TextureStudio

COLLABORATORS

	TITLE : TextureStudio		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		June 25, 2022	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

TextureStudio

1.1 TextureStudio.guide

                    Welcome to TextureStudio, shareware texture renderer for the ←
                Amiga.

   This document applies to version 1.0.x, written on 10th May 1995,
Copyright (C) 1995 Graham Dean, Andy Dean.

                Introduction
                        Welcome to TextureStudio

                Quick start
                         Worked example of some of TextureStudio's features

                Menu options
                        Detailed description of program's functions

                Floating windows
                    Description of all floating windows

                ARexx
                             Using ARexx to write scripts

                Tooltypes
                        Important preferences

                Known bugs
                        Bugs with the program and textures

                Common questions
                    Frequently asked questions

                The authors
                        How to contact us, reporting problems

                How to register
                     Receiving the full version

                    Credits
                            Thanks to...

                Future additions
                      Extra planned upgrades

                Example textures
                      Example texture modules included in this package

                ImageStudio
                        Also by the same authors...

1.2 TextureStudio.guide/Introduction

                    Introduction
****

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

                Copyright and Disclaimer

                Machine requirements

                Brief description

                List of features

                Shareware version

                Starting TextureStudio

1.3 TextureStudio.guide/Copyright and Disclaimer

Copyright and Disclaimer
========================

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

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

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

Permission is granted to include the unregistered package in
Public-Domain collections, especially in the excellent Fred Fish Amiga
Disk Library (including CD ROM versions of it). The distribution file
may be uploaded to Bulletin Board Systems or FTP servers. If you want
to distribute this program you must use the original unmodified
distribution archive.

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

The textures Blades.itx, Radar.itx, Target.itx and
WarningStripes.itx are freely distributable and may be used in any
pictures, renders or programs without permission from the authors.

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

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

Imagine and its texture format is (c) Copyright 1994 Impulse Inc.
Mpls, MN 55444.

1.4 TextureStudio.guide/Machine requirements

```
Machine requirements
====================
```

TextureStudio requires the following system to run:

* Workbench 2.04 or above.

* A machine equipped with a 68020 or greater and an FPU (floating
  point unit)

* Around half a megabyte of free memory.

1.5 TextureStudio.guide/Brief description

```
Brief description
=================
```

TextureStudio supports the loading of texture modules in Imagine3
format. The parameters of the texture can quickly and easily be
adjusted by means of slider gadgets or by typing in the numbers. The

texture can then be mapped onto a plane, cylinder or sphere and
rendered to a preview screen and/or as a 24-bit image to disk.

   Many aspects of the texture and render can be altered including axis
position/alignment/size, lighting settings, object colours, object size
etc.

   TextureStudio allows the user to quickly render the texture and
explore the effects of changing various parameters without the need to
ray-trace a new image each time something altered.

   TextureStudio can render images to disk in ILBM-IFF24, JPEG or Targa
format. This allows high quality images to be rendered and loaded into
other programs. 24-bit images of any size can be rendered, regardless
of memory available.

   Some example textures are included in the distribution but Imagine3
is required to be able to use it's textures.

1.6   TextureStudio.guide/List of features

List of features
================

   * Supports Imagine3 texture format.

   * Control of features via ARexx port.

   * Render unlimited number of textures simultaneously.

   * Render to HAM screen and 24-bit images on all Amigas.

   * Render 24-bit images of any size onto disk in IFF-ILBM24, JPEG or
     Targa format, regardless of memory available.

   * Saving of HAM preview screen to disk.

   * Support for colour, filter and bump type textures.

   * Easy adjustment of parameters by means of slider gadgets.

   * Map textures onto a plane, cylinder or sphere.

   * Control of light colour, distance, position, backlighting and
     intensity.

   * Adjustment of axis alignment, size and position.

   * Control of object size, visible width and image aspect ratio.

   * Multiple pass render to allow quick preview of image whilst it
     renders.

   * 5 levels of anti-aliasing available.

* Preview colours with colourbox window.

* Alter render screens width, height and screenmode.

* Control of all main functions from floating windows.

* Optimised code for 68881 and 68882 FPU's for maximum speed.

* Render plane, cylinder or sphere without any texture to quickly
  set up lighting etc.

* Loading and saving of textures settings, parameters and axis
  positions.

* Render preferences to alters speed and accuracy of render.

* Configure window positions, screenmode, default settings and then
  save preferences to disk.

* Runs on all Amigas with Workbench2.04 or above and an FPU
  (floating point unit).

* Standard Workbench2 interface.

* Uses public screen.

1.7 TextureStudio.guide/Shareware version

```
              Shareware version
==================
```

   The shareware version is limited in that only the first 8 parameters
of the texture can be adjusted. No other features have been removed.

   The full version allows all 16 parameters to be altered.

   For details on how to register, see
              How to register
                .

1.8 TextureStudio.guide/Starting TextureStudio

```
              Starting TextureStudio
=====================
```

   TextureStudio can be started from either the Workbench or CLI. From
the Workbench it is simply a case of double-clicking on the icon.

To start TextureStudio from the CLI, simply type:

    run TextureStudio

The tooltypes can also be typed in on the CLI. For example

    run TextureStudio "PUBSCREEN=TS" "PREFSFILE=Prefs/SmallScreen.prefs"

would start the program on a public screen named 'TS' and the preferences would be taken from the file 'Prefs/SmallScreen.prefs'. See

                Tooltypes
                , for a list of available tooltypes.

1.9 TextureStudio.guide/Quick start

                Quick start
****

   This tutorial is designed to quick introduction to some of TextureStudio's features. The Imagine3 textures are required later in the tutorial.

   Firstly, we shall demonstrate TextureStudio's ability to render a texture onto a plane.

   Load TextureStudio if you haven't already. Open the 'Object' window by selecting 'Show object' in the 'Windows' menu. Select the 'Object' to be a 'Plane'. Make sure 'Allow transparent object', 'Calculate surface normals', 'Full light calculations' and 'Multiple pass render' are all set in the 'Prefs' menu. Open the 'Colours' window and select the 'Object colour' to be light grey (150,150,150). You can see the actual colour being edited by opening the 'Colour box' window from the 'Windows' menu.

   Now we are ready to render the object, at the moment there are no textures open, but that's OK, we'll just render the plane with no textures mapped onto it. Click on 'Render' in the 'Infobar' window. A grey plane should be rendered fairly quickly to a HAM view screen (HAM8 on AGA machines, HAM6 otherwise). Click the right mouse button to bring TextureStudio to the front. The image should have been rendered in a series of passes, allowing you to see a general view of the image, more quickly than if it was rendered line by line sequentially, this is set by the 'Multiple pass render' preference. You may have also noticed that the plane had some shading on it, this is due to the light source and many aspects of the lighting can be altered, we shall discuss this later in the tutorial.

   Any time you want to stop the render, just press the right mouse button.

   OK, so we have now seen how TextureStudio can render to a flat plane, but it's not very exciting without a texture mapped on to it.

Open the 'Texture' window, if it is not already open. Now select 'Open texture...' from the 'Project' menu, or 'Open...' from the 'Texture' window to load in a texture module. Select 'Radar.itx' which is distributed with this package. Click on 'OK' to load it in. Now you should see that 'Radar' has appeared in the listview in the 'Texture' window. This list shows the all the textures that are currently open, it is possible to load as many as you like with TextureStudio. You should also see that in the 'Parameters' window, the parameters for the current texture are displayed, for the moment we shall use the default parameters.

Now click on 'Render' again, you should now see the image of the 'Radar' texture being mapped onto a plane. We can adjust the settings of the texture using the parameters window. If you are running TextureStudio on a screen of heigth 256 pixels or more, you can display the full parameters window and so make sure the 'Small parameters window' preference is turned off. If TextureStudio's screen height is less than 256 (ie screen size of 640 x 200) then you can only fit the small parameters window on the screen (the small parameters window doesn't have slider gadgets). The advantage of using the full parameters window is that the parameters can quickly be adjusted by the slider gadgets and there is no need to type in the numbers.

Let's adjust some of the parameters for the 'Radar' texture. Change the 'Sweep angle' parameter to about 45 degrees. Now change the 'Radar colour' to be bright red (255,0,0), again, if you have the colourbox window open, you will be able to see actual the colour being adjusted. Now click on 'Render' again to see the changes made. This is one of TextureStudio's most useful features, the ability to change one or more parameters, and then quickly re-render the texture to see the changes made. This makes the process of getting the texture parameters just right much easier than having to ray-trace the scene many times. You may have noticed that the new image was rendered over the old image, this can be turned off by the 'Use fresh render screen' preference.

Now let's try layering 2 textures together. Open the 'WarningStripes' texture supplied with this package. It should now appear at the bottom of the list in the 'Texture' window. Textures at the bottom of the list have the highest priority and so will appear on top of the textures in the render scene. The 'WarningStripes' texture should also be selected in the list, you can change which texture you want to alter by clicking in the list to select the texture. Try clicking on the 'Radar' texture in the list and then clicking back on the 'WarningStripes' texture, notice how the parameters window changes depending on the currently selected texture.

Now click on 'Render' again. You should be able to see how the 'WarningStripes' texture has been layered on top of the 'Radar' texture. Move the 'WarningStripes' texture up in the texture list by selecting the 'WarningStripes' texture and then clicking on 'Up' in the 'Texture' window. The 'Radar' texture should now be at the bottom of the list and so have the greatest priority. Click on 'Render' to see the changes. The 'WarningStripes' texture is now behind the 'Radar' texture.

TextureStudio can also map textures onto a cylinder or sphere, we

shall now demostrate mapping the 'GasGiant' texture (supplied with
Imagine3) onto a sphere. Close the 'WarningStripes' and 'Radar'
textures by selecting them and then clicking on 'Close' in the
'Texture' window. Now load in the 'GasGiant' texture. Change the
current object to a sphere by selecting the 'Sphere' in the 'Object'
window. Click on 'Render' again. You should see an image which looks
something like a planet. Now let's try altering the textures axis. Open
the 'Axis' window. Make sure 'View' is set to 'Front' and 'Edit' is set
to 'Alignment' in the 'Axis' window. Let's rotate the axis so that the
texture is on a slant. Change the Y-alignment to be about 30 degrees by
typing in the number or using the slider gadget. Now render the scene
again. You should be able to see how the textures alignment has changed.

Now let's look at the lighting options. Open the 'Light' window.
Move the light poition to the bottom right, do this by either clicking
on the bottom left of the box (on the left of the 'Light' window) or by
clicking on the 'Position' cycle gadget. Now re-render the scene to see
the changes made. Change the 'Backlight' setting to be 0% and
re-render. The backlight setting sets the ambient light, and so setting
it to 0% puts areas away from the light source in complete shadow. Set
the 'Backlight' to 25% and re-render to see the difference. You should
now have an image that looks like a fairly convincing planet.

Let's save the render screen to disk, so we can view it later or
load it in to other programs. Select 'Save...' from the 'Infobar'
window, choose a filename (e.g. "GasGiantPlanet.ilbm") and click on OK.
Now suppose we wanted to render this image again later, or just make a
few changes, it would be a pain to have to load in the texture again
and set up the object, axis and lighting as we have it now.
TextureStudio allows you to save all the current texture settings to
disk so you can easily load them back again another time. Select 'Save
texture settings...' from the 'Texture' menu, choose a filename (e.g.
"Planet.GasGiant") and click on OK. Anytime you want to re-render the
scene just load the settings back in, but you will have to make sure
that the 'GasGiant.itx' texture file is in your current texture
directory or else it won't be able to load the file, since
TextureStudio does not save the full filename, just the texture name.

Another one of TextureStudio's useful features is to be able to
render a texture image as a 24-bit image file for use with other
programs. Let's try and render a texture as an IFF-ILBM24 file. Close
the 'GasGiant' texture now. Open the 'Agate' texture supplied with
Imagine3. Set the current object to be a plane, turn the 'Full light
calculations' preference off, this removes any highlights due to the
light source. Select 'IFF-ILBM24' as the 'Render file format' from the
'Prefs' menu. Next, bring up the render options window by clicking on
'Options...' in the 'Infobar' window. Click on 'Render to file' and
choose a filename (e.g. "Ram:TestRender.ilbm"). Change the size of the
render if you like, the default is 160 x 128 which is quarter of a low
res PAL screen. You can use the screen mode requester to set the screen
and size by clicking on 'Choose...' level with the 'Render to screen'
gadget.

Now click on 'Render', it may take a while to render the image but
you can abort it by pressing the right mouse button if you like. Notice
that the 'Multiple pass render' preference has no affect when rendering
to a file. Once TextureStudio has finished, it will have created a

24-bit IFF file. This can then be loaded into other programs that
support 24-bit image files (e.g. ImageStudio, see
                 ImageStudio
).

   Well that's the end of the tutorial, I hope it has given you an
insight into TextureStudio's features. There are many other features
that have not been covered in this tutorial but hopefully you will be
able to pick them up fairly easily. Have fun!

1.10 TextureStudio.guide/Menu options

                 Menu options
****

   Description of all menu items.

                 Project

                 Texture

                 Windows

                 Prefs

1.11 TextureStudio.guide/Project

                 Project
=======

                 Open texture

                 Close texture

                 Render

                 Screen mode

                 About

                 Quit

1.12   TextureStudio.guide/Open texture

```
Open texture
------------
```

Keyboard shortcut - Amiga - O

This is how the user loads in a texture module from disk.

A file requester will appear prompting the user to select a texture
file. If the file is not a valid Imagine3 texture file, an error will
be displayed.

When the texture module has been loaded, it will appear in the
'Texture' window.

Loading a texture resets the parameters and axis settings.

1.13   TextureStudio.guide/Close texture

```
Close texture
-------------
```

Keyboard shortcut - Amiga - C

This closes and removes the currently selected texture.

If all the textures are closed, TextureStudio will render a plane,
cylinder or sphere with no texture mapped onto it.

1.14   TextureStudio.guide/Render

```
              Render
------
```

Keyboard shortcut - Amiga - R

This renders the textures to a HAM screen and/or a 24-bit image file.

If 'Render to screen' is selected in the 'Options' window, a HAM
screen is opened a brought to the front. TextureStudio will then render
the textures to the HAM viewer screen line by line. The render
operation can be aborted by pressing 'Esc' or the right mouse button.

If 'Render to file' is selected in the 'Options' window, a file is
opened and the image is rendered in 24-bits to the file. Since
TextureStudio renders to image line by line, very little memory is
required when only rendering to a file, regardless of the image size.
Note: when rendering to a file, the 'Multiple pass render' preference
is ignored and the image is render from top to bottom in one pass.

It is possible to render to the screen and file simultaneously.

See
                Render options
                .

1.15 TextureStudio.guide/Screen mode

                Screen mode

   This opens a screen mode requester to allow the user to change the
screen mode, width, height and number of colours of TextureStudio's
main screen.

   The screen mode used on startup can be saved with the 'Save prefs'
menu item in the 'Prefs menu', See
                Save prefs
                .

1.16 TextureStudio.guide/About

About

   Keyboard shortcut - Amiga - ?

   This opens the about window which displays the current version and
registered user name.

1.17 TextureStudio.guide/Quit

Quit

   Keyboard shortcut - Amiga - Q

   This closes all currently open textures and quits TextureStudio.

1.18   TextureStudio.guide/Texture

                      Texture
=======

                  Load texture settings

                  Save texture settings

                  Load axis positions

                  Save axis positions

                  Load parameters

                  Save parameters

1.19   TextureStudio.guide/Load texture settings

Load texture settings

   Keyboard shortcut – Amiga – D

   This loads in a TextureStudio settings format file.

   Loading in a settings file may alter currently loaded textures, axis
settings, object settings, lighting settings, colours and view settings.

   Settings files may contain any number of textures (including zero),
the textures are then attempted to be opened from the user's current
texture directory.

   If there are currently some open textures, then a requester will
appear asking the user whether they want to wipe the existing textures
or layer the new textures on top of the existing ones.

1.20   TextureStudio.guide/Save texture settings

Save texture settings

   Keyboard shortcut – Amiga – F

   This saves out a TextureStudio settings file with information about
the current settings of: all open textures, parameters, axis, object,
lighting, colours and view settings.

A settings file may contain information about any number of textures (including zero).

Settings files are particularly useful for saving out all the current information needed to reproduce the exact same effect another time.

Settings files are in ASCII text format.

1.21   TextureStudio.guide/Load axis positions

```
Load axis positions
-------------------
```

Keyboard shortcut - Amiga - G

Loads in a TextureStudio axis file.

The axis file contains the settings of the axis alignment, size and position.

1.22   TextureStudio.guide/Save axis positions

```
Save axis positions
-------------------
```

Keyboard shortcut - Amiga - H

Saves out current textures axis alignment, size and position in ASCII text format.

1.23   TextureStudio.guide/Load parameters

```
Load parameters
---------------
```

Keyboard shortcut - Amiga - J

Loads in 16 parameters from a TextureStudio parameters file.

Note: TextureStudio does not check to see whether the current texture is the same texture used to save out the parameters file.

1.24 TextureStudio.guide/Save parameters

```
Save parameters
---------------
```

    Keyboard shortcut – Amiga – K

    Saves out the 16 parameters of the currently selected texture.

    Note: Only the parameters of one texture are saved.

1.25 TextureStudio.guide/Windows

```
                Windows
=======
```

                    Show axis

                    Show colourbox

                    Show colours

                    Show light

                    Show object

                    Show parameters

                    Show texture

                    Show view

1.26 TextureStudio.guide/Show axis

```
            Show axis
---------
```

    Keyboard shortcut – Amiga – 1

    This opens or closes the axis window.

    See
            Axis window
                .

1.27 TextureStudio.guide/Show colourbox

```
              Show colourbox
-------------

   Keyboard shortcut - Amiga - 2

   This opens or closes the colourbox window.

   See
              Colourbox window
              .
```

1.28 TextureStudio.guide/Show colours

```
              Show colours
------------

   Keyboard shortcut - Amiga - 3

   This opens or closes the colours window.

   See
              Colours window
              .
```

1.29 TextureStudio.guide/Show light

```
              Show light
----------

   Keyboard shortcut - Amiga - 4

   This opens or closes the light window.

   See
              Light window
              .
```

1.30 TextureStudio.guide/Show object

```
              Show object
-----------
```

```
Keyboard shortcut - Amiga - 5

This opens or closes the object window.

See
            Object window
              .
```

1.31 TextureStudio.guide/Show parameters

```
            Show parameters
---------------

Keyboard shortcut - Amiga - 6

This opens or closes the parameters window.

See
            Parameters window
              .
```

1.32 TextureStudio.guide/Show texture

```
            Show texture
------------

Keyboard shortcut - Amiga - 7

This opens or closes the texture window.

See
            Texture window
              .
```

1.33 TextureStudio.guide/Show view

```
            Show view
---------

Keyboard shortcut - Amiga - 8

This opens or closes the view window.
```

See

View window

.

1.34 TextureStudio.guide/Prefs

Prefs
=====

Beep when finished

Flush textures on open

Multiple pass render

Small parameters window

Use fresh render screen

Allow transparent object

Calculate surface normals

Full light calculations

Anti-aliasing

Render file format

JPEG options

Save prefs

1.35 TextureStudio.guide/Beep when finished

Beep when finished

If this is on, when TextureStudio has finished rendering to the
screen, it will flash the screen and beep, to alert the user it has
finished. Since this could potentially get very annoying, it can be
turned off.

1.36 TextureStudio.guide/Flush textures on open

```
Flush textures on open
----------------------
```

   If this option is off, whenever a texture is opened from disk, it
will be added to the bottom of the texture list and all existing
textures will remain open.

   If this option is turned on, when opening a texture, all existing
textures will be flushed out (closed).

1.37 TextureStudio.guide/Multiple pass render

```
Multiple pass render
--------------------
```

   If this option is turned on, when rendering to the screen, the image
will be built up in 3 passes. The first pass renders every 4th line,
the second pass renders every 4th line and the third pass renders the
remaining lines. The idea of this option is that a general picture of
the texture can be seen much quicker than rendering line by line.

   If this option is turned off, the lines of the image will be
rendered sequentially.

1.38 TextureStudio.guide/Small parameters window

```
Small parameters window
-----------------------
```

   If this option is turned on, the parameters window used will only
contain one number gadget and text field for each parameter.

   If this option is turned off, the parameters window used will
contain slider gadgets to allow quick and easy adjustment of the
parameters. This window will not fit on a 200 height or less screen and
so the small parameters window will have to be used.

1.39 TextureStudio.guide/Use fresh render screen

```
Use fresh render screen
-----------------------
```

   If this option is turned on, the image will be rendered to a blank
screen each time.

   If this option is turned off, the image will be rendered on top of

the existing image (if it exists) and so it is easy to see any changes
made since the last render.

1.40 TextureStudio.guide/Allow transparent object

Allow transparent object

   If this option is turned off, the filter aspect of the texture and
object is ignored and so the object is totally opaque.

   If this options is turned on, objects can be transparent and filter
through light.

   The option slightly increases render time.

1.41 TextureStudio.guide/Calculate surface normals

Calculate surface normals

   If this option is turned off, the object will appear totally flat
and no shading due to the object shape, lighting or bump texture will
show up. Turning this option off is only really suitable when rendering
to a plane and using a non-bumpy texture.

   If this options is turned on, all object shape, lighting and bump
textures affect the shading as expected.

   This option slightly increases render time.

1.42 TextureStudio.guide/Full light calculations

               Full light calculations

   If this option is turned off, the object shading due to the light
source is slightly reduced and the distance of the light source from
the centre of the object is ignored.

   If this options is turned on, all the object shading from the light
source is present.

   You may wish to turn this option off if you want to render a texture
to a flat plane for use as a picture in another program, this will
eliminate any shading on the image and give a true representation of
the texture alone.

```
    See
                Light window
                    .
```

    This option slightly increases render time.

1.43   TextureStudio.guide/Anti-aliasing

```
Anti-aliasing
-------------
```

    Anti-aliasing is a process where the colours in the rendered image
are smoothed to reduce the 'jagged' affect of the pixels. The higher
the anti-aliasing setting, the less noticable the individual pixels are.

    Anti-aliasing greatly increases render time, the table below shows
the times taken to render relative to an anti-aliasing of none.

Anti-aliasing setting	Time taken to render
None	1
Low	4
Medium	9
High	16
Very high	25

    For most situations, a setting of none is sufficient and
anti-aliasing is only really recommended for the final render.

    For a demostration of anti-aliasing, render the Radar texture to a
plane firstly with no anti-aliasing and then with anti-aliasing set to
low. Notice the difference with areas of large contrast.

1.44   TextureStudio.guide/Render file format

```
                Render file format
-----------------
```

    This option sets the file format of the image file when 'Render to
file' is selected in the 'Render options' requester, See
                Render options
                    .

    If IFF-ILBM24 is selected, the image file is saved in 24 bits as a
standard Amiga compressed IFF-ILBM picture.

    If JPEG is selected, the image file is saved in 24 bits as a JPEG
compessed file. JPEG uses 'lossy' compression which offers excellent

compression ratios (very small files) but there is some loss in
quality. The quality of the image saved can be set with the 'JPEG
options' window, see
                JPEG options
                .

    If Targa is selected, the image is saved as an uncompressed 24-bit
Targa (Truevision) file.

1.45 TextureStudio.guide/JPEG options

JPEG options

    This window alters the settings when saving out in JPEG format.

    The 'Quality' setting alters the level of compression used in the
file, a high value (85 or more) will save out a good quality image but
with a relatively large file size. A small quality setting (50 or less)
will give a very small file but with significant loss in quality.

    Note: A quality setting of 25 or less may cause problems with some
JPEG readers.

1.46 TextureStudio.guide/Save prefs

                Save prefs

    This option saves out the current settings and preferences to a
prefs file on disk. The prefs filename is set by the 'PREFSFILE'
tooltype, see
                Tooltypes
                .

    TextureStudio saves out the following details in the prefs file:

    * Windows positions and whether they are opened or closed.

    * Current lighting settings.

    * Current object settings.

    * Current view settings.

    * Current render options.

    * Colours of background, light, object colour and object filter.

* TextureStudio's current screen mode, width and height.

* All items in the 'Prefs' menu.

* Current paths for textures, settings, axis, parameters, renders
  and render screens.

   After saving the prefs, when loading TextureStudio again, all these
details will be used as defaults.

1.47 TextureStudio.guide/Floating windows

                   Floating windows
****

                    Axis window

                    Colourbox window

                    Colours window

                    Infobar window

                    Light window

                    Object window

                    Parameters window

                    Texture window

                    View window

                    Render options requester
                       Notes.

   All windows can be opened or closed by selecting the relevant items
in the 'Windows' menu, see
                 Windows
                   .

   The position and status of all the windows can be saved by selecting
'Save prefs' in the 'Prefs' menu.

1.48 TextureStudio.guide/Axis window

```
Axis window
===========
```

   The axis window allows the user to alter the axis alignment, size
and position for the currently selected texture.

   The window visually shows the current axis settings in the box on
the left, the viewpoint of which is set by the 'View' cycle gadget.

   The 'View' cycle gadget alters direction from which the axis are
viewed from. Viewing the axis from the front shows the x-axis
horizontally, the z-axis vertically and the y-axis out of the screen,
when the axis have not been rotated.

   The 'Edit' cycle gadget adjusts the function of the slider gadgets
below. When 'Alignment' is selected, the slider gadgets adjust the
rotation about the x,y and z axis. When 'Size' is selected, the user
can alter the length of the x,y and z axis and when 'Position' is
selected, the slider gadgets adjust the offset of the base of the axis
from the central position.

   As the slider gadgets are moved, the axis are redrawn in real time
to give a representation of the changes made.

   The exact settings of the axis can be altered by typing the numbers
into the number gadgets.

   Each texture may have its own axis settings.

1.49   TextureStudio.guide/Colourbox window

```
Colourbox window
================
```

   The colourbox window contains a square in which the current colour
the user is editing is displayed. The current colour can either be from
the colour window or a parameter of the current texture. The colourbox
window also shows the red, green and blue values for the colour.

   The colour is displayed by setting the palette of TextureStudio's
current screen. If TextureStudio is being run on a 4-colour screen,
then the usual blue highlight colour will be altered. To set this back
to it's original colour, click on a parameter which does not affect a
colour value.

1.50   TextureStudio.guide/Colours window

```
                Colours window
==============
```

The colours window allows the user to alter the current object colour, object filter, background colour and light colour.

The cycle gadget adjusts which colour is currently being edited.

The three slider gadgets allow the red, green and blue components of the colour to be altered between 0 and 255.

The background colour alters the colour seen 'behind' a cylinder or sphere. If the current object is a plane, the background colour cannot be seen and has no affect unless the object has some transparency (see object filter below).

The light colour affects the colour of the light source. Setting it to 255,255,255 (pure white) gives a natural representation of colours of the texture. The actual intensity of the light can be altered in the 'light' window, see

                Light window

                .

The object colour affects the base colour of the object. With some textures, the base colour can be seen through the texture pattern, other textures are totally opaque and so the object's base colour cannot be seen.

The object filter adjusts the light the object filters through. Setting this to 0,0,0 gives a totally opaque object, setting it to 128,128,128 gives the appearance of slightly transparent object. Note that if the current object is a cylinder or sphere, the 'other side' of the object cannot be seen through the front half as you would expect. This is something that may be fixed in the future, see

                Future additions

                .

1.51   TextureStudio.guide/Infobar window

                Infobar window
==============

The infobar consists of a row of buttons to control aspects of the render screen, some text to display the size of the render screen currently open, a 'fuel gauge' to indicate the progress of a task and an abort button to stop a task mid-way through.

The 'Render' button starts rendering the texture(s) to the HAM screen, see

                Render

                .

The 'View' button views the HAM render screen, if it is already open. Pressing the right mouse button sends the sceen to the back again.

The 'Save' button saves the current HAM render screen to disk as a
HAM IFF-ILBM picture file. When clicking on the button, a file
requester will appear requesting the user to select a file to save to.

The 'Close' button closes the current render screen, if it is open.

The 'Options' button brings up the render options window, see

                Render options
                  .

1.52 TextureStudio.guide/Light window

                Light window
============

The light window adjusts the aspects of the light source used to
illuminated the scene.

The box to the left of the window shows the light source
(represented by a circle) relative to the centre of the object
(represented by a cross). The bounds of the box represent the bounds of
the image as viewed from the front. For example if the light source if
placed in the top left of the box, the scene will be lit from a light
source in the top left corner of the image.

The position of the light can be altered by clicking in the box to
the left of the window or by clicking on the 'Position' cycle gadget.

The 'Position' cycle gadget has nine preset positions for the light
source to be in.

The 'Intensity' slider gadget alters to intensity of the light in
percentage of the lights colour, See
                Colours window
                . Setting this to be
100% gives the actual light colour. A setting of 50% would give a light
source half as bright. A setting of 200% gives an 'over bright' light
source and will cause large highlights on the object, this gives the
effect of a very bright spotlight.

The 'Backlight' slider adjusts the backlighting or ambient light. A
setting of 0% gives an object lit from one light source with total
darkness in areas facing away from the light source. A setting of 50%
gives the impression of some ambient light in all directions around the
object, this will eliminate very dark shadows in areas that are not lit
by the main light source.

The 'Distance' gadget alters the distance of the light source from
the centre of the object in the y-axis (depth). This setting is ignored
if the preference 'Full light calculations' is off. For example if you
are rendering to a plane and the distance is set at around 20, there
will be a very defined highlight on the plane since the light is very

close to the object. If the light is moved to a distance of 200 or
more, the highlight will be barely visible.

   The 'Full light calculations' checkbox gadget alters the accuracy at
which the lighting is calculated. If the gadget is checked, the
lighting is calculated as expected and the light distance affects the
image. If the gadget is unchecked, the light distance is ignored and
the affect on the image is similar to having the light source a very
long distance away. This option was added to improve the speed of
rendering, since if the gadget is unchecked, the render speed is
slightly increased. See
                Full light calculations
                    .

1.53 TextureStudio.guide/Object window

Object window
=============

   The 'Object' cycle gadget alters which type of geometrical shape the
texture is mapped onto. If 'Plane' is selected, the texture is mapped
onto an infinitely big plane in the x and z directions, it is flat in
the y direction (depth). If 'Cylinder' is selected, the texture is
mapped onto a vertical cylinder with circular cross section in the x
and y directions. If 'Sphere' is selected, the texture is mapped onto a
sphere, with the radius being set with the gadget below.

   The 'Radius' gadget sets the radius of the object if 'Cylinder' or
'Sphere' is selected. If 'Plane' is selected, the radius is ignored.

1.54 TextureStudio.guide/Parameters window

                Parameters window
=================

   The parameters window allows the user to adjust the 16 number
parameters associated with the currently selected texture.

   There are 2 parameter windows available, the small parameters window
only allows the user to type the numbers directly in, it is designed
for a small screen. The larger parameters window has slider gadgets to
allow easier adjustment of the numbers, it is designed for a bigger
screen (interlaced). The type of parameters window can be toggled by
the 'Small parameters window' preference, see
                Small parameters window
                    .

   The minimum and maximum values for the slider gadgets in the larger
parameters window are, by default, -10 and 245 respectively. If the

text for the parameter has limits in brackets, (e.g. "Slope adjust
(-1..1)") then these are used for the limits.

1.55 TextureStudio.guide/Texture window

```
              Texture window
===============
```

   The texture window displays all the currently open textures. The
textures are ordered so that the texture at the bottom of the list has
highest priority and so is 'on top' when being mapped onto the object
(consistent with Imagine3's method).

   The 'Open' button opens a texture module from disk and adds it to
the list, see
               Open texture
                .

   The 'Close' button closes the currently selected texture and removes
it from the list, see
               Close texture
                .

   The 'Up' button moves the currently selected texture up in the list
and so gives it a lower priority.

   The 'Down' button moves the currently selected texture down in the
list and so gives it a higher priority.

1.56 TextureStudio.guide/View window

```
View window
===========
```

   This window adjusts how much of the 'world' can be seen and the
aspect ratio.

   The 'Visible width' gadget adjust the width of the 'window' in which
you see the 'world' through. For example, if you are rendering a sphere
of radius 50, and the visible width is set to 100, the left and right
edges of the sphere will just touch the edges of the render screen. The
positions of the top and bottom will depend on the aspect ratio.

   The 'X/Y aspect' gadget sets the X:Y ratio for the rendered image.
For example if you are rendering to a square screen (128 x 128), then
the X/Y aspect ratio is 128/128 = 1. However, most common screen sizes
are not square, for example a lores PAL screen is 320 x 256, therefore
for a sphere to look spherical and not 'stretched', the X/Y aspect

ratio must be set to 320/256 = 1.25. For a NTSC screen mode, the aspect
ratio will be 320/200 = 1.6.

1.57 TextureStudio.guide/Render options

                    Render options
==============

    The render options window allows the user to alter the render screen
width and height, the render image file name and whether to render to
the screen and/or a 24-bit image file.

    The 'Render to file' checkbox alters whether TextureStudio writes to
a 24-bit image file. If it is checked, the image will be saved out line
by line to a file decided by the filename in the text gadget. Clicking
on 'Choose' will bring up a file requester to make choosing a file
easier. When this option is selected, the 'Multiple pass render'
preference is ignored, see
                    Multiple pass render
                        .

    The 'Render to screen' checkbox alters whether the current
texture(s) should be rendered to a HAM render screen. The corresponding
'Choose' button chooses the render screen mode.

    The 'Width' and 'Height' number gadgets alter the width and height
of the image to be rendered. Note: The image width and height can also
be set from the screen 'Choose' requester.

1.58 TextureStudio.guide/ARexx

                    ARexx
*****

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

                    Introduction to ARexx

                    Basic ARexx

                    Command templates

                    Return values

                    Error checking

                              Common ARexx problems

                              ARexx tips

                              Example scripts

                              ARexx commands

1.59 TextureStudio.guide/Introduction to ARexx

                              Introduction to ARexx
====================

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

  1. Providing an easy and consistent method of adding macro facilities
     to programs.

  2. To allow ARexx aware programs to communicate with each other.

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

  1. Providing a ready-made script template which the user can just
     "fill in the blanks" to produce a fully working program.

  2. Providing many commands to perform commonly performed operations.
     This means the user needs to write less code in ARexx and doesn't
     need to rely on external utilities and libraries to perform the
     operations.

   Typical uses for ARexx in TextureStudio include:

  * Animation. A number of frames can be rendered, changing one or
    more parameters each frame. These frames can then be joined
    together to form an animation. See
                   RadarAnim script
                   .

  * Cataloguing. A number of textures can be rendered, one after
    another and the images saved to disk. This provides a quick
    reference as to what each texture looks like. See

                   RenderTextures script
                   .

  * Communication. ARexx can be used to link together 2 or more
    programs. For example, TextureStudio could render a texture and
    save the image out in 24 bits. An image processing program could
    then load in the image, reduce it to 32 colours with dithering and

save this new image out as a standard IFF-ILBM file for general
viewing.

   Several example files are given with TextureStudio (see

             Example scripts
             ), which can either be used directly or modified to
perform the desired operation.

1.60 TextureStudio.guide/Basic ARexx

                     Basic ARexx

===========

   This section is meant as a beginners guide to using ARexx with
TextureStudio. We cannot hope to teach you the ARexx language, although
it is only neccessary to the know the very basics to start using ARexx
scripts with TextureStudio. It is assumed that the user is familiar
with a text editor (for example MEmacs) for editing scripts.

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

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

    /* A simple ARexx program */

    REQUEST_MESSAGE TEXT '"Hello world!"'

    exit

  This shows some important things about an ARexx program:

 1. All ARexx programs must start with a comment line. A comment line
    is a line which starts with the '/*' sequence of characters and
    ends with the '*/' characters. Anything between these characters
    is ignored by ARexx.

 2. For clarity, all of TextureStudio's commands are shown
    CAPITALISED, ARexx commands are kept in lower case.
    REQUEST_MESSAGE is therefore an TextureStudio command that should
    be performed.

 3. The REQUEST_MESSAGE has some 'arguments' or 'parameters' following

it. These tell the REQUEST_MESSAGE command how to behave, in this
case they tell the command to pop up greeting message.

4. To stop an ARexx program, use the command 'exit'.

OK, lets enhance our program a little:

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

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

exit
```

From this example we learn:

1. To separate a long command line, place a comma ',' as the last
   character on the line. This tells ARexx to treat the next line as
   a continuation of the previous. Two line breaks are used in the
   above example.

2. ARexx loves to evaluate things. If we want to stop ARexx
   evaluating variables, the variable should be enclosed in single
   quotes ' ' '.

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

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

would be evaluated to:

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

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

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

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

options results

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

if RESULT == 0 then
    REQUEST_MESSAGE TEXT '"Sorry, I was trying very hard."'
```

```
      else if RESULT == 2 then
         REQUEST_MESSAGE TEXT '"It gets better."'
      else do
         REQUEST_MESSAGE TEXT '"We like happy users."'
         REQUEST_MESSAGE TEXT '"Treat yourself to a coffee."'
         end

      exit
```

   This shows:

1. Normally ARexx ignores the values returned by commands. To allow
   commands to return values, use "options results"; this is done for
   you in the blank ARexx script.

2. Unless otherwise specified (see
                Return values
) commands return the
   results of their operation in a variable called "RESULT". The
   command REQUEST_MESSAGE returns the value of the button that the
   user pressed. It is this value that we can test for.

3. The 'if' tests are shown above. Note that if you only want to
   perform one operation as part of the 'if', you can just place it
   after the 'then'. If you wish to perform more operations, they
   must be placed in a 'do / end' set.

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

1.61 TextureStudio.guide/Command templates

```
Command templates
=================
```

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

   Commands are always of the form:

      command [options]

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

      OPEN FILE/A,FLUSH/S

   The commands and options are not case sensitive, therefore 'OPEN',

'Open' or 'open' can be used to open a file. The options after the command name are separated by commas, and are named (e.g. FILE or FLUSH are option names). After the name, follows an optional modifier (e.g. /A or /S are modifiers) which describes what type of information the option specifies.

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

The following modifiers are used:

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

Multiple strings (/M)
     Many strings can be specified if an option uses this modifier.

Numeric (/N)
     Numeric options allow both positive and negative integers. Floating point numbers (decimals) are given as strings in TextureStudio.

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

Keyword (/K)
     A keyword option shows that the option name must be used to set this option.

Always (/A)
     This option must always be included in this command.

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

     OPEN FILE/A,FLUSH/S

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

The following would load in a texture module called 'Radar.itx' from the current directory and add it to any other textures that are already open.

     OPEN "Radar.itx"

The following would open a texture called 'Target.itx' from a drawer called 'Textures' in the current directory. It would also flush out (close) any textures that are already open.

```
    OPEN "Textures/Target.itx" FORCE

    COLOUR_SET COLOUR/A,R/N,G/N,B/N
```

This command is used to set the colour of the background, light, object or object filter.

'COLOUR' is a string which must always be supplied, it determines which of the above colours is to be altered.

'R/N','G/N','B/N' are integer numbers which set the red, green and blue components of the colour respectively. They are optional.

The following sets the background colour to be bright red.

```
    COLOUR_SET BACKGROUND 255 0 0
```

The following sets the green component of the light to be 128 (half brightness)

```
    COLOUR_SET LIGHT G 128
```

The following is an error, an incorrect 'COLOUR' parameters is used.

```
    COLOUR_SET 255 255 255
```

The following is not an error, but will do nothing.

```
    COLOUR_SET OBJECTCOLOUR
```

1.62 TextureStudio.guide/Return values

```
                Return values
=============
```

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

```
    options results
```

be placed near the start of the ARexx script.

Commands may return either strings, numbers or arrays of either. By default, all ARexx commands return their values in a variable called "RESULT". This is fine if the command returns a single number or string. For example, the following call to the FILE_JOIN command (see

```
            FILE_JOIN
            ) would return the string "T:Image.ilbm" in the RESULT
variable:
```

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

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

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

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

    LIGHT_GET

and RESULT might look something like this:

    −0.700000 −0.700000 100 0 150.000000

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

    LIGHT_GET STEM LIGHT.

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

    LIGHT.X_POS   = −0.700000
    LIGHT.Z_POS   = −0.700000
    LIGHT.INTENSITY = 100
    LIGHT.BACKLIGHT = 0
    LIGHT.DISTANCE = 150.000000

Now you can refer easily to the returned values.

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

    STEMNAME.RESULTNAME.NUMBER

with the variable STEMNAME.RESULTNAME.COUNT holding the number of
returned results. This example would display a multi−select file
requester and return the selected files.

    REQUEST_MULTIFILE PATHPART "Textures" PATTERN "#?.itx" STEM MATCHED.

which might return the following:

    MATCHED.FILES.COUNT = 4
    MATCHED.FILES.0 = Blades.itx
    MATCHED.FILES.1 = Radar.itx
    MATCHED.FILES.2 = Target.itx

```
    MATCHED.FILES.3 = WarningStripes.itx
```

1.63 TextureStudio.guide/Error checking

```
Error checking
==============
```

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

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

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

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

   If, for example the user was to try and close a render screen that
wasn't open, RC and RC2 would be set to the following:

```
    RC = 10
    RC2 = "RENDERSCREEN_CLOSE, No render screen open."
```

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

```
    COLSE
```

   the following values would be set:

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

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

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

```
    signal on error
```

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

   The user may wish to turn off the automatic error checking to

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

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

signal off error

/* Open the requester */

REQUEST_FILE

/* Check for the error condition */

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

1.64   TextureStudio.guide/Common ARexx problems

```
            Common ARexx problems
====================
```

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

              ARexx problem 2
                I can't set the same variable twice with VAR
```

1.65   TextureStudio.guide/ARexx problem 1

```
ARexx problem 1
--------------
```

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

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

   Consider the following example:

```
      REQUEST_MESSAGE TEXT "Hello"
```

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

```
      REQUEST_MESSAGE TEXT Hello
```

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

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

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

```
      REQUEST_MESSAGE TEXT Hello world
```

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

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

   would send TextureStudio the following command:

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

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

1.66 TextureStudio.guide/ARexx problem 2

ARexx problem 2

      "I can't set the same variable twice with VAR"

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

   For example, the following won't work:

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

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

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

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

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

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

1.67 TextureStudio.guide/ARexx tips

```
                ARexx tips
==========
```

```
               ARexx tip 1
                  Shortening command names
```

1.68 TextureStudio.guide/ARexx tip 1

```
ARexx tip 1
-----------
```

      "Shortening command names"

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

   * This behaviour may be removed in a future version of
     TextureStudio. Therefore we recommend that in ARexx scripts, the
     full command names should be used.

   * If the shortened command name is ambiguous, the first matching
     command will be executed. For example, if the shortened command
     'REQUEST' is used, 'REQUEST_DIR' will be executed.

1.69 TextureStudio.guide/Example scripts

```
               Example scripts
===============
```

   The following scripts require TextureStudio to already be running.

   To run the script, either double click on the script icon from the
Workbench, or from the shell, type

```
    rx RadarAnim.tsrx
 for example.
```

```
             RadarAnim script
                    Example of animation with ARexx

             RenderTextures script
                Useful cataloguing script

             RenderTexturesIS script
              Texture rendering and image processing

             RotatePlanetAnim script
              Another animation example
```

1.70   TextureStudio.guide/RadarAnim script

```
RadarAnim script
----------------
```

```
Description
     The user will be asked to select the 'Radar.itx' texture, the
     default directory will be the user's current texture path.

     The user will then be prompted to select a destination directory
     where the rendered pictures will be saved to.

     Finally, the user will be asked to select the number of frames
     from 10, 25, 50 or 75.

     TextureStudio will then render the frames of the animation as HAM
     screens to the destination directory.

     The images will be rendered to a screen at the back, this can be
     brought to the front if you wish, to see the image being rendered.

     The resulting frames can be joined together into an animation
     using a suitable program.

     The animation shows a radar scanner scanning through 360 degrees.
```

```
Known bugs
     None.
```

1.71   TextureStudio.guide/RenderTextures script

```
RenderTextures script
---------------------
```

Description
     The user will then be prompted to select some texture modules to
     render. By shift-clicking on the files, many textures can be
     selected.

     The user will then be prompted to give a destination directory for
     the resulting render screen files created.

     The script renders each texture in turn and then saves the HAM
     render screen to the given directory. All the settings e.g.
     object, light etc. are kept as the current settings. This is
     particularly useful for keeping a record of all the textures and
     what they look like for quick reference.

Known bugs
     None.

1.72 TextureStudio.guide/RenderTexturesIS script

```
                RenderTexturesIS script
-----------------------
```

   The ARexx script requires ImageStudio, see
                ImageStudio
                .

Description
     This scripts is similar to the 'RenderTextures' script but
     requires both TextureStudio and ImageStudio running at the same
     time.

     This scripts renders each texture, saves it out as a 24-bit for
     ImageStudio which then reduces the image down to 32 colour with
     dithering and then saves the new image as an IFF-ILBM file.

     See
                RenderTextures script
                .

Known bugs
     None.

1.73 TextureStudio.guide/RotatePlanetAnim script

```
                RotatePlanetAnim script
```

Description
      This script requires the 'GasGiant' texture supplied with Imagine3.

      The script is similar to the 'RadarAnim' script but produces an
      animation which looks similar to a planet (like Jupiter) spinning
      on its axis. See
                   RadarAnim script
                      .

Known bugs
      None.

1.74 TextureStudio.guide/ARexx commands

                      ARexx commands
==============

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

                      AXIS_GET

                      AXIS_SET

                      CLOSE

                      COLOUR_GET

                      COLOUR_SET

                      FILE_JOIN

                      FILE_SPLIT

                      GUI_BLOCK

                      GUI_UNBLOCK

                      LIGHT_GET

                      LIGHT_SET

                      OBJECT_GET

                      OBJECT_SET

                      OPEN

                      PARAMETER_GET

```
               PARAMETER_SET

               PREFS_GET

               PREFS_SET

               RENDER

               RENDEROPTIONS_GET

               RENDEROPTIONS_SET

               RENDERPATH_GET

               RENDERSCREENPATH_GET

               RENDERSCREEN_CLOSE

               RENDERSCREEN_SAVE

               RENDERSCREEN_VIEW

               REQUEST_DIR

               REQUEST_FILE

               REQUEST_MESSAGE

               REQUEST_MULTIFILE

               SCREEN_BACK

               SCREEN_FRONT

               TEXTUREPATH_GET

               TEXTURES_GET

               TEXTURE_SELECT

               VIEW_GET

               VIEW_SET
```

1.75 TextureStudio.guide/AXIS_GET

```
               AXIS_GET
--------

Command
     AXIS_GET

Parameters template
```

```
    None.


Return template
     X_ALIGNMENT,Y_ALIGNMENT,Z_ALIGNMENT,
     X_SIZE,Y_SIZE,Z_SIZE,
     X_POSITION,Y_POSITION,Z_POSITION


Description
     This command returns the axis settings for the currently selected
     texture.


Parameters
     None.


Returns
    X_ALIGNMENT,Y_ALIGNMENT,Z_ALIGNMENT
          These variables contain the axis alignment as floating point
          numbers.


    X_SIZE,Y_SIZE,Z_SIZE
          These variables contain the axis size as floating point
          numbers.


    X_POSITION,Y_POSITION,Z_POSITION
          These variables contain the axis position as floating point
          numbers.


Errors
     rc = 0 if the operation was successful.


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


Example
     The following example gets the current axis settings and puts them
     in a stem called axis. The settings can then be accessed by
     'axis.x_alignment', 'axis.y_alignment' ....


          AXIS_GET STEM 'axis.'


Known bugs
     None.


     See

                AXIS_SET
                     .
```

1.76 TextureStudio.guide/AXIS_SET

```
                AXIS_SET
--------


Command
```

```
     AXIS_SET
```

Parameters template
```
     X_ALIGNMENT,Y_ALIGNMENT,Z_ALIGNMENT,
     X_SIZE,Y_SIZE,Z_SIZE,
     X_POSITION,Y_POSITION,Z_POSITION
```

Return template
```
     None.
```

Description
```
     This command sets the axis settings for the currently selected
     texture.
```

Parameters
```
     X_ALIGNMENT,Y_ALIGNMENT,Z_ALIGNMENT
          These variables contain the axis alignment as floating point
          numbers.

     X_SIZE,Y_SIZE,Z_SIZE
          These variables contain the axis size as floating point
          numbers.

     X_POSITION,Y_POSITION,Z_POSITION
          These variables contain the axis position as floating point
          numbers.
```

Returns
```
     Nothing.
```

Errors
```
     rc = 0 if the operation was successful.

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

Example
```
     The following example sets the axis alignment so that the z-axis
     is perpendicular to the surface of a vertical plane.

          AXIS_SET X_ALIGNMENT -90 Y_ALIGNMENT 0 Z_ALIGNMENT 0
```

Known bugs
```
     None.

     See
               AXIS_GET
                    .
```

1.77 TextureStudio.guide/CLOSE

```
CLOSE
-----
```

```
Command
     CLOSE


Parameters template
     NAME,ALL/S


Return template
     None.


Description
     This commands closes one or all of the currently open texures.

     If no parameters are supplied, only the currently selected texture
     is closes.


Parameters
    NAME
         NAME is the name of the texture to be closed. This can be
         shortened to the first few characters in the name.

         The NAME string is case insensitve.


    FLUSH/S
         If FLUSH is given, all currently open textures are closed.


Returns
     Nothing.


Errors
     rc = 0 if the operation was successful.

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


Example
     The following example closes the currently selected texture.

         CLOSE

     The next example closes any textures beginning with 'Rad', which
     could close 'Radar', 'RadCheks' ....

         CLOSE "Rad"

     The following example flushes out all existing textures.

         CLOSE ALL


Known bugs
     None.
```

1.78   TextureStudio.guide/COLOUR_GET

```
                    COLOUR_GET
----------


Command
     COLOUR_GET


Parameters template
     COLOUR/A


Return template
     R/N,G/N,B/N


Description
     COLOUR_GET gets the red, green and blue values of the given colour.


Parameters
    COLOUR/A
          This specifies which colour is to be returned. Valid strings
          are:

          BACKGROUND (shortest abbreviation: BACK)
          LIGHT (shortest abbreviation: LIGHT)
          OBJECTCOLOUR (shortest abbreviation: OBJECTCOL)
          OBJECTFILTER (shortest abbreviation: OBJECTFIL)
Returns
    R/N,G/N,B/N
          These are the red, green and blue componets of the given
          COLOUR, the numbers are between 0 and 255.


Errors
     rc = 0 if the operation was successful.

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


Example
     The following is an example where the colour of the light source
     is returned in the 'light.' stem.

          COLOUR_GET LIGHT STEM 'light.'


Known bugs
     None.

     See

                    COLOUR_SET

                    .

     See

                    Colours window

                    .
```

1.79 TextureStudio.guide/COLOUR_SET

```
                COLOUR_SET
----------

Command
     COLOUR_SET

Parameters template
     COLOUR/A,R/N,G/N,B/N

Return template
     None.

Description
     This command sets the red, green and blue components of the given
     COLOUR.

Parameters
    COLOUR/A
         This specifies which colour is to be altered. Valid strings
         are:

         BACKGROUND (shortest abbreviation: BACK)
         LIGHT (shortest abbreviation: LIGHT)
         OBJECTCOLOUR (shortest abbreviation: OBJECTCOL)
         OBJECTFILTER (shortest abbreviation: OBJECTFIL)
    R/N,G/N,B/N
         These are the red, green and blue componets for the given
         COLOUR, the numbers should be between 0 and 255.

Returns
     Nothing.

Errors
     rc = 0 if the operation was successful.

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

Example
     The following example sets the colour of the background to be
     white.

         COLOUR_SET BACKGROUND 255 255 255

     The following example turns the red component of the light source
     off.

         COLOUR_SET LIGHT R 0

Known bugs
     None.

     See
                COLOUR_GET
```

.

See

Colours window

.

1.80 TextureStudio.guide/FILE_JOIN

                FILE_JOIN

Command
     FILE_JOIN

Parameters template
     PATHPART/A, FILEPART/A

Return template
     FILE

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

Parameters
    PATHPART/A
         The path (directory) part of the filename to be created.

    FILEPART/A
         The file part of the filename to be created.

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

Errors
     rc = 0 if the operation was successful.

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

Example
     The following creates the filename "Textures/Radar.itx" from the
     seperate path and fileparts – the result is put in a pop up
     requester:

          FILE_JOIN PATHPART "Textures" FILEPART "Radar.itx"

          REQUEST_MESSAGE TEXT '"'RESULT'"'

     The following creates the filename "Ram:Radar.itx" from the
     seperate path and fileparts (note how the '/' seperater is not

```
    needed) - the result is put in a pop up requester:

        FILE_JOIN PATHPART "Ram:" FILEPART "Radar.itx"

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

Known bugs
    None.

    See
            FILE_SPLIT
                .

1.81 TextureStudio.guide/FILE_SPLIT

```
                FILE_SPLIT
----------

Command
    FILE_SPLIT

Parameters template
    FILE/A

Return template
    PATHPART, FILEPART

Description
    Splits the given filename into seperate path and file parts.

Parameters
   FILE/A
        The full filename to be split.

Returns
   PATHPART
        The path (directory) part of the filename.

   FILEPART
        The file part of the filename.

Errors
    rc = 0 if the operation was successful.

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

Example
    The following seperates the filename "Textures/Radar.itx" into
    seperate path and fileparts - the result is put in a pop up
    requester:

        FILE_SPLIT FILE "Textures/Radar.itx" STEM FILENAME.
```

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

    The following seperates the filename "Ram:Radar.itx" into seperate
    path and fileparts - the result is put into the default settings
    of a file requester:

            FILE_SPLIT FILE "Ram:Radar.itx" STEM FILENAME.

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

Known bugs
     None.

     See
               FILE_JOIN
                  .

1.82 TextureStudio.guide/GUI_BLOCK

                GUI_BLOCK

Command
     GUI_BLOCK

Parameters template
     None.

Return template
     None.

Description
     Blocks any input to all of TextureStudio's windows. Stops the user
     from altering anything whilst a script is running.

     Note: Always remember to unblock the GUI once you have finished
     the script.

Parameters
     None.

Returns
     Nothing.

Errors
     rc = 0 if the operation was successful.

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

Example
     The following example blocks input to all the windows.

          GUI_BLOCK

Known bugs
     None.

     See

               GUI_UNBLOCK

                    .

1.83   **TextureStudio.guide/GUI_UNBLOCK**

               GUI_UNBLOCK

Command
     GUI_UNBLOCK

Parameters template
     None.

Return template
     None.

Description
     Unblocks all the windows after a GUI_BLOCK call. Allows the user
     to interact with the program again.

     Note: it is safe to call GUI_UNBLOCK, even if the windows are not
     currently blocked by GUI_BLOCK.

Parameters
     None.

Returns
     Nothing.

Errors
     rc = 0 if the operation was successful.

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

Example
     The following unblocks all the windows.

          GUI_UNBLOCK

Known bugs
     None.

```
    See
              GUI_BLOCK
                 .
```

1.84 TextureStudio.guide/LIGHT_GET

```
              LIGHT_GET
---------

Command
     LIGHT_GET

Parameters template
     None.

Return template
     X_POS,Z_POS,INTENSITY/N,BACKLIGHT/N,POSITION,DISTANCE

Description
     Gets the current settings about the light source.

Parameters
     None.

Returns
    X_POS,Z_POS
         These give the position of the light relative to the width of
         the scene visible. The floating point numbers are between -1
         and 1.

         -1 corresponds to the extreme left and bottom.   1
         corresponds to the extreme right and top.

    INTENSITY
         Gives the current light intensity from 0 to 200%.

    BACKLIGHT
         Gives the backlighting setting from 0 to 100%.

    POSITION
         This is a string containing the current setting of the
         'Position' slider in the light window. Can be 'TOPLEFT',
         'TOP', 'TOPRIGHT', 'LEFT', 'CENTRE', 'RIGHT', 'BOTTOMLEFT',
         'BOTTOM', 'BOTTOMRIGHT' or 'CUSTOM'.

    DISTANCE
         Gives the distance of the light source from the centre of the
         scene.

Errors
    rc = 0 if the operation was successful.

    rc = 10 if the operation failed for any reason, rc2 will contain a
```

```
    string describing the problem.
```

Example
     The following example gets the current light settings and puts
     them in a stem called 'light.'.

          LIGHT_GET STEM 'light.'

Known bugs
     None.

     See

               LIGHT_SET

               .

     See

               Light window

               .

1.85   TextureStudio.guide/LIGHT_SET

               LIGHT_SET

Command
     LIGHT_SET

Parameters template
     X_POS,Z_POS,INTENSITY/N,BACKLIGHT/N,POSITION,DISTANCE

Return template
     None.

Description
     Sets the current settings for the light source.

Parameters
    X_POS,Z_POS
          These set the position of the light relative to the width of
          the scene visible. The floating point numbers must be between
          -1 and 1.

          -1 corresponds to the extreme left and bottom.   1
          corresponds to the extreme right and top.

    INTENSITY
          Sets the current light intensity from 0 to 200%.

    BACKLIGHT
          Sets the backlighting setting from 0 to 100%.

    POSITION
          This is a string containing the current setting of the

'Position' slider in the light window. Must be 'TOPLEFT',
'TOP', 'TOPRIGHT', 'LEFT', 'CENTRE', 'RIGHT', 'BOTTOMLEFT',
'BOTTOM', 'BOTTOMRIGHT' or 'CUSTOM'.

   DISTANCE
        Sets the distance of the light source from the centre of the
        scene.

Returns
     Nothing.

Errors
     rc = 0 if the operation was successful.

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

Example
     The following example sets the light source to be in the extreme
     top right position.

        LIGHT_SET X_POS 1 Z_POS 1 POSITION "CUSTOM"

Known bugs
     None.

     See
                LIGHT_GET
                .

     See
                Light window
                .

1.86 TextureStudio.guide/OBJECT_GET

                OBJECT_GET

Command
     OBJECT_GET

Parameters template
     None.

Return template
     OBJECT,RADIUS

Description
     Gets the current object type and radius.

Parameters
     None.

```
Returns
    OBJECT
          Name of the type of object, can be 'PLANE', 'CYLINDER' or
          'SPHERE'.

    RADIUS
          A floating point number representing the radius of the object.

Errors
     rc = 0 if the operation was successful.

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

Example
     The following gets the current object settings and places them
     into a stem called 'object.'.

          OBJECT_GET STEM 'object.'

Known bugs
     None.

     See

                OBJECT_SET

                .

     See

                Object window

                .
```

1.87   TextureStudio.guide/OBJECT_SET

```
                OBJECT_SET
----------

Command
     OBJECT_SET

Parameters template
     OBJECT,RADIUS

Return template
     None.

Description
     Sets the current object type and radius.

Parameters
    OBJECT
          Name of the type of object, can be 'PLANE', 'CYLINDER' or
          'SPHERE'.
```

```
    RADIUS
          A floating point number representing the radius of the object.

Returns
     Nothing.

Errors
     rc = 0 if the operation was successful.

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

Example
     The following sets the current object to be a sphere of radius 30
     units.

          OBJECT_SET "SPHERE" 30

Known bugs
     None.

     See
                OBJECT_GET
                .

     See
                Object window
                .
```

1.88 TextureStudio.guide/OPEN

```
                OPEN
----

Command
     OPEN

Parameters template
     FILE/A,FLUSH/S

Return template
     None.

Description
     This command opens a standard Imagine3 texture module. See

                Open texture
                .

Parameters
    FILE/A
          This specifies the full filename of the texture module to be
```

```
        loaded.

   FLUSH/S
        This specifies whether any existing textures should be
        flushed out before loading in the specified file. Note that
        the 'Flush textures on open' preference is ignored from
        ARexx, see
            Flush textures on open
            .
```

Returns
     Nothing.

Errors
     rc = 0 if the operation was successful.

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

Example
     The following opens a texture module called 'Radar.itx' from the
     'Textures' drawer and closes all other textures.

         OPEN "Textures/Radar.itx" FLUSH

     The following loads the 'Target.itx' file from the user's current
     texture directory.

     TEXTUREPATH_GET VAR 'texture_path'

     FILE_JOIN '"'texture_path'"' '"'Target.itx'"' VAR 'texture_file'

     OPEN '"'texture_file'"'

Known bugs
     None.

1.89 TextureStudio.guide/PARAMETER_GET

                  PARAMETER_GET

Command
     PARAMETER_GET

Parameters template
     INDEX/N/A

Return template
     VALUE,TEXT

Description
     This command is used to get a parameter value (floating point

number) and text field (name of parameter). It gets the parameters
from the currently selected texture.

Parameters
    INDEX/N/A
          This is the index number of the parameter and must be from 0
          to 15.

Returns
    VALUE
          This is the floating point number associated with the given
          parameter INDEX.

    TEXT
          This is the text field of the parameter, e.g. 'Colour Red'.

Errors
    rc = 0 if the operation was successful.

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

Example
    The following gets the fifth (number 4) parameter from the current
    texture and returns the value and text in a stem called 'param.'.

          PARAMETER_GET 4 STEM 'param.'

    The following example adds 1 onto the first parameter.

          PARAMETER_GET 0 STEM 'param.'

          new_value = param.value + 1

          PARAMETER_SET 0 '"'new_value'"'

Known bugs
    None.

    See

              PARAMETER_SET

                  .

    See

              Parameters window

                  .

1.90   TextureStudio.guide/PARAMETER_SET

              PARAMETER_SET

Command

```
    PARAMETER_SET
```

Parameters template
    INDEX/N/A,VALUE/A

Return template
    None.

Description
    Sets the given parameter to a given value for the currently
    selected texture.

Parameters
    INDEX/N/A
        This is the index number of the parameter and must be from 0
        to 15.

    VALUE/A
        This is a floating point number associated with the given
        parameter INDEX.

Returns
    Nothing.

Errors
    rc = 0 if the operation was successful.

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

Example
    The following sets the last parameter (number 15) to be 0

        PARAMETER_SET 15 0

Known bugs
    None.

    See
            PARAMETER_GET
            .

    See
            Parameters window
            .

1.91 TextureStudio.guide/PREFS_GET

```
            PREFS_GET
```

Command
    PREFS_GET

Parameters template
     None.

Return template
     ALLOWTRANSOBJECT/N,CALCSURFNORMALS/N,FULLLIGHTCALCS/N,
     ANTIALIASING,FILEFORMAT,JPEG_QUALITY/N

Description
     Gets the current preferences.

Parameters
     None.

Returns
    ALLOWTRANSOBJECT/N
          0 for off, 1 for on. See
               Allow transparent object
               .

    CALCSURFNORMALS/N
          0 for off, 1 for on. See
               Calculate surface normals
               .

    FULLLIGHTCALCS/N
          0 for off, 1 for on. See
               Full light calculations
               .

    ANTIALIASING
          The anti-aliasing mode, can be 'NONE', 'LOW', 'MEDIUM',
          'HIGH' or 'VERYHIGH'. See
               Anti-aliasing
               .

    FILEFORMAT
          The current render file format. Can be 'IFF-ILBM', 'JPEG' or
          'TARGA'. See
               Render file format
               .

    JPEG_QUALITY
          The quality of the JPEG output. See
               JPEG options
               .

Errors
     rc = 0 if the operation was successful.

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

Example
     The following returns the current preference settings and returns
     them in a stem called 'pref.'.

```
            PREFS_GET STEM 'pref.'


Known bugs
     None.


     See

                PREFS_SET

                .
```

1.92 TextureStudio.guide/PREFS_SET

```
                PREFS_SET
---------

Command
     PREFS_SET

Parameters template
     ALLOWTRANSOBJECT/N,CALCSURFNORMALS/N,FULLLIGHTCALCS/N,
     ANTIALIASING,FILEFORMAT,JPEG_QUALITY/N

Return template
     None.

Description
     Sets any given preference(s).

Parameters
   ALLOWTRANSOBJECT/N
        0 for off, 1 for on. See
                Allow transparent object
                .

   CALCSURFNORMALS/N
        0 for off, 1 for on. See
                Calculate surface normals
                .

   FULLLIGHTCALCS/N
        0 for off, 1 for on. See
                Full light calculations
                .

   ANTIALIASING
        Sets the anti-aliasing mode, can be 'NONE', 'LOW', 'MEDIUM',
        'HIGH' or 'VERYHIGH'. See
                Anti-aliasing
                .

   FILEFORMAT
        Sets the current render file format. Can be 'IFF-ILBM',
        'JPEG' or 'TARGA'. See
                Render file format
```

.

    JPEG_QUALITY
            Sets the quality of the JPEG output. See
                    JPEG options
                    .

Returns
     Nothing

Errors
     rc = 0 if the operation was successful.

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

Example
     The following sets the 'Allow transparent object' preference off
     and sets the current render file format to be JPEG.

          PREFS_SET ALLOWTRANSOBJECT 0 FILEFORMAT "JPEG"

Known bugs
     None.

     See
                    PREFS_GET
                    .

1.93 TextureStudio.guide/ARexx_RENDER

                    RENDER

Command
     RENDER

Parameters template
     TOBACK/S

Return template
     None.

Description
     Renders the current texture(s) to the HAM render screen.

Parameters
     TOBACK/S
            If this is specified, the render screen is not brought to the
            front when rendering, it remains at the back.

Returns
     Nothing.

Errors
     rc = 0 if the operation was successful.

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

Example
     The following renders the current texture(s) to the HAM render
     screen in the background.

          RENDER TOBACK

Known bugs
     None.

     See

               Render

                    .

1.94   TextureStudio.guide/RENDEROPTIONS_GET

                    RENDEROPTIONS_GET

Command
     RENDEROPTIONS_GET

Parameters template
     None.

Return template
     WIDTH/N,HEIGHT/N,MODEID/N,TOSCREEN/N,TOFILE/N,FILE

Description
     Gets the current render options.

Parameters
     None.

Returns
    WIDTH/N
          The width of the render image in pixels.

    HEIGHT/N
          The height of the render image in pixels.

    MODEID/N
          A standard Amiga mode ID describing the screen mode.

    TOSCREEN/N
          Whether the image should be rendered to the screen. 0 for
          off, 1 for on.

```
    TOFILE/N
          Whether the image should be rendered to a file, given by
          FILE. 0 for off, 1 for on.

    FILE
          The filename of the render file.

Errors
     rc = 0 if the operation was successful.

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

Example
     The following gets the current render options and returns them in
     a stem called 'renderopts.'

          RENDEROPTIONS_GET STEM 'renderopts.'

Known bugs
     None.

    See
                RENDEROPTIONS_SET
                .

    See
                Render options
                .
```

1.95 TextureStudio.guide/RENDEROPTIONS_SET

```
                RENDEROPTIONS_SET
-----------------

Command
     RENDEROPTIONS_SET

Parameters template
     WIDTH/N,HEIGHT/N,MODEID/N,TOSCREEN/N,TOFILE/N,FILE

Return template
     None.

Description
     Sets any of the current render options.

Parameters
    WIDTH/N
          The width of the render image in pixels.

    HEIGHT/N
```

        The height of the render image in pixels.

    MODEID/N
        A standard Amiga mode ID describing the screen mode.

    TOSCREEN/N
        Whether the image should be rendered to the screen. 0 for
        off, 1 for on.

    TOFILE/N
        Whether the image should be rendered to a file, given by
        FILE. 0 for off, 1 for on.

    FILE
        The filename of the render file.

Returns
    Nothing.

Errors
    rc = 0 if the operation was successful.

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

Example
    The following sets the render width to be 320, the height to be
    256 and the screen mode to be LowRes (mode ID = 0).

        RENDEROPTIONS_SET WIDTH 320 HEIGHT 256 MODEID 0

    The following sets the render options so that TextureStudio will
    render to a file called 'Renders/Explosion.JPG' in JPEG format and
    not to render to the screen, see
            PREFS_SET
            .

        RENDEROPTIONS_SET TOSCREEN 0 TOFILE 1 FILE "Renders/Explosion.JPG"

        PREFS_SET FILEFORMAT "JPEG"

Known bugs
    None.

    See
            RENDEROPTIONS_GET
            .

    See
            Render options
            .

1.96   TextureStudio.guide/RENDERPATH_GET

```
              RENDERPATH_GET
--------------
```

Command
     RENDERPATH_GET

Parameters template
     None.

Return template
     PATH

Description
     This returns the user's current directory (path) for render files.

     This command is useful used in conjunction with FILE_JOIN, see

              FILE_JOIN
                 .

Parameters
     None.

Returns
    PATH
         The user's render path.

Errors
     rc = 0 if the operation was successful.

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

Example
     The following gets the user's render path and returns it in a
     varaible called 'pathpart'

          RENDERPATH_GET VAR 'pathpart'

Known bugs
     None.

1.97   TextureStudio.guide/RENDERSCREENPATH_GET

```
              RENDERSCREENPATH_GET
--------------------
```

Command
     RENDERSCREENPATH_GET

```
Parameters template
     None.

Return template
     PATH

Description
     This returns the user's current directory (path) for render
     screens.

Parameters
     None.

Returns
    PATH
          The user's render screen path.

          This command is useful used in conjunction with FILE_JOIN,
          see
               FILE_JOIN
                .

Errors
     rc = 0 if the operation was successful.

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

Example
     The following gets the user's render screen path and returns it in
     a varaible called 'pathpart'

          RENDERSCREENPATH_GET VAR 'pathpart'

Known bugs
     None.
```

1.98 TextureStudio.guide/RENDERSCREEN_CLOSE

```
                    RENDERSCREEN_CLOSE
     ------------------

Command
     RENDERSCREEN_CLOSE

Parameters template
     None.

Return template
     None.

Description
     Closes the current render screen. Returns an error if there is no
```

    render screen currently open.

    ARexx equivelent to the 'Close' gadget in the Infobar window.

Parameters
    None.

Returns
    Nothing.

Errors
    rc = 0 if the operation was successful.

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

Example
    The following closes the current render screen.

        RENDERSCREEN_CLOSE

Known bugs
    None.

    See

            Infobar window
            .

1.99   TextureStudio.guide/RENDERSCREEN_SAVE

            RENDERSCREEN_SAVE

Command
    RENDERSCREEN_SAVE

Parameters template
    FILE/A,FORCE/S

Return template
    None.

Description
    Saves the current render screen to the given FILE. Returns an
    error if there is no render screen currently open.

    ARexx equivelent to the 'Save' gadget in the Infobar window.

Parameters
   FILE/A
        Specifies the filename to save the HAM render screen as.

    FORCE/S

```
          If this is specified, files are automatically overwritten and
          no warning is given to the user.

Returns
     Nothing.

Errors
     rc = 0 if the operation was successful.

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

Example
     The following example saves the current render screen to a file
     called 'RenderScreens/Water.HAM8'. If the file already exists, it
     is overwritten.

          RENDERSCREEN_SAVE "RenderScreens/Water.HAM8" FORCE

Known bugs
     None.

     See

               Infobar window
               .
```

1.100 TextureStudio.guide/RENDERSCREEN_VIEW

```
                    RENDERSCREEN_VIEW
-----------------

Command
     RENDERSCREEN_VIEW

Parameters template
     None.

Return template
     None.

Description
     Views the current render screen. Returns an error if there is no
     render screen currently open.

     ARexx equivelent to the 'View' gadget in the Infobar window.

Parameters
     None.

Returns
     Nothing.

Errors
```

```
      rc = 0 if the operation was successful.

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

Example
      The following closes the current render screen.

            RENDERSCREEN_CLOSE

Known bugs
      None.

      See

                  Infobar window
                  .

1.101   TextureStudio.guide/REQUEST_DIR

                        REQUEST_DIR

Command
      REQUEST_DIR

Parameters template
      PATHPART, TITLE

Return template
      PATHPART

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

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

      In common with all TextureStudio requesters, if the user presses
      'Cancel', an error message is returned. For the script to trap
      this error, global error checking must be turned off. See

                  Error checking
                  , for more information.

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

      TITLE
            The text for the title bar of the requester.

Returns

```
    PATHPART
         The selected path from the requester.

Errors
     rc = 0 if the operation was successful.

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

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

          REQUEST_DIR STEM DIRINFO.

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

          REQUEST_DIR PATHPART "T:" STEM DIRINFO.

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

Known bugs
     None.
```

1.102 TextureStudio.guide/REQUEST_FILE

```
                    REQUEST_FILE
------------

Command
     REQUEST_FILE

Parameters template
     PATHPART, FILEPART, PATTERN, TITLE

Return template
     FILE

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

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

     In common with all TextureStudio requesters, if the user presses
     'Cancel', an error message is returned. For the script to trap
     this error, global error checking must be turned off. See

               Error checking
               , for more information.
```

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

    FILEPART
          The default filename to put in the requester.

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

    TITLE
          The text for the title bar of the requester.

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

Errors
    rc = 0 if the operation was successful.

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

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

          REQUEST_FILE STEM FILEINFO.

    The following puts up a file requester with the result being
    printed in a message requester. The default file is
    "Textures/Radar.itx":

          REQUEST_FILE PATHPART "Textures" FILEPART "Radar.itx",
            STEM FILEINFO.

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

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

          REQUEST_FILE PATTERN "#?.itx"

Known bugs
    None.

1.103 TextureStudio.guide/REQUEST_MESSAGE

                    REQUEST_MESSAGE

Command
      REQUEST_MESSAGE

Parameters template
      TEXT/A, BUTTONTEXT, AUTOCANCEL/S, TITLE

Return template
      NUMBER/N

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

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

         1. The "Negative" response should be placed on the far
            right-hand button. For example, the 'Cancel' button should be
            placed here.

         2. The "Positive" response should be placed on the far left-hand
            button. For example, the 'OK' button should be placed here.

         3. Try to word your requesters to keep the positive and negative
            text as "OK" and "Cancel". Using options like "Go to it" and
            "Stop right here" doesn't make for a very intuitive interface.

         4. Keep the request text short. The user shouldn't have to read
            a screen full of text to find out what to do next.

         5. You should NEVER swap the "OK" and "Cancel" buttons around.

         6. The last point is VERY important.

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

      If the AUTOCANCEL option is used and the user presses 'Cancel', an
      error message is returned. For the script to trap this error,
      global error checking must be turned off. See
                     Error checking
                     , for
      more information.

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

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

the requester.

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

TITLE
The text for the title bar of the requester.

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

Errors
rc = 0 if the operation was successful.

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

Example
The following puts up a message requester:

REQUEST_MESSAGE TEXT '"Operation finished"'

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

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

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

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

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

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

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

Known bugs
None.

1.104 TextureStudio.guide/REQUEST_MULTIFILE

```
                 REQUEST_MULTIFILE
----------------
```

Command
     REQUEST_MULTIFILE

Parameters template
     PATHPART, FILEPART, PATTERN, TITLE

Return template
     FILES/M

Description
     Opens a file requester, allowing the user to choose multiple
     filenames.

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

     In common with all TextureStudio requesters, if the user presses
     'Cancel', an error message is returned. For the script to trap
     this error, global error checking must be turned off. See

                 Error checking
                 , for more information.

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

    FILEPART
         The default filename to put in the requester.

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

    TITLE
         The text for the title bar of the requester.

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

Errors
     rc = 0 if the operation was successful.

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

Example

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

        REQUEST_MULTIFILE STEM MULTIFILEINFO.

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

        REQUEST_MULTIFILE PATHPART "Textures" STEM MULTIFILENFO.

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

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

1.105 TextureStudio.guide/SCREEN_BACK

                    SCREEN_BACK

Command
     SCREEN_BACK

Parameters template
     None.

Return template
     None.

Description
     This sends TextureStudio's main screen to the back.

Parameters
     None.

Returns
     Nothing.

Errors
     rc = 0 if the operation was successful.

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

Example
     The following sends TextureStudio's screen to the back.

         SCREEN_BACK

Known bugs
     None.

     See

               SCREEN_FRONT

                    .

1.106   TextureStudio.guide/SCREEN_FRONT

               SCREEN_FRONT

Command
     SCREEN_FRONT

Parameters template
     None.

Return template
     None.

Description
     This brings TextureStudio's main screen to the front.

Parameters
     None.

Returns
     Nothing.

Errors
     rc = 0 if the operation was successful.

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

Example
     The following brings TextureStudio's screen to the front.

          SCREEN_FRONT

Known bugs
     None.

     See

               SCREEN_BACK

                    .

1.107 TextureStudio.guide/TEXTUREPATH_GET

```
                TEXTUREPATH_GET
--------------
```

Command
      TEXTUREPATH_GET

Parameters template
      None.

Return template
      PATH

Description
      This returns the user's current directory (path) for textures.

      This command is useful used in conjunction with FILE_JOIN, see

                FILE_JOIN
                    .

Parameters
      None.

Returns
     PATH
          The user's render path.

Errors
      rc = 0 if the operation was successful.

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

Example
      The following gets the user's texture path and returns it in a
      varaible called 'pathpart'

          TEXTUREPATH_GET VAR 'pathpart'

Known bugs
      None.

1.108 TextureStudio.guide/TEXTURES_GET

```
TEXTURES_GET
------------
```

Command
      TEXTURES_GET

```
Parameters template
     None.

Return template
     NAME/M

Description
     This gets the names of all the currently open textures.

Parameters
     None.

Returns
    NAME/M
         The names of all the textures.

Errors
     rc = 0 if the operation was successful.

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

Example
     The following example gets the names of all the open textures and
     displays each one in turn with a requester.

         TEXTURES_GET STEM 'texture.'

         do l = 0 to (texture.name.count - 1)
            REQUEST_MESSAGE TEXT '"'texture.name.l'"',
               BUTTONTEXT '"More...|Cancel"' AUTOCANCEL
         end

Known bugs
     None.
```

1.109 TextureStudio.guide/TEXTURE_SELECT

```
TEXTURE_SELECT
--------------

Command
     TEXTURE_SELECT

Parameters template
     NAME/A

Return template
     None.

Description
     Selects a texture to make it the current texture.

Parameters
```

```
    NAME/A
          This is the name of texture. You need only supply as many
          characters in the name to distinguish it from other textures.
          For example, if you wanted to select the 'GasGiant' texture,
          you need only give a NAME of 'GasG'.

          The name is case insensitive.

Returns
     Nothing.

Errors
     rc = 0 if the operation was successful.

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

Example
     The following example selects the 'Radar' texture.

          TEXTURE_SELECT "Radar"

Known bugs
     None.
```

1.110 TextureStudio.guide/VIEW_GET

```
                    VIEW_GET
--------

Command
     VIEW_GET

Parameters template
     None.

Return template
     VISIBLEWIDTH,XYASPECT

Description
     Gets the current view settings.

Parameters
     None.

Returns
    VISIBLEWIDTH
          The current visible width as a floating point number.

    XYASPECT
          The current X/Y aspect ratio as a floating point number.

Errors
     rc = 0 if the operation was successful.
```

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

Example
    The following example gets the current view settings and returns
    them in a variable called 'view.'

        VIEW_GET STEM 'view.'

Known bugs
    None.

    See
                VIEW_SET
                    .

    See
                View window
                    .

1.111 TextureStudio.guide/VIEW_SET

```
                VIEW_SET
--------
```

Command
    VIEW_GET

Parameters template
    VISIBLEWIDTH,XYASPECT

Return template
    None.

Description
    Sets the current view settings.

Parameters
    VISIBLEWIDTH
        The current visible width as a floating point number.

    XYASPECT
        The current X/Y aspect ratio as a floating point number.

Returns
    Nothing.

Errors
    rc = 0 if the operation was successful.

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

Example
     The following example sets the current visible width to be 50
     units and the aspect ratio to be 1.

          VIEW_SET VISIBLEWIDTH 50 XYASPECT 1

Known bugs
     None.

     See

               VIEW_GET

                    .

     See

               View window

                    .

1.112 TextureStudio.guide/Tooltypes

               Tooltypes
****

   The tooltypes for TextureStudio set some important preferences for
the program such as the name of the public screen and ARexx port.

   The tootypes apply when starting TextureStudio from the Workbench by
double clicking on it's icon or when starting it form the shell and
supplying some arguments, see
               Starting TextureStudio
                    .

               PUBSCREEN
                         Public screen name

               PORTNAME
                          ARexx port name

               KEYFILE
                           Keyfile filename

               PREFSFILE
                         Prefs filename

1.113 TextureStudio.guide/PUBSCREEN

```
PUBSCREEN
=========
```

Name
     PUBSCREEN

Default
     TEXTURESTUDIO

Description
     This sets the public screen name for the screen that TextureStudio
     opens. This name must be unique, otherwise the screen will not
     open.

1.114 TextureStudio.guide/PORTNAME

```
PORTNAME
========
```

Name
     PORTNAME

Default
     TEXTURESTUDIO

Description
     The name of the ARexx portname used by the program.

     If a portname by that name already exists, the name is incremented
     until a free portname is found. For example, if 'TEXTURESTUDIO'
     was already in use, the following sequence of names would be
     tried: 'TEXTURESTUDIO.1', 'TEXTURESTUDIO.2', 'TEXTURESTUDIO.3' ...

     When choosing an ARexx portname, try to keep it fairly short.

1.115 TextureStudio.guide/KEYFILE

```
                  KEYFILE
=======
```

Name
     KEYFILE

Default
     TextureStudio.keyfile

Description
     The filename of the keyfile to use to unlock TextureStudio to use
     all 16 parameters. Keyfiles are obtaining by registering (see

How to register
).

1.116 TextureStudio.guide/PREFSFILE

```
PREFSFILE
=========
```

Name
    PREFSFILE

Default
    TextureStudio.prefs

Description
    The sets the filename of the preferences file that TextureStudio
    uses to set up it's defaults on startup.

    If TextureStudio cannot find the filename given, the settings will
    be reset to the built in defaults.

1.117 TextureStudio.guide/Known bugs

```
                Known bugs
**********
```

  * It seems that some of the textures supplied with Imagine3 that
    were originally with Imagine2; Bricks and Dots don't work
    correctly. The bricks texture doesn't affect the object at all and
    the Dots texture only creates one-quarter of complete circle for
    each dot.

    Other textures like Disturbed, Grid, Waves and Wood seem to work
    fine.

    Note: TextureStudio will not work with textures taken directly
    from Imagine2, it only supports Imagine3 textures.

  * When rendering to a transparent cylinder or sphere, the 'other
    side' of the object cannot be seen through the front surface. This
    gives the impression of a hemi-sphere and not a complete sphere.
    This is something that may get fixed in the future, see

            Future additions
            .

1.118 TextureStudio.guide/Common questions

```
             Common questions
****************
```

   If you have any questions about TextureStudio, make sure that they
haven't already been answered below:

```
             Common question 1
               Why doesn't TextureStudio support Imagine2 textures?

             Common question 2
               Why isn't there a non-FPU version?
```

1.119 TextureStudio.guide/Common question 1

```
Common question 1
=================
```

     "Why doesn't TextureStudio support Imagine2 textures?"

   The Imagine2 texture format differs to the Imagine3 format and the
details about the Imagine2 format were not released by Impulse.

1.120 TextureStudio.guide/Common question 2

```
Common question 2
=================
```

     "Why isn't there a non-FPU version?"

   All but the very basic textures supplied with Imagine3 require an
FPU.

1.121 TextureStudio.guide/The authors

```
             The authors
***********
```

   TextureStudio was written by Graham Dean and Andy Dean.

   Queries and orders (see
             How to register

```
             ) should be sent to Graham at:


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


 Andy can be reached for queries (no orders) via Internet Email at:


   adean@eleceng.ucl.ac.uk


 The rate at which TextureStudio progresses depends on a few things:


 1. You. If you like and use the program, please register it. If you
    like the program but think it is missing something that isn't
    already in our future additions list (see
             Future additions
             ) let us
    know!.


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


   If you find a bug in TextureStudio that is not covered in the 'Known
bugs' list (see
             Known bugs
             ), inform the authors at the above addresses.
Be sure to include as much information as possible, the version of
TextureStudio being used, a description of the Amiga system you are
running (model, amount of RAM, Workbench version, any expansion cards).
```

1.122 TextureStudio.guide/How to register

```
             How to register
***************

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


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


 We will accept the following methods of payment:
 * 10UK pounds cash.


 * A 10UK pounds cheque, drawn on a UK bank.
```

* A 10UK pounds postal order, purchased in the UK.

* 20US dollars cash.

* International money order.

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

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

In return you will receive the latest version of TextureStudio,
along with a personal keyfile to unlock the package. Each keyfile is
unique to the registered user, please do not distribute the keyfile to
others as it can be traced back to you. Allow a resonable time to allow
cheque clearance, the processing of the order, etc...

Upgrades will be offered to registered users free of charge. As we
are operating a keyfile concept, upgrades can be obtained by getting
the latest version from the Internet, Aminet, BBS's, PD houses etc...
If your local provider doesn't have the latest version, pester them
until they get it!

Upgrades will not be given by contacting the authors directly,
unless there is a very good reason for it (we're sorry, but we don't
have the resources to deal with lots of registered users all wanting
upgrades at the same time!).

The version number of TextureStudio (see
            About
) is to be interpreted
as:

    version.revision.subrevision

The 'version' shows the main version of the program, 'revision' will
be increased as small additions and improvements are made to the
program. The 'subrevision' value is incremented with bug fixes. All the
values are simple decimal, not floating point, so version 1.9.0 would
be followed by version 1.10.0.

New versions will be distributed with every change in revision
number, bug fixes are likely to be distributed as "patches" (more
details to follow).

1.123 TextureStudio.guide/Credits

Credits
****

The authors would like to thank:

* Commodore-Amiga.

* Our parents for their support (especially our mum for also helping
  with the posting and packing!!!).

* Impulse for Imagine3 and its wonderful textures.

* SAS Institute, for the 'SAS/C' C compiler.

* Ian OConner, for 'The Designer' – used to do all the GUI windows
  design.

* Michael Balzer, for 'ARexxBox' – used to implement the ARexx port.

* Jonathan Forbes, for 'LX' – used to decompress the .lha files in
  the distribution.

* All the public domain / freeware / shareware authors, for loads of
  great software.

* The Independant JPEG Group, for their essential JPEG code and
  information.

* All those involved with the excellent TeX and 'TeXinfo' packages.

TextureStudio has been tested on:

– A1200, Workbench 3.0, 2Mbyte CHIP RAM, 4MByte FAST RAM, Power
  PC1204 expansion card, 68882 FPU, 270Mbyte IDE hard drive.

– A1200, Workbench 3.0, 2Mbyte CHIP RAM, 8MByte FAST RAM, Power
  Viper 030 expansion card with MMU, 68882 FPU, 270Mbyte IDE hard
  drive.

– A4000/EC030, Workbench 3.0, 2Mbyte CHIP RAM, 8MByte FAST RAM,
  68882 FPU, 130Mbyte + 420Mbyte IDE hard drives.

TextureStudio shows no problems with either the 'Enforcer' or
'Mungwall' debugging tools.

1.124 TextureStudio.guide/Future additions

                 Future additions
****

   The following features may be added to future versions, they are
listed roughly in order.

* Proper transparent objects, see
              Known bugs
                  .

* Animation control from the GUI.

  * Better multiple pass render, i.e. blocky redraw.

  * Printing the texture's name onto the HAM render screen for
    identification purposes.

    Any ideas?

1.125 TextureStudio.guide/Example textures

                Example textures
****

    The following are descriptions of the example textures included in
this package.

    The textures are freely distributable so feel free to use as you
wish.

                Blades texture

                Radar texture

                Target texture

                WarningStripes texture

1.126 TextureStudio.guide/Blades texture

Blades texture
==============

Type
    Colour, filter

Description
    This texture creates circular blades similar to propeller or fan
    blades.

Parameters
    The parameters can alter the number of blades, the blade radius
    and amount of space the blades occupy. The blades can be made to
    fade away with the 'fade adjust' parameter.

    The colour and filter of the blades can be adjusted. For example
    if you set the object filter to be 255,255,255 and the blade
    filter to be 0,0,0, the object rendered will be totally

```
    transparent except for the blades themselves.

    This texture can be animated to simulate spinning blades by
    adjusting the 'Sweep angle' parameter.

Axis
    The axis position sets the centre of the blades.

    The axis sizes are ignored.

Sample object
    Plane or disk with default settings.
```

1.127 TextureStudio.guide/Radar texture

```
Radar texture
=============

Type
    Colour

Description
    This texture creates are radar screen similar to that on a
    submarine or airplane.

Parameters
    The radar image consists of a grid made of 2 lines, one
    horizontal, one vertical, concentric circles and the radar sweep
    itself. The width of the grid lines and concentric circles can be
    adjusted with the 'Grid width' and 'Circle width' parameters. The
    colour of the grid and radar sweep can be adjusted.

    The radar sweep fades to the object colour so it is possible to
    overlay the radar texture onto a brush map with Imagine. The 'Fade
    adjust' parameter alters how much the radar sweep fades away.

    This texture can be animated to simulate a sweeping radar by
    adjusting the 'Sweep angle' parameter.

Axis
    The axis position sets the centre of the radar.

    The axis sizes are ignored.

Sample object
    Plane or disk with default settings.
```

1.128 TextureStudio.guide/Target texture

```
Target texture
==============
```

```
Type
     Colour
```

```
Description
     This texture creates overlayed concentric filled circles which can
     be used to simulate an archer's target or the RAF markings.
```

```
Parameters
     The fourth circle is layered on top of the third circle which is
     layered on top of the second circle which is layered on top of the
     first circle. Therefore the first circle has the lowest priority.
     Any of the circle's radii may be set to 0 to remove it.
```

```
     The colours of each circle can be adjusted.
```

```
     Note: This texture is clipped by the y axis, this allows multiple
     copies of the texture to be applied to an object.
```

```
Axis
     The axis position marks the centre of the target.
```

```
     The y-axis size alters the depth at which the texture affects.
```

```
Sample object
     Plane or disk with default settings.
```

1.129   TextureStudio.guide/WarningStripes texture

```
WarningStripes texture
======================
```

```
Type
     Colour
```

```
Description
     The texture was designed for creating the
     seen-too-many-times-before warning stripes on spaceships etc. The
     texture consists of a rectangle with diagonal stripes of
     alternating colours.
```

```
     The rectangle is set by the bounding box of the axis and so
     multiple copies of this texture can be applied to one object.
```

```
Parameters
     The width of the stripes can be adjusted as can the amount of each
     colour with the 'Fraction on' parameter. The slope adj parameter
     adjusts the slope of the stripes, a setting of -1 gives a 45
     degree slope to the right, a setting of 1 gives a 45 degree slope
     to the left and a setting of 0 gives horizontal stripes.
```

The colour of each stripe can be adjusted.

Axis
     The axis size and position set the bounding box for the texture.

Sample object
     Plane with default settings.

1.130 TextureStudio.guide/ImageStudio

                 ImageStudio
****

   The authors of TextureStudio have also written a shareware image
processing package called "ImageStudio". ImageStudio is written for the
casual graphics user who wishes to manipulate images on a modest Amiga
system.

   ImageStudio includes the following features:

   General:
   * Full 24-bit image buffers, with optimizations for colour-mapped
     (palette based) images.

   * Up to 100 levels of undo / redo.

   * User configurable virtual memory.

   * Fully font sensitive, style guide complient, user interface.

   * Fully featured, easy to use, ARexx interface.

   * Internal / external viewers (external for third party 24-bit
     graphics cards).

   * Loading / saving / manipulating of AGA image formats (e.g. 256
     colours, HAM8) on non-AGA machines.

   * Max image size of 32000 x 32000 (limited to 250 x 250 in the
     unregistered version).

   * Runs on all Workbench 2.04+ Amiga's - utilises AGA chipset if
     available.

   * Online AmigaGuide help, as well as ASCII, TeX .dvi and PostScript
     documentation.

   * Requires no third party libraries or utilities.

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

Operations:
* Brightness, Contrast, Gamma, Convolution (includes user
  definable), Dynamic range expansion, FlipX, FlipY, RollX, RollY,
  Negative, Greyscale, Highlight, Shadow, Random, Pixelize, Remove
  isolated pixels, Crop, Scale, Colour reduction (with many
  dithers), Palette manipulation

ImageStudio costs 10UK pounds or 20US dollars and is available by
writing to the authors at the same address as TextureStudio (see

                How to register
). A demo version, limited to loading images of upto
250x250 pixels, is available from most PD libraries and available by
anonymous ftp from Aminet (gfx/conv directory).

ImageStudio has recently won the Amiga Shopper 1995 reader award for
'Best PD / Shareware Utiltity'.

Other reviews have said:

   "This program is superb."

Amiga Pro, Larry Hickmott, December '94

   "This is a real prize program. ... Registration is only 10UK
   pounds, a sound investment if you ask me... 96%"

Amiga User International, December '94

   "Perhaps the most impressive feature is the option to use a hard
   disk as virtual memory ... a feature that would be welcome in many
   commercial offerings."

Amiga Computing, December '94

   "It's a promising package... 88%"

Amiga Format, November '94

   "ImageStudio is an impressive program – all the more considering
   this is the first revision... 90%"

Amiga Shopper, December '94

   "It is impossible to choose between ImageStudio and Blackboard,
   [Blackboard] has better effects, but [ImageStudio] has better
   overall handling... 89%"

C.U.Amiga, December '94

   "This is a very stable and useful program with features which are
   worth a lot more that the asking price. ... I urge you to
   contribute your shareware fee as soon as possible to get the most
   from this excellent program."

Amiga Pro, Phil South, December '94

"Probably the most incredible thing about ImageStudio is that it
is as solid as a rock."

JAM, December '94

"It's been a long time since I've seen a shareware program as good
as ImageStudio."

EM, December '94