 to 20.
                   (Same as the AmigaDOS command ChangeTaskPri, except that
                   the range has been limited, instead of the usual -128 to
                   127 range).

Result:
        Returns the old task priority in RESULT.

Example:

                /* Reduce our Task Priority so that the user */
                /* can get on with other work in the foreground.
        'SET_PRI -1'
        OldPri=RESULT

        ...     /* Do some serious number crunching here */

        'SET_PRI '||OldPri      /*Finished work! restore old priority */

Menu equivalent:
        None.

See also:
        GET_PRI
```

1.139 setrenderrexx

```
Set a project's Render Options
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Synopsis:
        SET_RENDER <ProjectName> <Render Options>
```

```
Function:
        Set a project's render options.

Inputs:
        ProjectName    - Name of the project to set.
        Render Options - String of numbers describing the project's new
                         render settings.

Result:
        None.

Example:

        'TYPE COLOUR'   /* We're working in colour */

                /* Find out how the user would like us to render the image */
        'GET_RENDER COLOUR "How would like it rendered?"'
        RenderOptions=RESULT

        'OPEN "bono.iff"'       /* Open the image */
        BonoProject=RESULT

        'SET_RENDER' BonoProject RenderOptions
        'RENDER' BonoProject

Menu equivalent:
        Screen/Render Control

See also:
        GET_RENDER
```

1.140 sharpenrexx

```
Sharpen
~~~~~~~

Synopsis:
        SHARPEN <Project> <%>

Function:
        Apply a sharpening filter to an image.

Inputs:
        Project - Name of the project to sharpen.
        %       - Percentage to sharpen by.

Result:
        Returns the name of the newly created project in RESULT.

Menu equivalent:
        Filter/Sharpen

See also:
```

1.141 thresholdrexx

```
Threshold
~~~~~~~~~

Synopsis:
        THRESHOLD <Project> <Level>

Function:
        Threshold an 8 bit grey image.

Inputs:
        Project - Name of the project to threshold.
        Level   - Grey level threshold in the range of 0 to 255 inclusive.

Result:
        Returns the name of the newly created project in RESULT.

Menu equivalent:
        Filter/Threshold

See also:
```

1.142 typerexx

```
Set Load Type
~~~~~~~~~~~~~

Synopsis:
        TYPE <GREY/8/8BIT | COLOR/COLOUR/24/24BIT>

Function:
        Set whether images will be loaded as 8 bit grey or 24 bit colour.

Inputs:
        GREY/8/8BIT           - Specifies that images should be loaded as 8
                                bit grey.
  COLOR/COLOUR/24/24BIT - Specifies that images should be loaded as
                                24 bit colour.

Result:
        None.

Example:
        /* Set it up to load as 24 bit colour */
        'TYPE COLOUR'
        'OPEN "bono.iff"'
        BonoProject=RESULT

                /* Load the alpha channel as grey */
        'TYPE GREY'
        'OPEN "bono.alpha.iff"'
        BonoAlpha=RESULT
```

```
Menu equivalent:
        Project/Open/8 bit grey, 24 bit colour

See also:
```

1.143 wbtofrontrexx

```
WB to Front
~~~~~~~~~~~

Synopsis:
        WB_TO_FRONT

Function:
        Move the WorkBench screen to the front of the display.

Inputs:
        None.

Result:
        None.

Menu equivalent:
        None.

See also:
        IE_TO_FRONT
```

1.144 vmm

```
Use with Martin Apel's VMM
~~~~~~~~~~~~~~~~~~~~~~~~~~

Image Engineer allocates memory for image data with the MEMF_PUBLIC flag
clear.  The only problem is that superview.library's svobjects usually
don't allocate memory with the MEMF_PUBLIC flags (from what I can tell),
which means that you can still run out of memory when loading and saving
even if you have heaps of VM still left.  All I did to get around this was
to use the advanced options with 10240 set for both public and non-public
allocations.  Works great.
```

1.145 author

```
About Me (or "Who the hell are you anyway?")
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Well, you may have figured out that my name is Simon Edwards.  What you
probably don't know it that I'm currently studying Computer Science (2nd
year) at the Royal Melbourne Institute of Technology (RMIT) in cloudy
Melbourne Australia (Oz).
```

```
Email: s9407349@yallara.cs.rmit.edu.au

       (that's .au as in Australia, *not* Austria, yes we have kangaroos,
       that's the difference. ;-)

Snail Mail: Simon Edwards
            Student Villiage   <- that's where I live, student accomodation,
            Williamson Road                  never a dull moment \ensuremath{\pm})
            Maribyrnong 3032
            Victoria, Australia

(Email preferred), I can't guarantee a responce to snail mail).

IE was developed using Devpac 3 on a A1200 with a 50MHz 030 + 882, 4Mb
of fast ram, and a 250Mb Hard disk.

When reporting bugs give a detailed report of what the bug is whether it's
reproducable or not, what version of IE you using and what your system
consists of (A1200, A4000 etc), including expansions and additional
software that was running at the time.
```

1.146 thanx

```
Thanks and Greets
~~~~~~~~~~~~~~~~~

Thanks and greets go to the following people in no particular order...

* All programmers of excellent Amiga software.
* Nigel Steward, for putting up with all my questions
* Micheal Haigh, for the technical support
* Residents at the Villiage.
* U2, the greatest rock band in the world.
* to everyone I forgot...
```

1.147 future

```
The Future
~~~~~~~~~~

* Better Documentation
* More features !!!  (Image rotation next perhaps...)

This is where you can help.  Let me know what features you would like to
see in the next version (what would be even better would be a description
of how a given feature actually works (saves time in that I don't have to
run around trying to figure out how it works)).  Also, any other suggestions
as to how things can be improved or how things can be done better are also
very welcome.  If you come up with some wonderful convolve or ARexx script,
let me know and I'll include it in the next release.  Also drop me an email
if you want to be placed on a list of users interested in being informed by
```

email when future versions of IE are released.

I also have a rather unexciting page on the World Wide Web.

http://yallara.cs.rmit.edu.au/~s9407349/

If you would like to see the IE documentation made available on the Web,
or if you would like to see some sort of support page (and if you have
an suggestions as for what should be on it). Let me know and I'll see what
I can do. (That email address again is....) I can't read minds ya know..

Future versions also depend on how much time I have. (I'm doing 2nd year
Computer Science and that's going to consume most of my time).

"Drink when you are thirsty, rest when you are fatigued,
program when the moment is right."

1.148 history

```
History
~~~~~~~

V0.0   First the universe formed...

V1.0   21/3/95 Initial Release

V1.1   11/4/95
       Bug fixes:-
                 * Couple of minor bugs in the start up code.
                 * Nasty bug when opening 24bit files as 8bit grey.
                 * Fixed problem with the menus under OS 2.0. (hopefully)
                 * Numerous other small bug fixes.  (I've lost track ;-)
       Added ARexx macros from within IE and user configurable ARexx menu.
```

1.149 bibliography

Bibliography
~~~~~~~~~~~~

Gonzalez, R. C., and R. Woods. 1992. Digital Image Processing.
Reading, Massachusetts.: Addison Wesley.

Heckbert, P. 1982. "Color Image Quantization for Frame Buffer Display".
Computer Graphics, 16: pp 297-307.

Hill, F. S. Jr. 1990. Computer Graphics. New York.: Macmillan.

Kruger, Anton. 1994. "Median-Cut Color Quantization." Dr. Dobb's
Journal, September 19994, volume 19, issue 10.
~~~~~~~~~~~~