Sprite Plotting and Screen Display Independence on RISC OS

This document is an attempt to bring together previously disparate pieces of information concerning the rendering of sprites and use of 8-bit colour modes on RISC OS, particularly on the Raspberry Pi. My main reason for doing this, rose from the frustration of not having a single resource in the production of software such as games for the platform.

Here you will find extracts and/or weblinks from publicly available documentation on the use of commands such as `SYS OS SpriteOp`, and the `ColourTrans` module, and how to produce colour translation tables, in rendering 8-bit (256 colours) on modern graphics hardware. RISC OS was developed long before the advent of high resolution colour displays.

My hope in preparing this document is that it is much easier for new - as well as former developers of the platform - to prepare, revisit and revise their code for release on newer platforms, such as the Raspberry Pi.

1. Sprite Handling

Extracts from PRM 1-777 - Sprites - Technical details

The RISC OS Programmer's Reference Manual is now more widely available, and contains details on the use of sprites. I've provided links to the two main sources:

RISC OS Limited - Programmers Reference Manual
RISC OS PRMs: Overview of chapters

RISC OS Open - Programmers Reference Manual
https://www.riscosopen.org/wiki/documentation/show/Programmer's%20Reference%20Manuals

These will be designated as 'ROL' or 'ROOL' accordingly.

SYS OS SpriteOp

`SYS OS SpriteOp` (ROL/ROOL) is a general purpose command for sprite handling on RISC OS.

Sprite modes

To make your sprites readable by old releases of RISC OS, we recommend you use the following mode numbers in a sprite (see the section entitled Format of a sprite on page 1-777, ROL/ROOL):

	2 colours	4 colours	16 colours	256 colours
2 x 4 OS unit pixels	0	8	12	15
4 x 4 OS unit pixels	4	1	9	13
2 x 2 OS unit pixels	18	19	20	21

Scale factors

The scale factor will change the size of a sprite. It is a pointer to a block of four words with the following elements:

Offset	Meaning
0	x multiplier
4	y multiplier
8	x divisor
12	y divisor

The size of the specified sprite on the screen when it has been plotted in pixels (not OS units), is multiplied by the multiplier and divided by the divisor, ie:

x pixel size = x start size (in pixels) * x multiplier / x divisor
y pixel size = y start size (in pixels) * y multiplier / y divisor

If the plot action is using an ECF pattern, then the pattern will not be scaled up with the sprite. This is so that the patterning will be correct when used with a large scale factor. See the section entitled ECF patterns on page 1-552 for a description of ECF patterns (ROL/ROOL).

If the pointer is zero, then no scaling is performed; ie 1:1 scale.

The Wimp uses a similar system to provide mode independence. For details see `Wimp_ReadPixTrans` (ROL/ROOL) on page 3-205.

Pixel translation table

This allows a logical colour to be substituted for each colour in the sprite. It is a pointer to a table of bytes. The number of bytes in the table depends on the number of colours in the mode in which the sprite was created.

A pixel of colour n in the sprite will be translated to the nth entry in the pixel translation table. The first entry in the table is at offset 0 (ie the 0th colour). So colour 3 in a pixel will get the value 3 bytes into the table and use that as its logical colour.

If the pointer is zero, then the colours in the sprite will be used. However, if the destination bits per pixel is less than the source bits per pixel, you will get an error.

The Wimp uses a similar system to provide mode independence. For details see `Wimp_SetPalette`, `Wimp_ReadPalette`, and `Wimp_SetColour` from page 3-187 onwards.

The `ColourTrans` module provides facilities for translation table calculations. For more information refer to the chapter entitled `ColourTrans` on page 3-333, or via the RISC OS Open website.

Developers' Newsletter No. 19 - August 1990:
http://acorn.chriswhy.co.uk/docs/Acorn/DN/Acorn_DevNL19.pdf

Effective use of SpriteOp 52 (put sprite scaled)

Many programs use put sprite scaled as their sole method of displaying a sprite on the screen, surrounding it by appropriate calls of `ColourTrans` for a pixel translation table in order to be screen mode independent. All well and good, but this can result in a slower display of the information than necessary, since `SpriteOp 52` does not detect a pixel translation table that does nothing. Thus the application program must do it. The following BASIC code provides a pixel translation table pointer which will be fast whenever possible:

```
FOR I% = 0 TO 255 : PixTrans%?I% = I% : NEXT
SYS "ColourTrans_SelectTable",mode,sourcepal,destpal,PixTrans%
xPixTrans%=-1
FOR I% = 0 TO 255 : IF PixTrans%?I% o I%xPixTrans%=PixTrans%
NEXT
```

Then use `xPixTrans%` in every call to `SpriteOp 52`:

```
SYS "OS_SpriteOp",52,block,sprite,x,y,action,scale,xPixTrans%
```

This results in much quicker display of sprites where the sprite has the same palette as the current screen mode.

256 entry palettes in the RISC OS sprite format

http://acorn.riscos.com/riscos3/37/37DiscImage/Utilities/!ChangeFSI/Documents/256sprites

Wimp.Colours

http://www.chiark.greenend.org.uk/~theom/riscos/docs/RISCOS2/Wimp/Colours.txt

How to display a sprite (a) mode independently and (b) quickly

http://www.chiark.greenend.org.uk/~theom/riscos/docs/ultimate/a252spr1.txt

The key to doing this is to observe that the pixmap compiler which is driven by the put_sprite_scaled call (`OS_SpriteOp 52`) can do several clever things:

a. omit colour remapping
b. omit x and/or y size remapping
c. omit itself entirely and call put_sprite (`OS_SpriteOp 34`) (only when a) and b) completely omitted)

BUT it can only do these things if given some helpful hints.

In order to provoke action a), you need to tell it that the pixel translation table should not be used - this can be done in the 1-1 map situation by using code such as:

```
FOR I% = 0 TO 255 : PixTrans%?I% = I% : NEXT
... code as above
```

and then using `xPixTrans%` as the pixel translation table (r7) value for every call to SpriteOp 52.

```
SYS "OS_SpriteOp",52,block,sprite,x,y,action,scale,xPixTrans%
```

When `xPixTrans%` is -1 the pixel translation step will be omitted and this at least doubles the speed of the sprite plot.

Action b) comes from correctly specified scale factors: reduce them to the lowest terms and its happy.

Then action c) happens "as if by magic" and your sprites are plotted faster when the wind is behind them. And in any mode when it isn't. Action c) can be sped up by calling `SpriteOp 34` yourself having recognised that `SpriteOp 52` will be doing this - this avoids `SpriteOp 52` making several read mode variable calls.

Sophie Wilson

http://comp.sys.acorn.programmer.narkive.com/rnIDWszu/problem-converting-colours-into-gcol

Acorn Arcade Thread

http://www.acornarcade.com/forums/viewthread.php?threadid=1343

ColourTrans and GCOL

http://pl.digipedia.org/usenet/thread/1614/1292/

RISC OS Colour Format Converters

http://starfighter.acornarcade.com/convert/index.html

Gradients in the RISC OS 8-bit palette

http://www.starfighter.acornarcade.com/mysite/articles/gradr.html

RISC OS Select Documentation - Colour Mapping of Deep Sprites

http://select.riscos.com/prm/graphics/sprites/colourmapping.html

2. Resolution (mode) independence

Use "`MODE MODE`" at the start of your program to clear the current screen. There is no need to specify an actual mode number.

RISC OS Select Display Changes

http://select.riscos.com/prm/graphics/multidrivers.html

RISC OS Open Forum Thread: Should I be using MODE?

http://www.riscosopen.org/forum/forums/5/topics/1454

RISC OS 3.60 Release note - using MODE in BASIC.

http://acorn.chriswhy.co.uk/docs/Acorn/Tech/Acorn_RISCOS3Vsn360_RelNote.pdf

In BASIC V, the `MODE` command has been extended. Where before it used to take a mode number, now it can also take a pointer to a mode selector or a mode string. A mode selector is passed to `OS_ScreenMode`, whereas a mode string is passed to `*WimpMode`. The result is that if a program changes screen mode using a mode string, e.g. `MODE "X800 Y600 C256 EX 1 EY1"`, then when the program finishes that will be your desktop mode.

It should also be noted that the C/G option in the mode string is implemented by `*WimpMode`. Hence, `MODE MODE` will select a default palette mode for the given current mode depth. This means that if a grey mode is selected by `MODE "X800 Y600 G256 EX1 EX I"` then `MODE MODE` will revert back to the coloured palette.

comp.sys.acorn.programmer thread - Basic & Screenmodes:
https://groups.google.com/d/topic/comp.sys.acorn.programmer/cU6Vf7iGcVI/discussion

comp.sys.acorn.programmer thread - Changing screen mode
https://groups.google.com/d/topic/comp.sys.acorn.programmer/pQv1wBdSMsc/discussion

csa.programmer - Basic MODE question
http://compgroups.net/comp.sys.acorn.programmer/basic-mode-question/1304329

My thread on 'Getting rid of palette files for 16 colour game'
https://groups.google.com/d/topic/comp.sys.acorn.programmer/fjYq4f8fBrc/discussion

Wrong colour select when output is to a sprite
https://groups.google.com/d/topic/comp.sys.acorn.programmer/5WyjRyBryg4/discussion

porting old game
https://groups.google.com/d/topic/comp.sys.acorn.programmer/0Qj03slHVXM/discussion

Tile game help
https://groups.google.com/d/topic/comp.sys.acorn.programmer/JoQwN-41rBw/discussion

BASIC viewport problem
https://groups.google.com/d/topic/comp.sys.acorn.programmer/mFlUYMI_LXI/discussion

The csa.programmer usenet group is a gold mine of information, I'll hone down these links as and when time allows.

3. Screen Swapping

Also known as screen banking, or bank switching. RISC OS Pi RC7, released on 12th March 2013 contains some support for screen banking:

https://www.riscosopen.org/content/downloads/changes-for-bcm2835-rc6-to-rc7#mixed_RiscOS_Sources_Video_HWSupport_BCMVideo

According to the detail, a new system variable has been added, called `BCMVideo$ScreenBanksEnabled`:

> "Added a `BCMVideo$ScreenBanksEnabled` system variable. If set to '0' it will disable screen banks on the next mode change. Any other value (including unset) enables them. This is a (potentially temporary) compatability option for use with old games; the fact that switching screen bank requires the driver to wait until the next VSync (while waiting for the GPU response) means that having screen banks enabled may have a significant

framerate impact.”

The implication is that while such code is being added to support the manner in which old games perform their animation, because of the much higher resolution displays available, that this technique cannot be sustained in future, and other techniques must be used to achieve flicker free animation.

About Bank Switching and Screen Access

http://www.tofla.iconbar.com/tofla/arm/arm01/index.htm

RISC OS Open Thread: Screen Buffers on the Pi

https://www.riscosopen.org/forum/forums/5/topics/1494

This is an ongoing thread. At the time of writing, this thread has some interesting discussion and sample code that can be run on the Pi, as implementation of bank switching was being developed. The results of this are available from RC7 onwards.

The Icon Bar: Forum Thread ‘Double screen banking’

http://www.iconbar.com/forums/viewthread.php?threadid=11942

Appendix: Miscellaneous Research

Justin Fletcher’s RISC OS Rambles

Since December 2012, Justin has been publishing daily accounts of the work he performed at RISC OS Limited, where he worked for a number of years. The articles cover features of the OS in great detail, including sections relevant to screen display and sprite handling. This is an invaluable resource.
http://www.gerph.org/riscos/
http://www.gerph.org/riscos/ramble/graphics-core2.html (screen banking, legacy modes)
http://www.gerph.org/riscos/ramble/graphics-sprites.html (sprite handling)

About sprite plotting, with access to zipfiles of demonstration software

http://www.acornusers.org/cbsa/oldversion/Art.html
http://www.tofla.iconbar.com/tofla/gfx/spr01/index.htm

Access to PRMs online

RISC OS Limited - Programmers Reference Manual
http://www.riscos.com/support/developers/prm_index/prmindex.html

RISC OS Open - Programmers Reference Manual

https://www.riscosopen.org/wiki/documentation/show/Programmer's%20Reference%20Manuals

Google search for risc os PixTrans

https://www.google.co.uk/search?q=PixTrans%25%3FI%25&rlz=1C1SKPC_en-GBGB483GB483&oq=PixTrans%25%3FI%25&aqs=chrome.0.57.1748&sugexp=chrome,mod=13&sourceid=chrome&ie=UTF-8#q=risc+os+PixTrans%25%3FI%25&hl=en&tbo=d&rlz=1C1SKPC_en-GBGB483GB483&ei=TO-4UJbpIYfzsgan_YCoCw&start=0&sa=N&bav=on.2,or.r_gc.r_pw.r_cp.r_qf.&fp=e528a054a160ed8a&bpcl=39314241&biw=1280&bih=895

Google search for risc os spriteop colourtrans

https://www.google.co.uk/search?q=risc+os+spriteop+colourtrans&rlz=1C1SKPC_en-GBGB483GB483&oq=risc+os+spriteop+colourtrans&aqs=chrome.0.57j62j64.10488&sugexp=chrome,mod=13&sourceid=chrome&ie=UTF-8#q=risc+os+spriteop+colourtrans&hl=en&tbo=d&rlz=1C1SKPC_en-GBGB483GB483&ei=g6e4UICWJofysgaY5IH4BA&start=0&sa=N&bav=on.2,or.r_gc.r_pw.r_cp.r_qf.&fp=e528a054a160ed8a&bpcl=39314241&biw=1280&bih=895

Stephen Scott, November-December 2012, March, December 2013
www.sassquad.com/riscos