Composition

THE *image* MANIPULATION PACKAGE

Composition

Compo

Plug Compliant Apps

PCA

User Manuals

D R A F T

8 August 2024

Updated edition 2024

Rob Davison 2005

APDL 2003

Clares Micros 2003

Online resources

At the time of writing some Compo resources remain available online:

The IconBar: `https://compo.iconbar.com/compo.htm` which includes:

Some simple BASIC and BASIC/ARM programs that demonstrate an external utility modifying an image via CompoScript:
`http://www.compo.iconbar.com/compo/archives/csprog.zip`

and *CompoScript Reference Manual 1.22f*, by Lenny, dated 21 October 2003: `https://compo.iconbar.com/compo/csref.html`, supplied also in some distributions of Compo.

Jim Lesurf's pages:

Compo clues: `https://jcgl.orpheusweb.co.uk/Compo/clues.html`

and more clues: `https://jcgl.orpheusweb.co.uk/CompoRevival/CompoRevived.html`

plus Jim's six articles for *Archive* magazine between August 2000 and August 2001:

`https://orpheusweb.co.uk/Compo/Archive-1-6.pdf`

with its resources at:

`https://jcgl.orpheusweb.co.uk/Archive/index.html`

Editor's notes

This new edition of the 2005 version of the Composition and PCA Applets manual is updated in the following ways:

Mentions of the RiscPC, low resolution screen modes, restricted RAM and VRAM have been removed. Also Chapter 6 *PhotoCD* and Appendix 1 *Split* (for splitting files across several floppy discs) are gone.

The Appendix *Introduction to CompoScript* is now Chapter 7.

Chapter 8 *CompoScript language reference* has been added based on the Argonet/IconBar HTML page *CompoScript Reference Manual for Compo 1.22f* referred to above.

References to JPEG images that Acorn supplied with RiscPCs alongside the application *SlideShow* have been retained. The app and the images (00–99) can be downloaded from:
`https://www.4corn.co.uk/archive/acornftp/riscos3/37/37DiscImage/Images`

Images of windows and dialogues have been made from Compo 1.23c.

Credit and thanks are hereby given to Rob Davison for the software, for its user documentation and for its 2024 re-release for free via PlingStore.

This document is maintained in *OvationPro* and *TableMate* as a print-ready A5 double-sided document with binding margins, distributed as a PDF file with bookmarks, made using *PrintPDF* and a *PDFMark* file.

Bernard Boase, July 2024

Composition

Contents

1. Introduction

About Composition

Compo is an image composition program for the computers that run RISC OS. With Compo, you combine bitmaps together on a canvas to make a composition. Each bitmap remains completely independent from the rest. Links to art packages such as *ProArtisan24* enable you to edit the images making up your composition and to see the results of your changes immediately. In addition, the program allows a number of effects such as colour correction, shadows, opacity and tinting to be applied independently to each image. Using these effects the images are rendered to the canvas on the fly so that the effects do not modify (and therefore damage) the original image data.

Compo is useful not only to the graphics professional, it also allows those users with less artistic ability to produce complex graphical effects such as true shadows and photo montages at the click of a button. Because the images themselves are not modified you can quickly change the effect settings until the desired result is achieved.

Conventions used in this guide

We use the standard RISC OS conventions for the mouse buttons:

SELECT is the left button, and the most commonly used.

MENU is the middle button and is used primarily for opening menus.

ADJUST is the right mouse button.

Unless otherwise stated 'choose' means move the mouse pointer over something and click on it with **SELECT.**

Typographic conventions

Mouse buttons or keys on the keyboard appear like this: **SELECT**, **Return**, **Esc**, **F3**. **Enter** has the same effect as **Return**.

Special terms and jargon appear in italics when first introduced: '... called a *sprite*.'

Menu items or icons that can be chosen appear like this: choose **Select all.** This means choose the item called **Select all** from the menu.

Menu items which lead to submenus have an arrow after them: Save ➤.

Jargon

We have tried to keep the jargon to a minimum. If you find a term you are unfamiliar with please refer to Appendix A *Glossary*.

Three very important terms within Compo are:

A *Sprite* is a 32bit-per-pixel, 16 million colour bitmap image.

The *Canvas* is the work area within which a composition is created.

A *Mask* is a 256-level grey scale sprite used to attenuate an effect on a sprite. For example, the Tint mask is used to control the amount of colour tinting applied to a sprite.

The sprites in a composition are held internally in a *stack* which reflects their fore and aft positions in the display. Hence we may speak of a sprite being *in front of* or

behind other sprites in the display or, equivalently, *above* or *below* them in the stack. Similarly, a sprite may be moved *forward* or *backward* in the display, which is *upward* or *downward* in the stack. The *front* (*foremost*) and *back* (*rearmost*) sprites in the display are the *highest* and *lowest* of the stack. We reserve *top* and *bottom* for describing vertical positions of objects on the screen.

Keyboard shortcuts

In general, Compo follows normal RISC OS shortcuts whenever possible. Clicking **ADJUST** often has an alternative or opposite effect to **SELECT**, and pressing **Shift** or **Ctrl** while clicking a mouse button often modifies an operation. For example, holding down **Shift** or **Ctrl** when clicking on a bump arrow increases the speed of the change made to the related value. Some shortcuts require holding down both **Shift** and **Ctrl**.

When a dialogue box has a default action icon, clicking on the icon with **SELECT** carries out the action and closes the dialogue box. Clicking on the icon with **ADJUST** carries out the action but keeps the dialogue open.

In this manual, and on the program menus, keyboard shortcuts appear beside the icon name or menu item like this:

Copy (Ctrl-C)

This should be taken to mean that pressing **Ctrl-C** when the program has input focus has the same effect as choosing the **Copy** item from the menu. The program has the input focus when the title bar of a window belonging to it is coloured cream. Appendix B *Keyboard shortcuts* contains a full list.

It is assumed that you are familiar with the RISC OS Desktop environment.

2. Tutorials

This section begins with a tutorial which then carries on into a series of worked examples of some of the program's features. The example files used are in the program distribution and should be copied to your own directory.

The basics

We do not recommend using the program with a desktop of fewer than 32 thousand colours. Select a desktop with as much resolution and colour as appropriate for your image manipulation. With modern machines you will not be restricted to small screen sizes or low resolutions and will be able to use 16 million colours at high resolutions.

Compo runs slightly faster with the desktop at 16 million colours since internally it stores everything with this colour depth.

1. Load Compo in the usual way, by double clicking on the program icon.
2. Click **SELECT** on the Compo icon bar icon to open a canvas of the default size and colour.

   Clicking **ADJUST** instead of **SELECT** on the Compo icon bar icon opens a window for you to change the size and colour of the initial canvas created.

   Dragging an image onto the Compo icon bar icon before a canvas has been created creates a canvas the same size as the image and then loads it. This is useful if there is to be one background image to your composition.
3. Open the directory containing the tutorial files.
4. Drag the file called `Di` into the canvas window.
5. Change the window to full size by clicking on the window's toggle size icon.
6. Select the image by clicking on it: the dashed cyan frame indicates that it is the selected graphic.
7. Drag the image with **SELECT** to another place within the window. To abort a drag after you have started moving the image press **Esc** with your free hand. Use the arrow keys while holding down **SELECT** for coarse positioning.
8. Make a copy of the image by pressing **Ctrl-C** or clicking **MENU** and choosing **Select ➤ Copy** from the menu.

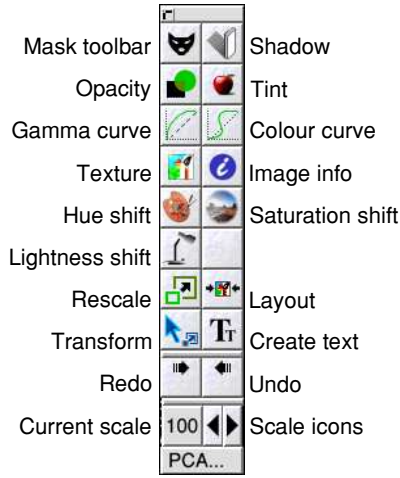
   The copy will be located to the right of and below the original and will be automatically selected in place of the original.

When selecting objects, Compo behaves like Draw in many ways.

9. Ensure that the second copy of `Di` is selected (the foremost one) and click on the **Opacity** icon in the toolbar to open the Opacity dialogue box.

   This dialogue box may also be opened by clicking **MENU** and following **Effects ➤ Opacity ➤** on the menu or by pressing **Ctrl-O**.

   We will normally describe using the toolbar (shown to the right) to open dialogue boxes, but you may prefer to use keyboard shortcuts.

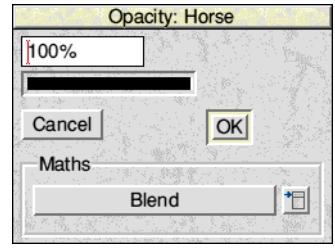


The main Compo toolbar

10. In the Opacity dialogue box drag the slider to alter the opacity of the image. Click on OK to apply the change. Clicking on OK with **ADJUST** keeps the dialogue box open. Clicking with **SELECT** closes it.

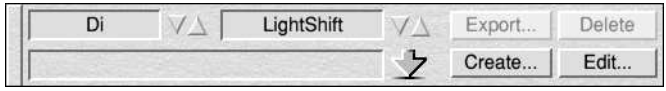
    If you drag the slider with **ADJUST** rather than **SELECT** the image will update as you drag. This effect is used in many parts of the program where a value can be altered by a slider.

    When working with large images it is best to use **SELECT** drag and then **OK**, rather than **ADJUST** drag.

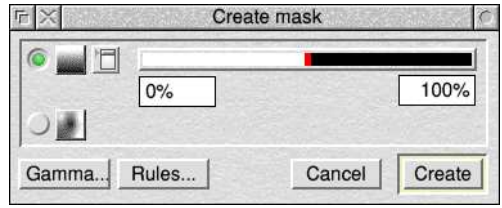


11. **ADJUST** drag the slider until the opacity reaches 50% and choose **OK**. The foremost copy of `Di` is now 50% transparent. Check that you can still move it around. Choose the opacity icon again and return the opacity to 100%.

12. Ensure that the foremost copy of `Di` is still selected and click on the **Mask** icon at the top left of the toolbar. A horizontal toolbar, shown below and known as the Mask toolbar will open at the bottom of the canvas window:



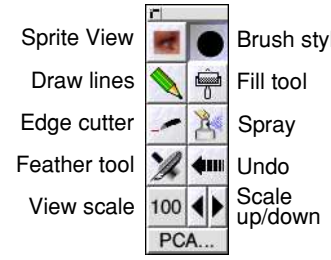
13. Choose **Create...** in this toolbar to open the **Create mask** dialogue box (right) which lets you create a number of simple masks. Using **ADJUST** choose **Create** in the dialogue box to create the default: a simple vertical fade.



The sprite will fade from completely transparent at the bottom to completely opaque at the top. Play with the sliders in the dialogue box to discover their relationship to the fade. You can **ADJUST** drag the sliders to create the mask as you go. Experiment with the other fades provided by clicking on the menu opener, choosing a fade from the menu and choosing **Create** once more.

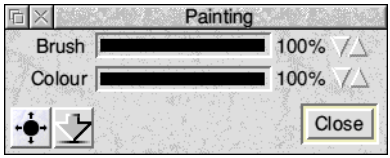
14. Choose the **Edit...** button in the Mask toolbar. This opens the Mask Edit window, displaying the Blend mask for *Di*. The blend mask controls the transparency of the sprite, as you have discovered. Pixels in the mask which are black equate to completely transparent pixels in the sprite. Areas of the mask which are white are completely opaque. Grey pixels range between the two extremes.
15. Click **MENU** over this window and choose **Clear to ➤ Black**. The mask will clear to black and the sprite will apparently disappear from the canvas.
16. Move over the Mask Edit window and drag the mouse across it. As you paint in the mask using the default circular drawing tool, related areas of the sprite will appear in the canvas. Drag with **ADJUST** over the mask to remove areas of the sprite.

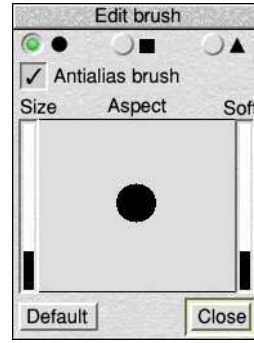
Most of the tools in the Mask Edit toolbar (shown on the right) have a small dialogue box associated with them which enables you to change the characteristics of the tool. Open a dialogue box by clicking **MENU** over a tool.



The Mask Edit toolbar

For example, click **MENU** over the **Brush style** tool and the window shown below left will open. This lets you set the characteristics of the mask, and by clicking on the button at the bottom left of this window the window shown below right will open where you can choose the shape and size of the brush.





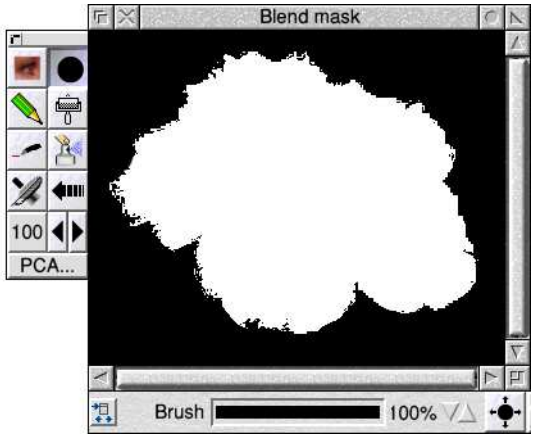
To undo the last operation on the mask choose the Undo button on the Mask Edit toolbar.

Depending on the number of undos configured in file `!Compo.Resources.prefs`, it may be possible to undo more than one operation by clicking on this icon again.

Cutting out objects: masking

1. Drag the file `Flower5` into the main window. Select it and position it suitably.
2. Choose the **Edit...** button in the mask toolbar. The program will create a default (black) Blend mask and open the Mask Edit window onto it.
3. Choose the **Eye** icon at the top left of the Mask Edit toolbar to open the Sprite View window.
4. Move over the flower in the Sprite View window and click **SELECT** to add pixels of a similar colour to the one you clicked on to the mask. You will see the flower shape build up in the Mask Edit window and, simultaneously, the flower will appear in the canvas window.

   Move around the flower, clicking on the differently coloured features on its surface. If parts of the background appear in the mask ignore them and carry on until most of the flower is selected, paying particular attention to its edges.
5. Move over the background surrounding the flower and click **ADJUST** to remove pixels from the mask. Continue until the picture in the Mask Edit window looks something like that shown below.

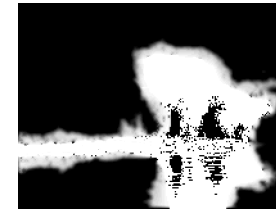


6. Use the circular brush in the Mask Edit window to white out the few remaining black pixels within the flower. Use the same tool or the edge cutters (see later) with **ADJUST** to remove white pixels remaining in the background.
7. To smooth the transition between the flower and what will be behind it, paint with the Feather tool around the edges of the mask or click **MENU** and choose **Anti-alias ➤ Smooth ➤ All levels**.
8. Close the mask window and move the flower over the `Di` image you loaded in the first tutorial. Try adjusting the flower's opacity.

Simple composition

1. Load the `Bridge` file into the main window.
2. Select it and press **Ctrl-I** to open the info window. Enter **0** into both of the writable position icons and press **Return**.
3. Load the `Bears` file and position it below the bridge as shown on the right.

4. Choose **Edit...** in the mask toolbar.
5. Choose the **Eye** icon in the Mask Edit window toolbar to open the Sprite View window.
6. Click over the bears in the Sprite View window until they appear solid in the mask. Remove any of the background from the mask with the drawing tools and/or by **ADJUST** clicking over it in the Sprite View window.
7. Select the Feather tool and Click **MENU** over it. Decrease the feather size to 8 pixels using the bump arrows and choose Strength **Silk** by clicking over the menu icon and choosing it from the menu which appears. Choose **OK** to close this dialogue box.
8. Drag around the top edge of the bear shapes in the Mask Edit window to smooth the edges of the mask. If the relevant area of the canvas is visible you will be able see the results of your changes as you work.

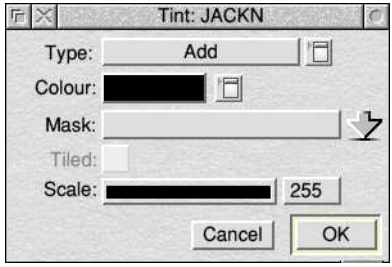


9. When you have gone around the edges of both bears click **MENU** over the Feather tool again, set the Strength to **Deluge!** and the size to **32** pixels.
10. Wipe the pointer repeatedly across the bears' reflection at the bottom of the mask to darken and blur it.

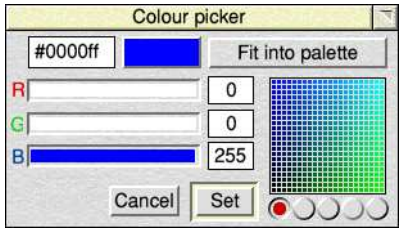
11. Finally, Click **MENU** over the Feather tool again and Choose Strength **Darken**. Drag over the land at the left of the mask a few times to darken and blur the edge.

Tints

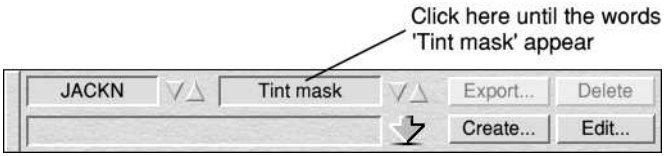
1. Open a new canvas and load the file called `JackN`. Select it and position it conveniently.



2. Choose the **Apple** in the main canvas toolbar to open the **Tint** dialogue box (shown on the right) for this sprite. Click on the menu button beside the Colour icon to open the **Colour Picker** which is similar in operation to the standard RISC OS colour picker.
3. Choose **blue** from the default palette and click on **OK**. The sprite will be strongly tinted towards blue



4. Click on the **Mask** icon and change the mask to be modified to Tint mask by clicking on the appropriate icon in the Mask toolbar. Choose **Edit...** to enable you to view the mask that will be created.



5. Choose the **Create...** button to open the default Fades dialogue box.
6. Choose the **Points** icon (the lower one) in this dialogue box. The **Mask fade** points dialogue box will open. Choose the Sprite button in this dialogue box to display a *greeked* version of the sprite. By default, there is one fade point in the middle of the area. If you choose **Create** (with **ADJUST** to keep the dialogue box open) a simple circular fade is created in the tint mask.

You can drag the fade point around, double click in the area to create more points and click **ADJUST** over a point to toggle it from white to black. Black points pull down the mask at that point.

When you choose Create the mask made is determined by the position and colour of the points in this window. If it is difficult to see the points over the sprite then click on the Sprite button again to hide the sprite.

7. To concentrate the tint around his eyes (for example) move the point already present on the left eye, double click **SELECT** over the right eye, double click **SELECT** over his chin and then click **ADJUST** over this point, double click **SELECT** over the middle of his forehead and then click **ADJUST** over this point.
8. Finally, choose Create.

More flexible fades may be created by using the Graduated fill tool in the Mask Edit window. See Chapter 3 *Toolbars*.

Shadows

1. Load the `Leafyl` image and move it to the bottom left of the canvas.
2. Load the `Horse` image. Because a suitable mask called `Horse_msk` exists in the same directory it will automatically load to the Horse's blend mask.
3. Position the horse on the path near to the bottom of the image.
4. Press **Ctrl-S** to turn on a simple shadow effect.
5. Choose the **Shadow** icon at the top right of the main toolbar to open the Shadow dialogue box.
6. Turn on the **Pseudo 3D** option and click on **OK** with **ADJUST.**
7. Alter the writable **X shear** value using the bump arrows or keyboard to **0.50**.
8. Click **OK** with **ADJUST**.
9. Use the bump arrows to reduce the **X** and **Y** offsets to zero.

10. Click on the **Colour** menu icon to open a colour selector, and choose a darker grey, then click **Set** with **SELECT**.

Text

1. Load the `MDHB` image and. if necessary, move it to the bottom left of the canvas as detailed above.
2. Choose the **Text** icon in the main toolbar.
3. Choose **Create** in the Text dialogue box to create the default text (the word 'Compo') at the bottom left of the canvas. Move the text until it is partly over the building and partly over the sky behind.

   You will notice that Compo anti-aliases text on a pixel by pixel basis. Use the zoom icons in the main toolbar to examine the effect in detail before returning to normal size by clicking **ADJUST** on the Current scale icon.
4. Give the text a shadow by pressing **Ctrl-S** or by using the Shadow dialogue box as described above.
5. Open the **Opacity** dialogue box and **ADJUST** drag the slider down to **50%**. The shadow appears behind the partly transparent text giving a 'Glass text' effect.

Text / Tints *(assumes carrying on from above)*

1. Choose the **Text** icon in the main toolbar to open the Text dialogue box.
2. Change the Text colour to Black.
3. Change the Text size to 100 point.
4. Choose **Create new**. The new text is positioned in the bottom left of the canvas.
5. Select the newly created text and open the Tint dialogue box by choosing the Apple in the main toolbar.
6. Choose a red tint and click on OK.
7. Change to the Tint mask in the horizontal Mask toolbar.
8. Choose **Create....** Select the Vertical fade from the menu if it is not ticked and then click on **Create** in the Create mask dialogue box.
9. Repeat with the Circular fade.
10. C**trl-SELECT** drag over the text in the canvas to move the Tint mask around behind the text.

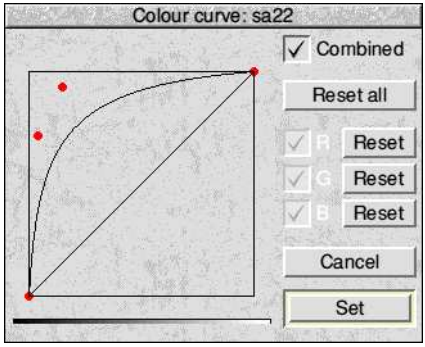
Experiment with text using different angles and fonts, with a shaded box, etc.

Colour correction

1. Load the image `sa22` from the directory `!SlideShow.Images.00-49`, or another that you like.
2. Select the image.
3. Click **MENU** and follow **Effects ➤ Scale ➤**
4. Select the first **By** item, enter **0.5** into the dialogue box and click on **Scale**.

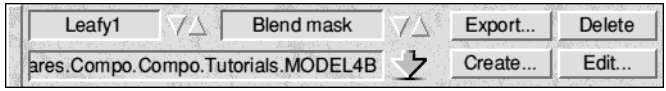
   Compo's Scale feature uses **ChangeFSI** to resample from the original image file. This provides a very high quality rescale at the expense of some speed and with the requirements of enough free memory and disc space.

   It is felt that image quality is of paramount importance in Compo and that a fast rescale on the fly would not produce high enough image quality. Scale is one of the very few effects in Compo which modifies the image data in memory directly and therefore has several important consequences which will be discussed later in this manual.
5. Press **Ctrl-C** to create a copy of the rescaled image.
6. Position the images side by side and select only the right one.
7. Choose the **Colour curve** icon in the main toolbar or press **Ctrl-V**.
8. Modify the curve to that shown on the right by moving the red control points.
9. Click on **Set** to see your changes. Select **Reset** all to restore the default.



Experimenting with masks

1. Press **Ctrl-A** and then **Ctrl-X** to select and delete all of the sprites.
2. Load the `Leafyl` image.
3. Select it and ensure that the Blend mask is selected in the horizontal Mask toolbar.
4. Drag the file called `MODEL4B` into the long box in this toolbar. After you have done so its filename will appear within the box as shown below:



Notice how the `Leafyl` image is restricted to the shape of the woman's head. Click **Edit...** in the Mask toolbar to see how the mask is constructed.

Compo can use just about any image as a mask. To load a file into a mask simply drag its icon into the box in the toolbar. If the image you drag is not suitable (e.g. a 256 grey level sprite) then Compo will load it and convert it for you. Compo can recognise Sprite, TIF, GIF, JPEG, TGA, PCX, PhotoCD and Clear files.

You can, of course, load an image to the Tint mask instead of the Blend mask, though the effect of this will not appear until there is a tint applied to the sprite.

Images masked in this way often look good with a slight tint or shadow to enhance details in the mask.

The instruction above to load image `sa22` refers to one of the 100 medium-resolution JPEG images that Acorn supplied with application *!SlideShow* on a RiscPC.

RISC OS 3.7 `Images.00-49.sa22`

3. Toolbars

This chapter describes all the toolbars provided by the program and the dialogue boxes associated with them.

General points

Pressing **Tab** while using the program hides all of the toolbars and editing windows apart from the canvas window. This can be useful for viewing the composition without the clutter of toolbars, etc. Press **Tab** again to reopen the toolbars. All of the toolbars may be moved by dragging their title bar in the same way that normal windows are moved.

Scale icons

These are essentially the same in all of the program's toolbars and are therefore described here. The left and right arrows scale the size of the attached window down and up respectively. **ADJUST** may be used to scale in an opposite direction to the norm. If **Shift** is held down the scale will change by 50% for each click, otherwise the scale change is 25% for each click.

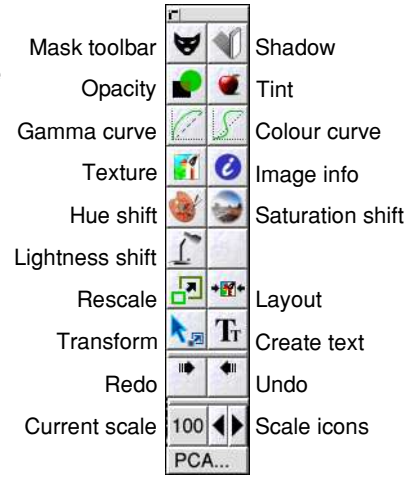
The icon displaying the current scale is writable and a new scale may be entered using the keyboard. **ADJUST** clicking on the current scale will toggle between 100% and the last value used that was not 100%. This is an easy way of moving between two scales.

Scale on load

If you hold down **Shift** while loading an image, the Scale dialogue opens allowing you to scale the image during loading.

Main toolbar

The main toolbar is attached to the canvas window and is shown on the right. The tools will now be detailed one by one. If more than one sprite is selected then the sprite modifying buttons enable you to make the same adjustments to all of the selected sprites. The values displayed initially in a dialogue box will be those for the last sprite added to the selection.



Selecting sprites

Compo's sprite selection functions are modelled on those of Draw. Click **SELECT** on a sprite in the canvas window to select it. Click **ADJUST** on a sprite to toggle it in or out of the selection. Double click **SELECT** on a sprite to move through the stack in order to select sprites which are behind another one. In a departure from Draw, double clicking **with ADJUST** with **Shift** held down will always add sprites to the selection while moving through the stack. Use this function to select layered sprites under the pointer. In addition the menu items and key shortcuts **Select ➤ All (Ctrl-A)** and **Select ➤ Clear (Ctrl-Z)** are available.

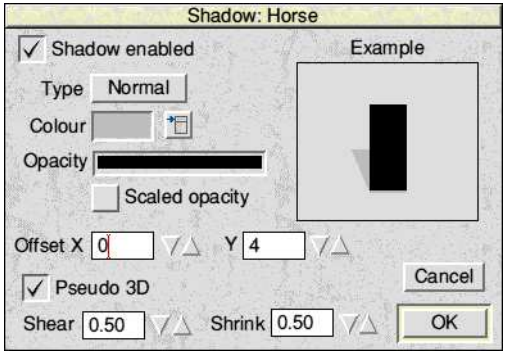
Mask

Clicking on the **Mask** icon toggles display of the horizontal Mask toolbar.

Shadow (Ctrl-Shift-S)

Clicking on the **Shadow** icon with **SELECT** opens the **Shadow** dialogue box if at least one sprite is selected. If no sprites are selected then clicking on the **Shadow** icon does nothing. Clicking on the **Shadow** icon with **ADJUST** opens the Mask Edit window for the last selected sprite's Shadow mask. If a Shadow mask has not previously been edited for the sprite then a copy of the sprite's **Blend mask** is automatically placed in the Shadow mask for you.

Shadow dialogue box



Shadow enabled controls display of shadows for the selected sprites. For the shadow to be visible, the sprite must have a Blend mask.

The **X** and **Y offset** values define the number of pixels to the right and below the object by which the shadow will project. Negative values may be used to project the shadow up and to the left.

The **Opacity** slider lets you set the opacity of the shadow.

The **Pseudo 3D** option, when on, distorts the shadow to produce a pseudo three dimensional effect. The example box at the right of the Shadow window provides a visual indication of what a particular set of values looks like.

The **Colour** option lets you change the colour of the shadow using a standard colour picker box. Normally you will use one of the greys but you might think a green shadow would look realistic. See Chapter 5 *Using Compo* for more information about Shadows.

Opacity (Ctrl-O)

Clicking on the **Opacity** icon with **SELECT** opens the **Opacity** dialogue box if at least one sprite has been selected. If no sprites are selected clicking on the Opacity icon does nothing. Clicking on the Opacity icon with **ADJUST** opens the Mask Edit window for the last selected sprite's Blend mask.

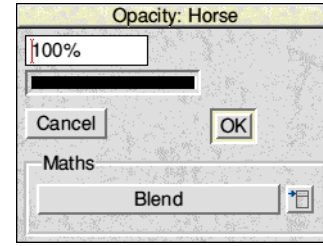
Opacity dialogue box

When you first open this dialogue box the bottom half is not visible. To display it, click on the toggle size icon at the top right of the dialogue box.

Drag the slider to change the opacity of the selected images, or enter a value into the writable icon and press **Return** to do the same. **ADJUST** drag the slider to change the opacity interactively.

Maths

Maths lets you change the way in which the selected images are integrated with the canvas. The default, **Blend**, is most commonly used. The **Opacity value** and **Blend mask** (grey level pixels) control the amount of effect the following functions have on the canvas.



For example **Add** takes the red value of the pixel being written and adds it to the red value for the pixel being overwritten. The same happens for the green and blue components of the pixel. All three values (red, green and blue) are then scaled down by the combination of the Opacity and Blend Mask settings before being plotted to the canvas.

In the case of **Multiply** the RGB values of each pixel to be written are multiplied by the RGB values of the pixel being overwritten. They are then scaled down by the combination of the Opacity and Blend Mask settings before being plotted to the canvas.

The full Maths menu is shown in Appendix C. Meanwhile, you should experiment while reading a selection of the descriptions below:

Blend mixes the sprite with whatever is underneath it.

Add adds the sprite to whatever is underneath it.

Subtract subtracts the sprite from whatever is underneath it.

Lighter lighter than whatever is underneath it.

Darker writes the sprite pixel only if it is darker than whatever is underneath it.

Multiply multiplies the sprite by whatever is underneath it.

NegMultiply multiplies the sprite by the inverse of whatever is underneath it.

MultAdd scales the sprite's pixel by the pixel underneath it and then adds the result.

Experiment uses a calculation related to the Blend and Multiply functions.

Sine Table produces false colour effects based on the mathematical function.

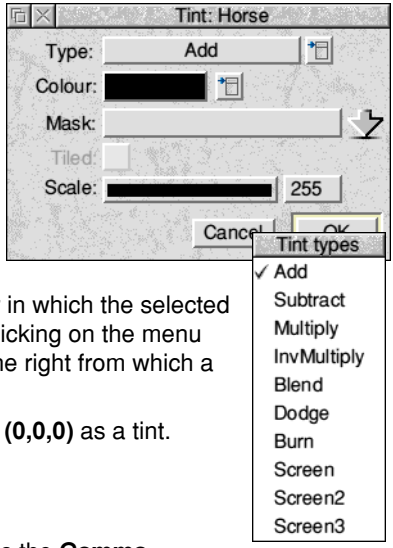
Cosine Table produces false colour effects based on the mathematical function.

Tan Table produces false colour effects based on the mathematical function.

Lin. Palette & Palette both false colour the image according to a palette definition. Compo uses the same palette format as ProArtisan24 and PA24 palettes may be used by dragging the palette file icon to the canvas window. Only one palette may be used at a time and Compo will redraw all affected images when a file is dropped in. For best effect, palettes should be defined with a gradual spread between colours. Some suitable files are provided in directory `Resources.Palettes`.See Chapter 5 *Using Compo* for more details and examples.

Tint

Clicking on the **Tint** icon with **SELECT** opens the **Tint** dialogue box if at least one sprite has been selected. If no sprites are selected then clicking on the Tint icon does nothing. Clicking on the Tint icon with **ADJUST** opens the Mask Edit window for the last selected sprite's Tint mask.



The dialogue box which opens when you click **SELECT** is the standard colour picker and operates in the same way as it does in other parts of the program.

Clicking on the menu icon beside **Colour** picker opens an RGB colour picker. The manner in which the selected colour is applied is shown by the top icon, and clicking on the menu button beside this will open a menu, shown on the right from which a variety of methods can be chosen.

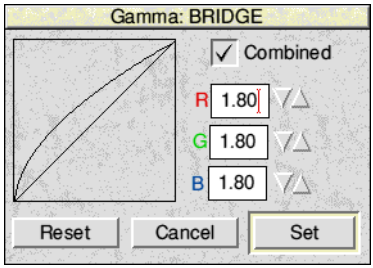
To remove all tinting from a sprite choose **Black (0,0,0)** as a tint.

Gamma correction (Ctrl-Shift-G)

Clicking on the **Gamma** icon with **SELECT** opens the **Gamma** dialogue box if at least one sprite has been selected. If no sprites are selected then clicking on the **Gamma** icon does nothing. Clicking on the Gamma icon with **ADJUST** opens the Mask Edit window for the last selected sprite's Curve mask.

Gamma dialogue box

Gamma correction is an industry standard way of modifying the brightness and contrast of an image.



Alter the gamma value either by entering a value in a writable field or clicking on the bump arrows. Remember you can press **Shift** or **Ctrl** to make larger changes to the value. If the **Combined** option is turned off you can alter red, green and blue gamma values separately. Values greater than one will lighten the selected images; values less than one will darken them.

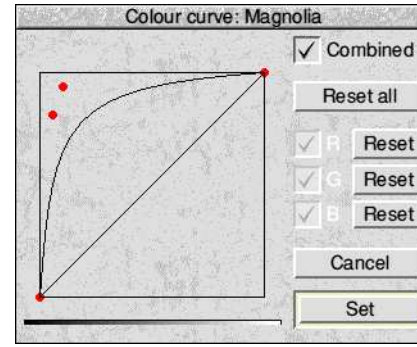
Choose **Reset all** or **Reset** to reset the gamma curve to the default of 1.00 (no Gamma correction)

Colour curve (Ctrl-V)

Clicking on the **Colour curve** icon with **SELECT** opens the **Colour curve** dialogue box if at least one sprite has been selected. If no sprites are selected then clicking on the **Curve** icon does nothing. Clicking on the **Curve** icon with **ADJUST** will open the Mask Edit window for the last selected sprite's **Curve mask**.

Colour curve dialogue box

The Colour curve function is a more flexible way of changing the colour and contrast of an image than Gamma correction. Gamma and Colour curve may both be used together in which case the gamma correction occurs first.



Drag the control points to change the contrast and brightness of the selected sprites. Turn the **Combined** option off to alter the red, green and blue channels separately.

If **Combined** is off you can choose which of the red, green or blue channels are displayed and alterable via the three small option buttons (labelled R,G and B). This is useful when the control points for more than one channel overlap.

Dragging a control point with **ADJUST** will attempt to update the image as you drag. This is generally slow on older hardware, but tapping **ADJUST** while dragging is a very quick way of seeing the changes you have made to the curve.

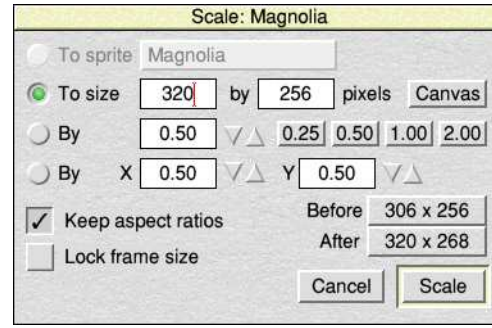
Scale (Ctrl-Shift-Z)

Clicking on the **Scale** icon opens the **Scale selection** dialogue box if at least one sprite has been selected. If no sprites are selected then clicking on the **Scale** icon does nothing.

In addition to the Scale selection dialogue box, it is possible to interactively resize the selected sprites by **ADJUST** dragging in the canvas. Holding **Shift** down locks the objects' aspect ratios as they are resized. To abort an inadvertent resize tap **Esc** while dragging.

Scale selection dialogue box

If more than one sprite is selected then clicking on the **To Sprite** radio button lets you scale all of the selected sprites to the same size as the sprite name beside this radio button: click on the sprite name to cycle through the selected sprites.



Choosing the **To size** button and entering a size into the width and height fields will rescale the selected sprites to the size entered. The **Canvas** button enters the canvas size into the writable fields enabling you to fill the canvas easily

Choose the first of the **By** buttons to scale the selection by the factor in the writable field. The three small buttons to the right of this field enable you to quickly enter scaling for one quarter, half and twice the current size.

Choose the second **By** button, and enter values into the two writable fields to scale the width and height of the selection separately.

The **Keep aspect ratios** option, when ticked, locks the aspect ratio of each sprite being rescaled. This means that the rescaling will probably not result in precisely the size chosen but it will be as close as possible while maintaining the aspect ratio of the original sprites.

To retain high image quality when rescaling it is advisable to choose the **Update bases** item from the **File** menu before any rescaling of objects is done. In this way, the program can always work from full resolution originals.

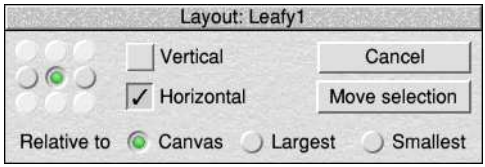
Rescale on load

If you hold down **Shift** while loading an image the **Scale** dialogue opens allowing you to scale the image during loading.

 Layout (Ctrl-L)

Clicking on the **Layout** icon opens the **Layout** dialogue box if at least one sprite has been selected. If no sprites are selected then clicking on the Layout icon does nothing.

Layout dialogue box



This dialogue box enables you to align the selected sprites in a number of ways, depending on the state of the Vertical and Horizontal option buttons.

Vertical

The Vertical button selects vertical alignment. This means that the sprites are aligned vertically and maintain their relative horizontal position. Only the three centre vertical radio buttons are available in this mode. Choose one to indicate the new position and click on **Move selection**.

Horizontal

The Horizontal button selects horizontal alignment. This means that the sprites are aligned horizontally and maintain their relative vertical position. Only the three centre horizontal radio buttons are available in this mode. Choose one to indicate the new position and click on **Move selection**.

With both Vertical and Horizontal selected all nine radio buttons are selectable.

Canvas

This aligns the selected sprites relative to the canvas. For example, choosing the centre radio button with **Canvas** and both **Vertical** and **Horizontal** selected before choosing **Move selection** will centre all of the selected objects around the middle of the canvas area.

Largest

This aligns the selected sprites relative to the largest of the selected sprites (the largest sprite will not move). For example, choosing the centre radio button with **Largest** and both **Vertical** and **Horizontal** selected before choosing **Move selection** will centre the smaller sprite around the larger one.

Smallest

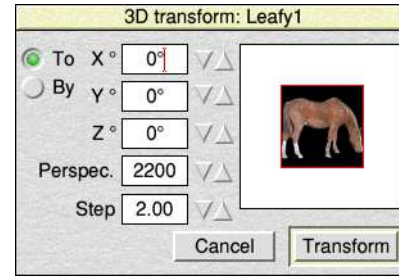
This aligns the selected sprites relative to the smallest of those selected.

The key short-cuts **F5**, **F6** and **F7** by themselves or with **Shift** or **Ctrl** allow quick access to most of the features provided by this dialogue box.

Transform (Ctrl-R)

Clicking on the **Transform** icon with MENU opens the **Sprite transformation** dialogue box if at least one sprite has been selected. If no sprites are selected then clicking on the icon does nothing.

3D Transform dialogue box



This dialogue box lets you transform the selected sprites by rotation about two or three axis. You can also add a Perspective effect. The default value is 2200 which has no effect on perspective. Try values between 300 and 2200 to see what effect they have in conjunction with Y and Z transforms. Low values tend to tie the image in knots or make it very large. See the image file `Resources.Perspec` for more information.

As you change the values using the bump arrows, the example box on the right changes. The black object represents the original sprite, the red one gives an idea of how the selected sprites will look when transformed. The order in which scaling and transformation functions are executed has a profound impact on the quality of the images. See the image file `Resources.ScaleRot` to clarify this.

As with the scaling option, it is advisable to choose the Update Bases item from the File menu before selecting this feature so that future transformations can work from the original images.

Transforming anti-aliased text does not generally look good. If you must do it, first scale by a factor of 2; do the transform; create a Blend mask and then scale by a factor of 0.5.

Transform discards any masks, clearing them to white.

Transforming

Clicking on the **Transform** button with **ADJUST** lets you directly transform the selected image without closing the Transform dialogue box. After clicking on the Transform button move the pointer over the selected image and click with **ADJUST**. As you move the pointer up and down or left and right the image will be transformed to match in real time.

Rotating

Clicking on the **Transform** icon with **SELECT** enables you to directly rotate the selected image without using the Transform Dialogue box. After clicking on the Transform icon move the pointer over the selected image and click with **ADJUST**. A small icon will appear in the centre of the image showing the degree of rotation. At first this will show 360°, which is the same as 0° rotation.

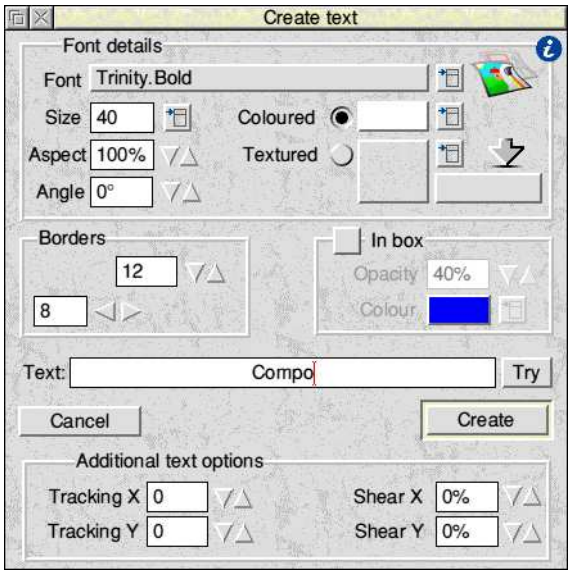
As you move the mouse pointer while holding down **ADJUST** the image will rotate and the angle of rotation is shown.

Text (Ctrl-Shift-E)

Clicking on the **Text** icon opens the **Create text** dialogue box. However, if the last sprite selected is text created with this dialogue box, then it is the **Edit text** dialogue box which opens. This dialogue box is very similar, but allows you to edit the piece of text rather than creating a new one.

Create text dialogue box

This dialogue box allows you to enter anti-aliased text into the composition via the writable **Text** icon. Once created, the text becomes a sprite, just like any other, and all of the normal effects may be applied to it. In addition, you can use the **Stamper** tool in the Mask Edit window to paste the text's **Blend mask** into another sprite in order to produce the very common effect in advertising, where an image appears through text.



Clicking this icon toggles between creating text in the bitmap and in the vector layer.

Although the specified text becomes a sprite, the program retains the information required to recreate the Edit text dialogue. This means that you can later edit any of the attributes associated with the text or even the text itself.

Font

The menu icon beside the Font field lets you choose a font from those available.

Colour

The menu icon beside the Colour field lets you choose the colour of the text to be created.

Size

The Size writable icon lets you set the size of the text. The value is in points and the menu icon allows you to choose from a list of preset sizes.

Aspect

Aspect lets you change the Aspect ratio of the text. Either enter a new value into the writable icon or use the bump arrows to change the value already present.

Angle

Angle enables you to change the angle at which the text is printed.

Borders

The vertical and horizontal values define a border of blank pixels surrounding the text. This is useful if the text is to have a shadow. This may be altered after the text has been created by using the **Trim** dialogue box. See Chapter 4 *Menus.*

In box

Superimposes the text over a partly transparent coloured tint box which can be used to make the text stand out over a detailed background. The **Opacity** and **Colour** items are used to alter the characteristics of the box.

Try

The Try button opens a window showing the text. This window can be left open while the attributes in the dialogue box are changed: it is updated automatically as you change attributes.

Cancel

Choosing Cancel closes the dialogue box.

Create new

The Create new button appears only if you are editing a previously created piece of text when it will force the program to create a new block of text rather than changing the edited one. This is useful as it enables you to read the attributes of a particular text sprite and then create another piece of text with the same settings.

Create / Change

The default action button will either contain the word **Create** as in the illustration if you are creating a new piece of text, or the word **Change** when you want to edit an existing text sprite.

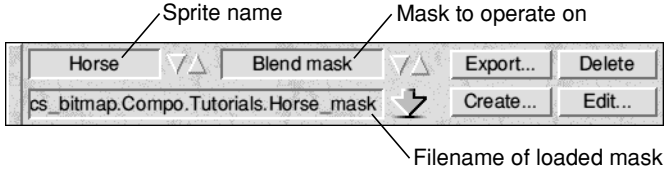
Pressing **Ctrl-E** while the pointer is over the canvas will **Create** text at the current pointer position in the canvas. The attributes of the last text block created or edited are used.

Undo / Redo

The **Undo** icon (on the right) and the **Redo** icon (on the left) enable you to undo and redo most of the operations described above. Compo keeps an extensive undo history and the change which can be undone or redone is displayed graphically below the relevant icon. Clicking on an icon with **ADJUST** has the same effect as clicking on the other icon. Clicking with the **Shift** key held down moves to the first or last undo recorded.

Horizontal Mask toolbar

This toolbar allows access to the various mask editing functions of Compo. The contents of the toolbar change as you select objects. It generally relates to the last sprite added to the selection. If no sprites are selected then the icons in the toolbar are greyed out and unselectable.



To select a sprite to work on, either select it in the canvas or use the bump arrows to toggle between the selected sprite names. Choose the mask you want to operate on by clicking on the relevant bump arrows. If there is already a mask of the type selected then its name will appear in the mask name area.

In all cases the amount of the relevant effect applied through the mask is governed by the grey level intensity of each pixel. White means no effect is applied, black means the full effect is applied and mid grey (value 128) means 50% effect is applied.

The available mask types are:

Blend mask

This mask controls the opacity of the sprite on a pixel by pixel basis, it also determines the shape of the shadow if no shadow mask exists. If this mask is not in use then the sprite's opacity is solely a function of the global opacity (see the Opacity dialogue box earlier in this chapter).

Tint mask

This mask controls the amount of tinting on a pixel by pixel basis. For this mask to have any effect a tint must be applied to the sprite (see earlier in this chapter). If no tint mask is in use and a tint is applied then all of the sprite is tinted by the same amount.

Curve mask

This mask controls the amount of colour curve and gamma correction applied to the sprite on a pixel by pixel basis. The Gamma or Colour curve values for the sprite must be different from the default for this mask to do anything. If this mask is not in use then the full amount of Gamma or Colour curves is applied to all of the sprite.

Displace mask

This mask causes a displacement or distortion of the sprite. Mid grey values in the mask produce no distortion, white values in the mask produce maximum positive distortion, dark values produce maximum negative distortion.

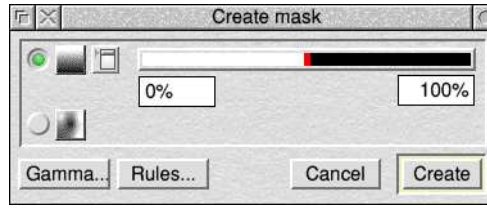
When first created (if you choose **Edit...**) the mask is automatically coloured mid grey. You will not notice any change until you paint into the mask. See Chapter 5 *Using Compo* for examples of how to use this unusual mask.

Shadow mask

If a shadow is active, this mask is used in preference to the Blend mask in determining the shape and density of the shadow. When this mask is first created (by choosing **Edit...** or **ADJUST** clicking on the **Shadow** icon in the main toolbar) it will inherit a copy of the Blend mask automatically which can then be modified by any of the tools in the Mask Edit window. This lets you have, for example, a very softly feathered shadow and a relatively sharp edge to the sprite at the same time. It also allows the more artistic user to create truly realistic three dimensional shadows by hand. Note that loading a file to the Shadow mask causes the mask to be scaled to fit the sprite; this is the only case where a file is always scaled to fit the sprite.

Create

Click on the **Create** button to open the **Create mask** dialogue.



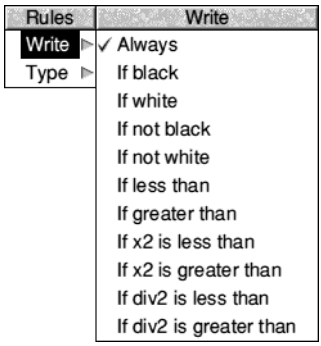
The top radio button has an associated menu opener which leads to a submenu containing different fill types such as vertical, circular, rectangular. These generate automatic fill patterns in the mask using a grey scale. The associated sliders control the intensity of the mask grey scale. 0% left and 100% right gives a pattern of dark to light. 100% left and 0% right gives the opposite.

50% on both gives a solid mid grey mask. When everything is set as you want it, click on Create to generate the mask.

Gamma

Click on this button to open a **Gamma correction** dialogue. In this case the **Combined** item cannot be unticked as the mask is grey scale and so you must change all components. The settings enable you to control the gradient used to blend between grey values when a new mask is created. Try settings of 0.2 and create a vertical fade. Next try settings of 2.0 with the same vertical fade.

Rules

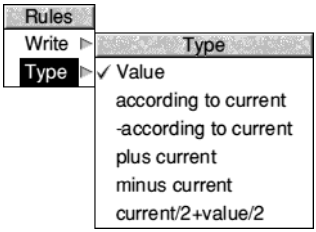


This item enables you to change the rules and tests used when creating a mask with the **Create mask** option. Its main use is to allow the mask that you create to modify an existing mask.

Write

This item leads to a submenu which enables you to choose the test applied to each pixel. The **Write** is applied only if the result of the Write test is true. For example, **Write ➤ If not black** will alter only those pixels in the mask that are not black.

Type



The results of this item are more subtle. Choosing an item other than the default **Value** item scales the value of the pixel being written to the mask based upon the value of the pixel in the mask.

For example, create some text, select it and the **Blend** mask, open the **Create** mask window, set **Rules ➤ Type ➤ According to current** and create a **Vertical fade**. You will notice that the vertical fade has been constrained to the shape of the text because black areas of the Text mask scale the fade being written to zero (anything multiplied by zero gives zero). The anti-aliasing around the edges of the text pulls the fade dark so you retain some of the edge anti-aliasing.

Compare this with **Rules ➤ Write ➤ If not black** instead to see the difference this makes.

Export

Click on the **Export...** button with **ADJUST** to open a standard save box allowing you to export the mask to disc or to another program. Clicking on the **Export...** button with **SELECT** sends the mask to a suitable editing application via OLE. Normally this would be RISC OS Paint.

Delete

Clicking on the **Delete** button deletes the mask and redraws the sprite. If the Warnings preference is on then you are asked to confirm the action.

Edit...

Click on **Edit...** to open an edit window for the mask selected. If there is no mask of the selected type present, then an empty one, the same size as the sprite, is created and the edit window opens onto it. See Mask Edit toolbar for more information.

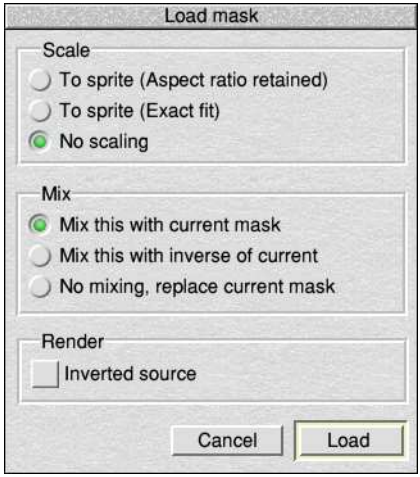
Loading a Mask

An existing mask or any image can be dropped onto the load target (the arrow) to load the mask. If the image is not a grey scale Compo converts it for you. If you hold down the **Shift** key while dropping a file to the Load target the Load mask dialogue opens.

You can then select if the file is scaled to fit the selected sprite, retaining aspect ratio if required. In addition, if there is already a mask in operation, you can choose to mix the masks.

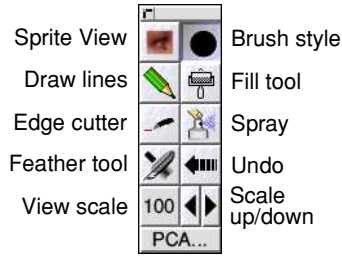
The Shadow mask is automatically scaled to the exact size of the sprite.

When loading a bitmap file called `image`, Compo automatically loads the related mask if it finds a file called `image_msk` in the same directory. Unfortunately this fails to work when either filename is longer than ten characters.



Mask Edit toolbar

This toolbar is attached to the Mask Edit window which is opened by choosing the **Edit...** button in the horizontal Mask toolbar or by **ADJUST** clicking on the **Shadow**, **Opacity**, **Tint**, **Gamma** or **Colour** curve buttons in the main toolbar.



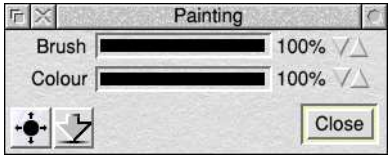
The Mask Edit toolbar

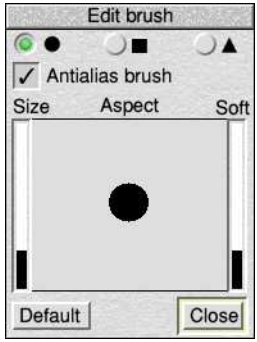
Sprite view

This opens the Sprite View window which shows the sprite in its original state, unmodified by effects and masks. It also lets you use the Magic wand and Snip tools to select areas of the sprite for inclusion in, or exclusion from, the mask.

Brush style

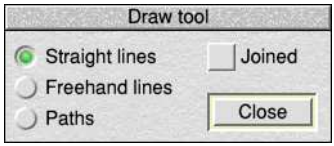
Clicking on this tool with **MENU** opens a dialogue box where you can set the shape and style of the brush which is used to paint into the mask.





Clicking on the button at the bottom left of this window opens the dialogue shown on the right. This lets you choose between three simple shapes, a triangle, square or circle, and set its size with the slider on the left. If the Soft slider on the right is dragged to a greater size than the slider on the left it will give the mask a soft edge.

Draw lines



This tool enables you to draw rubber banded lines in the mask. Click **SELECT** for the start point, drag the line to the end point and release. Dragging with **SELECT** draws a line in the current Mask paint colour. Dragging with **ADJUST** always draws a black line. This tool is useful for cutting off unwanted areas of a mask which can subsequently be removed with the **Fill** tool's Plain fill option.

Fill tool

The Fill tool enables you to fill an area of the mask with either a plain colour or a graduated range of colours. When you select this tool the dialogue box shown will appear at the bottom of the mask window and using this you can choose the fill type and the colours to use.

Plain fill fills in the current mask colour, if you initiate the fill with **SELECT**, or in black if you initiate the fill with **ADJUST**.

Angled and **Circular** are graduated fill types that require you to drag out an area to fill. The end point of this drag is the seed point from which the fill begins. The start point of the drag is the extent of the fill range and may be outside the object to be filled.

Angled fill enables you to have a fill at any angle or, by reversing the start and end points, in any direction e.g. dark to light or light to dark.

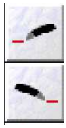
Circular fill enables you to have a circular fill and, by reversing the start and end points, you can have it or dark to light or light to dark.

Start and **End** enable you to choose the start and end shades. The fill is always a grey scale fill. The graduated fill effect is indicated in the small block on the right of the dialogue box.

Should the fill leak or not produce the desired effect, choose the Undo button to remove the fill (see below).

Edge cutters

The **Edge cutter** tools are used to trim rubbish from the edges of the mask. The tool draws a line from the left, right, top or bottom of the mask to the current pointer position. The direction of the cutter is chosen by repeatedly clicking on the tool. The four edges cutter tools are:


Cut to left

Cut to right


Cut to top

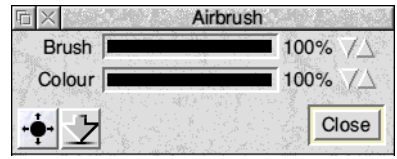

Cut to bottom

SELECT clicks draw lines in the current mask colour (see the Fill tool). **ADJUST** clicks draw in black.

Spray tool

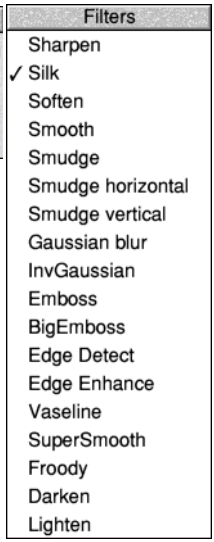
the **Spray** tool is a simple airbrush. Click **MENU** over the tool and the dialogue box shown will appear. This lets you change the characteristics of the spray. It can be particularly effective to use the Feather tool after spraying an area.

The icon at the bottom left opens a dialogue which lets you set the shape of the spray. This is the same as the one used to set the Brush shape.



Clicking **ADJUST** on the tool instead of **MENU** will attach the dialogue shown below to the bottom of the window. Clicking on the icon on the left toggles between **Brush shape** (shown), **Colour** and **Density**.

Feather tool



This can be used in a variety of ways to alter the mask's appearance. Most common uses involve darkening or softening an area of the mask, particularly its edges.

Press **MENU** over the icon to open the **Feather** dialogue box. This lets you change various aspects of the Feather tool, including the type of Strength filter used and the size of feathering. Clicking on the menu icon opens a menu showing the various filters available (shown at right). If Repeating is ticked the Strength filter is additive, that is it affects the pixels each time you wipe over them. If it is unticked then the filter is applied to the pixels only once.

As with many of the other tools if you click on the tool icon with **SELECT** instead of **MENU** then a version of this dialogue will attach to the bottom of the window.

Undo

The **Undo** icon lets you undo a modification made to the mask. Depending on the number of undo buffers configured and the amount of memory free in your computer, it may be possible to undo more than one step by repeated clicks on the Undo icon. If there is not enough memory free to create an undo buffer when you are about to modify a mask, you are warned and the action you were about to take is aborted. Either free some more memory or turn the Undo option off (see Chapter 4 *Menus* for description of Preferences).

Sprite View toolbar

This small toolbar is attached to the Sprite View window. In addition to the standard set of Scale icons it has the following two buttons:

Snip tool

This, when selected, causes clicks in the Sprite View window to draw a smoothed freehand line in the Mask Edit window. This lets you trace around the edges of an object to produce a mask outline which can then be filled. **SELECT** drags in the View window draw lines in the current Mask paint colour, **ADJUST** draws black lines.

Magic wand

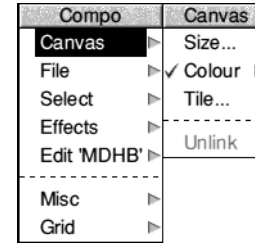
This is used to add pixels of specific colours to the mask. The associated menus let you set the search range, the colour value either side of the pixel colour that you dick on, and the size of the search area in pixels. Pixels in the sprite which are sufficiently close in colour to the one clicked on are added or removed from the mask depending on whether **SELECT** or **ADJUST** is used. For more information on the various settings available to let you adjust the sensitivity of this process see the next chapter.

4. Menus

This chapter describes the menus provided by the program.

Main menu

The M**ain menu** appears when you click **MENU** over the canvas window. The submenus and dialogue boxes associated with the menu entries are described in turn.

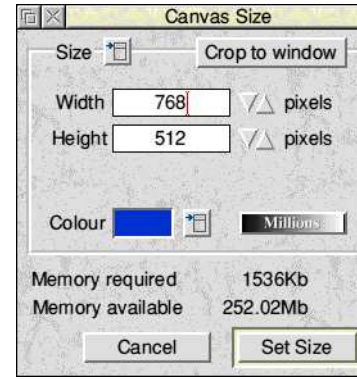


Canvas submenu

The canvas submenu lets you change aspects of the canvas even while working with it. This includes its size and whether is is plain colour or has a tiled background.

Size...

Choosing **Size...** opens the **Canvas size** window. The items in this window are:



Size This icon leads to a default set of canvas sizes that you may find useful.

Width You can alter the canvas width using the writable field or the bump arrows.

Height You can alter the canvas height using the writable field or the bump arrows.

Note that Compo works internally in pixels and so the width and height figures may be rounded up or down slightly from what you enter.

Colour Leads to a standard colour picker for the canvas colour.

Memory required Is the amount of memory required for the new canvas as shown in the dialogue.

Memory available Is the amount of free memory available (shown in red if the canvas is ikely to be too big for available memory).

Colour

This item leads to a standard colour picker for the canvas colour. Clicking on the menu item itself forces the program to use the last colour defined for the canvas. A tick appears beside this item when the canvas is a plain colour.

Tile...

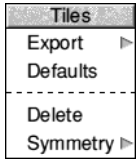
When chosen, this item opens the **Tile pool** window.Use the bump arrows to move through the tile pool. Choose the Use button to use the displayed tile as the canvas background. Choose this button with **ADJUST** to keep the tile window open, with **SELECT** to close it. A tick appears beside this item when the canvas colour is a tiled sprite.

You can drag any of the bitmap file formats supported by Compo (including JPEG, Sprite. TIF, GIF, PhotoCD etc.) onto the load target in the **Tile pool** window to add the image to the tile pool. Compo automatically scales and converts it to the Preferred tile Size. See the **Preferences** window below if you wish to change this. In some cases, e.g. if the sprite name is `Tile<nn>` or is exported from the **Edit** menu, the sprite is not automatically rescaled. To force rescaling, hold down **Shift** while dropping the sprite onto the Tile window.

For technical reasons it is not possible to load Squashed sprite, Draw or Artworks files directly into the Tile pool. Load the file into the canvas and use the **Edit ➤ Export** dialogue to transfer the image to the **Tile pool**.

To export a tile, drag the tile being displayed in the pool window to a Filer window or to another program for editing. The tile is saved with a name related to its position in the pool e.g. `Tile7`.

Click **MENU** over the tile window to open the menu shown.



Export This item enables you to edit all of the tiles in another program or to save a set of tiles to disc for later use.Tiles which have been exported may be re-imported into the program by saving from the editing program directly into the tile pool window: the tiles loaded will replace those in the program and, if necessary, the canvas is redrawn to reflect the changes.

Defaults Resets the Tile pool to the default tiles loaded with the program.

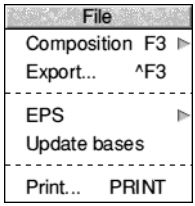
Delete Deletes the tile currently displayed in the Tile window. If the tile is being used in the canvas another tile is chosen from those remaining in the pool and the canvas is redrawn.

Symmetry This item leads to a submenu where you can set the tile symmetry in Horizontal, Vertical or Both dimensions. They work by mixing opposite edges of the tile together lo produce a symmetrical tile. Although these options are intended for use with nearly symmetrical tiles produced by the Colour block dialogue box (see Chapter 5 *Using Compo*). Interesting effects can be created by applying these items to a scanned image. Try loading a JPEG image, e.g. `!SlideShow.Images.00-49.sa24` butterfly, into the tile pool and choosing Symmetry ➤ Both.

Because these items modify the tile it is advisable to save a good tile to disc (as described above) before choosing one of the Symmetry items.

File submenu

This submenu lets you save the composition in a number of file formats. It also has an item which lets you save modified sprites and masks. An important general point about Save dialogue boxes within Compo is that you are allowed to transfer a file from one part of the program to another. For example, it is possible to load an image, use the Trim dialogue box to reduce its size and then choose Export (both in the Edit submenu detailed below) to transfer the trimmed sprite to the Tile pool for tiling the canvas.



Composition

This format is Compo's own format. It preserves all of Compo's effects and the independence of objects. On reloading a Composition file at a later date you can continue where you left off.

The file format has a couple of options which affect its internal structure and the size of file produced. To see them, click the Option arrow.

With images and masks This is on by default. The Composition file will contain all the image and mask data for the composition. This file format takes the most disc space but, assuming you have enough free memory, is guaranteed to reload all of the composition.

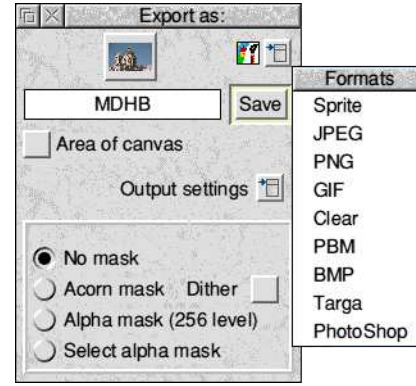
If this option is off then the Composition file saved will contain all of the effect and positioning information but not the image and mask data. It will attempt to locate image and mask files relative to the directory in which the Composition file is saved. When saving in this way you will are given the opportunity to save modified masks and sprites via an automatic **Update bases** command. See below.

When reloading this format (without images and masks), the program simulates the user dragging in the various files and masks to recreate the composition. This may fail if, for example, you have deleted or moved one of the files. If Compo fails to find a file as it loads it will open a dialogue box giving you the opportunity to find the file or a suitable alternative. If you cannot, the program will use a default sprite instead.

Squash internals This attempts to compress the Composition file internally as it is saved. Squashed Compositions are slower to save and reload but require less space on disc.

Export

This opens the Export dialogue.The menu icon at the top right opens a menu from which you can choose the format in which you wish to export the canvas. The options are:



Sprite Saves a standard 32bpp RISC OS format sprite containing the canvas.

JPEG Saves the canvas in industry standard JPEG format. Output quality may be changed in the Preferences window.

JPEG is a lossy format. It discards part of the image data to reduce file size. Because of this, you should not save compositions in JPEG format if some of the source files were JPEGs, since degradation of image quality becomes very noticeable after several iterations.

For the same reason you should not use JPEG format when a composition contains fine graduations, such as tints, colour blocks or imported vector format files. The JPEG algorithm degrades these noticeably.

PNG Portable Network Graphic. This is the bitmap graphic format intended for internet use.

GIF Graphics Interchange Format. A very widely used bitmap format for web use that is limited to 256 colours.

Clear This 24-bit file format is recognised by a number of RISC OS applications. Files are 25% smaller than Sprite format as the Alpha channel is not saved. Clear files may be translated to a number of foreign formats via the utility Creator.

PBM-Plus A standard file format in the Unix world. Compo outputs 'p6' PBM files.

Targa A-24 bit file standard in the PC world. Compo creates 'type 2' 24bpp images.

Photoshop Exports the file in the format used by the widely used bitmap editing program.

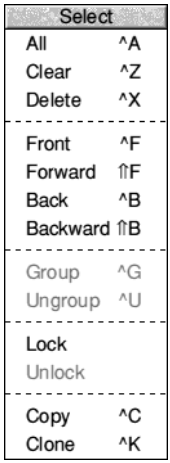
Update bases

Choosing this item (or saving a Composition without images and masks) causes the program to search through all of the sprites and masks in the composition. If an object has been modified in the program and not subsequently saved, Compo opens a dialogue box allowing you to save it.

This is an important feature. When rescaling and transforming objects Compo will use the original file whenever it can. To ensure the best possible image quality, it is important that you create masks and images and then choose this item before rescaling or transforming them. If you do this, scaling objects back up or transforming them again will result in much better image quality as the program has these full quality bases to work from.

Select submenu

This submenu is similar to Draw's submenu of the same name.



All Selects all of the sprites in the composition.

Clear Deselects all of the sprites which were selected.

Delete Deletes all of the selected sprites and their associated masks from the composition.

Front Moves the selected sprite(s) to the front.

Forward Moves the selected sprite(s) forward (up in the stack).

Back Moves the selected sprite(s) to the back.

Backward Moves the selected sprite(s) rearward (down in the stack). When there is more than one sprite selected the action of Front and Back cannot be guaranteed to produce any particular effect though the selected objects will move in the right direction.

Group Groups all the selected sprites so that they can be selected and deselected together as if they are one unit. This does not make them into one unit: actions such as scale, alignment, etc. act on each individual item independently.

Ungroup Reverses the action of the previous item, and removes grouping from all of the selected sprites (though they will still be selected).

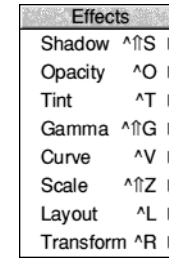
Lock Locks the selected sprites. Locked sprites cannot be moved, their opacity, shadow, etc. cannot be modified. Useful if you have finished part of the composition and do not want to modify it inadvertently.

Unlock Reverses the action of the previous item. Objects which were locked are now free to be modified.

Copy Makes copies of the selected sprites. The copies are then selected. Initially, the copies are identical to the originals.

Effects submenu

The **Effects** submenu duplicates the buttons in the Main window toolbar. These modify the selected sprite's effects.It serves as a textual version of this part of the toolbar which some users may prefer. It also provides reminders of the key shortcuts used to open these dialogue boxes

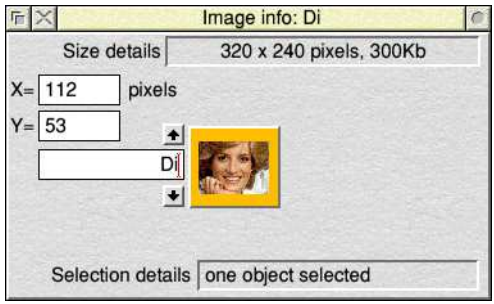


These were all fully described in Chapter 3 *Toolbars*, so the descriptions will not be repeated here.

Edit 'sprite' submenu

The exact appearance of this item in the main menu varies, depending on the sprites selected. Generally, the name of the last sprite added to a selection appears in the menu. It leads to a submenu (right) which holds effects and actions which cannot be logically applied to more than one sprite at a time. The actions are applied to only the named sprite.

Info Opens the **Image Information** window. This window not only provides useful information about the last selected sprite but also lets you enter the precise position of the sprite by hand.You can also use the small arrows to move selected sprites in the stack. Up to three sprites are shown at any one time.



It may be useful to keep the info window open while you work with the composition. If it becomes hidden as you work then pressing **Ctrl-I** will pop it to the front.

OLE Choosing this item has the same effect as **Ctrl** double clicking on a sprite in the canvas and sends it to a suitable editing application using the OLE standard. Saving the edited sprite from the editing application, using the standard save procedure, will return the edited image to Compo automatically.

Make text This opens the **Create text from sprite** dialogue. This is essentially the same as the Create text dialogue except that the colour options are greyed out. In addition there is a new button labelled **Fit to sprite**. This option creates a text mask which is attached to the Blend mask of the selected sprite. When you click on Create the text forms a window onto the sprite so that the sprite is visible only through the text.

The Fit to Sprite button scales the text to fit the selected sprite. The calculation is based upon the text size when the dialogue is opened so, if you change it, you may need to select this button twice. As with other masks you can move the mask on the sprite using **Shift** or **Ctrl** while dragging.

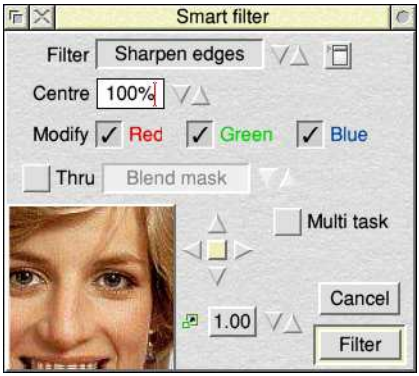
Save Leads to a **SaveAs** dialogue from which you can save the selection as a Sprite file.

Export This lets you export the sprite to another application or to disc. If the destination is a disc then the pathname for the sprite is updated. Compo also allows you to transfer the sprite to the Tile pool if the tile dialogue box is open. See Chapter 5 *Using Compo*.

To maintain image quality when rescaling sprites, you should use only one of the three items Filter, Horizontal flip and Vertical flip, on a sprite which has not been rescaled. Remember to choose Update bases from the File menu before any rescaling is done.

Filter... This opens the **Filter** dialogue box which lets you modify the image in a number of ways such as applying sharpening and embossing effects. Choose a filter by clicking on the bump arrows beside the filter name. The menu icon opens a filter definition window which is explained in detail in Chapter 5 *Using Compo*.

The **Centre** value is related to the strength of the filter and its effect varies from filter to filter. Generally, a larger centre value produces a weaker filter.



The **Modify** buttons let you restrict the filter to any combination of the red, green or blue components of the image.

Thru uses the named mask to control the strength of the filter on a pixel by pixel basis within the sprite. The bump arrows let you choose from the available masks.

If a given mask has not been created for the sprite then the filter will execute evenly over the whole sprite.

Click on **Filter** to process the image. This may take a few seconds with large images and complex filters. You can abort a filter while it is working by pressing **Esc**. The filter window will stay open if you choose Filter with **ADJUST**. This lets you select another sprite and execute another filter without reopening the dialogue box.

The Example box lets you see the effect of a filter on a part of the image. As you toggle through the filters the filter is applied to the Example window. The four bump arrows to the right of the Example box move the sprite relative to the example box. The bump arrows beneath the example box scale the image in the Example box.

Tick **Multi task** if you want to use other applications whilr a filter is in progress. This is required only when processing very large sprites. Filters work much faster with Multi task disabled.

Horiz. flip Flips the sprite and all associated masks horizontally. Choose this item again to reverse the effect.

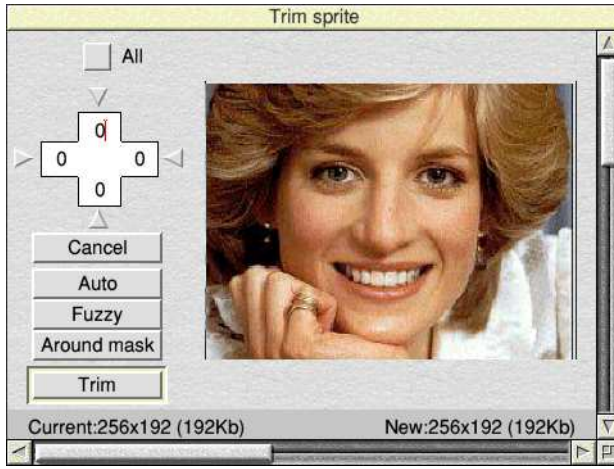
Vert. flip Flips the sprite and all associated masks vertically. Choose this item again to reverse the effect.

Trim This leads to the **Trim** dialogue box where you can increase or decrease the size of the sprite. It is also possible to make the program trim the sprite to the boundaries of the masks.

To reduce the sprite's size simply drag a box over the area you wish to keep. The surrounding portions of the image will be inverted and it is these areas and the associated areas of all of the masks that are discarded when you click on **Trim**. To fine tune the amount, use the four bump arrows or enter values in the writable icons.

To increase the size of the sprite click with **ADJUST** on the appropriate arrow icon. A negative value means that the number of rows or columns shown will be added to that side of the sprite.

Selecting **Around mask** sets the trim values to remove all areas of the sprite which relate to black areas in the masks.



Trimming a sprite may make it use less memory and allow it be redrawn faster.

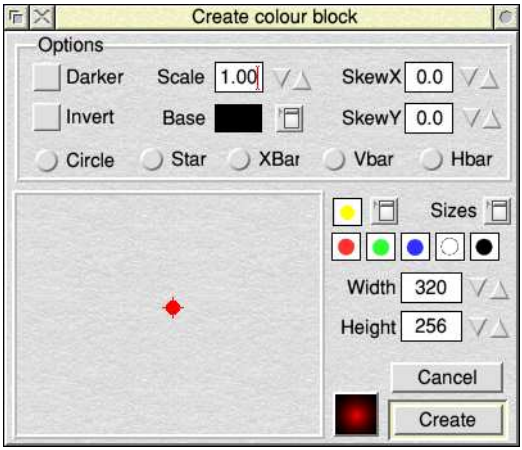
Trimmed sprites and masks are marked as modified. To retain image quality for rescale operations, you should do any trimming before rescaling the sprite and select **File ➤ Update bases** immediately after trimming.

Misc submenu

The **Misc** menu contains the following items.

Text This has the same effect as the text button in the main toolbar.

Block... This opens the **Colour block** dialogue box which lets you create a sprite which is filled with shaded areas of colour. The resulting sprites are useful for a number of purposes. Examples include multi-coloured text, canvas tiles and mixing effects. See Chapter 5 *Using Compo* for some detailed examples.



The colour of each pixel in the created sprite depends on its distance from each of a number of user defined colour points. Compo allows you to arrange up to 32 of these points and each may be any of the 16.7 million colours available. In addition, each point may be one of five different shapes which define how the strength of the colour point falls off with distance.

To add a point to the colour block drag one of the coloured circles onto the definition area at the left of the dialogue box. To move a point on the definition area simply drag it to where you want it. To remove a point from the definition area drag it off the area.

To change the shape of a point within the area double click on it with **SELECT** to advance through the available shapes. Double click **ADJUST** over a point to move backwards through the available shapes. The radio buttons in the Options section change to reflect the shape of the point. These options are also used to define the shape of a new point, which can then be dragged onto the area.

There is a set of standard colours and one user definable colour. To change the definition of this, use the standard RISC OS colour picker which is opened by clicking on the menu icon to the right of the user defined colour, which initially is shown yellow. This colour may be changed after each point has been created, allowing access to Options.

Width defines the width of the sprite to be created.

Height defines the height of the sprite to be created.

Create with **ADJUST** will create a colour block and keep the dialogue box open. Create with **SELECT** will create a colour block and close the dialogue box. The block created is initially positioned in the middle of the canvas.

A small thumbnail of the block beside the Create button gives an idea of what the block will look like when created. This thumbnail is updated as you make changes.

The Options at the top of the window enable you to change the way the colour block will look.

Darker has a quite subtle effect and, when on, tends to produce blocks in which the effect of each colour point drops off more rapidly.

Invert has a very strong effect. It causes colour points to have more influence the further away a pixel is from them. It produces a logical inversion of the colour block blend.

Scale can be thought of as a means of zooming on the colour block. Values greater than one produce less contrast in the resulting colour block. Values less than one produce more dramatic changes in the colour block.

Base colour letsyou set the base colour of the colour block via a standard RISC OS colour picker.

Skew X lets you distort the shape of the colour block horizontally.

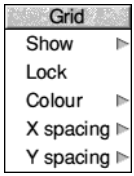
Skew Y lets you distort the shape of the colour block vertically

Experimenting with these very powerful features will soon give you a feel for what can be produced. In the next chapter we give a few examples.

The Block tool can also be used to create a box to place behind text. The box is not attached to the text unless you group the text and the block together. Both items can be scaled and moved independently. This is sometimes more convenient than having the block fixed to the text.

To create the block set the Base colour and the colour of the user definable point to the required colour and drag the user defined point onto the block. Drag the original point off the block to remove it and then click on Create. You can now scale and move the block as well as alter opacity, masks, etc.

Grid submenu



The Grid submenu lets you control aspects of a grid which can be overlaid on the canvas. Choosing the Grid item itself toggles the state of the Show and Lock items described below.

Show This toggles display of the grid over the canvas on or off.

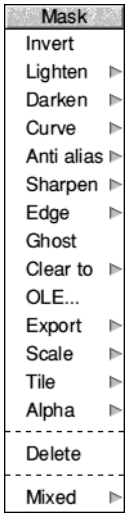
Lock This toggles the locking of objects to the grid. When dragging sprites their positions will snap to the grid points as they are dragged.

Colour This lets you change the grid colour via a standard RISC OS colour picker. Choose a grid colour which will stand out well against the sprites in the canvas.

X spacing lets you change the horizontal spacing size of the grid in units of one pixel.

Y spacing lets you change the vertical spacing size of the grid in units of one pixel.

Mask editing menu



This is the menu which appears when yu click MENU with the pointer over the Mask Edit window.

Invert ➤

Inverts the mask - black becomes white and vice versa. Choosing this item again reverses the effect.

Lighten ➤

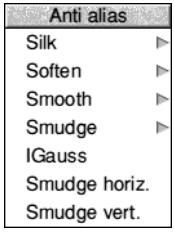
Lightens the mask.

Darken ➤

Darkens the mask.

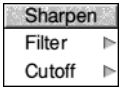
The submenus for Lighten and Darken enable you to choose a particular range of values in the mask to operate on. For example Darken ➤ All but white would tend to reduce anti-aliasing of the mask. If used repeatedly, it would eventually give a mask consisting of only white or black pixels. All of the pixels between black and white would have become black.

Not surprisingly, there is an option All but black and if you choose Lighten ➤ All but black repeatedly then all grey pixels would eventually become white. Many of the functions in this menu have an identical submenu and can be used in a similar way. For example. Anti-alias ä Smooth ä All but black will tend to erode the white edges of the mask.



Anti-alias ➤The Anti-alias submenu lets you soften the edges of the mask. Silk, Soften. Smooth and Smudge provide progressively more intense smoothing functions.

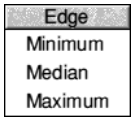
Sharpen ➤The Sharpen submenu has an opposite effect to the Anti-alias submenu and lets you remove anti-aliasing from a mask.



Filter This sharpens the mask using a simple filtering function

Cutoff This leads to a slider dialogue box and. when OK is chosen within this box all pixels darker than the cutoff value chosen are turned black. All pixels lighter are turned white. The slider value relates to the grey scale value of the pixel in the mask.

Edge ➤



Applies an edge smoothing routine which has the effect of reducing the size of the mask inwards. There are Minimum,Median and Maximum functions. Choosing Median is useful for removing stray pixels from the edge of the mask.

Ghost

This is unusual. It is a little like a violent smoothing function and produces a ghostly ethereal sort of effect. Choose Anti-alias ➤ Smooth after Ghost for best effect.

Clear to ➤

Allows you to clear the mask to black or white

OLE...

This sends the mask to a suitable editing application using the OLE standard.

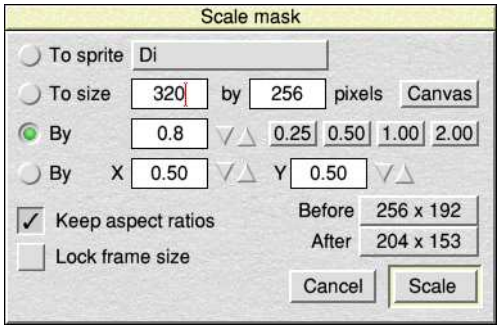
Export ➤

This leads to the standard Save dialogue box, allowing you to export the mask to disc or another application.

If you double-click on the file icon it will to toggle between sprite, JPEG and Squashed sprite formats.

Note: For best quality always use sprites or Squashed sprites. When rescaling a mask, the base file on disc is not used if it is a squashed tile. This will result in some loss of quality.

Scale ➤



This opens the dialogue box shown which lets you scale the mask. You can either enter the values directly or use the bump arrows. You can scale X (horizontal) and Y (vertical) sizes separately or together.

Clicking on Canvas will scale the mask to the same size as the current canvas.

Delete

Deletes the mask and removes the Mask Edit window.

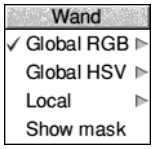
Mixed ➤

Choosing Mixed determines whether the program will mix the sprite and mask together in the Mask Edit window. This requires additional memory, (which is released when the Mask Edit window is closed), and slows down redraw of the edit window, but it can be useful for precise editing of the mask in relation to the sprite.

If there is not enough free memory or if the mask is not the same size as the sprite the mask is displayed by itself as if the option was off.

The submenu leads to a standard RISC OS colour picker, which allows you to select the colour used to represent the mask. Choose a colour which contrasts well with the colours in the sprite.

Sprite View menu

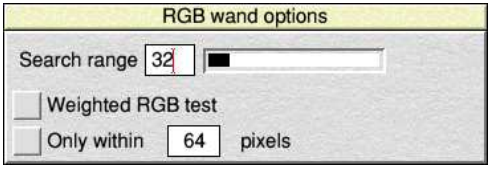


This small menu appears when you click MENU over the Sprite View window or the Wand icon. It allows you to change the settings of the Magic wand tool and to choose between a RGB wand and a HSV wand.

The RGB wand tests pixels in the red, green and blue colour space for their similarity to the one clicked on. The HSV wand tests pixels in terms of their distance in Hue Saturation and Value from the pixel clicked on. Which wand you use will depend on. the sprite being masked. The RGB wand is almost always satisfactory, and is much faster than the HSV wand to use.

There are separate options for Include (i.e. add pixels to the mask) and Exclude (i.e. remove pixels from the mask). Colours are included if you use SELECT to click on a pixel or they are excluded if you use ADJUST to click on a pixel.

Global RGB



This leads to a submenu with two items. Include options and Exclude options. Each leads to a window where you set the items to be included or excluded.

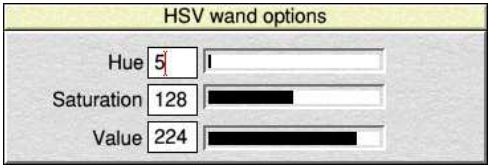
Search range Enter a value into the Search range writable icon or drag the slider. The value is a distance in a three dimensional colour cube, if a tested pixel in the sprite falls within this distance it is deemed acceptable for inclusion in, or exclusion from, the mask.

Weighted RGB test If this is on the search is weighted to more closely match the human visual system. The weights are set to provide a strong green bias as the human eye is much more sensitive to changes in green than red or blue.

Only within If this is on, the program considers only pixels within the radius set in the writable icon. In addition, as a tested pixel is further from the initial pixel its colour must be closer to the colour of the initial pixel.

Global HSV

As with RGB this leads to a submenu with Include and Exclude options, each of which lead to a dialogue box as shown.

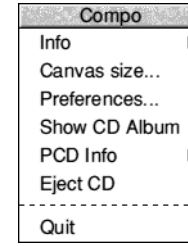


Hue This ranges from zero to 360 and is a distance around a colour wheel which will be accepted for inclusion or exclusion from the mask.

Saturation This defines the amount of change in the intensity of a pixel which is acceptable for inclusion in/exclusion from the mask. It can be thought of in terms of the amount of white in a colour.

Value This defines the amount of change in the brightness of a pixel which is acceptable for inclusion in/exclusion from the mask. It can be thought of in terms of the amount of black in a colour.

Icon bar menu

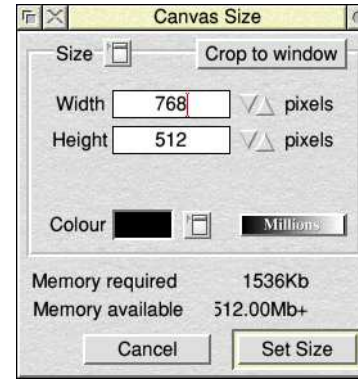


The icon bar menu appears when you click MENU over the application icon on the icon bar.

Info

This leads to an information box providing information about the program including a version number and date which you should mention in any correspondence. It also displays the serial number of the program.

Canvas size...



This opens the Canvas Size dialogue box.Clicking ADJUST on the application icon has the same effect. This item is available only if a canvas already exists.

Preferences...

Opens the Preferences window, which is described fully later.

Show CD Album

With a Kodak PhotoCD in the drive, clicking on this option produces a window containing the first ten thumbnail images on the disc. See Chapter 6 for more details.Note: Do not choose this item while the Busy light on the CD Player is lit.

PCD Info

After Show CD Album has been chosen this item will lead to a dialogue box giving information about the PhotoCD.

Eject CD

This ejects the CD from the drive. You must use this when removing a disc as the disc draw is locked to prevent problems from unmounted CDs.

Quit

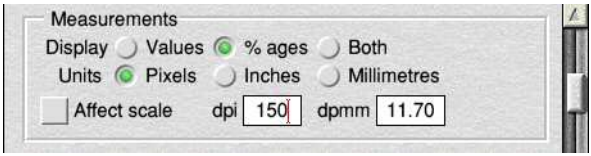
Quits Compo, prompting if you have a modified composition in memory.

Preferences window

The top section of the window may not be visible when you first open it. To see this area either use the vertical scroll bar.

Measurements

The window controls the format of the measurements and size values displayed by the program.



Display The Display radio buttons control whether component values or percentage values are used for dialogue boxes, such as the Opacity dialogue box. Component values usually range from zero to 255. Percentages from 0-100%.

Units The Units radio buttons control the display units for things like sprite positions, sizes of sprites and canvas size etc. Compo works internally in pixels and values entered in inches or millimetres may be rounded down to the nearest pixel. This means that the values in the dialogue box may change when you next view them.

You can also change the dots per inch and dots per millimetre values by altering the value and pressing **Return**. DPI is a device independent setting for the size of the image. A canvas set at 300dpi and 70mm by 50mm looks very much larger than 70mm * 50mm on screen. This is because monitor screens usually work at about 90dpi. The image will look bigger on a 17 inch monitor than on a 14 inch. However, when you send your image to a printer (the man not the device), he will output it at very high resolution onto film and it will then appear at the correct size.

In summary: the setting of the DPI value is a device independent setting which should be set to the dpi of the ultimate output device e.g. a Raster Image Processor. To complicate matters your printer may talk in terms of Lines Per Inch (LPI) or dot screen. Most quality litho printing is done at 150dpi and this equates nicely to using 300dpi. For lower quality or mono work, 150dpi can be used.

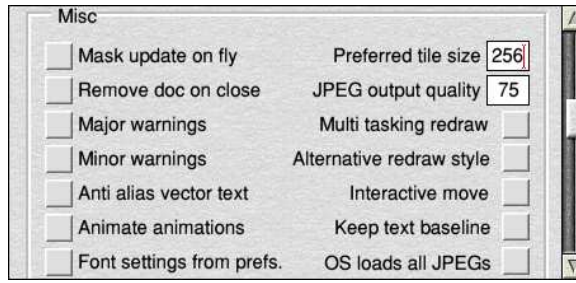
Misc

The second section of the Preferences window holds a variety of miscellaneous options and settings. They are:

Mask update on fly When this option is on (the default), painting in a Mask Edit window updates a corresponding area of the canvas to reflect your changes. When this option is oft. the canvas is updated only when you stop painting in the Mask Edit window.

Preferred tile size This is the preferred size to scale an image for use in the Tile pool. The size is in pixels and the image being converted is scaled to a size as close to this as possible while still maintaining its aspect ratio.

Remove doc on close If this is on then when you close the canvas window a warning is issued if you have not saved the canvas. You can choose to continue and lose the canvas or save it. If this option is off the canvas window is closed but the canvas remains in memory and can be redisplayed by clicking with SELECT on the icon bar icon



JPEG output quality This controls the level of compression and image quality when saving a JPEG file. Values between 10 and 99 are acceptable but normal ranges are from 70-90. The smaller the number, the smaller the final file size but the lower the image quality.

Multi-task redraw By default, Composition uses the entire resources of the computer while such operations as painting in a mask are underway. This improves the responsiveness of the drawing tools. If you want other programs to continue working while these operations are underway then turn this option on.

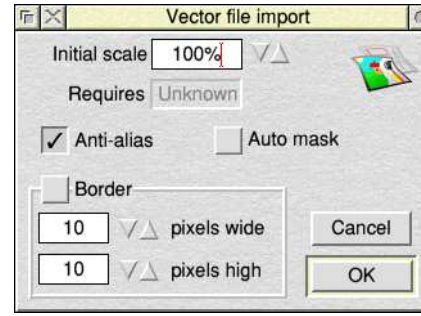
Major warnings When on, the program warns you before doing something potentially hazardous like deleting a mask.

Minor warnings When on the program warns you before doing things such as deleting a sprite, trimming a sprite etc.

Vector file import

Open dialogue box

If this radio button is selected a dialogue box (right) opens when you import a Draw or Artworks file. The dialogue box lets you select various settings before the file is loaded.

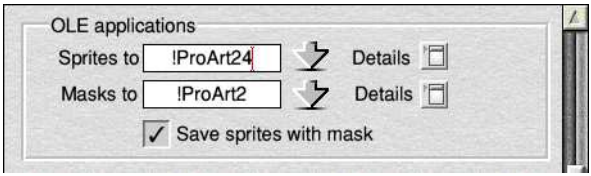


Use settings

If this radio button is selected, the settings defined in the associated dialogue box are used automatically.

OLE applications

The final section of the window concerns OLE - The method by which sprites and masks in Compo can be edited in other programs and quickly returned to the Composition with their effects untouched.



Composition's system of OLE is slightly different to other packages in that you specify a preferred program to edit images with. This is because, typically, many programs are able to load sprite files and it is not always clear which will get a particular file when double clicked on.

To change the default package for OLE drag the icon of the application you wish to use onto the appropriate load target icon in the Preferences window.

If the application conforms to the standard RISC OS name guidelines you will need to do nothing more. The text which appears beside the load target is the application name (e.g. !ProArt24). Click on the Details menu icon to change the task name and director) name tor the application. These are set automatically to a suitable default when you drag the application icon onto Compo.

By default, Paint will not allow you to edit sprites from Compo. See Appendix D *Technical issues* for details of a fix to make Paint edit these sprites. Compo will attempt to OLE a sprite to an appropriate editing package when you hold Ctrl down and double click on the sprite in the canvas.

Compo will attempt to OLE masks to ProArt2 or Paint when the OLE... button in the horizontal Mask toolbar is chosen.

Masks and Sprites modified in another package will automatically return to Compo when saved as long as the filename is not altered.

Applications which support the **F3** followed by **Return** short cut to save a file under the same name are particularly handy.

Note: Compo will cope if you change the size of the sprite in another program, although any masks in use by the sprite will be positioned incorrectly when the sprite is returned to Compo. It is recommended that you use Compo's Trim dialogue box to change the size of a sprite. See earlier in this chapter.

5. Using Compo

This Chapter is intended to give you ideas by explaining how some of the features and effects, described in the previous two chapters, may be used and combined to produce interesting effects. The ideas are presented as a series of tutorials, in a similar manner to the tutorials in the first chapter.

Shadows

Shadows for rectangular objects

1. Create an empty canvas. Click **MENU** over it, follow Canvas ➤ Colour and choose White from the colour picker.
2. Load the image `Magnolia` from the supplied tutorials.
3. Select Magnolia and **ADJUST** click on the Opacity icon in the main toolbar to open a blank Blend mask for the object.
4. Click **MENU** over the Mask Edit window and choose Clear to ➤ White. Close the Mask Edit window.
5. Choose the Shadow icon in the main toolbar, turn the shadow option on by clicking on the Active box And choose **OK** (Alternatively, Press Ctrl-S). The shadow will not be visible at present because all of the Blend mask is solid. To correct this, click **MENU** over the canvas and follow Edit Magnolia ➤ Trim. **ADJUST** click on the bottom and right hand arrows in the Trim dialogue box to extend the appropriate sides of the sprite to reveal the shadow. A value of -10 should be enough. Remember to **ADJUST** click on Trim in the trim dialogue box to keep it open so that you can repeat the operation if all of the shadow is not visible.

Soft shadows using the Shadow mask

1. Load the `Horse` image from the supplied tutorials or use the shadowed magnolia image created in the previous tutorial.
2. Turn Shadows on for the selected object(s) if necessary.
3. **ADJUST** click on the Shadow icon in the main toolbar. A Shadow mask is created and a copy of the object's Blend mask is automatically placed in it.
4. Click **MENU** over the Mask Edit window and choose Ghost or Anti-alias ➤ Smooth a couple of times. If desired, the shadow may be altered in a more localised way by using the Feather tool with Strength ➤ Deluge! or Darken.

The actual intensity of the shadow calculated at each point is quite complex and is based on the intensity of the Blend mask value (brighter values in the Blend mask produce less shadow), the intensity of the Shadow mask pixel and the intensity of the shadow colour.

Creating true 3D shadows

The Pseudo 3D shadow options often suffice for creating realistic looking shadow effects. If you have the time and the artistic ability it is possible, however, to recreate the exact shape of the shadow in the original image.

1. Load the Horse tutorial image
2. Select the Horse and turn on its shadow.

3. ADJUST click on the shadow icon in the main toolbar.
4. Click MENU over the Mask Edit window and choose Clear to ➤ Black with ADJUST to keep the menu open.
5. Choose the Mixed item in the menu to turn on mixing of the horse and its shadow mask.
6. Using the Line drawing tool, draw around the shadow shape on the grass behind the horse.
7. Ensure that the outline of the shadow is complete, with no missing pixels. The zoom buttons may help with this.
8. Choose the Fill tool (and set it to Plain if necessary) and click SELECT within the shadow outline you have created. If the fill Floods outside the shadow shape choose the Undo button. Find the pixel missing from the shadow outline, fill it with the Line drawing tool and try again.
9. Either use the menu item Anti-alias ➤ Smooth or the Feather tool to soften the edges of the shadow.

All around shadows

1. Load the Leafy1 tutorial image
2. Load sa24 from the standard Images directory supplied with your computer.
3. Select sa24 and click on the Mask icon to open the horizontal Mask toolbar. Select the Blend mask if necessary.
4. Drag sa24_msk from the Tutorials.std_msks directory into the horizontal Mask toolbar.
5. Open the Shadow dialogue box by clicking on the Shadow icon in the main toolbar, turn shadows on and reduce the x and y offsets to zero. Click OK in the Shadow dialogue box.
6. Click with ADJUST on the Shadow icon to open the Shadow mask for the image.
7. Click MENU over the Mask Edit window and choose Edge ➤ Maximum twice with ADJUST then choose Anti-alias ➤ Smooth or Ghost.

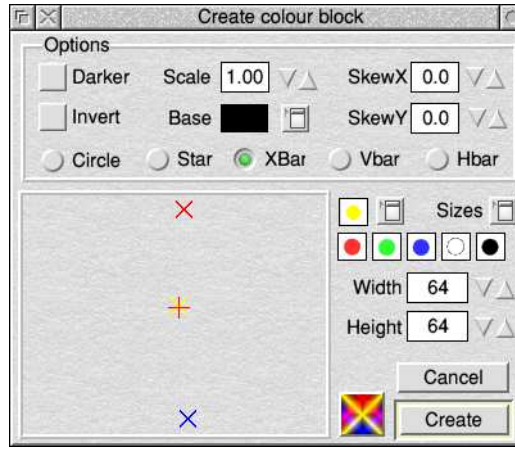
Colour blocks

This feature is one of the most flexible in the program. Here we give a few examples of what it can be used for.

Canvas Tiles

1. Open the Tile and Colour block dialogue boxes by choosing Canvas ä Tile... with ADJUST and then Misc ä Block... from the main canvas menu.
2. Change the Width and Height fields in the Block dialogue box to 64 pixels. Using smaller tiles slows down redraw of the canvas considerably.
3. Drag the default red colour point to the top of the block definition area and double click on it twice to turn it into a X-bar point.
4. Drag a blue point onto the definition area and place it at the bottom of the area.

5. Drag a yellow point to the middle of the area and double click on it until it turns into a Star shape. The Create colour block window should look like the illustration below at this point.



6. Choose Create with ADJUST.
7. Select the small colour block which appears in the middle of the canvas. (If it is not visible choose the toggle size icon in the canvas window with ADJUST)
8. Click MENU over the object and follow Edit ColBlock ➤ Export. Drag the file icon to the Tile window and choose the Use icon in the Tile window with ADJUST.
9. Click MENU over the tile window and choose Symmetry ä Vertical.

Colour blocks as canvas background

1. Open the Colour block window, click MENU over it and choose Reset with ADJUST and then Size ➤ Canvas from the menu.
2. Drag the default red circle to a point about 7/8 of the way up and 1/8 of the way in from the left of the definition area. Drag a blue colour point on and drop it at about 7/8 of the way across the area and 1/8 of the way up. Drag a white point to the centre of the area. A line at about 45° should connect all three points.
3. Choose Create with ADJUST to create the colour block.
4. Experiment with different colour points and shapes. The thumbnail will give you a good idea of how a block will look before it is created. Remember that each block you create will use memory. You may need to select and delete unwanted blocks if memory becomes short.

Mask creation techniques

There are several shortcuts and techniques for creating masks of irregular real world« Multi task

If the object to be masked has a fairly simple background colour (for example, a blue sky with a multi coloured object, such as a person, in the foreground) then it is often easier to select areas of the image you do not want and then choose the Invert item from the Mask Edit window menu.

Invert is also useful for fine tuning a mask when used in conjunction with Mixed viewing in the Mask Edit window menu. Turn Mixed on, choose Invert and then cut holes in the solid mask until all of the object you want to display is visible (using the Feather tool to soften the edges of the object). When complete, choose Invert once more.

Creating smooth fades of irregular objects.

A common requirement when creating a mask is for it to fade smoothly from an angle. The Angled fill tool in the Mask Edit window can fill areas of single colour within a mask but if anti-aliasing has been used an unsightly border of pixels is not filled.

To avoid this, use the Sharpen ➤ Cutoff dialogue box in the Mask Edit window menu to remove any anti-aliasing from the mask, then fill and anti-alias it.

Maths options

The Maths options in the Opacity window are often good for special effects. Here are a few examples, there are many possibilities.

Luminous 3D text

1. Starting with a blank canvas. Load the supplied file. Landscape.
2. Choose the Text button in the main toolbar.
3. In the Create text dialogue box change the font size to 140 point and the text colour to White if it is not set to white already. You can also change the default text if you wish but stick to roughly the same size word.
4. Choose the Create button and move the text to partly cover sky and land, about two thirds of the way up the landscape image.

Note: If the text is not anti-aliased select it, delete it, run the Tfonts utility and then recreate the text by pressing Ctrl-E with the pointer over the canvas.

5. Choose the Shadow icon and select the Active box. Increase the X and Y offsets to 10 pixels. Choose the Colour menu opener in the Shadow dialogue box and choose a slightly darker grey from the default set in the colour picker. Choose OK to close the dialogue box.
6. Choose the Shadow icon with ADJUST to open the Mask Edit window for the Shadow mask. Click MENU over the Mask Edit window and choose Edge ä Maximum twice and then Anti-alias ä Smudge twice. Close the Mask Edit window.
7. Choose the Opacity icon and click on the toggle window size gadget to reveal the Maths options. Drag the Opacity down to about 35%, Change the Maths type to Add and then choose OK.

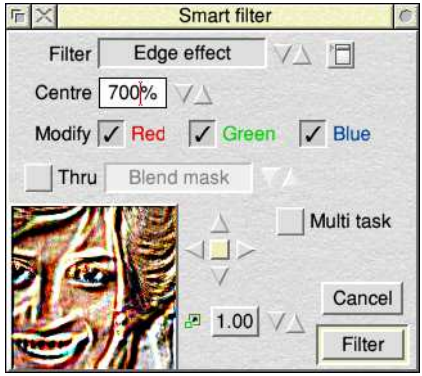
8. Experiment with different coloured text and varying the amount of blurring in the Shadow mask by using the Ghost and Anti-alias filters a different number of times.
9. Try changing the Maths type and varying the colour and intensity of the shadow colour.

The file 3DText (shown right) is an example of what can be produced.

Filters

Some examples:

1. Starting with a blank canvas, load the `Di` tutorial image. Select it and make a number of copies by pressing Ctrl-C a few times or choosing Select ä Copy
2. Select one of the images and open the filters dialogue box by choosing Edit 'Di' ➤ Filter... from the main menu.
3. Change the filter type to Edge effect by clicking on the bump arrows.
4. Increase the Centre value to 700%, either by entering a value into the writable field or clicking on me associated bump arrows with **Shift** or **Ctrl**.
5. Choose Filter with ADJUST so the Filter dialogue box stays open.
6. Select one of the other `Di` images, try changing the Centre value to 200% and turning off the Modify Red and Green buttons, before ADJUST clicking on Filter again.



Before

After

Defining filters

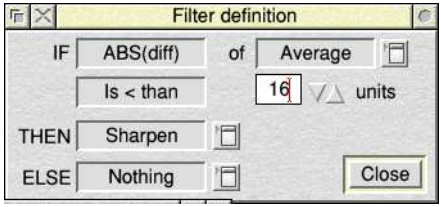
The Smart filters in Compo can be made to work differently on different parts of the image being processed. The Filter definition dialogue is used to construct a User filter using a BASIC like IF...THEN...ELSE construct. The program decides which of two Action filters to use. based upon the results of the Test filter.

If the result of the Test filter, as tested by the surrounding three fields, is positive, the THEN filter is the one executed for that pixel, otherwise the ELSE filter is executed. Anyone who understands the basics of programming will have no difficulty in working out what is going on.

A good example to look at is the Unsharp filter. Its brief is to execute a sharpening filter on the areas of the image that are blurry and to leave alone areas of the image that are already sharp. This helps to prevent unsightly artefacts left by the sharpen filters of other applications. Ironically, the filter decides which parts

of the image are blurry by blurring it some more internally. This method takes advantage of the fact that areas which are blurred do not change much when blurred further, while those areas of high contrast (the sharp bits) change considerably when subjected to a blurring filter. Try the following Unsharp filter.

1. Ensure Modify - Red, Green and Blue are all ticked
2. Change the filter type to User by clicking on the bump arrows.
3. Click on the menu icon to open the Filter definition window
4. Change the window to look like that below by clicking on the relevant fields then choose the Close button and click on Filter in the Smart filter dialogue box.



The above filter definition can be written as:

- Take an average of the pixel being changed and the pixels surrounding it.
- If the absolute value of the difference between the averaged result and the current value of the pixel is less than the threshold value (16 in this case) then the pixel must be part of a blurred area, so sharpen it.
- If the above is not true then leave it alone.

Put another way it can be read as:

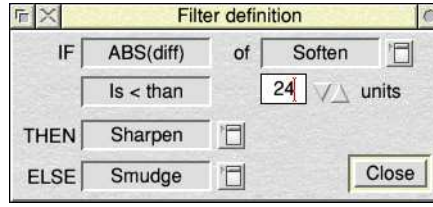
- IF the Absolute difference of the Averaged pixels is less than 16 units
- THEN apply the Sharpen filter
- ELSE do nothing

You might like to change the value of 16 to 8 and see what happens. Now only the most blurry areas are sharpened.

This is such a powerful feature that you should spend a little time acquainting yourself with the possibilities.

Try to work out what is happening in the next example.

1. Change the filter type to User by clicking on the bump arrows.
2. Click on the menu icon to open the Filter definition window.
3. Change the window to look like that below by clicking on the relevant fields then choose the Close button.



The filter you have created produces an impressionist painting effect, sharpening the blurry parts of the image and smudging those parts of the image that were sharp. Repeated several times on an image, this filter produces a quite realistic painting.

Effects of the filter described above

Displace mask

Melting moments

1. Load the tutorial image Magnolia and select it.
2. Open the horizontal Mask toolbar, if necessary, and select Displace mask.
3. Choose the Edit... button.
4. In the Mask Edit window toolbar select the Spray tool
5. Change the Mask paint colour to 140 units by clicking on the Current mask colour icon and dragging the slider in the Mask intensity dialogue box.
6. Paint randomly across the mask using the Spray tool.
7. Change the mask colour to about 100 units and repeat step 6.

8. Select the Feather tool and paint across the mask. The flower will quickly reform where you paint and then proceed to melt. The more often you go over an area the more it will melt. You may like to experiment using different feather strengths.

With practice, you will find you can melt any part of an image in any direction simply by the way in which you use the Feather tool.

Magic mirrors

1. Load the tutorial image **Di**, select it and choose the Displace mask on the horizontal Mask toolbar. Choose the Edit... button in the horizontal Mask toolbar
2. Choose the Fill tool with MENU and select the Circular fill type from the dialogue box.
3. Try circular fills of various sizes. Remember to use the Undo button in the Mask Edit window toolbar to undo a fill before trying the next one.50

6. More examples

This chapter contains additional examples for using Composition. In these examples it is assumed that you are familiar with both your computer and the Composition tools. The examples are written tersely so that you can follow them easily and complete them quickly.

Tiles 1

1. Open empty canvas
2. Drag the file Daisies to the Tile window
3. Choose Use in Tile window with ADJUST
4. Click MENU over Tile window
5. Choose Symmetry ➤ Horizontal
6. Choose Symmetry ➤ Vertical

Tiles 2

1. Open an empty canvas
2. Load Images.00-49.sa01 from your hard drive
3. Select the image
4. Click MENU over canvas
5. Choose Edit "sa01" ➤ Trim
6. Drag box to surround left eye of tiger
7. Choose Trim with ADJUST and fine tune trim if necessary
8. Press Ctrl-I
9. Click MENU over canvas and choose Canvas ➤ Tile
10. Drag over thumbnail image of the eye in the info window - a sprite icon appears
11. Drag the sprite icon to the Tile window
12. Choose Use in the Tile window
13. Try Symmetry > Horizontal in the Tile window menu.

Opacity & Maths options

1. Ensure that you are working in a 16 million colour mode
2. Open a new canvas
3. Click MENU over the canvas, select Canvas ➤ Colour and choose white
4. Load MDHB from the supplied images and move to the bottom of the canvas
5. Click on the Text icon in main toolbar
6. Create the word ‘Compo’ in red 140pt Homerton Bold text
7. Select the newly created text
8. Choose the Opacity icon and reduce opacity to 50%
9. Click on the Shadow icon and enable shadows. Set X & Y offsets to 8 and click on OK
10. Click on the Shadow icon with ADJUST to open a Shadow mask

11. Click MENU over Shadow mask and choose Anti-alias ➤ Smudge twice (use ADJUST the first time)
12. Click on the Shadow icon and turn on Pseudo 3D. Change the X Shear value to 0.50 and choose OK with ADJUST
13. Change the Y Shrink to 0.60 and choose OK
14. Select the MDHB sprite, choose the Opacity icon and set opacity to 50%
15. Return the opacity of MDHB to 100% after viewing the effect
16. Click on the Mask icon to open the horizontal Mask toolbar
17. Ensure the Blend mask is selected and click on Create
18. Choose Create within the Create Mask dialogue
19. Click on the Opacity icon and then on the Size icon in the top right of the box
20. Use the bump arrows to select different Maths options and use ADJUST to click on OK so that you can view the effect and immediately try another one. Some of the Maths options will have no effect on this image
21. When you get to the Palette item click on OK and then try dropping the supplied palette files onto the canvas.

Colour curve and curve mask

1. Open a new canvas
2. Load House, select it and choose the Colour curve icon in the main toolbar
3. Move both of the control points to the bottom right of the curve area and choose Set
4. Click on the Curve icon with ADJUST to open a Curve mask
5. Click MENU over Curve mask and choose Mixed
6. Choose the Eye icon in the Mask Edit window toolbar and click over the sky with the Wand in the Sprite View - you will find it easier if you set the Wand RGB ➤ Include Options to Within 64 pixels. You may need to click several times to select all of the sky
7. Tidy up the mask using the Painting and Feather tools in the Mask window
8. Choose the curve icon with SELECT, turn Combined and the Green and Blue buttons off. Drag the Red curve to the top left and choose Set

Maths options and the Blend mask

1. Open a new canvas
2. Load SwanCygnet
3. Select the image and press Ctrl-C to make a copy
4. Select background copy of the image
5. Choose the Opacity icon, click toggle size if necessary and set Maths options to Lin. Palette by clicking the down arrow twice from Blend. Click on OK
6. Select the front copy and move to exactly cover the back copy of image (use Info window for this - Ctrl-1)
7. Ensure the front copy is selected and click ADJUST on the Opacity icon to get a blank Blend mask

8. Click MENU over Blend mask and turn Mixed on
9. Paint across Blend mask - a colour version of the image will replace the grey scale version
10. Paint so that the area surrounding the swan and cygnets is filled - result in canvas is quite subtle
11. Choose Invert from Mask Edit window menu to produce a colour swan on a grey scale background
12. Try the Angled and Circular fills in the Blend mask to produce images that graduate from colour in one print to mono in others

Colour blocks / Text / Shadow mask

1. Open a new canvas
2. Click MENU on Misc ➤ Block
3. Choose the menu icon beside Sizes and choose Size ➤ Canvas from the menu
4. Drag the red point to 1/4 of the way across and 3/4 of the way up the definition area
5. Drag a yellow point to 3/4 of the way across and halfway up the definition area
6. Drag a blue point to halfway across and about 1/3 of the way up the definition area
7. Choose Create
8. Choose the Text icon in the main toolbar
9. Set font Size to 100 point and font to Trinity.Bold.italic
10. Turn In box on and reduce Opacity to 20%
11. Change the text to Composition and choose Create
12. Select text, select Shadow icon, make shadows Active and set X offset and Y Offset to 8 pixels. Click on OK
13. Select the Shadow icon with ADJUST to create a Shadow mask
14. Click MENU over the Fill tool and set to Plain fill. Ensure Mixed is not ticked
15. Click ADJUST over dark grey background around text and inside the ‘o’s and the ‘p’
16. Click MENU over Shadow mask. Choose Anti-alias ➤ Smudge ➤ All but black twice (both times with ADJUST)
17. Choose Edge ➤ Maximum twice and then close the Mask Edit window
18. Select the text
19. Choose the Layout icon in main toolbar
20. Turn Horizontal and Vertical on. select the central radio button and choose Move selection
21. Select background image, click on the Opacity icon and change the Maths type to Palette - one downward click from Blend should do it
22. Try dropping the different files in the Palettes directory onto the canvas

Stonehenge

1. Load Images.00-49.sa00 from your hard drive
2. Select the sprite and click on the Mask icon
3. Ensure that Blend mask is selected
4. Drop the file Images.50-99.sa51 from your hard drive onto the load target in the Mask toolbar

You can Try dropping the supplied mask, sa51_msk, onto the load target with **Shift** held down and opt to mix the masks

All screwed up

1. Load Images.00-49.sa05 from your hard drive
2. Select the sprite and click on the Mask icon
3. Ensure that Blend mask is selected
4. Drop the supplied mask file, Screw, onto the load target in the Mask toolbar

Try some other images such as sa07.

Torn paper 1

1. Load the supplied image Daisies and select it
2. Click on the Mask icon
3. Ensure Blend mask is selected
4. Drop the file TornPaper.Torn1 onto the mask load target
5. Click on Edit and experiment with the tools in the Mask Edit window

Torn paper 2

1. Load the supplied image Daisies and select it
2. Load the file Torn1 into the canvas - the file Torn1_msk is also loaded automatically
3. Select the Torn1 sprite and click on the Opacity icon
4. Use ADJUST on the slider to alter the opacity interactively

Torn paper 3

The torn paper sprite was created from the supplied Draw file. The sprite was then loaded into Compo where a mask was created. The sprite was then saved out as a JPEG file and the mask was also saved.

7. Introduction to CompoScript

CompoScript is a language for automating many tasks when using Compo. It is a scripting or macro language, defining a series of actions within Compo to be collected together into a single file. The file can then be activated either by clicking on an icon in a toolbar, or by dropping it onto a Compo window.

CompoScript can also be used to batch process files. It is possible to drop a directory onto a CompoScript program and have every file in that directory processed and resaved. If you commonly perform the same operation on a large number of files, CompoScript can save a very considerable amount of time.

You can write your CompoScript in whichever editor you normally use, either Zap or StrongED or even Edit will do.

Basics

Like most languages, CompoScript uses variables, loops and procedures. If you are familiar with a modern programming language, you will find the ways CompoScript handles these easy to learn.

Variables

There are two types of variable used by CompoScript. You can declare variables yourself. These are either integers or strings. There are also a number of built-in variables, which give information about objects and which are called *dotvars*. Dotvars are distinguished by the dot (full stop) at the beginning of the object's name:

```
Let fred = .canvaswidth / 4
```

Here, fred is a user-defined variable and .canvaswidth is the dotvar that contains the width of the current canvas.

Loops

There are a number of loops available. The For...Next and Repeat...Until loops are unsurprising:

```
let fred = 1
For i = 1 to 5
   fred = fred * i
next

repeat
   fred = fred + 1
until fred > 100000
```

Procedures

Procedures work on the currently selected object. They are defined in a similar manner to BASIC, but do not take parameters:

```
def procbloggs
let fred = fred + 1
endproc
```

and are called by

```
let fred = 1
procbloggs
```

Selected objects

Many of the CompoScript commands operate on the selected object. To select an object, you use the `select` command. Examples are:

```
select "fred"
select CANVAS
select FIRST
select ALL
select NONE
```

These select the object called `fred`, the canvas, the first object, all objects and no objects respectively.

One frequently used feature involves a variable, a loop and the `select` command:

```
for loop = 1 to 5
   select "fred<loop>"
next
```

This selects the objects `fred1`, `fred2`, `fred3`, `fred4` and `fred5`.

There are several dotvars associated with the selected object. To examine these, you use a very useful command:

```
report "hello world"
```

The `report` command displays its argument in a window.

```
select "fred"
report "object is <.fullname>"
report "height is <.height>"
report "Width is <.width>"
report "opacity is <.opacity>"
```

will display information about the object `fred`.

As will become apparent, you can do a great deal more with these variables than just display them.

Attributes

Some of the attributes of selected objects are more complex than height and width. A selected object may have a shadow for example. In this case, setting attributes can done as follows:

```
select "fred"
shadow ON
{colour:&a0a0a0
 XOffset:3
 YOffset:3
}
```

This sets various attributes of the shadow mask, and is really just a shortcut for writing:

```
.shadow.type = 0
.shadow.colour = &a0a0a0
.shadow.xoffset = 3
.shadow.yoffset = 3
```

Menus

CompoScript allows you to provide menus for the user to select from. This is done via the `definemenu` command. Here is a complete CompoScript example. Note that the first line of every CompoScript file is:

```
#CompoScript
```

Since CompoScript files are just ordinary text files, there needs to be a way of recognising a CompoScript file. The above line is how this is done. Here then is a complete CompoScript program:

```
#CompoScript
#  test menus
definemenu Shadows
{soft Superdooper-soft
 Rather hard really
}
openmenu Shadows to result
if result <> "" then
   report "I think the user selected item <result>"
else
   report "Oops! the user didn't select any item!"
endif
end
```

Let's look at this program line by line.

```
#CompoScript
```

This is the first line of every CompoScript file.

```
#  test menus
```

Lines beginning with # are comments.

```
definemenu Shadows
{soft Superdooper-soft
 Rather hard really
{
```

These lines define a menu. `Shadows` is the name of the menu.

```
openmenu Shadows to result
```

The `openmenu` command opens the specified menu. The item selected will be placed in the variable called `result`. When the user selects an item, you can examine the variable to see which item, if any, was selected.

```
if result <> "" then
   report "I think the user selected item <result>"
else
   report "Oops! the user didn't select any item!"
endif
```

Here, the selected item is just displayed, but you could take different actions for each option.

Note that variable names are enclosed in <> when they could be confused with a string.

```
end
```

This indicates the end of the program.

Similar to the `menu` command is the `prompt` command. This allows you to ask the user a question:

```
#CompoScript
prompt to result
{you really want to do this pal?
 Buttonl:Proceed
 Button2:Crazy!
 Button3:NoWay
}
report "Result is <result>"
end
```

CompoScript examples

This section introduces some additional features by means of examples. Many of these examples are provided with Compo, so you can try them out for yourself.

```
#CompoScript
```

The standard header line.

```
let myname = "Rob Davison"
```

creates a variable containing the name of the illustrious author of Compo.

```
maketext "Titletext"
{is <myname>
 Font:Homerton.Bold
 At: 128 128
 Size:48
 Angle:0
 Colour:&FF00
 Border:16x16
}
```

This creates a text object, specifying the font, position, size, rotation, colour and border.

```
select "Titletext"
```

Select the newly created object.

```
let .shadow = 1
```

Give the object a shadow.

```
makemask shadow copy blend
```

makes a mask. In this case a `shadow` mask, which initially is a copy of the `blend` mask.

```
processmask shadow Antialias Smudge AllButBlack
```

```
processmask shadow Edge Maximum
processmask shadow Antialias Smudge All
```

`Processmask` is a powerful command which allows you to perform actions on any of the masks. The parameters are the actions to take, which are those found in the mask editing section of Compo.

```
let .opacity = 127
```

This line sets the opacity of the selected object. The variable has a value between 0 and 256.

```
let .shadow.colour = &707070
let .shadow.xoffset = 4
let .shadow.yoffset = -4
```

These lines set attributes of the shadow.

```
end
```

The standard end marker.

A more complex example

Here is an example which generates an animation.

```
#CompoScript
```

The standard header line.

```
star "remove <.composcript$dir>.tempfile"
```

The star command passes its argument to RISC OS command line processor. In this case it removes a file called `tempfile` in the directory where the CompoScript file is located.

```
select "Text0"
```

Select the object called `Text0`.

```
makemask tint copy blend
```

Creates a tint mask, which is initially a copy of the blend mask.

```
for xpos = -40 to .width+40 step 10
```

This starts a loop based on the object's width. The loop steps up by 10 each time round.

```
select "Text0"
```

Selects the object.

```
PROC blended_animation
```

Calls a procedure.

```
next
```

ends the loop.

```
for xpos = .width+40 to -40 step -10
select "Text0"
PROC blended_animation
```

```
next
```

The same loop, but with `xpos` decreasing each time round:

```
star "wimptask intergif -loop -d 6 -best 255 -o
<.composcript$dir>.anim/gif <.composcript$dir>.tempfile"
```

This line (it should be all on one line) calls the !**InterGif** utility to generate an animation from the files saved by this script. See later for how the files are created.

```
report "All done"
```

Report success.

```
select "Text0"
makemask blend copy tint
```

Select the object and restore the mask to normal.

```
end
```

End of the main program.

```
defproc blended_animation
```

Start the definition of a procedure.

```
nib up
```

Stop the screen updating while the masking is in progress.

```
makemask blend copy tint
```

Reset the mask to normal.

```
processmask BLEND Gradfill
   {blended:yes Opacity:255
    Centre: <xpos>,<.height>/2
    Size:40
    Edgelntensity:0
    Centrelntensity:255
   }
```

Do a graduated fill in the mask. The fill is circular. The centre, size and opacity are specified. In this case, opacity has a value between 0 and 255. The effect of this is to make a small part of the text visible.

```
nib down
```

Allows the screen to be updated.

```
PROC export_snapshot
```

Calls a procedure.

```
endproc
```

Ends the procedure `blended_animation`.

```
defproc export_snapshot
# definition of procedure
   select CANVAS
   export sprite
```

```
{mask:1
BlendCanvas:1
Append:YES
 spritename:<loop>
}
```

Selects the whole canvas and exports the sprite as a file. The filename is specified. A mask is used. The sprite is appended to the existing file and the name of the sprite is the position in the loop. It is also possible to export as other types:

```
select first
```

Selects the object.

```
endproc
```

Ends the procedure.

Loading/Saving dialogues

For batch processing, you need to find the directory you want the script to process. You may also need to ask for a destination directory. This example does both.

```
#CompoScript
```

The standard header line

```
repeat
   dragload directory "Drag an input directory of images to
process." to indir
until indir <> ""
```

Put up a dialogue and wait for a directory to be dropped onto it. When it is, the variable `indir` contains its name:

```
repeat
   dragsave directory "Output" "Drag the output directory to
create." to outdir
until outdir <> ""
```

Put up a dialogue and wait for the user to drag the icon to disc. The default name in this case is `Output`. When the operation is complete, `outdir` contains the name of the directory.

```
fileinfo outdir
```

`fileinfo` gets information about a file or directory:

```
if .fileinfo.there = 0 then
   proc create_output_directory
endif
fileinfo outdir
if .fileinfo.there <> 2 then
   report "Output already exists (and is not a directory). Giving
up."
stop
endif
```

Check that it really has been created okay.

```
repeat
```

Starts a `repeat... until` loop.

```
scandirectory indir to found
```

`scandirectory` reads the next available filename from a directory; in this case, from the directory `indir`, and places the result in the directory `found`.

```
if found <> "" then
   proc process_image
endif
```

If the last file has been read, `found` will be "", otherwise it will contain a filename. If it is a file, call a procedure:

```
until found = ""
```

Continue the loop until all files have been processed.

```
end
```

End of the main program.

```
defproc process_image
```

Start of procedure definition.

```
if .objects < 0 then
   proc make_canvas
endif
```

`.objects` holds the number of object on a canvas. If it is less than zero, there is no canvas. If that's the case, call a procedure to create one.

```
select all
delete
```

Delete any objects on the canvas.

```
loadimage found
```

Load the image with the filename in the variable `found`.

```
select first
moveto centre centre
```

Select the object and centre it.

```
maketext "sometext"
{<.directoryscan.leafname> Font:Homerton.Bold
 Size:25
 At: 0 0
 Angle:0
 Colour:&FF0000
 Border:10x10
 Rubout:ON 30% &FF
}
```

Create a text object called `sometext`. The text is taken from the variable `.directoryscan.leafname` which holds the leafname of the last file read from a

directory.

```
select "sometext"
moveto centre 0
```

Select the text and centre it in the x direction.

```
proc save_in_output_directory
```

Call a procedure to save it.

```
endproc
```

End of the procedure.

```
defproc save_in_output_directory
   select canvas
   export jpeg
   {quality:75 }
endproc
```

A procedure to export the canvas as a JPEG at 75% quality.

```
defproc create_output_directory
   star "cdir <outdir>"
endproc
```

A procedure to create a directory.

```
defproc make_canvas
   makecanvas 768 576 0
endproc
```

Make a canvas.

8. CompoScript language reference

File format

A CompoScript file is a simple text file created using your favourite text editor. It *must* begin with the line (not case sensitive):

```
#CompoScript
```

Blank lines are ignored by the interpreter, as are lines beginning with the hash character (#) which may be used for comments. Commands, keywords, options and variable names are not case sensitive. Command parameters are separated by one or more spaces.

Commands

CompoScript provides the following sets of commands:

Assignment

`let`

Logic flow

`do`	`endproc`	`to`	`else`
`doto`	`case`	`step`	`endif`
`breakdo`	`when`	`next`	`repeat`
`enddo`	`otherwise`	`if`	`end`
`proc`	`endcase`	`then`	`stop`
`defproc`	`for`	`elseif`	

RISC OS interface

`star`	`input**`	`dragload**`	`fileinfo`
`definemenu`	`prompt**`	`dragsave**`	`colourpicker**`
`openmenu**`	`report`	`scandirectory**`	`slider**`

`**` indicates those commands with which the keyword `to` is used to indicate a return value This is distinct from its uses in the `for...next` construct and in the `=(interpolate)` function.

Image composition

`loadcompo`	**`processmask`**	`scriptoptions`	**`scaleto`**
`savecompo`	**`shadow`**	**`readpixel`**	**`scaleby`**
`select`	**`makecopy`**	`writepixel`	`trim`
`makecanvas`	**`redraw`**	`nib`	**`export`**
`makeaoi`	`rename`	`flipimage`	`loadimage`
`makemask`	`mouseover`	**`move`**	`ifexists`
`maketext`	`delete`	**`moveto`**	
`loadmask`	`include`	**`moveby`**	
`savemask`	`library`	**`rotateto`**	

Commands shown in **`bold underlined`** are those which affect only the currently selected object i.e. the *Current object*.

Parameters (if present) may be literals or variables and are separated by one or more spaces. You cannot have more than one command per line of code.

Command options

There are two command structures : single-line and multi-line. The latter have the general form :

```
<command>
{
   <option>: <value>
   <option>: <value> <value>
}
```

Options are not case sensitive: just make sure you spell them correctly and include the colon. Parameter values follow the colon.

Many options are switches or flags and take one of two values: `yes` | `no`, `1` | `0`, or `true` | `false` are all acceptable.

Spaces should be used only to separate parameters. Any parameter that is an expression (rather than a solitary variable or literal) should not contain any spaces within it. Parameters comprised of more than one word (space separated) must be enclosed in quotes, though this is an issue only with the **processmask** command and the built-in variable `.opacity.type`.

Some options take a pair of values representing a width and height. In such cases, the values can be separated with a comma, a space or the character 'x'.

A few accept a percentage value. The % symbol is optional though its presence aids clarity.

Note that this structure is distinct from multi-line program flow structures, such as:

```
repeat
   <commands>
until <condition>
```

where the `{` and `}` characters are not required.

Current object

Some of CompoScript's commands operate on only a single selected object (referred to as the **Current object**), whereas others operate on all selected objects at once.

Compo and CompoScript both use the concept of **Primary Selected object** (PSO) to refer to the one object within a multiple selection that will be affected by commands that operate on only a single object. The PSO is also referred to as the **Single object Focus**.

The Current object can refer to one of three things :

- A single selected object.
- The Primary Selected object (PSO) of a multiple selection.
- Each selected object, in turn, of a multiple selection when in a `doto` selection block of code.

The PSO is determined by how a selection is made or altered see the `select` command for details.

From Compo's user interface, using Ctrl-A to select all objects, or `select all` in

CompoScript, results in the foremost object being the PSO.

In the user interface, dragging the pointer to select a group of objects will result in the hindmost object being the PSO.

When using **ADJUST** to add to a selection then that most recently added object will become the PSO, whereas using `select + xxx` in CompoScript causes no change in the PSO.

When using **ADJUST** to remove an object from a selection, the hindmost object becomes the PSO, whereas when using `select - xxx` in CompoScript, this occurs only if it was the PSO that was removed, otherwise the PSO is left unchanged.

In CompoScript, the only command that operates on all objects within a selection is `delete`.

Conventions

`.dotvar`	is a built-in or dot variable
`<keyword>`	is to be replaced by a keyword e.g. an option or parameter value
`<text>`	is to be replaced by text
`<filename>`	is to be replaced by the full path of a file
`Keyword`	underlining indicates it is the default value for that option
`[...]`	brackets indicate an optional parameter

Constants

`TRUE`, `ON` and `1` all mean 'true' and
`FALSE`, `OFF`, `0` and, in some cases, `None` all mean 'false'

Variables

Three types of variables are available. Strings, integers and special cases of these called built-in variables or *dotvars* which are described below. Real variables (fractional numbers) are not possible, though there are a few instances where pseduo-reals can be used (see below).

CompoScript does not enforce any particular naming style on different types of variables. However in general you should stick to using upper and lower case letters and numbers. Those familiar with BASIC may like to include a terminal $ character in string variable names, though this is not necessary.

Variables are all global: there is no concept of local variables. Indeed, as they are actually RISC OS system variables, they can be accessed by external routines, run from within CompoScript, as well as persisting between scripts.

Variables are not case sensitive.

Assignment command

let

Declares and assigns a value to a variable, including any dotvar that is writable.

```
let <variable> <assignment operator> <expression>
```

that is, three space-separated elements follow the `let` command, where the spaces are necessary and where `<assignment operator>` is one of:

```
=     +=    -=    *=    /=
```

and `<expression>` can be a single variable, a literal value, or a compound expression. In the latter case spaces may be included to separate the various components. Examples:

```
let x = 12
let y = x +3
let z = 4 *5
```

For integers, the four standard arithmetic operators are available, for use within the `<expression>` element:

+	-	*	/
add	subtract	multiply	divide

along with their related assignment operators:

+=	-=	*=	/=
increment by	decrement by	multiply assign	divide assign

For string variables, there is just the one operator:

+
concatenate

along with its related assignment operator:

+=
append

Examples:

```
let x += 3
let y *= <multiple>
let .leafname += "/gif"
```

Hexadecimal numbers are denoted by prefixing them with an ampersand “&”.

Conditional operators

CompoScript allows only simple conditional tests, i.e. a single comparison using one of the following operators:

```
=     equal to
<>    not equal to
<     less than
>     greater than
<=    less than or equal to
>=    greater than or equal to
```

Compound conditions (using AND, OR, etc.) are not possible. As mentioned above, `false = 0`, and `true = 1`.

Pseudo-arrays

Although arrays are not possible in CompoScript, they can be approximated with this syntax:

```
let foobar<var> = <something>
for loop = 1 to 10
   let myvariable<loop> = a *loop +c
next
```

This assigns values to ten variables, named `myvariable1`, `myvariable2`, etc.

Interpolation

There is a special form of the `let` command that provides a way of obtaining non-integer step sizes within a `for...next` loop. This is useful for smoothly interpolating things like opacities over a fixed number of frames in an animation:

```
let <value> =(interpolate) <type> <start> to <end> <step> of <max>
```

where:

`<value>` (pseudo-real, var) is the interpolated result
`<type>` is one of : `Linear` | `Gamma_<value>` | `Bezier_<value`

`Flat` is a synonym for `Linear`.

`<start>`, `<end>` (integer) is the range from which to obtain the interpolated value.
`<step>`, `<max>` (integer) is the required step and the total number of steps.

This maps a position (`<step>`) into one range (`1` to `<max>`) against another range (`<start>` to `<end>`), using floating point to return the exact interval. The mapping can be linear, a gamma scale or even a Bézier-like S-curve.

Note that the `=(interpolate)` element is not optional, and should be written with the brackets. Example:

```
let frames = 30
for loop = 1 to frames
   # move 'opacity' from 0 and 255 in 'frames' equally sized steps
   let opacity =(interpolate) flat 0 to 255 loop of frames
   # or, a more interesting alternative,
   let opacity =(interpolate) gamma_1.6 0 to 255 loop of frames
next
```

Pseudo-reals

Some command parameters accept pseudo-reals and, although it is possible to specify these parameters using variables, you should note that these variables are in fact strings, and so arithmetic operations upon them are not possible:

```
let textsize = 12.5
```

```
maketext "line1"
{
 ...
 Size: <textsize>
 ...
}
```

is okay, but you would not be able to do (say, immediately before a `maketext` command) :

```
let textsize = textsize * 2
```

because `textsize` is a string variable. Note that there is a trap for the unwary here:

```
let textsize = textsize + 2
```

will work, though probably not as intended. Since `textsize` is a string variable, the string "2" would be appended, and textsize would become "12.52", i.e. still a valid pseudo-real quantity, though in this particular example there would be no apparent difference, as `maketext`'s Size: parameter is rounded to the nearest 1/16th of a point.

```
let textsize = textsize + 2.5
```

would produce nonsense, as `textsize` would then end up as string "12.52.5".

Commands that make use of pseudo-reals are : `export` (various parameters), `maketext` (`Size:`), scaleby and `shadow` (`3D:`).

Literal strings

Literal strings do not need to be enclosed in quotes, even if they contain spaces, though it is strongly recommended that you do quote such strings, both for clarity and to ensure that CompoScript interprets them as you intend. Consider the following command :

```
let foobar = foo
```

The content (and type) of `foobar` will depend on whether `foo` is defined or not. If it's not, you'll get a string variable with the characters "foo". If it's defined, e.g. if the above line was preceded by:

```
let foo = 12
```

then `foobar` will become an integer variable with value 12.

Enclosing something in angle brackets <,> forces CompoScript to try to turn it into a variable. If it fails you'll get a string variable containing the whole lot. Enclosing something in quotes forces it to treat it all as a string unless it is angle bracketed.

Using string variables takes a little getting used to but are very neat once you're familiar with them. Example:

```
let first = "Rob "
let last = "Davison"
let fullname$ = "<first><last> wrote this."
```

`fullname$` is now the string "Rob Davison wrote this."

Variable types

CompoScript is a loosely typed language. That is, there is no hard distinction between integers and strings, or even commands.

Any string variable that contains only digits can be used in arithmetic expressions. Conversely, integer variables can be directly expressed in a string: there is no need for BASIC's VAL() and STR$() functions. Examples:

```
let stringnumber = "21"
let answer = stringnumber * 2
report "Life, etc equals <answer>"


let foo = 1
let foobar = foo
let foobar = <foo>
let foobar = "<foo>"
let foobar = "foo"
```

In the first of the above cases, `foobar` is the number 1. Only in the last case is it the string "foo".

Built-in variables (dotvars)

n addition to user defined variables, CompoScript makes available a number of internal variables. These are distinguished by being prefixed with a full stop, and so are referred to as *dotvars*.

Global dotvars

dotvar	Description	Read / Write status
`.compo$version`	Returns the Compo version number as a string, e.g. "1.23c". Returns a null string for Compo 1.21e, or earlier	Read only
`.objects`	Total number of objects on the canvas	Read only
`.selected`	Primary selected object (1 is oldest), or -1 if none. Only use this to test that there is a selection: its value (when not = -1) is not of any real use. Use `.stackdepth` if you want to know its position within Compo's image stack	Read only
`.canvaswidth`	The canvas width (pixels)	Read only

.canvasheight	The canvas height (pixels)	Read only
.canvascolour	The canvas background colour (decimal value of &rrggbb)	Read / Write
.canvas.tile	A null string means that .canvascolour is being used. Otherwise a string var of the form "tile4", being the internal name of the tile sprite	Read only
.composcript$dir	The path where the current script resides	Read only

Selected object: General dotvars

dotvar	Description	Read / Write status
.fullname	Full pathname.leafname of the object's file	Read / Write
.gamma	The object's global gamma * 100 (150 = a gamma of 1.5) Altering this will affect the following three variables equally	Read / Write
.gamma.red	The object's red gamma * 100. Alter to adjust the red component	Read / Write
.gamma.green	The object's green gamma * 100. Ditto the green component	Read / Write
.gamma.blue	The object's blue gamma * 100. Ditto the blue component	Read / Write
.height	The object's height (pixels)	Read only
.width	The object's width (pixels)	Read only
.istext	0 = not a text object 1 = is a text object	Read only
.leafname	Leafname of the object's file	Read / Write

.mouseover	The name of the object clicked on. See also command mouseover	Read only

```
#CompoScript
# Demonstrates the use of the 'mouseover' command.
# Remember, in CompoScript, '0' is 'false' and '1' is 'true'

makecanvas 640 480 0
let the_cows_come_home = 0
maketext "Daisy"
{
 Text: "Moo!"
 Font: Homerton.Bold
 Colour: &00cc00
 At: 100 120
}
mouseover "Daisy" on click proc something
repeat
   # an empty loop which will continue until the
   # mouseover has executed
until the_cows_come_home
end

defproc something
 select <.mouseover>
 report "The user clicked on <.leafname>. "
 report "It is at <.xpos> <.ypos>"
 let the_cows_come_home = 1
endproc#CompoScript
# Demonstrates the assignment of '.opacity.type' using the textual
# name rather than a numeric value. The textual values are defined
# in the '!Compo.Resources.Messages' file, using the tags
# 'math0' to 'math15'.

select first
let .opacity.type = "SoftLight"

# Is equivalent to:

select first
let .opacity.type = 10
```

dotvar	Description	Read / Write status
.selected.imagebase	Base of image data. Used when an external program processes Compo's image data. See star command for an example	Read only

.stackdepth	Position in the stack of overlapping objects (1 is the hindmost)	Read only
.xpos	The position of the object's left edge (pixels). 0 is left of canvas	Read / Write
.ypos	The position of the object's bottom edge (pixels). 0 is bottom of canvas	Read / Write

Selected object: Animation dotvars

The following .animation.xxx variables are available if an animation is selected.

dotvar	Description	Read / Write status
.animation.frames	Number of frames in the animation. =0 if the object is not an animation	Read only
.animation.currentframe	The current frame. Changing this for an animated object lets you combine and manipulate GIF animations (e.g. adding text to a frame, or varying its opacity)	Read / Write

Selected object: Mask dotvars

The following .mask.xxx variables, when non-zero, are an offset from .selected.imagebase to the start of the mask object data structure. For Data masks, which refer to a vector layer (a Draw or Artworks file), the format is non-standard and is best not played about with. For Blend masks etc. it points to a sprite area containing a single greyscale sprite (which need not necessarily be the same size as the image).

dotvar	Description	Read / Write status
.mask.blend	0 = No blend mask. Non-zero = offset to the object's blend mask	Read only
.mask.curve	0 = No curve mask. Non-zero = offset to the object's curve mask	Read only

.mask.data	0 = No data mask (the object is a simple bitmap. Non-zero = offset to the object's data mask (the object has a vector layer)	Read only
.mask.displace	0 = No displace mask. Non-zero = offset to the object's displace mask	Read only
.mask.shadow	0 = No shadow mask. Non-zero = offset to the object's shadow mask	Read only
.mask.texture	0 = No texture mask. Non-zero = offset to the object's texture mask	Read only
.mask.tint	0 = No tint mask. Non-zero = offset to the object's tint mask	Read only

Selected object: Shadow dotvars

dotvar	Description	Read / Write status
.shadow	Whether there is a shadow: 0=No, 1=Yes, 257=Yes with its opacity linked to the object's global opacity	Read / Write
.shadow.colour	The colour of the shadow (&rrggbb)	Read / Write
.shadow.opacity	The opacity of the shadow (0–255)	Read / Write
.shadow.type	0=Normal 1=Subtractive 2=Blend	Read / Write
.shadow.xoffset	The horizontal displacement of the shadow (pixels)	Read / Write
.shadow.yoffset	The vertical displacement of the shadow (pixels)	Read / Write

Selected object: Text dotvars

The following `.text.xxx` variables are set if the selected object is a text object (i.e.if `.istext` = 1). See also the `maketext` command.

dotvar	Description	Read / Write status
`.text.angle`	Amount of rotation (degrees). Negative is clockwise	Read only
`.text.aspect`	The text's aspect ratio (%)	Read only
`.text.baseline`	The text's baseline as an offset from the object's bottom edge (pixels)	Read only
`.text.colour`	Colour of the text (`&rrggbb`)	Read only
`.text.font`	Name of the font used	Read only
`.text.size`	Height of the text (points). Note, internally CompoScript works to a resolution of sixteenths of a point, though this variable is rounded down to the nearest integer value. See example	Read only
`.text.text`	Textual content	Read only
`.text.texture`	Full pathname of the texture file, or a null string if the text is not textured. See example	Read only
`.text.xborder`	Left and right border (pixels)	Read only
`.text.xshear`	Amount of horizontal shear (%)	Read only
`.text.xtracking`	Amount of horizontal tracking	Read only
`.text.yborder`	Top and bottom border (pixels)	Read only
`.text.yshear`	Amount of vertical shear (%)	Read only
`.text.ytracking`	Amount of vertical tracking	Read only

Selected object: Texture dotvars

dotvar	Description	Read / Write status
`.texture.angle`	Angle of rotation (0–360 degrees anticlockwise)	Read / Write
`.texture.opacity`	Transparency value (0–255)	Read / Write
`.texture.type`	0=Normal 1=Inverse 2=ThruBlend 3=InvThru 4=Cutoff 5=InvCutoff 6=ToColour 7=3D 8=3DBlended 9=3DInvBlend 10=Alt3D 11=Alt3DBlend	Read / Write

Selected object: Tint dotvars

dotvar	Description	Read / Write status
`.tint.colour`	The colour of an object's tint (`&rrggbb`)	Read / Write
`.tint.opacity`	Transparency level (0–255)	Read / Write
`.tint.tile`	0=Is not tiled 1=Is tiled	Read / Write
`.tint.type`	0=Add 1=Subtract 2=Multiply 3=InvMultiply 4=Blend 5=Dodge 6=Burn 7=Screen 8=Screen2 9=Screen3	Read / Write

On return from command dragload

dotvar	Description	Read / Write status
`.dragload.leafname`	The object's leafname	Read only
`.dragload.filetype`	The object's filetype (numeric) or &1000 for dirs &2000 for apps	Read only
`.dragload.length`	The file's length (bytes). Not relevant for directories	Read only
`.dragload.there`	0=Not found 1=File 2=Directory 255=Invalid	Read only

On return from command dragsave

dotvar	Description	Read / Write status
`.dragsave.leafname`	The object's leafname	Read only
`.dragsave.filetype`	The object's filetype (numeric) or &1000 for dirs &2000 for apps	Read only
`.dragsave.length`	The file's length (bytes). Not relevant for directories	Read only
`.dragsave.there`	0=Not found 1=File 2=Directory 255=Invalid	Read only

On return from command fileinfo

dotvar	Description	Read / Write status
`.fileinfo.filetype`	The object's filetype (numeric)	Read only
`.fileinfo.length`	The object's length (bytes)	Read only
`.fileinfo.there`	0=Not found 1=File 2=Directory 255=Invalid	Read only

On return from command scandirectory

dotvar	Description	Read / Write status
`.directoryscan.leafname`	The object's leafname	Read only
`.directoryscan.filetype`	The object's filetype (numeric) or &1000 for directories &2000 for applications	Read only
`.directoryscan.length`	The file's length (bytes). Not relevant for directories	Read only
`.directoryscan.there`	0=Not found 1=File 2=Directory 255=Invalid	Read only

On return from command readpixel

dotvar	Description	Read / Write status
`.r`	The red component (0–255)	Read / Write
`.g`	The green component (0–255)	Read / Write
`.b`	The blue component (0–255)	Read / Write
`.p`	The pixel value (Acorn 32-bit sprite format, `&xxbbggrr`) where xx is the alpha channel data. Readable/writable via `processmask <mask> readalpha | writealpha`	Read / Write

On return from command trim

dotvar	Description	Read / Write status
`.trimx0`	The number of pixels removed from the left edge	Read only
`.trimx1`	The number of pixels removed from the right edge	Read only
`.trimy0`	The number of pixels removed from the bottom edge	Read only
`.trimy1`	The number of pixels removed from the top edge	Read only

do breakdo enddo

Repeat a series of instructions a number of times. There is a nesting limit of 30.

```
# do example 1
let loops = 5
do loops
   ...
enddo

# do example 2
do 100
   ...
   if a = <premature_exit> then
      breakdo
   endif
   ...
enddo
```

proc defproc endproc

`defproc` and `endproc` are used to define a procedure which is a self contained section of code that can be called by name. `proc` is used to call a procedure. There is a nesting limit of 30.

Procedures in CompoScript neither take parameters nor return results. There is no concept of local variables. Note also that `defproc` is one word, and that there should be a space between the `proc` or `defproc` commands and the procedure's name.

You must use the `end` command to separate the main section of code from any procedure definitions that follow.

```
# proc example 1
proc caption_object
...
end

# proc example 2
defproc caption_object
   ...
endproc

# proc example 3
let fred = 2
case fred of
   when 1 proc fred_one
   when 2 proc fred_two
   otherwise proc fred_common
endcase
end
```

```
defproc fred_one
   let jim$ = "fred is one"
endproc

defproc fred_two
   let jim$ = "fred is two"
endproc

defproc fred_common
   let jim$ = "fred is <fred>"
endproc
```

---

case when otherwise endcase

Provide a case structure which cannot, however, be nested and only one command per `when` is allowed (which can be a procedure).

---

for variable = <counter> to <startval> step <endval> next

`For...to...step...next` loop. There is a nesting limit of 30.
`<counter>` is an integer or variable, counting cycles around the loop.

`<startval>` and `<endval>` are integers, namely the initial and final values of the counter.

`<inc>` is an integer, namely the step value, i.e. the increment that is applied to the loop counter on each cycle. The step value can be positive or negative.

```
for i = 60 to 0 step -2
   ...
next
```

---

if...then elseif...then else endif

Multi-line enhanced `if...then` construct. Must be multi-line: single line form is not supported. Immediate nesting (within the same procedure) is not yet supported (it's there, but not reliable).

```
let fred = 12
if fred = 2 then
   proc fred_two
elseif fred = 1 then
   proc fred_one
else
   proc fred_common
endif
```

The multi-line `if..elseif..` structure is similar to the case structure, and indeed the previous example could be coded using such a structure. The difference between the two is that a case structure tests for different values of a single variable, whereas each `if` and `elseif` can test a different variable. To a limited extent this can be used to compensate for the lack of logical comparisons (AND, OR, etc.).

```
if fred = 1 then
```

```
   # here fred is 1
elseif jim = 2 then
   # here fred<>1 AND jim=2
elseif sheila = 3 then
   # here fred<>1 AND jim<>2 AND sheila=3
else
   # here none of the above were true
endif
```

---

repeat...until

`Repeat...until` structure.There is a nesting limit of 30.

```
repeat
...
until x = 5
```

---

end

This ends the program. It need not be physically at the end of the code. If the code contains any procedure definitions, then this command must occur after the main code and before the procedure definitions.

---

stop

Halt script execution prematurely. Unlike the `end` command, `stop` may be used inside `if..then` constructs without generating an error. Example:

```
if .blendmask = 0 then
   # object has no blend mask
   stop
endif
```

RISC OS interface commands

---

star

Passes a command to the standard RISC OS command line interpreter (OSCLI):

```
star "Filer_OpenDir <fred>"
```

`star` can be used to run external programs to process an image held in Compo or to input data into the selected image.

```
#CompoScript
select first
star "WimpTask <.composcript$dir>.myprogram <.selected.imagebase>"
redraw
```

This script selects the first image on the canvas and then runs a program (assumed to be in the same directory as the script), with the base address of the image data passed on the command line.

Note the use of the dotvar `.composcript$dir` to obtain the path of the current script, and `.selected.imagebase` to pass the base address of the currently selected image data.

Also of use are the `.mask.xxx` variables which, when non-zero, contain offsets from `.selected.imagebase` to the various mask data structures.

Great care should be taken in the writing and use of such programs lest they overwrite something they shouldn't.

It is not possible to call other CompoScript files from within a CompoScript.

---

definemenu

Defines a RISC OS menu.Only one menu structure is held: you can't define more than one menu concurrently and then choose which one to open. This shouldn't be a restriction, as RISC OS can only have one menu open at a time; you simply define a menu immediately before you want to open it. Syntax:

```
definemenu <title>
{
 list of items
}
```

Where `<title>` (string) is the text to be used as the menu's title, and the identifier to be used to refer to this menu, when you later wish to display it, using the **openmenu** command.

---

openmenu

Popup the named menu, that was previously created using definemenu. Syntax:

```
openmenu <title> to <result>
```

Where `<title>` (string) is the menu identifier, as specified using **definemenu**.

`<result>` (string var, return) contains the text of the menu item that the user selected.

Although there can only be one menu definition at any one time, the title needs to be specified both for clarity and also to allow for the special case of a font menu:

```
openmenu "Fonts" to result Default "Trinity.Medium"
```

This also illustrates the use of the default option to pre-tick a menu entry.

---

input

This provides a means of getting keyboard input from the user. `Syntax:`

```
input <default> to <result> <prompt>
```

Where `<default>` (string) is the default text that will be returned by just pressing **Return**. Use "" for no default text.
`<result>` (string var, return) returns the user's input.
`<prompt>` (string) is the message prompting the user.

```
input "Purple" to colour "Enter your favourite colour"
```

---

prompt

Pop up a dialog box containing three buttons and some prompt text. As well as the prompt text, the text for each button needs to be specified. Example:

```
prompt to result
{
 Message:Do you really want to do this pal?
 Button1:Proceed
 Button2:Crazy!
 Button3:NoWay
}
```

After this call, `result` holds the text of the button pressed.

---

report

Report a message in a popup window. An optional second parameter specifies a delay time before automatically closing the window. Syntax:

```
report <message> (<timeout>)
```

Where:

`<message>` (string) is the message to be reported.
`<timeout>` (integer) is a time delay in centiseconds before closing the window. If omitted, a default value of 500 (i.e. 5 seconds) is used.

```
if .selected = -1 then
   report "There is no selected object"
   stop
endif
```

---

dragload

This command opens a window which asks for a file or directory to be dropped onto it.

```
dragload <fileicon> <message> to <filename>
```

`<fileicon>` (string) is the name of the icon to represent the type of file wanted.
`<message>` (string) is the message to be displayed.
`<filename>` (string var, return) is the full pathname of the file or directory the user dropped onto this window. It is "" if no file is dropped.

Here as elsewhere, the `to` keyword refers to a return variable, not a destination.

Note that specifying the sprite for the filetype you want doesn't mean that only that filetype is accepted. You must therefore check that the expected type has been dropped.

Following a `dragload`, several dotvars are set. In particular, you should use `.dragload.there` to establish that it is in fact a file (or a directory), and before using

any of the .dragload.leafname, .dragload.filetype or .dragload.length variables. Also, .dragload.length doesn't make sense in the case of directories.

```
dragload file_aff "Drag a Draw file here" to pathname$
dragload directory "Drop a directory here" to pathname$
```

---

dragsave

This command is used to display a SaveAs box with a user supplied message. No data is actually saved by this command, instead it allows the user to choose where something is to be saved.

```
dragsave <fileicon> <leafname> <message> to <filename>
```

<fileicon> (string) is the name of the icon to be used in the window.
<leafname> (string) is the default leafname put into the SaveAs window.
<message> (string) is the message to be displayed.
<filename> (string var, return) is the full pathname after the user has dragged the icon to a directory, or a null string if the window was closed without performing a drag operation.

Following a dragsave command, several dotvars are set. You can use .dragsave.there to establish whether a file or directory with the proposed name already exists, and certainly before using any of the .dragsave.leafname, .dragsave.filetype or .dragsave.length variables, which all refer to this pre-existing object. Thus they can be used to avoid overwriting something that already exists.

```
dragsave file_ff9 abc "Drag the icon to save" to where
```

---

scandirectory

Returns the names of files in a specified directory.

```
scandirectory <dirname> to <filename>
```

<dirname> (string) is the pathname of the directory to be scanned.
<filename> (string var, return) is the full pathname, or null ("") if no more.

It should be used in a loop, to read the names one at a time. It is not possible to scan more than one directory concurrently.

Following a scandirectory command, several dotvars are set. In particular, you should use .directoryscan.there to establish whether an object is a file or a directory, before using any of the .directoryscan.leafname, .directoryscan.filetype or .directoryscan.length variables. Also, .directoryscan.length doesn't make sense in the case of directories.

```
repeat
   scandirectory "ADFS::4.$.Images" to result
   if result <> "" then
      report "found file <result>"
   endif
until result = ""
```

fileinfo

```
This provides information about a file.
```

fileinfo `<filename>`

`<filename>` (string) is the full pathname of the file.

Following a fileinfo command, several dotvars are set. In particular, you should use `.fileinfo.there` to establish that it is in fact a file before using either of the `.fileinfo.filetype` or `.fileinfo.length` variables. Unlike the commands that have similar dotvars (**dragload, dragsave** and **scandirectory**) if the object is a directory, then `.fileinfo.filetype` will be invalid and so cannot be used to differentiate between normal and application directories

colourpicker

Opens a colour picker window, allowing the user to choose a colour. A colour is coded as hexadecimal `&rrggbb` for its red, green and blue values.

colourpicker `<type> <title> <default>` **to** `<colour>`

`<type>` (string) is either ‘Acorn’ or ‘Compo’.
`<title>` (string) is the popup window’s title.
`<default>` (integer) is the initial colour shown when the window is opened (`&rrggbb`).
`<colour>` (integer var, return) is the colour value chosen (`&rrggbb`).

slider

Opens a window containing a draggable slider, allowing the user to enter a numeric value in the range 0–255.

```
slider <title> <default> to <value>
```

`<title>` (string) is the popup window’s title.
`<default>` (integer) is the initial value shown when the window is opened (0–255).
`<value>` (integer var, return) is the value shown when the window was closed.

```
slider "Intensity:" 128 to result
```

Image composition commands

Select

Compo specific commands mostly reference a selected object and the `select` command is used to select objects.

```
select [+|-] <object>
```

Option	Data type	Description and values
`<object>`	keyword	`Canvas | None | All | First | Last | Previous | Next | "Name" | <integer>`

`select + <object>` adds to the current selection; the single object focus remains unchanged.

`select - <object>` removes it from the current selection. If the removed object had the single object focus, focus is transferred to the hindmost object in the remainder of the selection).

`select Canvas`
Selects the canvas, deselecting all other objects.

`select None`
Is a synonym for the above command. It deselects all selected objects, and the selection focus (for exporting) is moved to the canvas.

`select All`
Selects all of the objects on the canvas, the single object focus is the foremost object. Most commonly used before a delete command.

`select First`
Selects the first (hindmost) object. If there are no objects in the composition then the canvas is selected (and no error returned).

`select Last`
Selects the foremost object. If there are no objects in the composition then the canvas is selected (and no error returned).

`select Previous`
Selects the previous object (moving towards the back of the stack). If the first object in the composition is already selected this command does nothing.

select Next

Selects the next object (moving towards the front of the stack). If the last object in the composition is already selected this command does nothing.

`select "Name"`
Selects an object by its leafname. The search is case sensitive and if more than one object with the same leafname exist this command selects only the first (hindmost) one. If the referenced object cannot be found an error is returned and the CompoScript halted.

`select <integer>`
Selects an object by its depth in the stack, 1 being the hindmost object.

Any vector layer objects (and that includes AOIs) will always be in above any bitmap objects in the stack. Thus using `select Last` to select a just created bitmap object will work as intended only if there are no vector layer objects or AOIs on the canvas. If there are any objects on the vector layer then `select Last` will always select the topmost of those.

```
# Example:
# Move through all the objects, appending "/gif" to their names

select first
let lastselect = 0
repeat
   let lastselect = <.selected>
   let .leafname += "/gif"
   select next
until .selected = <lastselect>
end
```

or, more succinctly:

```
select all
doto selection
{
   let .leafname += "/gif"
}
end
```

In addition to `.selected`, there are several dotvars relevant to the currently selected object.

---

makecanvas

```
makecanvas <xsize> <ysize> (<colour>)
```

Creates a Compo canvas. It takes two compulsory and one optional parameters.

`<xsize>`, `<ysize>` (integers) are the width and height of the canvas (pixels).
`<colour>` (integer) is the colour (&rrggbb).

If a canvas already exists, it will be resized and coloured as specified. Existing objects will be left intact.

```
# Example
makecanvas 640 480 0
```

creates a black canvas of 640 × 480 pixels.

---

makeaoi

```
makeaoi <xlow> <ylow> <width> <height> [<name>]
```

Creates an Area Of Interest (AOI) on the canvas. This has no effect on any underlying image on the canvas. Once created, an AOI may be selected, moved, resized, exported, just like any other object.

`<xlow>`, `<ylow>` (integers) are the left x and bottom y canvas coordinates of the

AOI.

<width>, <height> (integers) are the width and height of the AOI (pixels).
<name> (string) is the name for the AOI. If no name is provided one is automatically generated.

Masks (shadows, tints, etc.) for AOIs are undefined, so avoid using mask-related commands with AOIs.

---

maketext

```
maketext [<object>]
```

Creates a text object or modifies the current object. Which of these two actions is performed is determined by whether the object name is specified:

```
maketext
{
 options
}
```

modifies an existing (the currently selected) object, whereas:

```
maketext <title>
{
 options
}
```

creates a new text object with the name specified in <title>.

If a *sprite* is selected then the *text through image* effect is applied to generate a blend mask from the given text. Any options that are omitted will be given default values.

If a *piece of text* is selected then the options supplied replace those settings for the piece of text, and the object is updated. This lets you easily change the font, size, colour, angle etc. of a bit of text without having to delete and recreate it (and then put it back at the right depth in the stack.

The text attribute dotvars are useful in this situation, particularly if you want to make a relative change (e.g. doubling the height of a piece of text). Any options that are omitted will be left unmodified. The options are:

Option	Data type	Description and values
Text:	string	New text object's name, or null to modify existing text object.
Font: <fontname>	string	Name of the font to be used.

`Size: <w> <h>`	integer or pseudo-real	Width and height of the text (pts). If only one parameter is given it specifird both width and height (i.e. aspect ratio is 100%). When masking a sprite the text size field has been enhanced to support **fitting** whereby either or both width and height may be replaced with the keyword `FIT` which will set the font size to fit into the sprite (taking account of any borders you specify).
`Size: FIT`	string	Fit as well as possible, while keeping the aspect ratio to 100%.
`Size: FIT,FIT`	string	Fit exactly, allowing non-100% text aspect ratio.
`At: <x> <y>`	integer	Position on the canvas at which to place the text object (bottom left corner)
`Angle: <deg>`	integer	Amount of rotation (degrees). Negative is clockwise.
`KeepCentre: YES|NO or ON|OFF`	string	Is only of use when modifying a text object such that the width changes. When ON the text will remain centred about the same point. When OFF the left edge will remain fixed.
`Colour: <textcolour> | <backcolour>`	integer `&rrggbb`	24-bit colour values (`&rrggbb`)
`Texture: <filename> | OFF`	string	Specifies the source of a texture, or `OFF` if, when modiying text, you wish to explicitly remove the texture. The default is `<textcolur>`.
`Border: <bx> <by>`	integer	Amount of additional surrounding space (pixels).
`Opacity: <x> <y>`	integer	Amount of transparency (0–100%). The % symbol is required. This parameter can be omitted in which case a value of 100% will be assumed.
`Tracking: <tx> <ty>`	integer	Amount of x and y tracking.
`Shear: <sx> <sy>`	integer	Amount of x and y shear (%).
`Rubout: (<opacity>%) <backcolour> | OFF`	integer	Amount of transparency (0–100%). The % symbol is required. If omitted in a default of 100% is assumed. 24-bit colour values. Or `OFF` to disable `Rubout`.

Texture:OFF and Rubout:OFF are only needed when modifying text and you wish to explicitly remove either option. There is no ON associated with these options - just specifying them will do that. If Texture: is not specified (or is set OFF, or the texture file cannot be found), then the <textcolour> will be used. Note that OFF, 0 and

FALSE are all synonymous.

Several dotvars are set if the selected object is a text object (i.e. if `.istext = 1`), but they are all read-only. It is possible however to pass them in as part of a modifying `maketext` command, as in this example script which doubles the size of all text objects on a canvas:

```
let yellow = &FFFF00
maketext "Titletext"
{
 Text: "Squat red text"
 Font: Homerton.Bold
 Size: 50x10
 At: 64 64
 Colour: &ff0000
 Border: 16,16
 Rubout: 50% <yellow>
}

maketext "linel"
{
 Text: "12.5 pt text"
 Font: Homerton.Medium
 Size: 12.5
 Colour: &aa00aa
 At: 50 50
}

maketext "banner"
{
 Text: "Fluffy Clouds Inc"
 Font: Homerton.Bold
 Size: 60 40
 At: 50 50
 Texture: "<Compo$Resources>.textures.n056/jpg"
}

select "sometext"
# Force Rubout OFF for this piece of text
maketext
{
 Rubout: off
}
```

Compo is supplied with a directory for texture tiles in `<Compo$Resources>`. These may be added to with any tiling images for use with the `Texture:` option.

---

makemask

Creates a mask for the current object, either by copying an existing mask (belonging to the same object) or by creating a blank white or black mask:

```
makemask <newmask> copy <mask>
```

copies a mask giving it a new name.

```
makemask <newmask> <colour>
```

creates a mask.

Option	Data type	Description and values
`<mask>` `<newmask>`	keyword	`Blend | Tint | Curve | Shadow | Displace | Texture`
`<colour>`	keyword	`White | Black`

Examples:

```
makemask Tint copy Blend
makemask Blend white
makemask Shadow black
```

---

loadmask

Loads an image file into the specified mask for the current object:

```
loadmask <mask> <filename>
```

An extended form allows the image to be scaled and/or mixed during the load operation:

```
loadmask <mask> <filename>
{
 Mix: <mix>
 Scale: <scale>
}
```

Option	Data type	Description and values
`<mask>`	keyword	`Blend | Tint | Curve | Shadow | Displace | Texture`
`<mix>`	keyword	`No | Yes | Invert` `True` and `1 = Yes, Inverse` and `2 = Invert`

`<scale>`	keyword	`No | Yes | Aspect`
		`Fit, ToFit, Exact,` and `1 = Yes`
		`AspectLock, ToAspect` and `2 = Aspect`

If `AspectLock` is used with vector objects, they are rendered centred in the mask. Example:

```
loadmask BLEND "adfs::4.$.Images.softvignette"
```

savemask

Saves the specified mask of the current object.

```
savemask <mask> <filename> [<filetype>]
```

Option / parameter	Data type	Description and values
`<mask>`	keyword	`Blend | Tint | Curve | Shadow | Displace | Texture | Data`
`<filetype>`	keyword	`Sprite | JPEG | PNG | Squash`

The vector file data for **vector in sprite** objects may be extracted by saving the Data mask, in which case `<filetype>` is ignored as it will save a Draw or ArtWorks file, as appropriate. (This will not work with TopModel files, as they are heavily modified when loaded.) Examples:

```
savemask BLEND "adfs::4.$.Images.Spider" JPEG
savemask Data "adfs::4.$.Images.ViSMask"
```

---

processmask

Applies image processing to the specified mask of the current object. There are three forms: (1) a single-line form for mask processing operations, (2) a single-line form for alpha channel operations, and (3) a multi-line form for graduated fills.

Most of the processes from Compo's **Edit Mask** menu are available, with any submenu options being specified as additional parameters.

The list of processes is extendable: any of the filters specified in the file !Compo.Resources.Filters may be used, though it's not immediately obvious which can be used as a process in its own right, and which are a subprocess of Antialias.

Note that any parameter that is made up of more than one word (i.e. space separated) must be quoted.

(1) Simple process

```
processmask <mask> <process> [<processparams>]
```

Option / parameter	Data type	Description and values
`<mask>`	keyword	`Blend | Tint | Curve | Shadow | Displace | Texture`
`<process>`	keyword	`Antialias | Filter | Darken | Edge | Ghost | Invert | Lighten | Sharpen` `Filter = Antialias`

The value of `<processparams>` depends on the value of `<process>`:

When `<process> = Antialias` or `Filter`:

<processparams>	keyword	Silk | Soften | Smooth | Smudge | InvGaussian | "Smudge horizontal" | "Smudge vertical" | "Gaussian blur" | Emboss | BigEmboss | Vaseline | SuperSmooth | Froody | "Edge detect" | "Edge enhance"

Silk, Soften, Smooth and Smudge can each take an optional second parameter :

	keyword	All | AllButBlack | AllButWhite | AllButWhiteOrBlack | OnlyBlack | OnlyWhite | All (specifies nothing)

When <process> = Darken:

<processparams>	keyword	All | AllButBlack | AllButWhite | AllButWhiteOrBlack | OnlyBlack | OnlyWhite

When <process> = Edge:

<processparams>	keyword	Minimum | Median | Maximum

When <process> = Ghost: no parameters

When <process> = Invert: no parameters

<When <process> = Lighten:

<processparams>	keyword	All | AllButBlack | AllButWhite | AllButWhiteOrBlack | OnlyBlack | OnlyWhite

When <process> = Sharpen:

<processparams>	keyword	All | AllButBlack | AllButWhite | AllButWhiteOrBlack | OnlyBlack | OnlyWhite | Cutoff

Cutoff requires a second parameter, an integer <value> of 0–255 which passes a cut-off filter over the mask. Values in the mask less than <value> are set to zero (i.e. transparent), those equal or greater are set to 255 (i.e. solid).

(2) Alpha channel process

```
processmask <mask> readalpha
```

Reads the alpha channel into the specified mask.

```
processmask <mask> writealpha
```

Writes the specified mask to the alpha channel of the image.

Option	Data type	Description and values
<mask>	keyword	Blend | Tint | Curve | Shadow | Displace | Texture

(3) Graduated fill process

```
processmask <mask> GradFill

{
 Type: Circular|Angled
 Blended: YES
 Opacity: <opacity>

 # If Circular :
 Centre: <cx>,<cy>
 Size: <size>
 EdgeIntensity: <ei>
 CentreIntensity: <ci>

 # If Angular :
 Startpos: <xl>,<yl>
 Endpos: <x2>,<y2>
 StartIntensity: <il>
 EndIntensity: <i2>
}
```

Option	Data type	Description and values
<opacity>	integer	Amount of transparency (0–255).
cx> <cy>	integer	Centre coordinates (pixels).
<size>	integer	Radius (pixels).
<xl>,<yl>, <x2>,<y2>	integer	Start and end positions (pixels).
<ei>,<ci>, <il>,<i2>	integer	Intensity levels (0–255).

Examples:

```
processmask shadow antialias smudge
processmask blend edge maximum
processmask tint antialias smooth allbutblack
processmask blend antialias "edge detect"
processmask blend sharpen cutoff 128
```

shadow

Creates a shadow mask for the current object.

```
shadow ON
```

```
{
 Type: <type>
 Opacity: <opacity>
 Colour: &rrggbb
 XOffset: <xoff>
 YOffset: <yoff>
 3D: OFF|ON,<shear>,<shrink>
}
```

Option	Data type	Description and values
`<type>`	keyword	`Normal | Subtractive | Blend`
`<opacity>`	integer	Transparency value (0–255).
`<xoff>,<yoff>`	integer	Displacement of the shadow (pixels).
`<Shear>`	integer	Amount of shear.
`<Shrink>`	integer	Amount of shrinkage.

If values for `<shear>` and `<shrink>` are omitted, a value of zero will be assigned, even when setting `3D: OFF`. In the `3D:` option its parameters must be comma separated. The shadow's opacity can be linked to the opacity of the object itself, using the dotvar `.shadow`.

```
# Example
shadow ON
{
 Type: Normal
 Opacity: 255
 Colour: &a0a0a0
 XOffset: 3
 YOffset: 3
 3D: OFF,0.30,0.50
}
```

---

readpixel

Reads the specified pixel from the current object. The pixel is from the original image, before any of effects have been applied. So settings of opacity, mask, shadow and math have no effect on the returned values.

```
readpixel <x> <y>
```

Option	Data type	Description and values
`<x>,<y>`	integer	Coordinates of pixel.

Following a `readpixel` command, several dotvars are set.

writepixel

Sets the specified pixel in the current object. It has two forms.

```
writepixel <x> <y> (<p>)
writepixel <x> <y> (<r> <g> <b>)
```

Option / parameter	Data type	Description and values
`<x>,<y>`	integer	Coordinates of pixel to be updated.
`<p>`	integer	Pixel value in Acorn 32-bit sprite format `&xxbbggrr`.
`<r>,<g>,<b>`	integer	Red, green and blue components (0–255).

The dotvars set after a call to `readpixel` will be used by default in a `writepixel` command if you don't specify them.

`readpixel` and `writepixel` are slow in execution, and hence not recommended for processing an entire image. An alternative approach would be to write a program to process an image and call that. See the **star** command for details.

mouseover

Attaches procedures to mouse events. It is a powerful command that also makes possible the creation of presentations using CompoScript. You can't switch off a mouseover, and active mouseovers slow execution speed. You may have up to ten mouseovers active.

```
mouseover <object> on <event> proc <procname>
```

Option	Data type	Description and values
`<object>`	string	Name of object to associate with mouseover event.
`<event>`	keyword	`Click | Over`
`<procname>`	string	Name of procedure called when mouseover event occurs.

```
mouseover "fred" on over proc flash
```

```
mouseover "fred" on click proc something
```

When the pointer is over the bounding box of the object called "fred", `proc flash` is called, and when there is a mouse click on this object, `proc something` is called.

Within the called procedure, the dotvar `.mouseover` will hold the name of the object that the pointer was over or clicked on. The usual variables can then be used. This is best illustrated by a complete script:

```
#CompoScript
```

```
makecanvas 640 480 0
let the_cows_come_home = 0
maketext "Daisy"
{
 Text: "Moo!"
 Font: Homerton.Bold
 Colour: &00cc00
 At: 100 120
}
mouseover "Daisy" on click proc something
repeat
   # an empty loop which will continue until the
   # mouseover has executed
until the_cows_come_home
end

defproc something
   select <.mouseover>
   report "The user clicked on <.leafname>"
   report "it is at <.xpos> <.ypos>"
   let the_cows_come_home = 1
endproc
```

---

include | **library**

The commands **include** and **library** are synonymous. They take a single parameter which should resolve to a string pointing to the library file to include.

```
include "<filename>"
library "<filename>"
```

Libraries are text files containing script procedures. When searching for a proc to execute, Compo first searches for the name of the procedure in the main script and then in the libraries in the order in which they were included.

Scripts which use libraries should be properly formed, i.e. they must end with an **end** or **stop** command) otherwise execution may stray into the libraries with unexpected results.

The system variable `Compo$Scripts` is defined pointing to the scripts directory. Library files are best stored in directory within this directory, say **libs**. The command to include a library of procedures then takes the form :

```
library "<Compo$Scripts>.libs.shadow"
```

thus avoiding problems inherent with absolute paths.

scriptoptions

```
scriptoptions <option>
```

Sets global script options:

`scriptoptions HideCanvas`
Moves the canvas to the back of the window stack.

`scriptoptions HidePanes`
Hides the pane windows.

`scriptoptions ShowPanes`
Shows the pane windows.

`scriptoptions FullScreen "X800 Y600 C32K"`
Selects the screen mode, removes title and scrollbars, and centres the canvas in the display. An error is returned if the specified mode cannot be selected.

`scriptoptions ShowPointer`
`scriptoptions HidePointer`
`scriptoptions BigPointer`
Control the visibility and size of the mouse pointer.

`scriptoptions HourGlassOn`
`scriptoptions HourGlassOff`
Control the visibility of the hourglass.

`scriptoptions CT_Cols`
Use ColourTrans colours.

---

loadcompo

Loads a Compo canvas from disc.

```
loadcompo <file>

# Example
loadcompo "Boot:^.Images.Scene"
```

---

savecompo

Saves a Compo canvas to disc.

```
savecompo <file>

# Example
savecompo "Boot:^.Images.Scene"
```

---

nib

Controls whether the screen is updated.

```
nib <option>
```

Option / parameter	Data type	Description and values
`<option>`	keyword	`Up | Down | NoBox`

`nib up` disables screen updates. This is useful when generating animations, to stop redraws until the changes to an object are complete.

`nib down` updates the screen and allows subsequent operations to update the screen immediately.

`nib nobox` After this command is issued, selecting an object won't place the dotted GUI selection box around it: it will just be invisibly selected. This is is the option taken automatically at the start of a script.

rredraw

```
rredraw (<x> <y> <width> <height>)
```

Causes a redraw of the specified region of the current object. If no parameters are specified, the entire object is redrawn. If no object is selected it regenerates the given area of the canvas. It is typically called after processing an image with `writepixel` or calling an external program that alters it.

Option	Data type	Description and values
`<x>,<y>`	integer	Coordinates of bottom left corner of the region to be redrawn.
`<width>, <height>`	integer	Width and height of the region to be redrawn.

```
#CompoScript
# Demonstrates the use of the '.text.texture' dotvar
# to establish whether or not a text object is textured.

select "mytext"
if .text.texture = "" then
   report "this text is not textured"
else
   report "this text is textured with <.text.texture>"
endif
end

#CompoScript
# Doubles the size of all text objects on a canvas, retaining
# their aspect ratio. Non-text objects are left untouched.
# Demonstrates the use of the '.istext' and '.text.size' dot
# variables.

select all
doto selection
{
 if <.istext> then
   # Only text objects are processed.
   proc doublesize
 endif
}
end
# ============================================================
defproc doublesize
   let newheight = .text.size * 2
   let newwidth = (newheight * .text.aspect) / 100
   maketext
   {
   Size: <newheight> <newwidth>
   }
endproc
```

```
# ============================================================
# Alternative form ...

# Note that spaces aren't allowed within the expressions

defproc doublesize_alt
 maketext
 {
   Size: <.text.size>*2 (<.text.size>*2*<.text.aspect>)/100
 }
endproc

# ============================================================
```

---

doto

Apply a series of commands to the objects referenced.

```
# doto example 1
doto (selection|"name")
{
   commands
}

# doto example 2
doto "fred"
{
   proc do_something
   proc do_something_else
}

# doto example 3
doto selection
{
   proc do_something_else_entirely
   proc do_something_different
}
```

Errors which can arise are:

```
Nothing selected for doto command
Object passed with doto command could not be found
Doto commands cannot be nested
Select command may not be used inside a "doto"
Delete command may not be used inside a "doto"
```

---

delete

Deletes all currently selected objects.

---

flipimage

Produces a left-right or top-bottom mirror image of the current object.

```
flipimage <action>
```

Option	Data type	Description and values
`<action>`	keyword	`Horizontal | Vertical`

makecopy

```
makecopy <name>
```

Copies the current object, including its masks and other attributes (shadow, opacity, etc.) and names it `<name>`.

```
makecopy "fred"
```

Makes a copy of the current object and calls it "fred".

rename

```
rename <name1> <name2>
```

Renames object `name1` as `name2`.

```
rename "fred" "bert"
```

Renames the object ‘fred’ as ‘bert’.

move

```
move <action>
```

Moves the current object within the stack of objects.

Option / parameter	Data type	Description and values
`<action>`	keyword	`Front | Back | Forwards | Backward`

moveto

```
moveto <x> <y> [middle]
```

Moves the current object.

Option / parameter	Data type	Description and values
`<x>`	keyword	`Centre | Left | Right | <integer>`
`<y>`	keyword	`Centre | Top | Bottom | <integer>`

`Left` places the left edge of the object against the left edge of the canvas. `Right` places the right edge of the object against the right edge of the canvas. Similarly for `Top` and `Bottom`.

If the optional third parameter `middle` is used, then it is the centre of the object that is placed against the relevant edge. If numeric values are given for `x` or `y`, then the centre of the object is placed at that coordinate.

```
moveto centre top
moveto 200 300
moveto 500 bottom
moveto 200 300 middle
moveto left 200 middle
```

---

moveby

```
moveby <dx> <dy>
```

Moves the current object relative to its current position.

Option / parameter	Data type	Description and values
`<dx>`	integer	Number of pixels to move horizontally: positive to the right, negative to the left.
`<dy>`	integer	Number of pixels to move vertically: positive upwards, negative downwards.

```
moveby 300 -50
```

---

rotateto

```
rotateto <direction>
```

Rotates the current object to the specified angle (degrees).

Option / parameter	Data type	Description and values
`<direction>`	integer	Absolute angle with respect to x-axis

```
#Example

rotateto 30
```

---

scaleto

```
scaleto <width> <height> [AspectLock]
```

Scales the current object to a specific size, optionally keeping the aspect ratio of the image unchanged.

Option / parameter	Data type	Description and values
`<width>`, `<height>`	integer	New width and height (pixels).

```
# Examples
scaleto 500 650
scaleto 500 200 AspectLock
```

scaleby

```
scaleby <xscale> [<yscale>]
```

Scales the current object by the specified factors.

Option / parameter	Data type	Description and values
`<xscale>`, `<yscale>`	pseudo-real	x and y scale factors. If `<yscale>` is omitted then `<xscale>` is used to scale both x and y, keeping the original aspect ratio.

```
# Examples:
scaleby 1.1 0.75
scaleby 0.5
```

trim

Trims the size of the current object. It has two forms:

```
trim <x0> <y0> <xl> <yl>
trim around
```

Option / parameter	Data type	Description and values
`<x0>,<y0>` `<xl>,<yl>`	integer	left edge, bottom edge (pixels) right edge, top edge (pixels)

Use negative values to extend the size.

The second form is the equivalent of the `trim round` mask option in the Trim dialogue box.

Following a **trim** command, several dotvars are set, which are useful after a trim around so you can find the same image centre pixel.

export

```
export <type>
```

Saves the current object as a bitmap image file, or copy a flattened version back onto the canvas. Options:

```
{
 Filename:
}
```

Option	Data type	Description and values
`<type>`	keyword	`PNG | Sprite | JPEG | GIF | Clear | PBM | TGA | PSD | BMP | ToCanvas`

The options are type-specific and are mostly optional apart from Filename:. There are sensible defaults for options not specified.

CanvasArea: is automatically turned on.

The type `ToCanvas` differs from the other types in that no file is saved, but instead the object is exported back onto the canvas.

export PNG

Saves the current object as a PNG file.

```
{
 Filename: "<string>"
 CanvasArea: NO|YES
 Gamma: 1.0
 Interlace: NO|YES
 Mask: NO|YES
 Shadow: NO|YES
 Palettise: NO|YES
 CheckMask: NO|YES
 TrimMask: NO|YES
 CheckGrey: NO|YES
 More: "<string>"
}
```

Option	Data type	Description and values
`CheckMask:`	keyword	`YES` to simplify the mask `| NO`
`TrimMask:`	keyword	`YES` to to trim transparent rows and columns `| NO`
`CheckGrey:`	keyword	`YES` to to produce a 1 channel greyscale PNG `| NO`
`More:`	string	Add the string to the command line of program Spr2png (q.v.)

export Sprite

Saves the current object as a Sprite file.

```
export Sprite
{
 Filename: "<string>"
 CanvasArea: NO|YES
 Mask: 0|1|2 (none, 1bpp, alpha)
 Append: NO|YES
 Spritename: "<spritename>"
 Dither: NO|YES
 BlendCanvas: NO|<value>
 OutputFSI: NO|YES
 # if yes :
 CFSICols: 0|1|2|3|4|5 (16m, 32k, 256, 256g, 16c, b/w)
 CFSIScale: <scale>
 # and/or :
 CFSIx: <scale>
 CFSIy: <scale>
 CFSIAspect: NO|YES
```

```
 CFSISharp: NO|<value>
 CFSISmooth: NO|<value>
 CFSIGamma: NO|<value>
 CFSIDither: NO|YES
 CFSIRange: NO|YES
}
```

Option	Data type	Description and values
`CFSICols:`	integer or keyword	Number of colours in output sprite: `0 | "16m" | "millions"` for 16M colours `1 | "32k" | "thousands"` for 32K colours `2 | "256" | "riscos" | "255"` for 256 colours `3 | "256g"| "greys" | "greyscale"` for 256 greys `4 | "16c" | "16" | "riscos16"` for 16 colours `5 | "b/w" | "b&w" | "mono" | "black and white"` for black and white sprite

To export as a 16 colour sprite, the desktop must first be put in a 16 colour mode (this is a ChangeFSI requirement).

---

export JPEG

Save the current object as a JPEG file.

```
export JPEG
{
 Filename: "<string>"
 CanvasArea: NO|YES
 Quality: <number>
 Progressive: NO|YES
}
```

---

export GIF

Save the current object as a GIF file.

```
export GIF
{
 Filename: "<string>"
 CanvasArea: NO|YES
 Palette: 0|1|2 (Best, RISC OS, RO, NetScape, NS)
 Mask: NO|BLEND|<value>
 BlendCanvas: NO|<value>
 DitherMask: NO|YES
 DitherSprite: NO|YES
 Interlace: NO|YES
 TrimToMask: NO|YES
}
```

---

export Clear

Save the current object as a Clear file.

```
export Clear
{
 Filename: "<string>"
 CanvasArea: NO|YES
}
```

---

export PBM

Save the current object as a PBM file.

```
export PBM

{
 Filename: "<string>"
 CanvasArea: NO|YES
}
```

---

export TGA

Save the current object as a TGA file.

```
export TGA
{
 Filename: "<string>"
 CanvasArea: NO|YES
}
```

---

export PSD

Save the current object as a PSD file.

```
export PSD
{
 Filename: "<string>"
 CanvasArea: NO|YES
}
```

---

export BMP

Save the current object as an uncompressed 24bpp BMP file.

```
export BMP
{
 Filename: "<string>"
 CanvasArea: NO|YES
}
```

---

export ToCanvas

Produces a new canvas object from the currently selected object. It is like saving an image and then reloading it, but as a new object with default attributes.

If `CanvasArea:` is specified and defined as YES then the result will be flattened, i.e. it will be a single object containing the results of all the layered objects present within the selected object's bounding box.

This command is particularly useful with AOIs to make a single bitmap object (e.g. a Sprite) from whatever is on the canvas within the AOI's bounding box and with the option of composited masks.

```
export ToCanvas
{
 Filename: "<string>"
 CanvasArea: NO|YES
 Mask: 0|1|2 (none, lbpp, alpha)
 At: <x>,<y>
}
```

Synonyms for `ToCanvas` are `AsObject` and `Object`.

For this export type `Filename:` should be just the leafname of the new object; any path elements will be ignored.

If `Mask:` is present and set to '2', then all of the masks present in the target area (the PSO) will be combined and given to the new object.

If the `At:` line is present, then the bottom left of the new image will be placed at the specified pixel coordinates otherwise it will be at 0,0 (bottom left of the canvas).

Example applications of `export ToCanvas` include: image stacking, where a number of dark frames are added together to produce a well exposed image, and using a script to produce an average of a series of frames, e.g. for digital camera sensor noise reduction.

---

loadimage

Loads a graphics object. There are two forms, depending on whether the object is a bitmap image file, or a vector file in Draw or Artworks format:

```
loadimage <filename>
{Vector: YES|NO
 Scale: <scale>
 Angle: <deg>
 Automask: NO|YES
 Border: <bx>,<by>
}
```

Option / parameter	Data type	Description and values
`<scale>`	integer	Scale factor to apply (%). 100%
`<deg>`	integer	Rotation to apply (degrees). 0. positive anticlockwise, negative clockwise

`<bx>,<by>`	integer	Width of surrounding border (pixels). 0,0

If any options are omitted, or even if the first form is used with a vector file, default values are used. These are :

If the second form is used with a bitmap file, the values given are ignored.

---

ifexists

Checks for the existence of an object.

```
ifexists "textl" then
   proc something
endif
```

tests for the existence of an object called "text1". It is possible to use `else` and `elseif` with this command.

OLE Object Linking and Embedding

Any current selection can be exported to an editor by double-clicking SELECT with CTRL pressed. This will usually send a sprite to RISC OS Paint, and the edited sprite will be automatically returned to Compo on saving.

A. Glossary

24-bit colour or so-called True Colour

A method of defining the colour of a pixel in an image where eight binary bits (giving 256 possible levels) are used to define each of the three primary colours (Red, Green and Blue) which, when mixed, produce the colour of the pixel. Called *True Colour* because the human eye cannot distinguish all of the shades available.

Aliasing

Aliasing occurs most often when scaling the size of a bitmap. It turns smooth curves into ugly jagged ‘staircases’ and degrades image quality, Anti-aliasing may be used to correct it.

Anti-aliasing

Anti-aliasing is a technique used to improve the quality of rescaled bitmaps by replacing pixels which would be ideally ‘half on’ with an intermediate shade. RISC OS users will be most familiar with anti-abased fonts which improve the quality of fonts displayed on screen at different sizes.

Alpha channel

The RISC OS 24-bit sprite format is actually 32 bits in size. 24bits are used in the definition of the Red, Green and Blue values for each pixel. The remaining eight bits are known as the Alpha channel and are used for video overlay.

Bitmap

A bitmap is an image made up of pixels having a fixed amount of detail. Most accurately, a bitmap contains pixels of only two colours (black and white) though the term is commonly used to describe images with many colours. Increasing the size of a bitmap can only increase the size of the pixels from which it is made up (producing the familiar jagged stall-casing effect called aliasing). Compare *Vector image*, *Pixmap*.

Curve correction

A technique used in Compo and similar to Gamma correction whereby the output colour levels of an image are controlled in relation to the shape of a Bézier curve.

Dots per inch (dpi)

The value used to link the width of an image, in pixels, to the width of the image when printed. See *Resolution*.

Gamma correction

An industry standard way of changing the contrast and brightness of an image. Gamma values greater than 1.0 lighten the image. Gamma values less than 1.0 darken it. Definition: Output=Input (1/Gamma).

A standard value for Gamma used in the Television industry is 2.2.

Mask

Usually, a mask is a 256 grey level image which is used to control the mixing of one image with another. Compo expands this term to an image which is used to attenuate or control an effect. For example, the *Tint mask* is a mask used to control the amount

of tinting applied to the pixels in a sprite.

OLE: Object Linking and Embedding

Allows you to edit a file in another program and see the results of your changes in Compo very quickly by simply saving the file from the editor without changing its filename. Compo automatically reloads the modified image and displays the changes in the composition.

Pixel

A single dot on the computer screen, in a sprite or bitmap image. Short for *picture element*.

Pixmap

A colour image made up of pixels; a narrower definition of Bitmap. Compare *Bitmap*, *Sprite*, *Vector image*.

Rasterising

The process of converting a vector image into an array of pixels (a bitmap). When printing, a vector image is rasterised to the output resolution of the printer in use.

Resolution

The amount of detail in an image. A function of the physical size of the image when displayed and the number of pixels used to make up the image. For example, an image 600 pixels wide printed at a resolution of 300 dots per inch would appear two inches wide. The same image printed at a resolution of 600 dots per inch would be one inch wide.

Sprite

A Sprite, within Compo, is an image, defined as a two dimensional array of values representing the colour of individual pixels within that image. Compo Sprites are always 24 bit colour images. Compare *Bitmap, Vector image*.

Tint

Adding or subtracting a certain shade of colour from every pixel in an image. It is used to produce effects similar to viewing the image through a coloured filter.

Vector image

A vector image is defined in terms of the positions of discrete objects that are lines, points and curves. The details of these objects are held to high precision so that increasing the displayed size of a vector image will often reveal more detail. Although Compo can load vector images from Draw and Artworks files, it does so by converting them into bitmaps of fixed resolution. See *Rasterising*.

B. Keyboard shortcuts

Ctrl-A	Select all sprites
Ctrl-B	Move selected sprites to back
Ctrl-C	Copy selection
Ctrl-E	Add text to canvas at pointer position (using settings from last use of **Ctrl-Shift-E** Create text dialogue)
Ctrl-F	Move selected sprite(s) to front
Ctrl-G	Group selection
Ctrl-I	Open Info dialogue box/bring Info dialogue box to front
Ctrl-L	Open Layout dialogue box
Ctrl-O	Open Opacity dialogue box
Ctrl-R	Open Transform dialogue box (rotate)
Ctrl-S	Toggle Shadow active state of selected objects on/off
Ctrl-T	Open Colour picker dialogue box
Ctrl-U	Ungroup selection
Ctrl-V	Open Colour curve dialogue box
Ctrl-X	Delete selection
Ctrl-Z	Clear selection
Shift-B	Move selected sprite(s) back one layer
Shift-F	Move selected sprite(s) forward one layer
Ctrl-Shift-E	Open Create/Edit text dialogue box
Ctrl-Shift-S	Open Shadow dialogue box
Ctrl-Shift-G	Open Gamma dialogue box
Ctrl-Shift-Z	Open Scale dialogue box
Tab	Toggle toolbars on/off
Shift-Tab	Toggle toolbars and tool windows (mask edit, etc.) on/off
Shift	Holding down Shift while clicking on bump arrows to change values in dialogue boxes causes the values to move in larger increments
Ctrl	Holding down Ctrl while clicking on bump arrows to change values in dialogue boxes causes the values to move in even larger increments
Shift & Load	Holding down Shift while loading an image brings up a Scale dialogue allowing you to scale the image to fit the canvas or to an existing object.
F3	Open save dialogue box. If the pointer is over the Mask edit window then the save box is for the mask sprite. Otherwise the save box is for the composition.

The action of the following group of shortcuts is dependent on the setting of the Vertical and Horizontal options in the Layout dialogue box.

F5	Left/Top align selection on canvas
Shift-F5	Left/Top align .selection on largest sprite in selection
Ctrl-F5	Left/Top align selection on smallest sprite in selection
F6	Centre selection on canvas
Shift-F6	Centre selection on largest sprite in selection
Ctrl-F6	Centre selection on smallest sprite in selection
F7	Right/Bottom align selection on canvas
Shift-F7	Right/Bottom align selection on largest sprite in selection
Ctrl-F7	Right/Bottom align selection on smallest sprite in selection
F8	effect/move Undo when the main window has input focus
F9	effect/move Redo when the main window has input focus

C. Opacity maths options

This list of maths options for the Opacity function needs explanation.

Blend	mixes the sprite with whatever is underneath
Add	adds the sprite to whatever is underneath
Subtract	subtracts the sprite from whatever is underneath
Darker	writes sprite pixel if it is darker than underneath
Lighter	
Multiply	multiplies the sprite by whatever is underneath
NegMultiply	multiplies the sprite by the inverse of underneath
MultAdd	scales sprite pixel by the pixel underneath & adds
Experiment	calculates related to Blend and Multiply functions
Screen	
SoftLight	
HardLight	
Lin. Palette	false colour the image by a palette definition
Palette	false colour the image by a palette definition
PhotoNeg	
XFadeSoftLight	
Dodge	
Burn	
Soft dodge	
Soft burn	
Reflect	
Glow	
Freeze	
Heat	
Interpol	
Stamp	
Red	
Green	
Blue	
Hue	
Saturation	
Lightness	
Hue & Saturation	
Hue & Lightness	
Sat & Lightness	
SatDark & LightLight	
SatLight & LightDark	
Lightness of Saturation	
Saturation of Lightness	
Lightness by Saturation	
Saturation by Lightness	
MyHardLight	
MySoftLight	
Crystal	
Crystal2	
Crystal3	
Undef1	
Undef2	

D. Technical issues

Anti-aliased Fonts

Text is anti-aliased only up to the size set in your RISC OS Font Configuration. To run Configure, double click on the !Boot application on your hard disc. Unfortunately, RISC OS permits anti-aliasing of text only to a maximum of 255 points. When used with Composition it is often desirable to anti-alias text larger than 255 points. For this reason we have created a very simple application called !Tfonts. When run, Tfonts sets the Font manager to anti-alias text much larger than the 255 point default. The maximum size available depends on the amount of memory in your machine and the complexity of a particular font. Tfonts makes a temporary modification only. The next time your computer is reset the configured defaults will be used. For this reason, we recommend setting the anti-aliasing size to the maximum possible using Configure and using Tfonts only if text at a particular size does not appear anti-aliased.

ChangeFSI

Compo uses the excellent !ChangeFSI program found in your RISC OS boot disc directory Utilities. Use of ChangeFSI is transparent and your need never be aware of it unless you have moved it on your hard drive or stopped it being seen by the Filer when the computer is booted. When that is the case Compo reports an error. You can overcome this problem by opening the directory containing ChangeFSI. For a permanent fix you should ensure that ChangeFSI is Filer hooted when the computer is turned on.

Compo's !Run file contains a list of directories where Compo looks to find ChangeFSI. If you store it in a different directory you can add its path to this command so that Compo knows where to find it.

One advantage of using ChangeFSI in this way is that as newer versions become available you can replace your copy of ChangeFSI and Compo will automatically use the new version.

Note: A bug in CHANGEFSI results in a single pixel black line appearing at some edges of an object that has been scaled. If this happens you can remove it by selecting the object and using the Edit <object> ➔ Trim ➔.

!Paint and 32-bit sprites

By default, Paint does not permit the editing of true colour sprites. To force Paint to edit true colour sprites the command *Set Paint$Options X must be placed in an Obey or Command file within your !Boot application. We recommend the file !Boot.Choices.Boot.PreDcsktop. This is a relatively trivial task for an experienced Acorn user. It is not something that we recommend a non technically aware user attempt.

Foreign filetypes

Composition is able to load image files directly from DOS format floppy discs, hard disc partitions and CD-ROMs. To do this mappings between DOS style three character file extensions and RISC OS filetypes need to be established. Composition has commands which do this within the applications !Run file. If you need to add new files in the future simply add them to this list, e.g. &F98 PhotoShp for PhotoShop.

Transformations

To get the best possible quality from the Transform function use the following rules:

1. If an object has been created or modified with Compo you must save the object first. The Update bases command in the File submenu is useful for this.
2. Scale the object you wish to transform by a factor of 2.00
3. Transform the object. You can do this repeatedly bv clicking on Transform with **ADJUST** to get the angles right.
4. Create a Blend mask for the transformed object if one does not exist. (Quick way; click ADJUST on the Opacity icon, click on the Eye icon, click on the black area outside the transformed sprite with SELECT, tidy up the mask then choose Invert from the Mask edit window menu.)
5. Scale the object by a factor of 0.50

Housekeeping

In some circumstances Compo is not allowed to quit cleanly and this may result in files being left in the 'Scrap directory as well as Dynamic memory areas not being released. To clear out both these areas you can start Compo while holding down the Alt key. This clears Compo's scrap files and also checks the Dynamic areas for files use by Compo. While clearing the Dynamic area, it sends messages to the other applications to check if it can clear the memory. The Dynamic areas are also cleared on a reset but the scrap directories are not cleared.

An example that could cause this situation to arise would be a machine crash. This can be caused by any application, a power surge etc. Switching off the machine without quitting Compo may leave files in the scrap directory.

PCA
Applets

Contents

What is PCA?

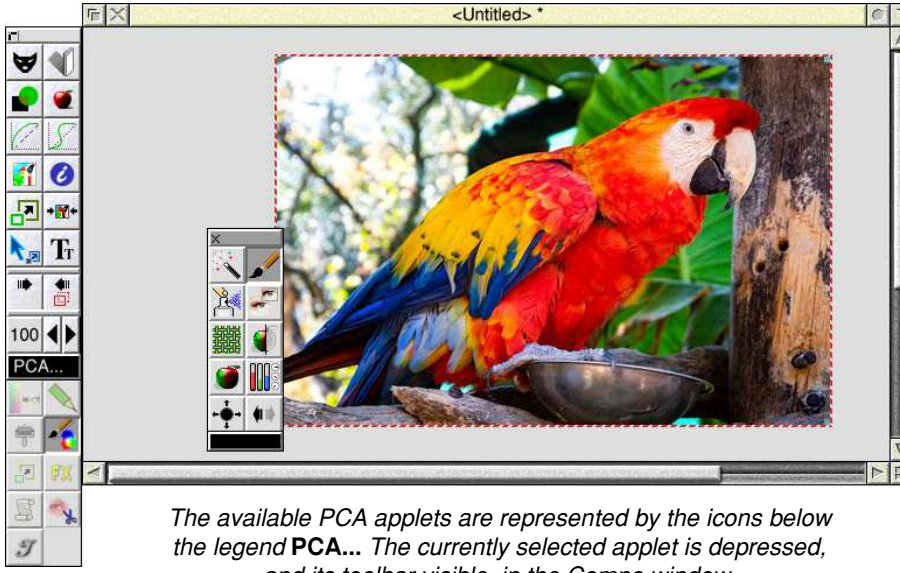
The Plug In Compliant Application (PCA) specification provides an easy way to implement of enabling multiple applications to share common objects in shared memory areas. A program written to the PCA specification will work with any other program which supports the standard and uses the same types of objects.

What does PCA do for me as a user?

The PCA standard works at two levels. In each case a document or object in a common memory area can be shared between two or more applications. This is not like the versions of OLE implemented on the RISC OS platform. There is a major difference in that OLE copies the object for the second application to work on, thus your data takes up twice as much RAM. With PCA all applications work on the same object thus incurring no RAM overheads.

The application which owns the object is called the **Local** application and a compatible tool working on it is called the **Remote** application. When an object is edited in a Remote application the view of it in the Local application is automatically updated as you make changes.

What we have up to now is a more efficient way of doing OLE. However, PCA does not stop there. Take an application like Composition which provides a page make-up environment for bit image and vector images, all of which can be moved and edited: a bit like Draw but for bit images, and with a lot of extras. Compo does not provide painting tools but because it supports PCA's range of tools, which include painting, filling and drawing tools. They can be used to modify images in Compo, dramatically extending the capabilities of the program. The available tools add themselves to Compo's normal application toolbar and appear as if they were part of the program.



The available PCA applets are represented by the icons below the legend **PCA...** *The currently selected applet is depressed, and its toolbar visible in the Compo window.*

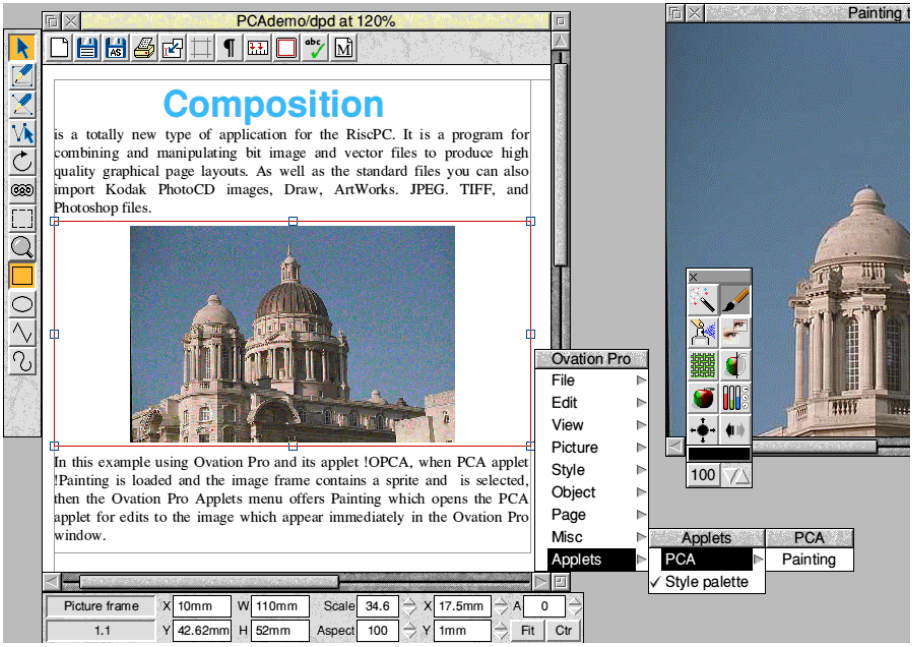
All of these tools will work with any other PCA compliant application.

For example, imagine a program that does nothing but display a sprite but which conforms to the PCA standard. Another application could provide painting tools, another could provide filters, yet another might provide warping tools, and so on. From this range of applets you can construct a toolbox that suits the way you work and has the functions you need. This toolbox can be extended at any time and all of the tools will work with all compliant applications.

Furthermore, more than one tool can work on an object at the same time and changes made in any one of these will immediately be reflected in all of them. This is better than RISC OS OLE, and there is still more functionality.

Instead of each applet opening its own window onto the object we can use In- place editing. This means that only the Local application displays the object and the selected tool merely attaches itself to the object. Any action on the object (such as a mouse click) then goes to the tool and its functions are used to modify the object until you decide to break the link (or choose another tool). Now you are really able to construct an working environment from a range of applets. All you will see is one window displaying the document.

Imagine a DTP package supporting the PCA standard. It could have a frame containing a graphic image and you could link this to any of the tools you have. Any changes made with a selected tool would appear in the DTP window.



A PCA toolbar acting on an image in a DTP frame

What we now have is a system whereby an application can have an unlimited number of additional tools and all applications conforming to PCA can be used with and by all other PCA applications.

PCA links can be nested so that tools can use other tools. For example, an image in a DTP frame could be linked to Compo which uses this image as a ‘canvas’ for a composite picture. From Compo one of these images could also be linked to a painting tool and changes made with this would immediately be reflected in Compo and the combined result would appear in the DTP package.

PCA applets do not need to be loaded to be available to applications. As soon as the directory they are in is ‘seen’ by the filer they become available in any PCA compliant application. All the applets for a particular filetype will be listed by PCA compliant applications, whether they are loaded or not. This means you don’t need to waste memory having applets loaded when you don’t need them. An applet is loaded only when you select it for use with an object. Once you have finished with it, the applet will automatically quit after a period of inactivity. If you are short of memory you can, of course, quit it yourself. In either case, it will remain available, and should you need it again later it will be loaded automatically when selected. Applets which are currently not loaded are shown with faded icons. In applications which use a menu rather than a toolbar to list applets, those loaded into memory are listed first and separated from the rest by a dotted line.

This facility is only possible with applications which support version two of the PCA protocol. Applets written using version two of the protocol will work will applications which support version one of PCA (e.g. OvationPro at the time of writing) - you will however, have to load them yourself.

Esoteric features

Many applications that support PCA provide a number of related options which affect the way the protocol works. Typical options (those provided by Composition) are given below with a brief explanation of what they do.

Compo asks to redraw

When on, Compo asks a selected tool to let it redraw the image when a part of it is modified by the tool. This is because Compo can display the image using a number of special effects such as masking and variable transparency. This is however, only a request and the applet is free to ignore it if it needs to draw user graphics over the image. At the time of writing, only the Painting applet takes note of this option. When it is on, the depth, transparency, masking and other special effects of Compo are visible even when editing with the Painting applet but redraw is slower. When off, the Painting applet redraws the image but Compo’s special effects are disabled until you break the editing link.

Track selected object

When on Compo will move the last applet selected as you select images in Compo so that it is always working on the selected image.

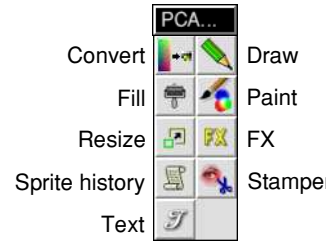
Allow in place edits

When on, Compo allows those tools that can do so to edit the image *1n place’ on the canvas instead of opening their own view onto the image. Consequently you can only use one tool on an image at once and all mouse clicks over the image are sent to the tool. Turning this option off makes the selected tool open its own window onto the object and lets you edit an image with more than one tool at once.

The Applets

Applets toolbar

Shown here is a PCA applets toolbar when all those applets are loaded.



The PCA applets toolbar

Current Colour

The current colour is displayed in a block beneath the Tool Pane. Depending on the tool selected there may be two colours. Primary and Secondary. The Primary colour is the main colour and is used when only one colour is required e.g. Paint brush, single colour fill etc. The secondary colour is used for graduated fills, shadows, fill colour in Draw objects etc. Use SELECT to select the Primary colour and the ADJUST to select the Secondary colour.

The current colour block is displayed as one solid block when only one colour is used, or as a block of three when two colours are used. This enables a representation of the graduated set of colours. The colours can be edited by clicking on them to bring up a Colour Picker.

n some cases a tool will be a single colour but it can also have a secondary colour, e.g. Airbrush. By default this is set to a single colour, to mimic a real airbrush, but pleasant effects can be generated by using a secondary colour. To achieve this you must click on the Secondary section, the right hand block, to bring up the Colour Picker. You can then change the colour. Once changed you :an then select the Secondary colour from the Palette using ADJUST as usual.

A return to a flat colour (where primary and secondary are the same) click in :he middle of the colour block - to set both colours to the same setting. If the current tool does not use a colour then a ‘hatched’ pattern is displayed in the colour block.

n the Drawing applet the colour block is shown with a diagonal line of the primary colour with the block filled with the Secondary colour. The Primary colour is used for the lines and outline and the secondary colour is the fill colour. The diagonal line also indicates the line thickness.

Bump arrows

n most cases the action of the bump icons is altered by using Shift or Ctrl. This alters the step size used so you can increase or decrease numeric values faster.

Zoom

Note that when used in non-in place mode applets have an additional pane below the colour pane which lets you zoom into the picture being edited.

Palettes

Some applications provide a palette, which is shown below the main window of the application. This shows a selection of colours which can be altered to suit the picture currently being worked on. It also shows the most recent colours used. It has an eye dropper associated with it. This can be used to pick up colours off the graphic in the window. The colour can then be dropped either on the palette, or into the colour bar in the current applet’s tool pane.

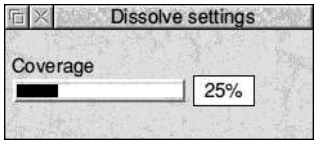
Painting applet

This applet, also known as **!Painting** or the Painting Tool (and not to be confused with !PCAPaint), provides a general set of painting functions. All tools are applied using the current brush settings, which can also be edited. In all cases **SELECT** paints and **ADJUST** undoes any change made.

The paint undo using **ADJUST** obeys the brush settings, including opacity. Set Opacity to 100% to undo completely, lower values reduce the effect of the change instead of undoing completely.

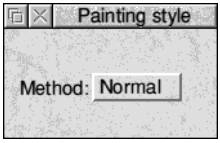
All items in this applet are affected by the Brush Designer (the icon at the bottom left of the tool pane). The size, shape, density and opacity of all tools can be altered using the designer. It is a very powerful tool, and is described below.

Dissolve



This tool has an additional dialogue box. **SELECT** reads the area under the brush, breaks it up and writes it back. Use **ADJUST** to undo. Coverage controls the number of pixels used in the brush.

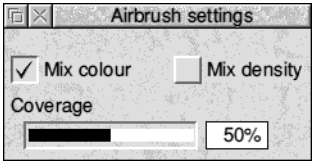
Paint



Click on the icon in the dialogue box to change modes. This can be:

- Normal paint style as defined by Brush Designer. SELECT paints, use ADJUST to undo.
- Additive i.e. if red is current colour then a red tint is added to the image. Colour must be a pure colour i.e. not a mixture of two colours. Use the Colour picker to select the colour.
- Subtractive i.e. the current colour is subtracted from the image. Colour must be a pure colour i.e. not a mixture of two colours. Use the Colour picker to select the colour.

Airbrush



Provides a user definable Airbrush.

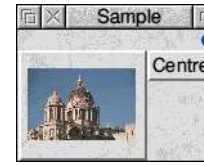
The inner and outer colours can be chosen but initially they are set to the same colour. To select different inner and outer colours you must first click on the left section of the ‘Current colour’ block and select a colour for the Outer area. Do the same for the Inner area by clicking on the right block of the ‘Current colour’ block. Once this has been done you can change the colours by clicking on the Palette with the Left and Right mouse buttons. If the ‘Mix colour’

button is selected then the inner and outer colours are mixed together randomly. If 'Mixed colour' is off then the colour used is determined by the intensity of the brush at the relevant point, the first colour is used at the outside of the brush (where brush pixels are dark), the second colour on the inside (where brush pixels are white) with a blend between the two extremes.

Obviously this will only take effect if the brush is 'soft' and has grey pixels in it.

The airbrush effect is much nicer with a soft brush, so open up the Brush Designer and select one of the other brushes using the bump arrows. You can also design your own brush by clicking on the Edit button (this is only available on the initial brush). The 'Mix density' button randomizes the brush density. The 'Coverage' slider controls the number of pixels used in the brush.

Sample

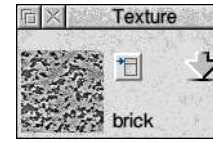


This tool enables part of an image to be cloned. Click with SELECT to position the sample point on the thumbnail image in the dialogue box. Alternatively, while the Sample tool is selected, Ctrl clicking on the main image will set the start point.

When you paint the canvas is cloned from the start point. Remember that the tool uses the current Brush design. Each time you paint on the canvas the sample re-starts. For example, you could sample a flower and repeat it several times. However, if you hold down Shift whilst painting you can release the Left Mouse button, move to a new position and continue cloning the image i.e. you do not re-start the sample.

Click on 'Centre' to put the sample point at the centre of the image.

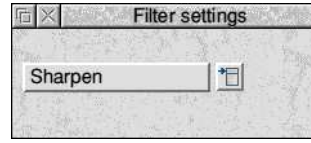
Texture

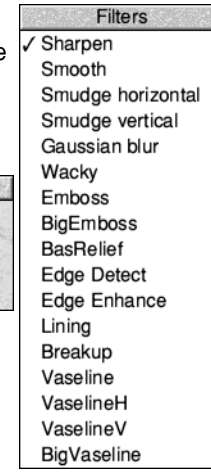


Applies a transparent texture e.g. canvas, to the image. Choose textures from the menu which is opened by clicking on the Menu icon. The best effect is achieved with Brush Opacity set to about 20% using the Define Brush option

Textures may be loaded by dragging a 256 level greyscale sprite or a JPEG to the load target icon on the right hand side of this dialogue box.

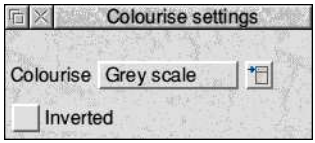
Filters





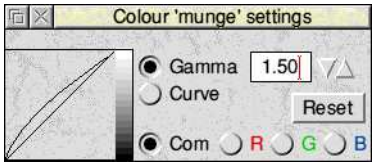
Various filters can be applied using the current brush. Click on the Menu opener to open the list and select the required filter. Most of the filters have self explanatory names but for the others it is best to just try them.

Colourise



Applies various tints (supplied or user defined) to the image via the current brush. To select a 'Tint set' click on the Menu icon and select from the list. To define your own Tint set, click on the left and right blocks in the 'Current colour' block and use the colour pickers to select the colours. To use this set select 'User defined' from the list. The 'Inverted' button, swaps the start and end colours. Initially you should have the Brush Opacity set to 100% as the effect will be more predictable though lower opacity settings can be useful to tint an image.

Munge



Provides Gamma correction and Colour curve correction via brush. For Gamma correction select 'Gamma' and drag the curve to the required position or drag over the graduated bar to the right of the curve box. You can also edit the numeric setting if required. You can edit Combined colours or Red, Green and Blue individually. When modifying the image, the Combined settings is taken into account.

Example: Gamma is selected with a Combined value of 1.5. Turn Combined off, switch to Red and set the gamma to 2.0. Switch to Blue and set gamma to 0.8. When the Process button is clicked the Red channel is set to a gamma value of 1.5 * 2.0 = 3.00 of the original, Green is 1.5 (because we did not set an individual value). Blue is 1.5 × 0.8 = 1.2 of the original image.

Curve works in a similar way.

Colour Curve is chosen by selecting the Curve button. There are 4 points on the line and these can all be dragged independently. The graduated bar indicates the effect that the tool will have.

These tools are useful for enhancing photographs, especially if they are a little dark or light.

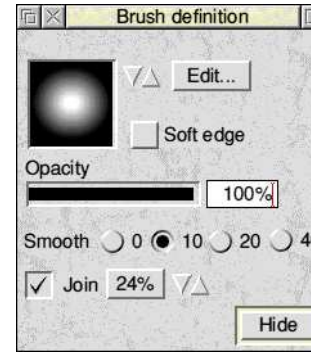
Define brush

This is probably the most important and powerful tool in this applet. It lets you select from various supplied brush styles or create a user defined one. Various 'Vignettes' are supplied and can be accessed using the bump arrows. A representation of the brush is shown.

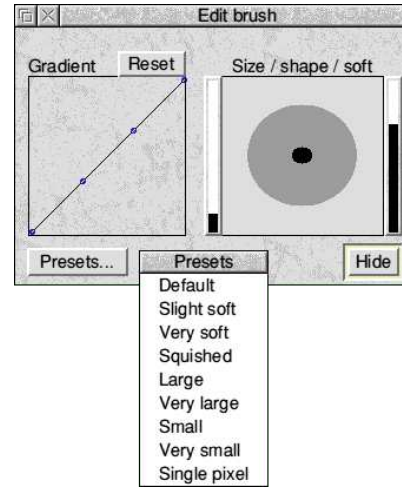
The Opacity slider lets you set the brush opacity so that you can have semi transparent brushes. More rarely required settings are available by toggling the box to full size as shown in the picture above.

Interpolation can be turned on or off using the 'Join' button. When on, this joins up painting points, producing continuous lines. This is most evident on small brushes.

‘Smooth’ averages mouse coordinates so that you can draw smooth curves Smooth 0 turns it off, the other settings smooth the output position more and tend to eliminate fine detail (like the wobbles you get when painting with the mouse). You can use any combination of Smooth and Join settings.



To define a brush click on the bump arrows until an Edit button appears. Click on the Edit button to open the edit window. The window consists of an area for editing the Gradient and another for editing the size, shape and feathering properties. You can alter the brush shape by dragging the brush representation in the right half of the window. You can also control the size this way, but it is much easier to use the left hand slider to control the size of the solid core of the brush and the right slider to control the size of the feathered area. It is this that gives a smooth outline to the brush. The Gradient section has 4 points that can be dragged and this controls the rate of change between the solid core of the brush and the feathered edge. This is best viewed in the brush representation of the Brush Edit window i.e. the one where you select the vignettes using the bump arrows.



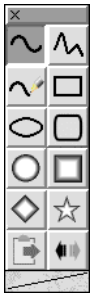
The Presets button opens a menu of useful preset brush shapes and sizes.

Undo / Redo

Provides a one step undo. All work done with a particular tool will be undone but as soon as another tool is selected the image is fixed. Clicking again will perform a Redo, that is it undoes the Undo. A paintable Undo is available by using ADJUST to paint. This also obeys the opacity and brush settings.

Whilst working with a tool you can ‘confirm’ your changes to date by pressing the Spacebar. Undo will then return you to that point.

Draw applet



Colours are selected in the normal manner. A dialogue indicates the line width and enables you to alter it using the bump arrows.

The line thickness is indicated by a diagonal line in the 'Current colour' block. The colour of the diagonal line is the Line colour and the colour of the box is the Fill colour. The default Grey means that there is no Fill colour. Selecting a colour with SELECT sets the Line colour and using ADJUST selects the Fill colour. Colours can be changed whilst an object is still being edited.

When an object is completed you have the option to edit it in various ways. A bounding box appears around the object with 3 points joined by lines. The centre point is the 'Origin' and this is the point about which the object will be rotated or scaled. Double click on the point for a graphical input dialogue.

The top point is the 'Rotation' point and dragging on it rotates the object about the Origin. Holding down Ctrl rotates in steps of 15 degrees. Double click on the point for a dialogue box where you can enter the number of degrees of rotation required. Hold down Shift to step 5 degrees at a time when using the bump arrows or Ctrl to step 10 degrees at a time.

The lower point is the 'Scale' point. Drag on it to scale the object about the origin. Hold down Ctrl to retain the Aspect ratio.

The object is fixed to the canvas by Double clicking over it, or single clicking outside it. To abandon the object before fixing it to the canvas double click ADJUST

The Drawing tools are:

Bézier curves

Plot start point, move and plot end of curve segment. You can now move the pointer to modify the curve. When the curve is satisfactory plot another point and do the same for that segment. At any time you can click ADJUST to modify the existing curves by dragging the anchor points around. To resume adding points click with ADJUST again. Double click to complete the object or double click the Right Mouse button to cancel the object.

Objects will be automatically closed when the curve being added gets close to the starting point. You can then cut holes inside a closed, filled object by creating another closed object inside it.

Straight lines

Plot start point of the line segment with SELECT and then plot end point with SELECT. Continue dragging and clicking until you have the required shape. Click ADJUST to edit the control points, which are draggable. Click again with ADJUST to continue adding line segments with SELECT. Double click to complete the object or

double click ADJUST to cancel the object.

Freehand

Hold Left Mouse Button down and draw. Release to stop drawing. The objects created with this tool cannot be rotated or scaled and are deleted using the Undo button.

Rectangle

Plot start point and drag, in any direction, to required size. Hold down Ctrl whilst dragging to create a square. Click with Left button to complete object. Click to fix the object or double click the Right Mouse button to cancel the object.

Ellipse

Plot start point and drag, in any direction, to required size. Hold down Ctrl whilst dragging to create a circle. Click with Left button to complete object. Click to fix the object or double click the Right Mouse button to cancel the object.

Rectangle with round corners

Plot as for normal rectangle.

Graduated ellipse

As for ordinary ellipse but colours are graduated.

Graduated rectangle

As for normal rectangle but colours are graduated.

Graduated diamond

Plots a graduated diamond shape. Ctrl maintains aspect ratio.

Graduated star

Plots a graduated star shape. Ctrl maintains aspect ratio.

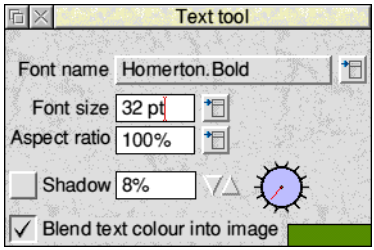
A Draw object

If the content of the Clip Board is a Drawfile then choosing this icon reads the drawing from the clipboard and plots the object. Ctrl will retain the original aspect ratio.

You can also load and plot any Drawfile by simply dragging its file icon to the drawing tools toolbar.

Text applet

There is no Tool pane associated with this applet as there is just one dialogue box where you can select Font, Size, Aspect ratio and Shadow for use in adding text to the canvas. Once fixed the text becomes part of the image and cannot be edited though it can be undone by the undo button on the main toolbar.



To enter text, select the required settings and click on the canvas. A box appears and as you type the box expands to encompass the text.

You can insert carriage returns, if required, to produce multi line text and the text remains editable until you click outside the text box or double click on the text. Whilst the text is editable any of the settings can be changed and viewed.

The text bounding box can be dragged to re-position the text. There is also a diagonal line with a scaling point. This point can be dragged to alter the scale of the text. If you hold down Ctrl whilst scaling the Aspect ratio is maintained. The Blend Text button is used to provide true anti aliasing to each pixel. If the text is on a single colour background or is going to be filled using the graduated fill tool after it is fixed to the canvas then this option should be turned off.

You can also add a Shadow by selecting the Shadow button. The percentage figure is the size of the shadow in relation to the point size of the font. The Shadow Dial is used to set the direction of the shadow by dragging the pointer to the required angle.

When shadow is on, clicking the Left button over the colour patch selects the text colour and the Right button selects the Shadow colour.

Importing text

If the Clipboard contains text then it can be pasted onto the canvas using the Paste from Clipboard icon. The text is scaled to fit a sensible amount on the screen but you can still edit any of the settings and the text itself before fixing it to the canvas. The text is clipped after 99 lines and is indicated in the text by ‘Clipped here’.

Alternatively, drag a text file onto the text applets dialogue box to load, format and display it.

6. Fill applet

Five fill types are supported. Note that fills can only be carried out on single blocks of plain colour e.g. a whole screen of one colour, a Draw object filled with a colour or connected pixels of exactly the same colour.

Angle graduated fill

Fills the area with a graduated fill using the graduations in the Current colour block. The start colour is selected with the Left mouse button and the end colour with the Right Mouse button. Plot the start point, drag out line and release. Fill direction is controlled by start and end points. The length of the line controls the rate of colour change. Click the Right mouse button to undo the last fill made.

Flat fill

Fills an area with the current colour. Click with Left Mouse Button to select the colour from the palette. Click on the canvas with the Left Mouse button to start the fill. Click the Right mouse button to undo the last fill made.

Bitmap fill

Click on this icon to open a dialogue that shows a thumbnail of the current bitmap. Use the Menu opener to view the menu and select the required bitmap. To fill an area with the tiled bitmap click on the canvas. Click the Right mouse button to undo the last fill made. You can also fill with any JPEG file by dragging it onto the load target in this dialogue box.

Circular fill

Fills the area with a circular graduated fill using the graduations in the Current colour block. The start colour is selected with the Left mouse button and the end colour with the Right Mouse button. Plot the start point, drag out line and release. Fill direction is controlled by start and end points. The length of the line controls the rate of colour change and the size of the circular fill. Click the Right mouse button to undo the last fill made.

Sprite plot

If a suitable bitmap image is on the Clipboard then this icon lets you fill with the image. The image is tiled to cover the area being filled. Click the Right mouse button to undo the last fill made

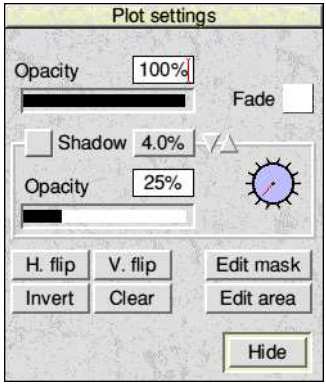
7. Stamper applet

This tool is used to cut sections of the canvas and paste them back in a different position or use the stamp as a brush. The user interface is very similar to that used by the Draw applet. It is also used, by default, when pasting a sprite or JPEG picture from the clipboard. When a stamp has been created using one of the tools, a dialogue becomes available where you can change various settings.

NOTE: These settings only take effect with 32K and 16M colour images and when the image and the stamp have the same number of colours.

Opacity: Use the slider to alter the Stamp's opacity.

Fade: You can click on the Fade button to cycle through the available fades. These are Solid, top to bottom, bottom to top, right to left, left to right and circular. The stamp is then plotted according to the fade type selected.



Shadow: You can add Shadow by ticking the box and alter the amount of shadow using the bump arrows. Holding down Ctrl or Shift will advance in larger steps.

Shadow opacity: This can be altered by dragging the slider. Shadow colour is black at 100% opacity and graduates through shades of grey as opacity is reduced. Shadow Angle: Drag the pointer on the dial to change the angle of the shadow.

Horizontal flip: This flips the stamp in the horizontal plain i.e. right to left.

Vertical flip: This flips the stamp in the vertical plain i.e. top to bottom.

Both flips can be used together.

Invert: This reverses the relationship of the stamp and the mask. A stamp is created from a rectangular object that has a transparent mask applied to it in order to provide the irregular shape. Using Invert makes the initial stamp transparent and the mask visible, so you end up with a rectangle with an irregular shape cut out of it.

Clear: This clears the area of the selected region covered by the mask to a flat colour. The colour can be changed by clicking on it and using the standard Colour Picker. You will not see the effect of this until you Invert the mask.

If you have cleared a mask and then done an Invert you can then clear the 'new' mask to give two blocks of solid colour, the stamp and the mask.

Edit mask allows you to edit the mask using any of the other available PCA tools (the mask is used to define the irregularity of the selected region).

Edit area lets you edit the selected region using any other available PCA tools.

Bézier curves

Plot start point, move and plot end of curve segment. You can now move the pointer to modify the curve. When the curve is satisfactory plot another point and do the same for that segment. At any time you can press ADJUST to modify the existing curves by dragging the anchor points around. To resume adding points, click with ADJUST again. Double click to complete the object or double click the Right Mouse button to cancel the object.

Objects will be automatically closed when the curve being added gets close to the starting point. You can then cut holes inside a closed, filled object by creating another closed object inside it.

Once an object is cut it can be used to paint with by using the Left Mouse button. A single click of the Right Mouse button returns you to editing the object and a double click of the Right Mouse button abandons the object.

Selecting irregular objects

(You may like to increase the view scale to 200% or 400% to allow for accurate placement).

Trace round the edges of the object, plotting another point whenever the straight line cuts too much of the object out. Don't worry too much where there are smooth curves, just plot from one side to the other. Where object edge complexity increases add more control points and make them closer together.

After you have traced completely around the object and closed the shape the Stamper tool will automatically switch to path edit mode (pointer changes shape). You can then drag the path control points to fine tune the shape of the path. In all cases, try to adjust the path so that it passes over the very edge of the object. Increasing the view scale even further to 800% or 16:1 will help you do this and tidy up finer points.

To cut a hole in an irregular object (for example, in the gap between a persons arm and their body) click the right mouse button to switch back to path create mode and create the 'subpath' inside the original. The area inside this path will be made transparent.

To see this more clearly, you can make Stamper display the selected area inverted by pressing the RETURN key. However, this tends to obscure the object you are interested in so press RETURN again to go back to outline.

Straight lines

Plot start point of the line segment with Left Mouse Button and then plot end point with Left Mouse Button. Continue dragging and clicking until you have the required shape. Click the Right button to edit the control points, which are draggable. Click again with the Right button to continue adding line segments with the Left button. Double click to complete the object.

Once an object is cut it can be used to paint with by using the Left Mouse button. A single click of the Right Mouse button returns you to editing the object and a double click of the Right Mouse button abandons the object.

Edit

Select this icon to edit the control points of the current stamp.

Rectangular stamp

Plot start point and drag in any direction to required size. Hold Ctrl whilst dragging to create a square. Click with SELECT to complete the object Once an object is cut it can be used to paint with by using SELECT. A single click with ADJUST returns you to editing the object; the Left Mouse button moves the position and ADJUST changes its size and shape. Holding down Ctrl creates a square. At any time double click ADJUST to abandon the object.

Elliptical stamp

Plot start point and drag, in any direction, to required size. Hold down Ctrl whilst dragging to create a circle. Click with SELECT to complete the object Once an object is cut it can be used to paint with by using SELECT. A single click of ADJUST returns you to editing the object. SELECT moves the position and ADJUST changes its size and shape. Holding down Ctrl creates a circle. At any time a double click with ADJUST to abandon the object.

Text stamp

Click this icon to open a dialogue where you can select the Font and its size. The X and Y values let you alter the aspect ratio of the font making it taller or fatter. The text is used as the shape for the Stamp. Select the font and size you wish to use (this can be changed at any point before the stamp is fixed). Now click on the canvas and type some text, which appears in white. You can drag the text around with the mouse and when it is in the correct place click SELECT to complete the stamp.

Once an object is cut it can be used to paint with by using SELECT. A single click with ADJUST lets you add more text. After editing a double click with SELECT is required to complete the stamp. At any time a double click with ADJUST abandons the object.

Cut stamp

Click on this icon to cut the currently selected area as a stamp. This duplicates the double click action

Paint

Click on this icon to use the current stamp as a brush. This is selected automatically when painting with the stamp and is for information only.

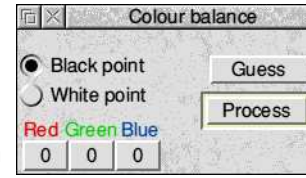
Paste from clipboard

If there is a sprite or JPEG on the clipboard then choosing this reads a copy of it and paints with it. Alternatively a sprite file may be dragged onto the Stamper tool to plot with it.

Remember that it must be in the same number of colours as the image being stamped into before special effects like opacity will work

FX applet

This tool provides features that work on the whole canvas rather than just parts of it.



Colour balance

This tool lets you modify the colour balance of the whole image by selecting a point of reference and then adjusting the image so that the point of reference appears black or white.

For example, select ‘White point’ and drag over the image to set the reference point.

As you drag, the colour of the pixel under this reference point is displayed as numeric values in the Red, Green and Blue boxes.

Colours can thus be described in numeric terms e.g. white is 255, 255, 255 (full red, full green and full blue). Conversely, black is 0, 0, 0.

Dragging to a point that is nearly white and then choosing Process sets that point to full white and adjusts the rest of the image accordingly. Try ‘Process’ with the reference point over different ‘bright spots’ in a digitised image. You will see the colour cast and brightness of the image change with different points of reference. If you choose ‘Guess’ the program will attempt to find a colour in the image as near to perfect white as possible for you. Often the scanning or digitising process introduces a slight colour cast to the image so if the best white point found was 255,255,230 there could be a slight yellow cast to the image. Processing this reference point to full white would correct it.

Alternatively, the tool can be used to introduce a slight tint or cast to the image by your deliberately choosing a point that is not quite white in a balanced image and processing based on that point.

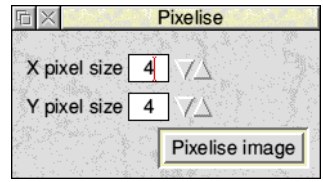
The Black point selection is similar but tends to darken the image instead of lightening. Drag the reference point to shadows and dark areas for best results.

Clear

Clears the screen to the selected colour. Click on the colour block to choose a new colour.

Filter

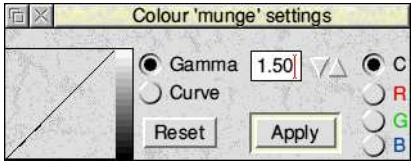
Applies the selected filter to the canvas. The filter can be applied to all colour channels or those selected by the RGB buttons.



Pixelise

Pixelises the canvas by increasing the pixel size. The pixel size is selected using the bump arrows. Try it to see the effect, you can always use Undo.

Colour munge



This is similar to the Colour Munge tool in the Painting applet and provides Gamma correction and Colour curve correction to the whole image.

To do Gamma correction select the Gamma button and drag the curve to the required position or drag over the graduated bar to the right of the curve box. You can also edit the numeric setting manually or by using the bump arrows. You can edit Combined colours or Red, Green and Blue individually. When modifying the image, the Combined settings is taken into account.

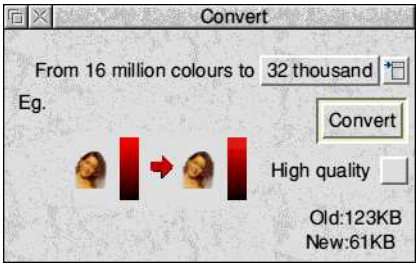
Colour Curve is chosen by selecting the Curve button. There are 4 points on the line and these can all be dragged independently. The graduated bar indicates the effect that the tool will have.

These tools are useful for enhancing photographs, especially if they are a little dark or light.

Convert applet

This tool enables you to convert an image from one colour mode into any other mode. This can be done using a ‘quick and dirty’ method or a high quality option can be selected. The change in file size is also indicated.

The current number of colours is shown and you use the menu opener to select from the available options. Select the High Quality option if required and click on Convert. The Undo button in the main toolbar can be used to reverse this process, but only until you select another tool.



Note that if you convert to a smaller number of colours or greys and then try to convert back to a larger number of colours, you will not get back to the original image. For example, from 32k colours to 256 greys and then 256 greys to 32k colours will result in a 32k image but it will be grey.

You should also be aware that some applications, Compo for example, only allow a limited range of colour depths.

Resize applet

This provides tools that transform the whole image in terms of size, crop and rotation.

Crop

This tool lets you cut an area from the canvas or enlarge the area of the canvas. You can use the bump arrows on the dialogue to increase the canvas area - the size is indicated in pixels in the ‘New’ line of the dialogue.

You can reduce the canvas size by clicking on the bump arrows with ADJUST. The effect of this is visible on the thumbnail image.

To define the area for cropping you can drag a box over the required area on the thumbnail or on the main image itself. The old and new values are shown. NOTE: Cropping is not the same as Scaling. Cropping means things stay the same scale but part of the image is thrown away.

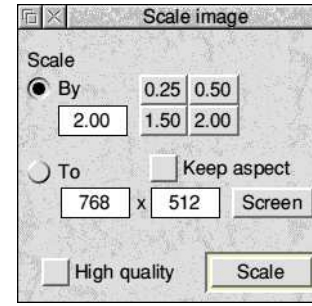
Scale

The image can be scaled using preset ratios or user applied ratios when the ’By’ button is selected. An option to retain aspect ratio is also supplied.

Scaling can be ‘quick and dirty’ or a ‘High quality’ option can be selected.

The ‘To’ option enables you to set a size in pixels. Clicking on the ‘Screen’ button reads the screen size and puts it in the boxes.

Click on the ‘Scale’ button to carry out the scale. Undo will reverse the scaling.



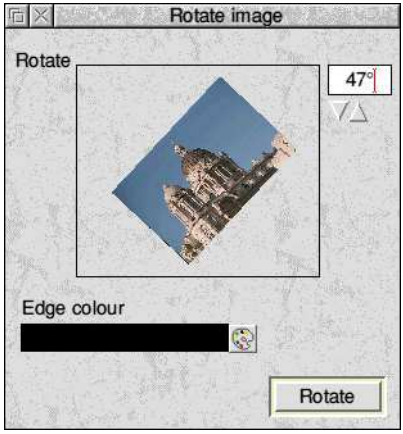
Rotate

This tool allows you to rotate the canvas by setting a value, in degrees, for rotation. The value can be entered manually or using the bump arrows. Alternatively, you can drag the image in the Rotate window. The canvas is enlarged to accommodate the rotated image.

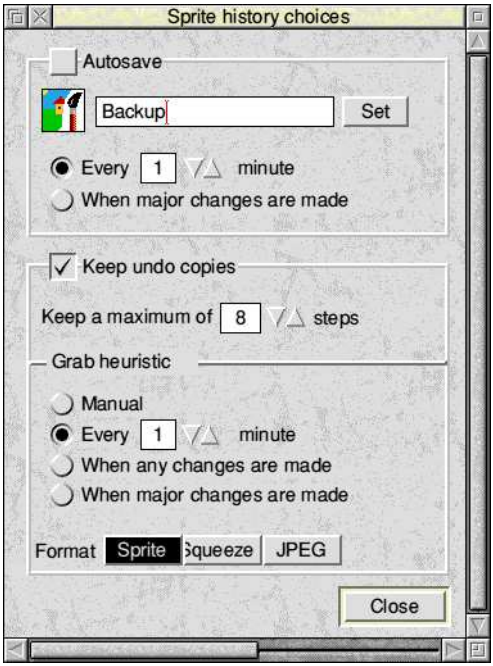
When you rotate the image there may be blank areas, so an Edge colour can be selected for use as the background after rotation. The colour can be selected from the palette or the 'Previous' palette using the Left Mouse button. Or you can use the Colour Picker available from the Rotate dialogue window.

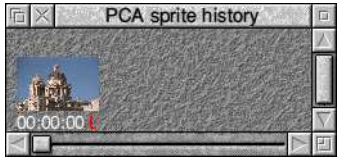
If you want to match a colour exactly, the dropper can be used to select a colour, which you can then drop into the 'Previous' palette. You can then select the Edge colour from the Previous palette.

Click the Rotate button to rotate the image.



Sprite history applet





This applet has not been finished.