Windows

The AOZ programmer expects to be able to produce file selectors, warning boxes and on-screen control panels with a few simple lines of code. The range of windowing command featured in this Chapter allow you to create interactive dialogue boxes by restricting text and graphics operations to selected areas of the current screen.

A "window" is simply a rectangular display area, which must first be opened before electronic life can course through it. Your current screen is treated as a window, and opened automatically by the AOZ system as window number zero. All other windows have to be opened by you, and you are advised not to re-open window zero or change its size or position.

Creating windows

WIND OPEN

instruction: create a window

```
Wind Open number,x,y,width,height
Wind Open number,x,y,width,height,border
```

The window opened by this instruction will be displayed on screen and used for all subsequent text operation until you command otherwise. WIND OPEN must be qualified by a window number (don't forget that zero has already been allocated to the current screen), followed by the x,y graphic coordinates setting the top left-hand corner of the new window, followed by the width and height of the new window in the number of characters needed. You may also specify an optional border style, with values ranging from 1 to 16.

If the manifest of the AOZ application indicates an Amiga hardware limitation emulation, the horizontal coordinates will be rounded to the nearest multiple of 16. If the emulation is set to 'PC' then there is not limitation.

Additionally, if you have specified a border for your window, the x and y-coordinates will be incremented by an additional 8 pixels. In this way, you can be sure that your windows always start at the correct screen boundary. There are no boundary restrictions on the y- coordinates. Titles can also be included in window borders, which will be dealt with a little later. Try this example:

```
For W=1 To 3
    Wind Open W,(W-1)*96,50,10,15,W
    Paper W+3 : Pen W+6 : Clw
    Print "Window";W
Next W
```

WINDOW

instruction: change the current window

```
Window number
```

This command sets the window number specified as the active window, to be used for all future text operations. There is an automatic saving system for re-drawing the contents of windows, which is explained below.

For now, run the last example from Direct mode and enter the following statements:

```
Window 1: Print "AOZ"
Window 3: Print "open windows on the world"
Window 2: "lets me"
```

The active window is host to the flashing text cursor, unless it has been made invisible with a CURS OFF command.

BORDER

instruction: change window border

```
Border number,paper, pen
```

This command allows you to change the style and colour of the current window border. Border style numbers range from 1 to 16, and the paper and pen colours can be selected from any available colour index numbers. Any of these parameters can be omitted from the BORDER instruction as long as the commas are included for any missing values: If the last example is still on screen, enter these lines from direct mode:

```
Border 3,2,3
Border 2,,
```

TITLE TOP

instruction: set title at top of current window

```
Title Top title$
```

Use this command to set a border title at the top of the current window to your chosen title string. This facility will only operate with bordered windows, as follows:

```
Cls: Wind Open 4,1,1,20,10,1
Title Top "Top of the morning"
```

TITLE BOTTOM

instruction: set title at bottom of current window

```
Title Bottom title$
```

Similarly, this instruction assigns a string to the bottom title of the current window, like this:

```
Cls : Wind Open 5,75,50,24,15
Border 5,6,
Title Bottom "Bottom of the barrel"
```

Manipulating windows

WINDON

function: return the value of the current window

```
w=Windon
```

Before using windows in your programs, you will need to refer to their identification numbers. This function returns the value of the current window. For example:

```
Do
    Cls : Wind Open Rnd(99)+1,1,1,25,5,1
    Print "Window number ";Windon : Wait Key
Loop
```

WIND SAVE

instruction: save the contents of the current window

```
Wind Save
```

This command is extremely valuable for the AOZ programmer. Once activated, the WIND SAVE feature allows you to move your windows anywhere on screen without corrupting the existing display, by the following method. The contents of the current window is saved as soon as the command is used, and then every time a new window is opened, the contents of the windows underneath get saved automatically. The screen is then re-drawn whenever a window is moved to a new position or closed.

As you begin a new program, the current window (the default screen) consumes 32k of valuable memory, and this would be wasted if you were to save it as background beneath a small dialogue box. To solve this problem, create a dummy window of the size you need, and place it over the zone you want to save. Now execute your WIND SAVE command and continue with your program. When this dialogue box is called up, the area beneath it will be saved as part of your dummy window, so it will automatically be restored after your box has been removed.

WIND CLOSE

instruction: close the current window

```
Wind Close
```

The WIND CLOSE command deletes the current window. If the WIND SAVE command has been activated, the
deleted window will be replaced by the saved graphics, otherwise the area will be totally erased from the screen. Here is an example:

```
Wind Open 1,1,8,35,18,1 : Print "Press a key to close this window"
Wait Key
Wind Close
```

WIND MOVE

instruction: move the current window

```
Wind Move x,y
```

The current window can 'be moved to any acceptable graphic coordinates. Give the new x,y-coordinates after the WIND MOVE command, and the x-coordinate will be rounded to the nearest 16-pixel boundary automatically. Here is an example:

```
Wind Save : Wind Open 1,0,2,30,10,1 : Wind Save
For M=1 To 100
    Pen Rnd(15) : Paper Rnd(15) : Print : Centre "Making Movies"
    Wind Move Rnd(30)+1,Rnd(100)+1
    Wait VbI
Next M
```

SCROLL ON/OFF

instructions: toggle window scrolling on and off

```
Scroll On
Scroll Off
```

The SCROLL commands are used to control the scrolling of the current window. SCROLL OFF turns off the
scrolling, and whenever the cursor passes beyond the bottom of the window it will reappear from the top. SCROLL ON starts the scrolling process again, so that a new line is inserted when the cursor tries to pass beyond the bottom of the window.

WIND SIZE

instruction: change the size of the current window

```
Wind Size width, height
```

To change the size of the current window, specify the new width and new height in terms of the number of
characters. If WIND SAVE has been activated, the original contents of the window will be re-drawn by this instruction. If the new window size is smaller than the original, any parts of the original image that lie outside of the new window boundaries will be lost. Alternatively, if the new window is larger, the space around the saved area will be filled with the current paper colour. Please note that the text cursor is always re-set to coordinates 0,0. For example:

```
Wind Open 1,16,16,22,10,2
Print "I want to be wider!"
Wind Save
Wait 50
Wind Size 30,10
```

CLW

instruction: clear the current window

```
Clw
```

This simple command erases the contents of the current window and replaces it with a block of the current PAPER colour. Like this:

```
Cls: Paper 4 : Wind Open 1,1,1,12,5,1
Window 1: Print "Clear Off" : Wait 75
Paper 9 : Clw
```

Creating slider bars

One of the standard uses of windows is to create interactive slider bars.

HSLIDER

instruction : draw a horizontal slider bar

```
Hslider x1 ,y1 To x2,y2, units, position, length
```

Horizontal slider bars are set up by giving the HSLIDER command, qualified by the following parameters: the x,ycoordinates of the top left-hand corner of the bar in pixels followed by the x,y-coordinates of the bottom right-hand corner, then the number of individual units that the slider is divided into. Next, you must specify the position of the active slider box or control button from the left-hand end of the slider, measured in the same sized units as the slider divisions. Finally, give the length of the slider control box in these units. The size of each unit is calculated with this formula:

(x2-x1)/number of units

Here is an example:

```
Hslider 10,10 To 100,20,100,20,5
Hslider 10,50 To 150,100,25,10,10
```

VSLIDER

instruction: draw a vertical slider

```
Vslider x1 ,y1 To x2,y2,units,position,length
```

This works in the same way as Hslider, and sets up vertical slider bars. For a working demonstration, examine the vertical slider in the Editor window, where the number of units into which the slider is divided is set to the number of lines in the current program. Here is a simpler example:

```
Vslider 10,10 To 20,100,100,20,5
Vslider 250,0 To 319,199,10,2,6
```

SET SLIDER

instruction: set fill pattern for slider bar

```
Set Slider ink1,paper1,outline1,pattern1,ink2,paper2,outline2,pattern2
```

SET SLIDER is used to set up the colours and patterns used for your slider bars and their control boxes. Simply give the index numbers of the ink, paper, outline and pattern to be used for the slider bar, followed by the ink paper, outline and pattern to be used by the slider control box. If negative values are used for either pattern, a sprite image will be commandeered from the sprite bank, allowing even more spectacular effects. Try this example:

```
Centre "<Press a key>" : Curs Off
Do
    A1=Rnd(15) : B1=Rnd(15) : C1=Rnd(15) : D1=Rnd(24)
    A2=Rnd(15) : B2=Rnd(15) : C2=Rnd(15) : D2=Rnd(24)
    Set Slider Al ,B1,C1,D1,A2,B2,C2,D2
    Hslider 10,10 To 300,60,100,20,25
    Vslider 10,60 To 20,190,100,20,25
    Wait Key
Loop
```

Having set up your slider bars, you will want to activate them using the mouse. A simple routine to create working slider bars needs to be included in your main program. As always, remember to test out the ready-made example programs, for a working guide.

Displaying a text window

To end this Chapter, here is an extremely useful AOZ feature that allows the display of a text file directly on screen. Text can be displayed in its own independent screen, it may be scrolled through at will, the display window can be dragged around the screen and there is even a facility to include a title line.

READ TEXT$

instruction: display a text window on screen

```
Read Text$ name$
Read Text$ name$,address, length
```

In its simplest form, the READ TEXT$ command reads the text held in a specified filename on disc, for example:

```
Read Text$ Fsel$("**")
```

You can move through the displayed text using scroll bars, the arrow icons or via the following key combinations:

- Key Press Effect
- [Up Arrow]/[Down Arrow] Move up/down by one line
- [Shift]+[Up Arrow]/[Down Arrow] Scroll up/down by one page
- [Ctrl]+[Up Arrow]/[Down Arrow] Jump directly to top/bottom of text
  *[Esc] or [Return] Exit

To read some text from an address in memory, there is an alternative version of the READ TEXT$ command. In this case the name$ parameter refers to a title line that will be printed at the top of the viewing window. Address holds the address of the first line of the text to be read. Length specifies the length of the text to be read, in bytes.