Graphics

In this Chapter, you will learn how to master the arts of form and colour.

AOZ allows the programmer to harness the computer full graphic potential, and all aspects of design
can be controlled simply, accurately and almost instantaneously.

Graphic coordinates

PLOT

instruction : Plot a single point

```
Plot x,y
Plot x,y,colour
```

This is the simplest drawing command of all, by plotting a single pixel of ink colour between graphic coordinates 0,0 and the width and height of the screen. When followed by specific x,y-coordinates, the current ink colour will be plotted at this new position.

You are allowed to omit either the x or the y- coordinate, provided the comma is left in the correct position. If an optional colour index number is added, the new colour will be used for this and all subsequent drawing operations. For example:

```
Cls: Curs Off
Do
    Plot Rnd(319),Rnd(199),Rnd(15)
Loop
```

POINT

function: Return the colour of a point

```
c=Point(x,y)
```

Use this function to find the index number of the colour occupying your chosen coordinates, like this:

```
Cls : Plot 160,100
Print "The colour is ";Point(160,100)
```

Setting the graphics cursor

GR LOCATE

instruction: Position the graphics cursor

```
Gr Locate x,y
```

The graphics cursor sets the starting point for most drawing operations. To establish this point, use GR LOCATE to position the graphics cursor at your chosen coordinates. For example:

```
X=150 : Y=10
For R=3 To 87 Step 3
    Gr Locate X,Y+R
    Circle ,,R
Next R
```

XGR / YGR

functions: Return the relevant coordinate of the graphics cursor

```
x=Xgr
y=Ygr
```

Use these functions to find the current coordinates of the graphics cursor, which is the default location for future drawing operations. For example:

```
Cls : Circle 100,100,50
Print Xgr,Ygr
```

Drawing lines

DRAW

instruction: Draw a line

```
Draw x1 ,y1 To x2,y2
Draw To x,y
```

Line drawing is extremely simple. Pick two sets of graphic coordinates, and draw your line from one to the other. To draw a line from the current position of the graphics cursor, use DRAW TO followed by a single set of coordinates. For example:

```
Cls: Ink 2
Draw 50,50 To 250,150
Draw To 275,175
```

Line styles

Changing the appearance of straight lines is very simple with AOZ. Each line pattern is held in the form of a binary number made up of 16 bits, with 'zeros' setting blank spaces in the current background colour, and 'ones' setting the solid parts of the pattern in the current ink colour. So a normal solid line can be imagined as having all its bits set to one, like this:

```
%01111111111111111
```

SET LINE

instruction: Set a line style

```
Set Line binary mask
```

This command sets the style of all straight lines that are subsequently drawn. Theoretically, the 16-bit mask can generate values for different patterns between 0 and 65535, but here is a more practical example:

```
Cls : Ink 2
Set Line $F0F0
Box 50,100 To 150,140
Set Line %1100110011001100
Box 60,110 To 160,160
```

Drawing outline shapes

Here is a range of AOZ short-cuts for drawing outline shapes on the screen.

POLYLINE

instruction: Draw multiple line

```
Polyline x1 ,y1 To x2,y2 To x3,y3
Polyline To x1 ,y1 To x2,y2
```

The POLYLINE is identical to DRAW except that it accepts multiple coordinate settings at the same time. In this way, complex 'many-sided' outlines can be drawn with a single statement. In its POLYLINE TO form, drawing begins at the current graphic cursor position. For example:

```
Circle 160,100,95
Polyline 160,6 To 100,173 To 250,69 To 71,69 To 222,173 To 160,6
```

BOX

instruction: Draw a rectangular outline

```
Box x1 ,y1 To x2,y2
```

Boxed outlines are drawn at settings determined by the top left-hand and bottom right-hand coordinates, as in the last example.

CIRCLE

instruction: Draw a circular outline

```
Circle x,y,radius
```

To draw circles, a pair of coordinates sets the position of the centre point around which the shape is to be drawn, followed by the radius of the circle (the distance between the centre point and the circumference or rim of the circle.) If the x,y-coordinates are omitted, the circle will be drawn from the current graphic cursor position. For example:

```
Cls : Curs Off : Ink 3
Gr Locate 160,100
Circle ,,45 : Wait 100: Flash Off
Do
    Ink Rnd(15) : X=Rnd(250) : Y=Rnd(150) : R=Rnd(90)+1
    Circle X,Y,R
Loop
```

ELLIPSE

instruction: Draw an elliptical outline

```
Ellipse x,y,radius1,radius2
```

An ellipse is drawn in a similar way. After the x,y-coordinates have set the centre location, two radii must be given, one to set the horizontal width and the second to set the height of the ellipse. Coordinates may be omitted as usual, providing the commas remain in place. For example:

```
Ellipse 100,100,50,20
Ellipse ,,20,50
```

CLIP

instruction: Restrict drawing to a limited screen area

```
Clip Clip x1 ,y1 To x2,y2
```

This command is used to set an invisible rectangular boundary on the screen, using the normal top left-hand corner to bottom right-hand corner coordinates. All subsequent drawing operations will be clipped off when they reach these boundaries. To toggle the command and restore the normal screen display area, use CLIP and omit the coordinates. Areas that are preserved outside of the clipped zone can be used for items such as borders and control panels. For example:

```
Clip 150,5 To 280,199
For R=4 To 96 Step 4
    Gr Locate 150,R+5
    Ellipse ,,R+9,R
Next R
```

Selecting colours

In Amiga hardware emulaton mode, AOZ allows the use of colour numbers ranging from 0 to 63. This is in order to make full use of the extra colours available from the Half-Bright and HAM modes, as explained in Chapter XX.X (6.1).

The number of colors is non relevant in "PC" mode.

INK

instruction: Set drawing colour

```
Ink number
Ink number,pattern,border
```

You are not restricted to the pre-set colours that have been allocated for drawing operations. The INK command is used to specify which colour is to be used for subsequent drawing, and the number of the colour register is set like this:

```
Cls: Ink 5
Draw To 319,199
```

The INK instruction can also be used to set patterns for filling shapes, as well as colours for borders around shapes, and this will be explained later. The next concept to understand is how different colours are mixed.

Every shade of colour displayed on your television set or monitor is composed of various mixtures of the same three primary colours: Red, Green and Blue (RGB for short). There is a range of 16 intensities available for each of the RGB levels in every colour. A zero level is equivalent to "none" of that colour (black), and the maximum intensity of 16 is the equivalent of "all" of that colour. Because there are three separate components each with 16 possible strengths, the maximum range of available shades is 16 times 16 times 16, in other words 4096 possible colours.

Colors are defined by their RGB components, given in hexadecimal values, known as "hex".

The following table shows the equivalent decimal and hex values for the 16 numbers involved:
Hex digit 0 1 2 3 4 5 6 7 8 9 A B C D E F
Decimal 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

In Amiga hardware emulation mode, colors are encoded on 12 bits, with colors like $ABC. In "PC" mode, colors are true RGB on 3x8 bits.

COLOUR

function: Read the colour assignment

```
c=Colour(index)
```

It is not difficult to find which colours are occupying the colour index, and analyse how much Red, Green and Blue is used to make up each shade. The COLOUR function can take an index number from 0 to 31, and returns the colour value assigned to that number. Hex$ is used for this purpose, as follows:

```
Curs Off : Flash Off
For C=0 To 15: Ink C
    Print Hex$(Colour(C),3)
    Circle 160,75,(C+1)*4
Next C
```

That example creates a list of 16 colour values in hex code, alongside a ripple of circles in those colours. Note that the $ symbol is always used to introduce a hex number for the Amiga to recognise. The first hex value in the example table should be $000, meaning no Red, no Green and no Blue component is present in colour index 0. Sure enough, the innermost circle is drawn in black ink.

Here are some other examples in this form of notation:

Colour - Hex value - RGB components
White - $FFF - R=F G=F B=F
Grey - $666 - R=6 G=6 B=6
Green - $0F0 - R=0 G=F B=0
Violet - $FOF - R=F G=0 B=F
Ox blood - $801 - R=8 G=0 B=1
Pig foot - $A74 - R=A G=7 B=4

COLOUR

instruction: assign a colour to an index

```
Colour number,$RGB
```

Used as an instruction, COLOUR allows you to assign the RGB components of a colour to each of the colour registers. For example, if you wanted to load colour number 1 with a subtle shade of pig's feet, you would use this:

```
Cls : Colour 1,$A74
```

COLOUR BACK

instruction: assign a colour to the screen background

```
Colour Back $RGB
Colour Back (number)
```

This command is used to assign your choice of RGB components for the screen's background colour, which fills unused areas at the top and bottom of the visible screen. Alternatively, existing colours may also be specified when enclosed in brackets.

Setting several colours

Impressive effects can be programmed using multi-colour changes, but assigning individual colours to every colour index would be a tedious business. AOZ handles all the donkey work as usual.

PALETTE

instruction: Set the current screen colours

```
Palette colour list
```

This is a much more powerful command than COLOUR, and it can be used to set as few or as many colours in your artist's palette as are needed. Your programs always begin using a list of default colour values, and these values may be changed as in the next example.

Remember that only the colours specifically set with this command will be affected, and any others will retain their original values.

```
Palette $FFF : Rem set colour 0 to white
Palette ,,,,$F00,$D00,$A00,$700,$400 : Rem colours 4 to 8 graded reds
Palette $000,,$000: Rem colours 0 and 2 both black
```

PALETTE can also be used to set the colours used by the Half-Bright and HAM modes, and some superb readymade examples are available, care of Chapter XX.X (6.1). For a little light relief, try his routine, which changes the first five colours in the palette with a hexadecimal poem, and displays the result on screen. Feel free to change the values or the poetry.

```
Palette $BAD,$0DD,$B0D,$FAB,$F1B
Curs Off : Flash Off
For C=0 To 4: Ink C
    Print Hex$(Colour(C,3)
    Bar 50,8*C To 150,8*C+8
Next C
```

Filled shapes

You should now be familiar with drawing basic shapes and setting choices of colour. The next stage explains how to combine these skills. Reset your colours now, by getting rid of any customised PALETTE commands, before continuing.

PAINT

instruction: Fill a screen area with colour

```
Paint x,y
Paint x,y,mode
```

The PAINT command allows you to fill any section of your screen with a solid block of colour. You may also fill areas with various patterns previously selected with the SET PATTERN command, which is explained later. Decide which area is to be filled, and follow the PAINT command by a set of coordinates located anywhere inside the section of screen you want to paint with the current ink colour. Try this, which if all goes well should result in the Japanese national flag:

```
Palette 0,$F00
Circle 160,100,50
Paint 50,50
```

The optional mode setting can be set to either zero or one. A value of 0 ends your PAINT operation at the first pixel encountered of the current border colour. A mode of 1 stops the painting operation at any colour which is different from the existing ink colour. If there are any gaps in the boundaries of the sections you wish to fill, colour will leak out and stain the adjoining area.

BAR

instruction: Draw a filled rectangle

```
Bar x1,y1 To x2,y2
```

This is used to draw solid bars of colour by the familiar method of setting the top left-hand and bottom right-hand graphic coordinates.

POLYGON

instruction: Draw a filled polygon

```
Polygon x1,y1 To x2,y2 To x3,y3
Polygon To x1,y1, To x2,y2
```

This can be regarded as creating a solid version of the POLYLINE command, and your shape will begin at the current graphic coordinates if you commence the command in its POLYGON TO form. Provided that your single statement does not exceed the maximum allowed line length of 255 characters, there is no limit to the number of pairs of coordinates you can use. Try this example:

```
Do
    Ink Rnd(15)
    X1=Rnd(250) : Y1=Rnd(150) : H=Rnd(200) : W=Rnd(150)
    Polygon X1,Y1 To X1+W,Y1 To X1+W/2,Y1+H To X1,Y1
Loop
```

Alternative fill style

Filling shapes with plain colours is a useful technique, but the AOZ programmer has a much wider choice of fill effects.

SET PATTERN

instruction: Select a fill pattern

```
Set Pattern number
```

Use this command to select from a range of pattern styles. The default status fills shapes with the current ink colour, and is set with a zero, like this:

```
Set Pattern 0
```

If SET PATTERN is followed by a positive number from 1 to 34, shapes are filled from a ready-made selection of patterns.

View them now, by running this routine:

```
Do

    For N=0 To 34
        Set Pattern N
        Ink 0,1,2: Set Paint 1
        Bar 50,50 To 150,150
        Locate 0,0: Print N ;" "
        Wait 50
    Next N
Loop
```

If SET PATTERN is followed by a negative number, shapes will be filled with a pattern grabbed from a Sprite or Bob image, taken from the Object Bank (memory bank 1). Because these patterns can be very complex, AOZ will simplify them automatically, as follows.

The width of the image is clipped to 16 pixels (Amiga hardware emulation mode only), and the height is rounded to the nearest power of two (2, 4, 8, 16, 32 and so on.)

The original colours of the image are discarded, and the pattern is drawn using the current ink and paper colours.

Two-colour patterns are drawn as monochrome images.

If multi-coloured images are required using the original Object colours, the INK must first be set up, as follows:

```
Ink 15,0
Set Pattern -1
Paint 100,100
```

That example fills the screen area around the given coordinates with any of the Object colours, except the transparent colour zero. The colour index number 15 acts as a mask, defining which colours are to be used, and sets the range from 1 to 15. If the INK command is changed to the following line, the object will be drawn with the normally transparent colour filled by colour 1:

```
Ink 15,1
```

Before making use of sprite images as fill patterns, remember to use GET SPRITE PALETTE to avoid messy
displays. Here is an example:

```
Flash Off : Cls 0
Load "AMOSPro_Tutorial:Objects/Pattern.Abk"
Get Sprite Palette
Box 1,1 To 319,199
Ink 15,0
Set Pattern -1
Paint 102,102
```

SET PAINT

instruction: Toggle outline mode

```
Set Paint mode
```

This is a simple command that toggles outlines off and on for any shapes drawn using the POLYGON and BAR
instructions. Follow SET PAINT with a mode value of 1, and borders will appear in the previous ink colour. If the mode is set by a zero, the default setting applies, with no borders shown. For example:

```
Ink 0,1,2 : Set Paint 1
Bar 5,5 To 200,100
Set paint 0: Bar 210,75 To 310,190
```

In the last example, the INK command carried additional parameters. These optional settings follow the usual colour number, and are used to determine paper and border colours. In other words, they can set the colours to be used for fill patterns and outlines of bars and polygons. Remember to include any commas for unused options, as follows:

```
Ink 3: Rem Set ink colour
Ink ,,5: Rem Set border outline only
Ink 0,8,2: Rem Set ink, fill colour and border
Ink 6,13: Rem Set ink and background fill colour
```

Overwriting styles

When graphics are drawn, they normally get "written" over what is already displayed on the screen.

There are four alternative drawing modes that change the way your graphics appear, and they may be used individually or combined to generate a whole range of effects.

GR WRITING

instruction: Change graphic writing mode

```
Gr Writing bitpattern
```

This command is used to set the various modes used for drawing lines, shapes, filled shapes and graphical text. Settings are made using a bit pattern, whose values give the following results:

- Bit 0 = 0 only draw graphics using the current ink colour.
- Bit 0 = 1 replace any existing graphics with new graphics (default condition).
- Bit 1 = 1 change old graphics that overlap with new graphics, using XOR.
- Bit 2 = 1 reverse ink and paper colours, creating inverse video effect.

The normal drawing state is where new graphics overwrite old graphics, like this:

```
Ink 2,5 : Text 100,80, "NORMAL TEXT"
Wait 100 : Gr Writing 1
Text 10 ,80, "REPLACE"
```

Try the next example for some simple demonstrations of alternative settings:

```
Ink 2,5 : Text 100,80,"NORMAL TEXT"
Wait 100 : Gr Writing 0
Text 100,80, "MERGED"
Wait 100 : Gr Writing 4
Text 100,90, "STENCIL"
Wait 100 : Gr Writing 5
Text 100,100, "REVERSE"
```

Advanced techniques

SET TEMPRAS

instruction: Set Temporary Raster (obsolete, no action)

```
Set Tempras
Set Tempras buffer address,buffer size
```

This command allowed the AOZ programmer to adjust the amount of memory used by the various
graphics operations and is no longer needed n modern machines.