Screen Effects

IMPORTANT NOTE: color rotation and animation and copper effects all depend on the original hardware of the Amiga. AOZ has to emulate the Copper and display chip to enable these features. This process is not yet implemented (version 13/05/2019).

This emulation will necessitate an analysis of everything that chains into each screen, and may be a long process for large (modern sized) screens. This is the reason why palette animation and fades will only be available in Amiga hardware emulation mode, where the screen is limited in size...

The AOZ programmer expects to achieve superb visual effects with simple, economic commands. Classic cinematic and video techniques are readily available, as well as more spectacular routines that are only made possible by the power of the computer.

When the image of one screen dissolves and melts into the image of another screen, various "fade" effects are produced.

APPEAR

instruction: Fade between two screens

```
Appear source screen To destination screen, pixels
Appear source screen To destination screen, number pixels, range
```

This command creates a fade between two pictures. Choose the number of the source screen where the original picture comes from, then the number of the destination screen whose picture it fades into. LOGIC and PHYSIC functions can be substituted for screen numbers, if required.

Next determine a value that will cause the desired effect, by setting the number of pixel points on the screen, ranging from 1 pixel all the way up to every pixel in the display.

Normally APPEAR affects the whole of your screen area, but there is an optional parameter that causes only part of the screen to be faded. Because screens are drawn from top to bottom, set the area to be faded by adding the range of the number of pixels from the top of the screen. For example:

TODO - "Change 'Load' address below to AOZ_Tutorial:objects etc"

```
Load "AMOSPro_Tutorial:Objects/Bobs.Abk"
Flash Off : Get Bob Palette
Paste Bob 100,0,1
Wait 100
Screen Open 1,320,90,16,Lowres
Flash Off : Get Bob Palette
Appear 0 To 1,1,28800
```

That example fades the top part of your default screen into the newly opened Screen 1. Obviously, the appearance of fades will vary, depending on the screen mode being used.

FADE

instruction: Blend colours to new values

```
Fade speed
Fade speed,colour list
Fade speed To screen number
Fade speed To screen number,mask
```

The classic "fade to black" movie effect takes the current palette and gradually fades all values to zero. Set the speed of the fade by choosing the number of vertical blank periods between each colour change. Try this:

```
Flash Off : Curs Off
Centre "GOOD NIGHT"
Fade 5
```

Fade effects are executed using interrupts, so it is sensible to wait until the fade has ended before going on to the next program instruction. The length of wait required can be calculated with this formula:

```
wait = fade speed * 15
```

So that last example is sure to work with the rest of your program if the third line is changed to this:

```
Fade 5 : Wait 75
```

By adding a list of colour values, the fade effect will generate a new palette directly from your list, and it is used like this:

```
Flash Off : Curs Off
Centre "RED SKY AT NIGHT"
Fade 10,$100,$F00,$300
Wait 150
```

Any number of new colours can be set up like this, depending on the maximum number allowed in your current screen mode. Any settings that are omitted will leave those colours completely unaffected by the fade, as long as you include the right number of commas. For example:

```
Fade 5,,$100,,,$200,$300
```

There is an even more powerful use of the FADE command, which takes the palette from another screen and fades it into the colours of the current screen. Set the speed of the fade as usual, then give the number of the screen whose palette is to be accessed. By using a negative number instead of a screen number, the palette from the sprite bank will be loaded instead.

There is one more parameter that can be added, and this creates a mask that only permits certain colours to be faded in. Each colour is associated with a single bit in the pattern, numbered from 0 to 15, and any bit that is set to 1 will be affected by a colour change. For example:

TODO "Change Address below 'AMOSPro_Tutorial' etc"

```
Load "AMOSPro_Tutorial:Objects/Bobs.Abk"
Screen Open 1,320,90,16,Lowres
Flash Off : Get Object Palette
Paste Bob 100,0,1
Wait 100
Fade 1 To 0,%01111000011001010
Wait 15
```

Flashing colours

You will already be aware that colour index number 3 is pre-set to flash on and off, and is the default setting for the text cursor. By using interrupts, any colour index can be cycled through a series of colour changes, producing complex flashing effects.

FLASH

instruction: Set flashing colour sequence

```
Flash index number,"(colour,delay)(colour,delay)(colour,delay)..."
Flash Off
```

When FLASH is followed by the index number of any colour, that colour will display animated flashing every time it is used, until FLASH OFF is called. Up to 16 colours can be cycled to customise your flashing effects, and the rate of delay from one colour change to the next can be individually set. Try this:

```
Flash 1 ,"(0A0,10)(F0F,40)"
```

In that example, the colour to be affected is set to index number 1. After the comma, the set of quotation marks can contain up to 16 pairs of brackets, and each pair of brackets contains the colour that is next on the list to be flashed, and the time it will appear for. Colour is set in RGB component values, which are fully explained in the next Chapter. Delay time is set in units of a 50th of a second. So the last example has the effect of flashing colour number 1 between a green value and a violet value once every second. The next example is more subtle:

```
Cls : Centre "SILENT MOVIES"
Flash 1,"(111,4)(333,4)(555,4)(777,4)(555,7)(333,7)
Curs Off : Wait 250 : Flash Off
```

SHIFT UP

instruction: Rotate colour values upwards

```
Shift Up delay,first,last,flag
```

This command takes the values held in the colour registers and shunts them forwards. The delay between colour shifts is set in 50ths of a second, similarly to the FADE command.

Next the values of the colours to be affected are set, from the first colour to the last colour in the sequence. The first colour in the list will be copied to the second, the second to the third, and so on until the last colour in the series is reached.

Finally, a flag is set to either 0 or 1. When this flag is set to zero, the last colour is discarded, and the rotation will cycle for the number of times it takes to replace all colours with the first colour in the list. Alternatively, if the flag is set to one, the last colour index in the list is copied into the first, causing the colours to rotate continuously on screen.

Each of your screens can have its own set of animated colour rotations, and because they are executed using interrupts they will not affect the running of your programs.

SHIFT DOWN

instruction: Rotate colour values downwards

```
Shift Down delay,first,last,flag
```

This command is identical to SHIFT UP, except for the fact that colours are rotated in the opposite direction, so that the second colour is copied into the first, the third to the second, and so on. With the final flag set to zero, all colours are eventually replaced with the last colour in the list.

SHIFT OFF

instruction: Turn off all colour shifts for current screen

```
Shift Off
```

Use this command to terminate all colour rotations previously set by the SHIFT UP and SHIFT DOWN instructions.

Rainbow effects

So far, most of the screen effects in this Chapter take a colour index and change its value over a set period of time. AOZ offers an alternative system, where colour indexes are changed depending on specific screen locations. This means that a single colour index can be used to generate hundreds of colours in some spectacular rainbow effects. Before any rainbows can be conjured up, their parameters must first be set.

SET RAINBOW

instruction: Define a rainbow

```
Set Rainbow number,index,height,red$,green$,blue$
```

Try the next example before analysing how it works:

```
Set Rainbow 0,1,16,"(1,1,15)","",""
Rainbow 0,56,1,255
Curs Off : Flash Off
Locate ,12 : Centre "RED STRIPE"
```

Up to four different rainbows may be set up for later use, and SET RAINBOW is followed by an identification number for this rainbow, from 0 to 3.

The next parameter is the colour index that is to be changed, and only colours 0 to 15 can be affected. In practice, this colour can be assigned a different value for each horizontal screen scan line, if necessary.

Following this, the height parameter sets the size of the table to be used for colour storage, in other words, it sets the height of the rainbow in units, with each unit ready to hold one scan line of colour. The size of this table can range from 16 to 65500, but only the first 280 or so lines can be displayed on screen at once. So if your table is less than the physical height of your rainbow, the colour pattern will be repeated on the screen.

Finally, the Red, Blue and Green components of the rainbow colours are set up as strings, each within their own brackets. The last example leaves out any reference to the Green and Blue components, which is why the resulting effect is completely in the red. These strings will be 06.03.04Screen Effects cycled to produce the final rainbow pattern, and their format comprises three values contained in each relevant pair of brackets, as follows:

(number,step,count)

- Number refers to the number of scan lines assigned to one colour value. Think of it as controlling the "speed" of the sequence.
- Step is a value to be added to the colour, which controls the colour change. Count is simply the number of times this whole process is performed.

RAINBOW

instruction: Display a rainbow

```
Rainbow number,offset,vertical position,height
```

The last example also demonstrates the parameters of the RAINBOW command, which is used to display one of the rainbows created with SET RAINBOW.

The rainbow number is obvious, and refers to one of the four possible rainbow patterns from 0 to 3. The offset sets the value for the first colour in the table created with SET RAINBOW, and it governs the
cycling or repetition of the rainbow on screen.

The vertical position is a coordinate which must have a minimum value of 40, and it affects the starting point of the rainbow's vertical display on screen. If a lower coordinate is used, the rainbow will be displayed from line number 40 onwards.

Finally, the height number sets the rainbow's vertical height in screen scan lines.

Please note that normally only one rainbow at a time can be displayed at a particular scan line, and the one with the lowest identification number will be drawn in front of any others However, experienced Amiga users will be able to start more than one rainbow at the same line, using the Copper. See Appendix F for an explanation of this technique.

RAINBOW DEL

instruction: Delete a rainbow

```
Rainbow Del
Rainbow Del number
```

Use this command on its own to get rid of all rainbows that have been set up. If a rainbow identity number is added, then only that particular rainbow will be deleted.

RAIN

function: Change the colour of a rainbow line

```
Rain(number,line)=colour
```

This powerful rainbow instruction allows you to change the colour of any rainbow line to the value you choose. RAIN is followed by a pair of brackets containing the number of the rainbow to be changed and the scan line number that is to be affected.

The next example demonstrates the following technique. Rainbow number 1, with colour index 1, is given a colour table length of 4097, which is one entry for every colour value that will be displayed on screen. The RGB values are left blank, to be set up by the first FOR ... NEXT routine, that contains the RAIN command. The second FOR ... NEXT routine uses RAINBOW to display a pattern 255 lines long, starting at scan line 40. The DO ... LOOP structure is used to repeat the process.

```
Curs Off : Centre "OVER THE RAINBOW"
Set Rainbow 1,1,4097,"","",""
For L=0 To 4095
    Rain(1,L)=L
Next L
Do
    For 0=0 To 4095-255 Step 4
        Rainbow 1,C,40,255
        Wait Vbl
    Next C
Loop

## The copper list

All the Copper instructions will only be available in Amiga hardware emulation
mode.

The appearance of every line displayed on your screen is controlled by the
Amiga's co- processor, known as the "copper". The copper is a self-contained
processor with its own special set of instructions, and its own internal memory.
A massive number of special effects can be created by programming the copper,
but the copper list is notoriously difficult to manipulate, and many competent
programmers have failed to master its mysteries.

A full discussion of the copper lists may be found in Appendix XX.X (F) of this
User Guide.
```