Macromedia Authorware 6.0

Training Notes

Tel/Fax 01772 751465

http://www.castlecs.co.uk

Version 2.0

Introduction

These notes are intended to accompany a training course to cover the fundamentals of producing multimedia-training packages using Macromedia Authorware

The course assumes no prior knowledge of Authorware, but does expect the student to be reasonably familiar with PC Computers in general, Windows 9x, NT or later, and to have an understanding of conventional methods of producing training courses.

Authoring

This course refers to Macromedia Authorware version 6.0 and how this program can be used to produce multimedia-training programs. The full manual for the program runs to over 500 pages, and a number of books are available on the subject. The course and these notes are not intended to replace these reference sources and the reader is recommended to consult the official manual or the on-line help that comes with the program in any cases of doubt.

The classroom work will consist of a series of exercises, increasing in complexity, until a small but complete training course is produced which will incorporate all of the items covered throughout the week. These notes can be consulted for information and guidance on the specific topics covered within the course but it must be emphasised that they are only notes and are intended to be neither complete nor definitive.

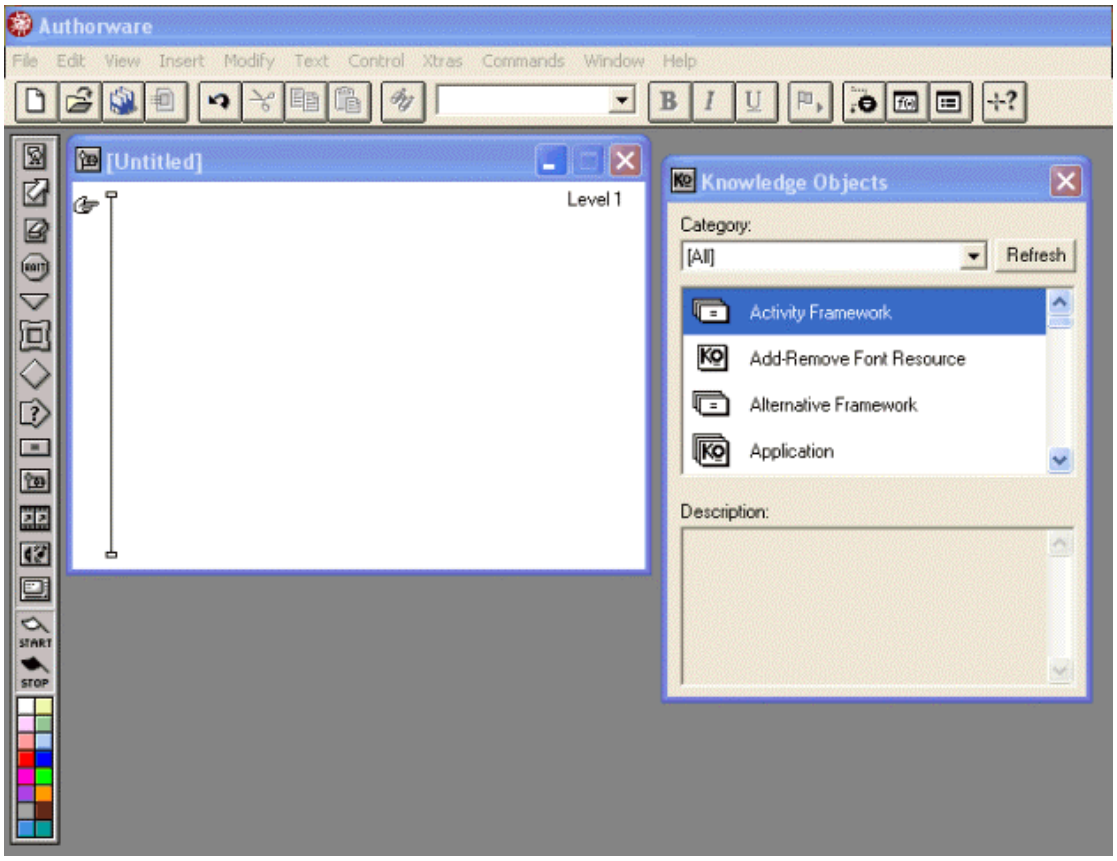
Introduction

Authorware is an authoring program that is intended to allow trainers with the minimum of computing skills to produce useful and effective training packages. While doing this it also includes a highly complex and flexible programming language that allows the more computer proficient user to produce advanced and highly complex packages.

This section of the notes will be split into the various options that have been covered in the training room.

Authorware Environment

The main Authorware screen is basically similar to other Windows screens, there is a menu bar across the top and a button or tool bar below it, the main working area is inside a window and there is a further tool bar down the left side of the screen. If the mouse is pointed at any of the buttons on screen and held still for a couple of seconds a small tool tip appears identifying the button.



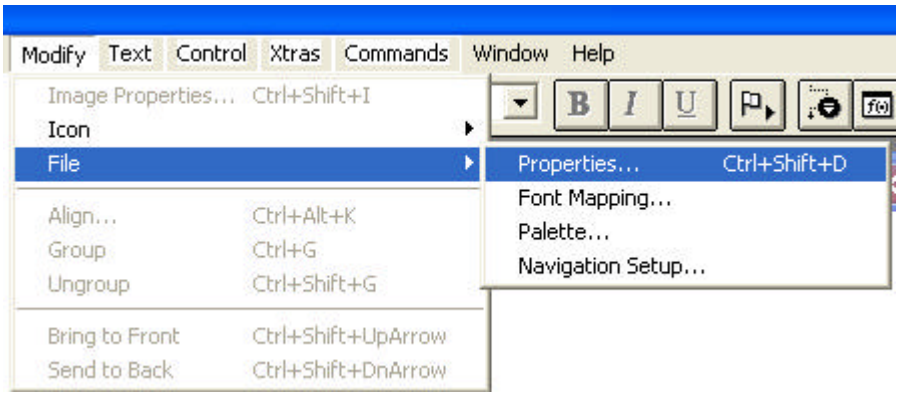
The button at the extreme right edge of the main tool bar is a help button, click on this and then point and click at other items on the screen and Help will be started with an explanation of whatever you have clicked on.

In common with most Windows based programs, pressing the 'F1' key will start the help system. Help is usually context sensitive – it should start in that area of the help files that most closely matches whatever you are doing on screen at the time.

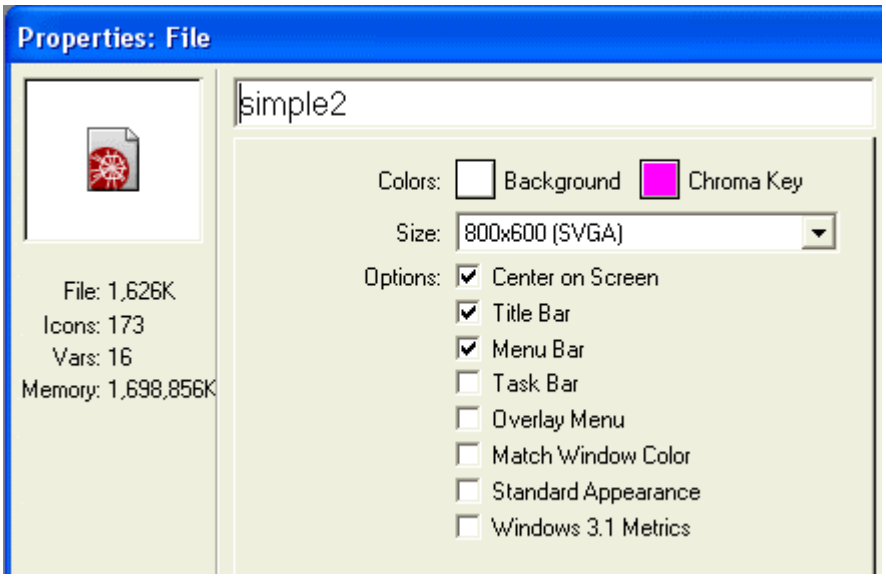
As the items on the menu and toolbars are used in the course they will be explained briefly in this document. For a fuller explanation you are recommended to refer to the main Authorware manual. In particular where there is a discrepancy between these notes and Macromedia documentation you should accept the Macromedia documentation.

File Properties

The first thing that needs to be set up before starting to create any Authorware application is the 'File Properties'. On the menu bar click on 'Modify' then 'File' then 'Properties'.



This will cause the File Properties dialogue box to be displayed.



There is no need to enter a Title, as this will be created when we save the file later.

Click on the small white box next to 'Background' and you will be shown a palette of colours from which to choose the basic background colour of the application you are making, then click on 'Okay'.

Don't bother with 'Chroma' as this only really applies to video overlay colours.

Next you can choose the size of the application screen you are creating, this will usually be '640x480 VGA' or '800x600SVGA'. You can choose any of the sizes from the drop down depending on the type of application you want to make. You can also choose 'Use full screen' or 'Variable' but you should be careful with these, as you then have no control over the actual appearance of the application when it is run and some applications will have a problem with this.

Always make sure that 'Center on Screen' is clicked, this will make sure that your application will automatically centre itself on the screen when it is run.

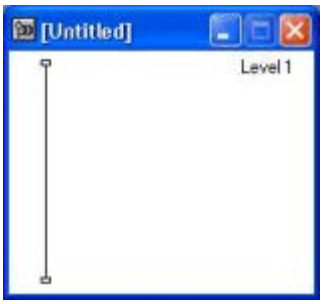
Generally you should remove the remainder of the 'tick' marks unless there is a specific reason for using those features.

Click on 'Okay' when this is done and you will be returned to the main screen.

You can change any of the above settings at any time after starting the development but be careful about changing the screen size as this will *not* automatically move objects which are displayed and they will either be in the wrong place or even hidden off the screen altogether.

Flow line

When you start a new Authorware application you will b presented with an empty window with the word 'Untitled' displayed in the title bar. This is the working window:



Notice the vertical line down the left of the window, this is called the flow line, it will contain all of the icons that you will need to produce your application.

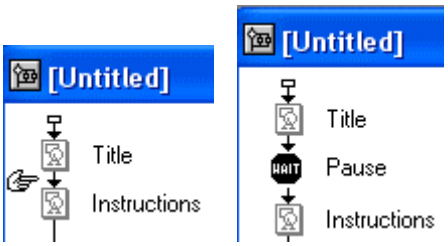
While you are here, notice also the label 'Level 1' in the top right of this window.

These working windows can contain other windows, each 'nested' inside another. You will invariably have several of these windows and often very many. When one window is nested inside another its level will be increased, it is not unusual to have seven or eight levels of nesting and this label is a useful way of reminding yourself where you are in your application.

All applications in Authorware follow this 'flow line' metaphor. A vertical flow line is displayed in the editing window and the program effectively follows the line from top to bottom as the application is processed. It's possible to create branches from this flow line to make the application take a less than truly linear path through the inserted icons, but in effect each separate piece of the application is actually made up of sections of these linear flow lines.

To insert any icon on the flow line simply drag the icon from the Icon Bar and drop it in the correct place on the flow line. You can insert an icon at any point on the line that you wish. You do not need to always place an icon at the bottom of the existing flow line.

If you simply click on a flow line a hand symbol will be placed there. This indicates the point at which a paste function will insert an icon that you have copied or cut from elsewhere, or where an imported media item will appear.



Icons

You are now in a position to start producing your application. Click and dragging icons from the toolbar on the left of the screen to the flow line will let you do this almost entirely. Remember the flow line is the thin vertical line that is visible on the working window.

You can 'open' an icon by double clicking on it or by running the application. When an application is running it will usually stop when it reaches an icon that has no content, a display icon that has nothing to display for example.

New to version 6 of Authorware is the ability to pre-define the properties of icons. You can if you wish, put content or change the properties of an icon on the flow-line and if you then drag and drop that icon back on the toolbar every time that you use the icon it will 'remember' how you left it and start like that – this can apply to names, colours and even content.

Click on File – Preferences – Reset Icon Palette to return everything back to normal.

Icon Names

It is very important that unique names are given to the icons as they are used; this helps not only you when you are creating the application, but also helps others when they come to maintain it later. Several of the functions that you will use in later development will actually refer to icons by name and will refuse to work if the icons are not named or if two or more icons have the same name.

The icons will be explained briefly here. The explanations are given in the order in which the icons appear on the tool bar, from top to bottom.

Display

The display icon is the fundamental icon in Authorware and is used to show both images and text. Click and drag a display icon onto flow line then release it. Give it a name by typing the name alongside the icon and pressing the 'Return' key.

When you first place any icon on the flow line it will naturally be 'empty' because you have not given it any content. An empty icon will be displayed in a pale grey colour so that you can see which icons do not yet have content.

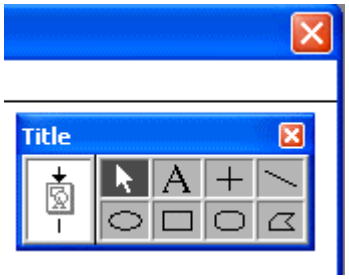
You can drop as many icons as you wish onto the flow line one after another, although it is always a good idea to name them as you do this.

When you run your Authorware application in editing mode (we usually call this 'Authoring' mode) it will stop when it reaches an empty icon so that you can edit it. (This includes display icons).

Alternatively from the flow line you can double click on the display icon to open it for editing at any time.

Once the display icon is open you will see a window at the size that you set up in File Properties and a small floating tool bar. The tool bar will contain the title that you gave to the icon.

The drawing tools in Authorware are fairly basic and self-explanatory. You can draw rectangles, lines and ovals as well as polygons and rounded squares by selecting the appropriate tool from the palette and drawing on the display area.



Double clicking on the items in the tool bar will allow you to change things like line thickness and the fill properties of the rectangle and ellipse. Double clicking on the ellipse icon will display a colour palette.

In general we use only the text entry that is shown by the letter 'A' and the pointer arrow. The other drawing tools are occasionally used for drawing simple panels as a quick background.

Text is such an important function that it will be dealt with below in its own section.

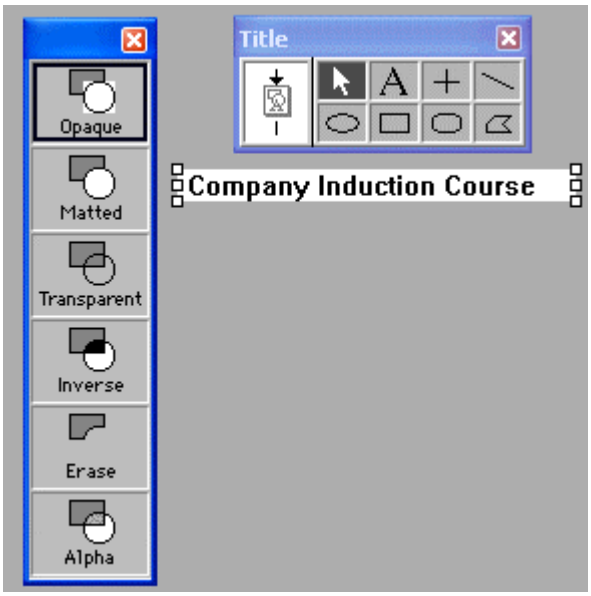
Using the arrow pointer and then clicking on any object you can move it on the display area. Objects can then be dragged anywhere, on or off screen, this applies to text as well as drawn objects and imported objects such as graphics.

Holding down the 'Ctrl' key and pressing 'K' will bring up a colour palette to allow you to change the colours of the drawing objects.

'Ctrl' and 'T' will display the Transition dialogue box that will allow you to define the special effect used to display the contents of the icon. You are encouraged **NOT** to use too many different transitions in the same project, as a large variety can be distracting.

For simplicities sake in your early days with the program just use the 'Internal' transitions, the other transitions are provided by external functions called 'Xtras' which require other files to be included when packaging your finished program.

Modes



'Ctrl' and 'M' will display the mode palette which is very useful. This can be used to make the background of text and the white part of images invisible, giving a much neater and clearer finish to your displays.

Notice in this example where the application background is grey, the text is currently set to be Opaque, changing it to Transparent will remove the white box and allow the text to be displayed directly on the background.

A similar effect also applies to images, but you should note that for this to take effect areas that you want to be transparent must be 'pure' white, that is they must have an RGB value of 255,255,255.

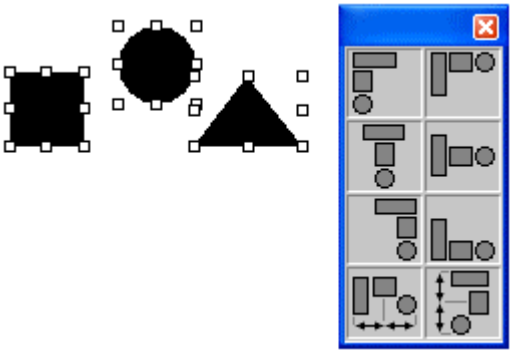
The most common other option chosen is Matted, this is similar to Transparent except that only areas that are pure white and *not completely surrounded* by any other colour will be displayed as transparent.

Alpha will allow you to take advantage of alpha masks produced in graphics programs these can include graduated shadows, but their production is beyond the scope of this course.

Aligning Objects

You can select a number of objects in the display area, by either click and dragging across several of them, or by holding down the Shift key while clicking on individual objects.

A separate palette is available to align objects on the display area, once the objects have been selected you can display the alignment palette by holding down both the 'Ctrl' and 'Alt' keys and then pressing the 'K' key.



There are several ways in which objects can be aligned; either vertically or horizontally and by either edge or by their centres.

The two options on the bottom of the palette are used to distribute the selected objects evenly; again this can be either vertically or horizontally.

NB There is a known bug in Authorware on this palette. You should always save your work *before using the vertical distribution option*. In certain circumstances clicking on the vertical distribution can cause Authorware to crash! This invariably happens when you have done a considerable amount of work since the last save.

It is usually a good idea not to align objects both vertically and horizontally at the same time. Try this once and you will see why.

In addition to aligning objects on screen you can also move them in and out of the screen, this is usually referred to as setting layers. The effect is to have one object in the display icon in front of or behind another. For example you can place some text in front of a graphic object as in this example.

Authorware allows you to set layers for objects with a display icon and also to set the layer of the whole display icon in relation to other displays.

By default, the later an object is placed on the screen the higher its layer will be, this applies within a display icon, so in the example above, the graphic was drawn first and the text added later.

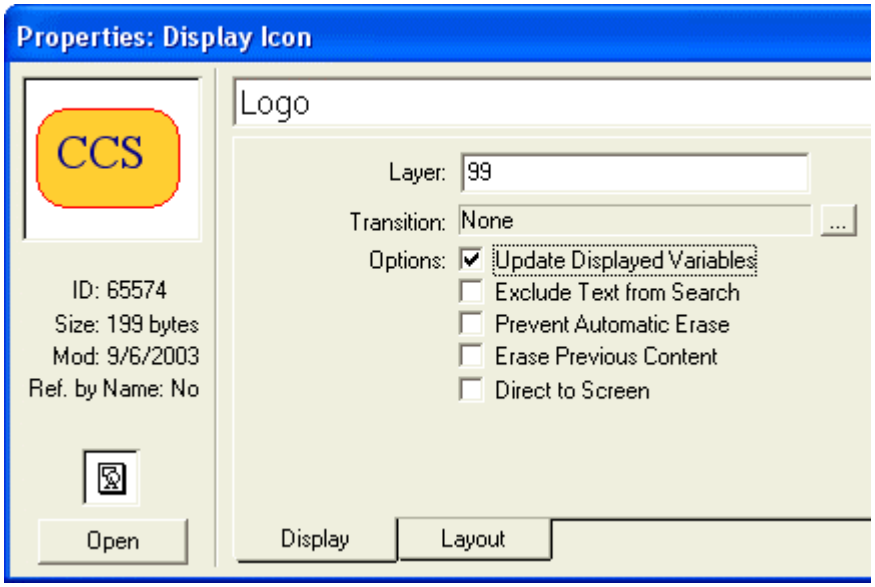
To move an object up or down in *within* a display icon you can select it and then use 'Ctrl – Shift – Up Arrow' or 'Ctrl – Shift –Down Arrow' as required. The effect of these is to move the object to the front or the back of the display.

The default rule about later objects sitting in front of earlier ones also applies to the whole display icon.

For example you can set a display icon to contain a background image and then any subsequent display icons will by default sit in front of it.

This default rule is of course not appropriate for all circumstances; you may want a display to sit behind a part of previously displayed icon. Or you may want to have an earlier image sit in front of later ones.

Control of the layering of display icons is applied using the properties window for the icon.



To display the properties palette for a display icon (and indeed any icon) first select the icon, or even double click it to open it and then press 'Ctrl – I' on the keyboard and the properties window will be opened.

Ensure that you are in the 'Display' tab and you will see the Layer field. By default this will be blank. You can enter any arbitrary value you wish into this field, the higher values would be displayed in front of any display icons that do not have a value set. To ensure that a particular icon is always in front of everything you could set a very high value.

It is quite normal also to set a negative value to ensure that a display is always behind other icons.

It is a 'bad thing' to set a value on every display icon as there is a small performance loss when too many layer values are set.

While we are looking at this window lets also consider a couple of the other items that could be changed.

You have direct access to the Transitions dialogue box from here, this works in exactly the same way as by pressing Ctrl – T.

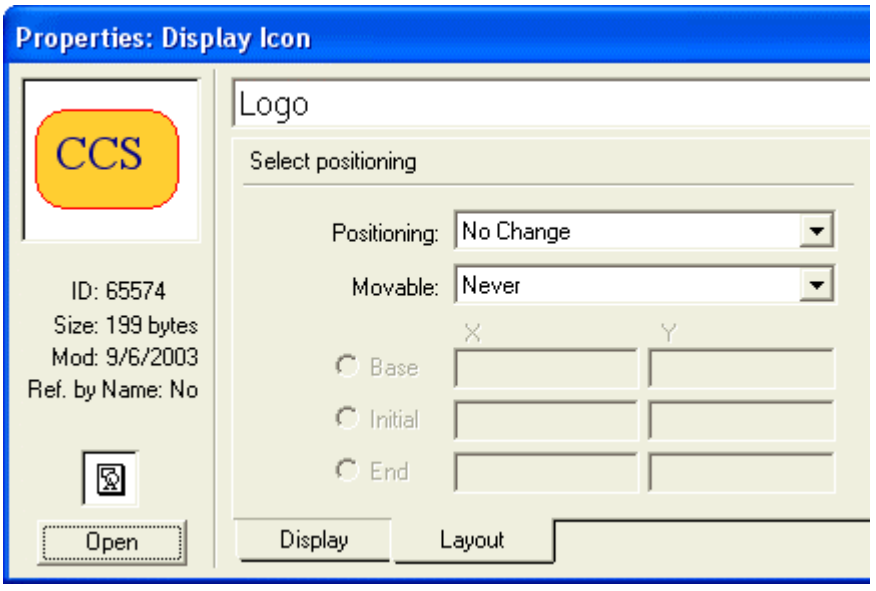
Of the options shown the most important is 'Update Displayed Variables'. While we have not yet discussed variables you should know that one of the things you might want to do with the application that you are building is display new information on screen, for example the current score, or the length of time taken. A variable can contain this information and be displayed on screen, if the Update Displayed Variables option is set then whenever the value changes the display will reflect that change, otherwise it will not.

There is again a very small performance loss when this option is set so while you shouldn't hesitate to use it if needed you should not automatically set it for every display icon.

None of the other options are of major significance at this time, in particular you should be wary of setting the 'Prevent Automatic Erase' until you are completely sure of what you are doing. This option would cause the contents of the current display icon to remain on screen when it would otherwise have been automatically removed. If you are not careful you will find that an object remains on screen unexpectedly and you can waste large amounts of time trying to find out why.

You are very unlikely to use the option 'Erase Previous Contents' as almost always there is a background graphic that you would want to remain on screen.

'Direct to Screen' would cause an object to be drawn directly to the screen rather than be processed by the Authorware program. This is rarely needed. Probably the most likely use would be when displaying digital movies or Flash pieces within Authorware.



The second tab within the properties window is 'Display'. This allows us to set some properties that you may not use very often, but which will be vital when they are needed.

When you are in authoring mode any object on screen can be moved as much as you wish, simply

click and drag to re-position it in the location that you want. By default once the piece has been packaged no objects can be moved so that the user can't move things around.

There may well be occasions when you might want the user to be able to move objects from one place to another, a calculator or a glossary window for example.

Equally there may be occasions when you want to display an object on screen at a specific location depending on what the user has done elsewhere in the piece.

Positioning allows us to have control over the location of an object (almost invariably a display icon) once a value different to 'No Change' has been selected the greyed out fields will be made available. You can enter values into these fields, but generally you will place variables in which will allow you to position an object exactly as required simply by changing the value of the variable before displaying the icon. Precise details of this process are beyond the scope of an introductory course.

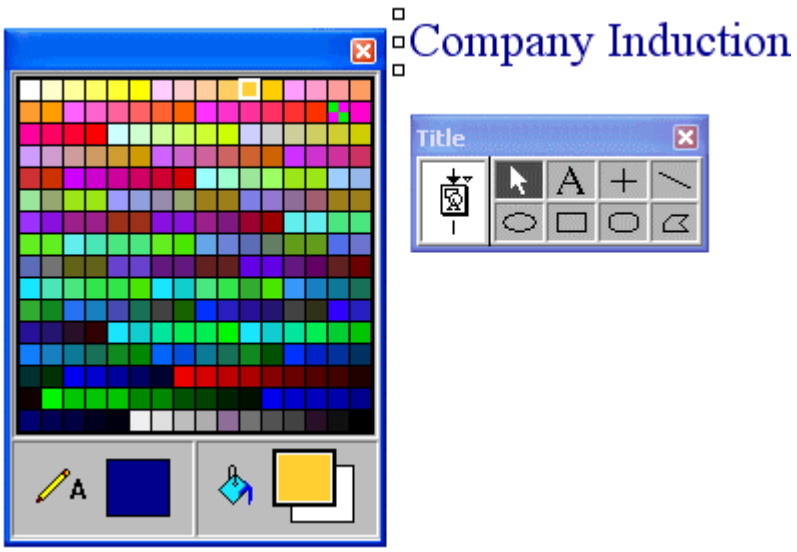
To make an object movable by the user, choose either 'On Screen' or 'Anywhere' and the user will then be able to click and drag the object as they wish.

Note that both positioning and moving an icon will apply to all of the objects within a specific display icon. If there is text and a graphic for example, in the same icon then both text and graphic will move together. For this reason it is best to ensure that if you want to use either of these facilities then you should ensure that the object to be affected is in a display icon by itself.

Text

There are a number of ways in which you can place text on the display area the most obvious way is simply opening a display icon and using the 'A' tool to type in what you want. It can become rather more complicated than that so we'll deal with the various aspects in turn.

Colour



Ctrl and K will display the colour palette this is the simplest and is the least useful way of controlling your text. Notice the button next to the Pencil image, this is what controls the foreground colour of the selected object, click on this and then select a different colour for the selected object to change its foreground colour, in this case the colour of the text. Although they have no effect on text, the fill buttons to the right of this area will control the foreground and background colours of any graphic object that you draw with the Authorware drawing tools.

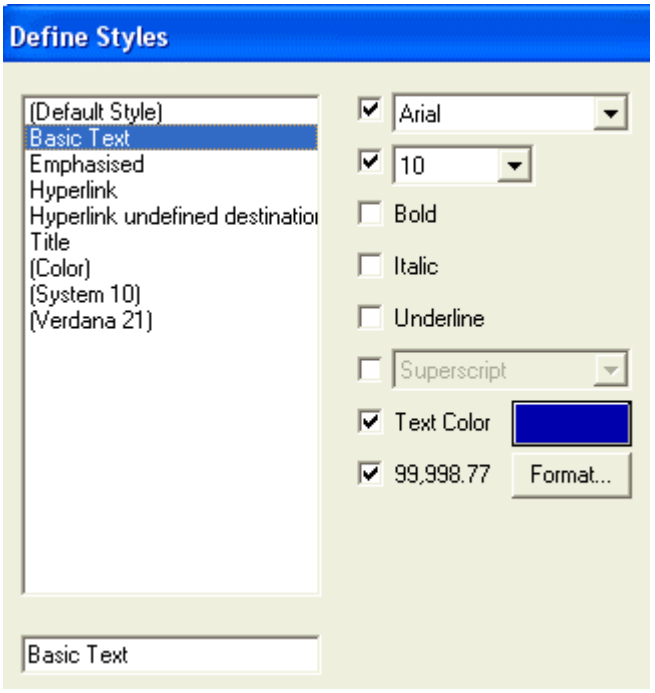
There are a couple of points that should be noted here.

1. Once selected, the foreground colour will affect every new text or drawing object that you put on screen until the colours are changed.
2. Every time that you change the colour of a text object a new text style is created, you will see shortly how to avoid this.

Styles

Clicking on 'Text – Define Styles' will display the styles definition window.

There is always a [Default Style], you cannot delete this; typically it will be System font, 10 pitch, black colour. Unless you make changes all text that you enter on screen will appear in this format.



The square brackets that surround the words Default Style indicate that this is a system-generated style. Any other styles that are created directly on the screen will also be displayed as a title within square brackets; these other styles can be edited and renamed as you wish.

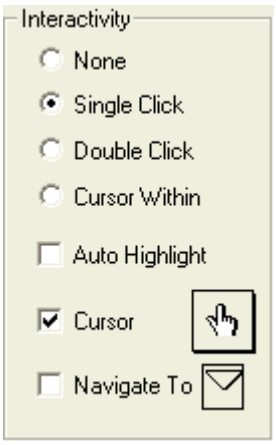
You can remove a style *that is not being used*, by clicking on the Remove button. This also applies to any styles enclosed in the square brackets. While a style is in use it cannot be removed.

To see in how many places the icon titles use these styles click on the References button and a list of the icon titles will be displayed (another good reason to name your icons). By selecting one of the icon titles and clicking on the Show Icon button you will be shown the icon that uses that particular style (you will not be able to edit that icon until the Styles dialogue window is closed).

To create or change an attribute of a style you must ensure that the tick mark alongside the attribute is selected, once this is displayed you can choose a font, size or colour by using the drop down or pop-up windows alongside the attribute.

Text styles extend to more than just the displayed attributes. We can also set up the way in which numbers are displayed, leading and trailing zeroes, whether a thousand separator is used or not and what decimal separator is used.

Note that this will apply only to numbers that are displayed as the value held in a variable. Simply typing a number on screen will not cause the numerical format to be applied.



One of the most powerful uses for styles is using them to create *hyperlinks.* A hyper-linked piece of text is a word which, when you click on it will cause the program to jump to a different part. For example if you have a block of text, some of the words may need separate explanations, these could be held in a glossary. By hyper-linking those words you could allow the user to activate the glossary just by clicking on the word.

Once the style is selected, choose how the hyperlink is going to be activated, by clicking on single or double click, or just cursor within (when the cursor is simply pointing at the word). And the style will automatically be classed as a hyperlink. You should at least ensure that the style is in a different colour, so that the user can see that there is something different about it, you may also want to set it to be underlined as many users are now used to seeing hyperlinks on the Internet in which a different colour (usually blue) and underlining are used.

If you want a different cursor to appear when the mouse pointer is over the link then clicking on the Cursor option will allow you to choose one.

The final option in this section is the Navigate To, to activate the navigation you can set up that destination by clicking in the Navigate To selector.

Now you must make a choice. If this style will *always* go to the same place in the program click on the navigation icon (the small triangle) and you will be shown the navigation dialogue window. If you want to allow different words to share the same hyperlink style but go to different destinations then you would leave this option untouched (you will then be able to set the destination on each occasion that you apply the style).

Navigation in these circumstances is usually to a Destination that might be defined as 'Anywhere – Call and Return – Icon attached to a Framework'. Don't worry too much about these terms they will be explained in more detail later. Just remember that you should have at least one framework icon with at least one option hanging off it and that this should have a unique name.

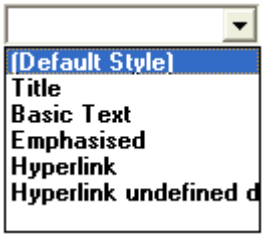
In the example above we have a framework icon called ‘Subroutines’ and the ‘Glossary’ map icon is hanging off it. It is there we will take the user.

There are a number of different options that we might choose from the navigation window, but they will be explained in more detail in the Navigation section later.

It is this navigation set up that will appear on each use of a hyperlink style when we have not pre-defined the destination.

Once the styles have been defined you must ensure that you click on the Modify button to cause the changes to be saved.

When you then close the styles definition window any text that has a style that has been change will have the new style reflecting the changes applied to it.



To apply a style to a piece of text the text must be selected in the normal Windows fashion, either while in text editing mode by selecting a section of the text with the mouse, or by selecting the text block with a pointer.

Once selected click on the style drop down list on the menu bar and select the style that you want to apply.

Not all of the available styles may be displayed in this window; you can use either the initial letter of the style or the cursor up and down arrows to scroll through the full list.

It is also possible to apply styles using the styles floating palette. Click on Text – Apply Styles. This is not recommended, as it is possible to apply multiple styles to the same text object with completely unpredictable results.

To remove all styles from a text object, simply select it and set the style to be Default.

In addition to entering text directly on screen you can place the contents of variables on screen, this is particularly useful to show values that change through the course of the program, such as questions, possible answers, feedback and scores.

To display the contents of a variable on screen you should enclose the name of the variable in braces ‘{‘ and ‘}’ like this:

Your current score is {StudentScore} out of a possible {MaxScore} points.

Remember to set the display property to be ‘Update Displayed Variables’ if the value of the variable is likely to change *after* it has been displayed.

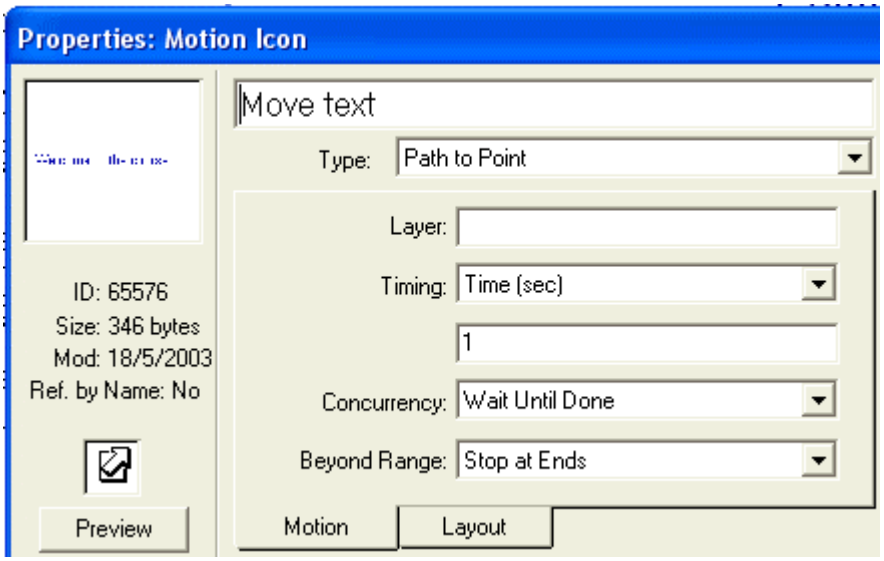
Motion

The motion icon will allow you to move the contents of a previously inserted display icon. It will move all of the contents of the display icon and will only move one icon.

If you want to move just a portion of the screen, make sure that the portion is in a display icon all by itself.

If you want to move more than one item at the same time then use two or more motion icons, each one moving a specific display. To do this you must make sure that the 'Concurrency' option in the motion icon control box is set to 'Concurrent'.

The motion icon will expect you to click on the item to be moved and then to drag it to its intended destination. You can also set the timing to control how long the movement should take.



Shown here are the default settings for a motion icon. Just as with most icons in Authorware you can simply drop the icon on to the flow line and run the piece. When Authorware encounters an empty icon it will stop and prompt you for some content.

In this case the minimum content that is required is the icon to be moved. To select the icon to be moved, simply click on it on the display screen.

If you simply want to move the icon to a different location on screen then you should simply drag the icon to that location and drop it.

You can see the effect of the motion icon by clicking on the 'Preview' button.

We will consider the 'Type' of motion shortly, for now leave it at 'Direct to Point'.

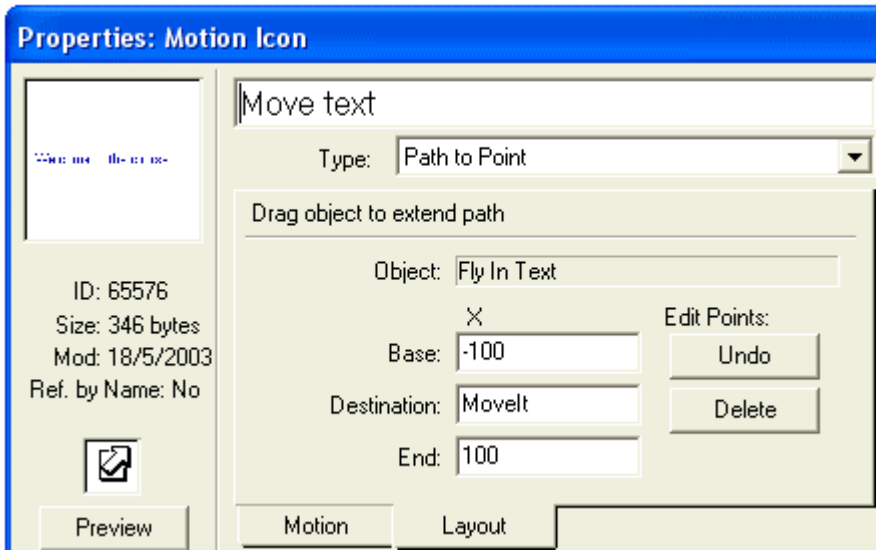
Layer is by default left blank. You should only enter a value in this field if you really need to as there is a performance hit if too many motions are on none zero values.

Timing is either by 'Time (sec)' or 'Rate (sec/in)'.

The difference is that:

- Time will indicate how long in seconds or parts of a second the motion will take, regardless of the distance on screen, so that for a given time an object moving a shorter distance will travel more slowly.
- Rate will cause an object to always move at the same speed regardless of how far it needs to move. The rate is defined in inches per second (this is an American

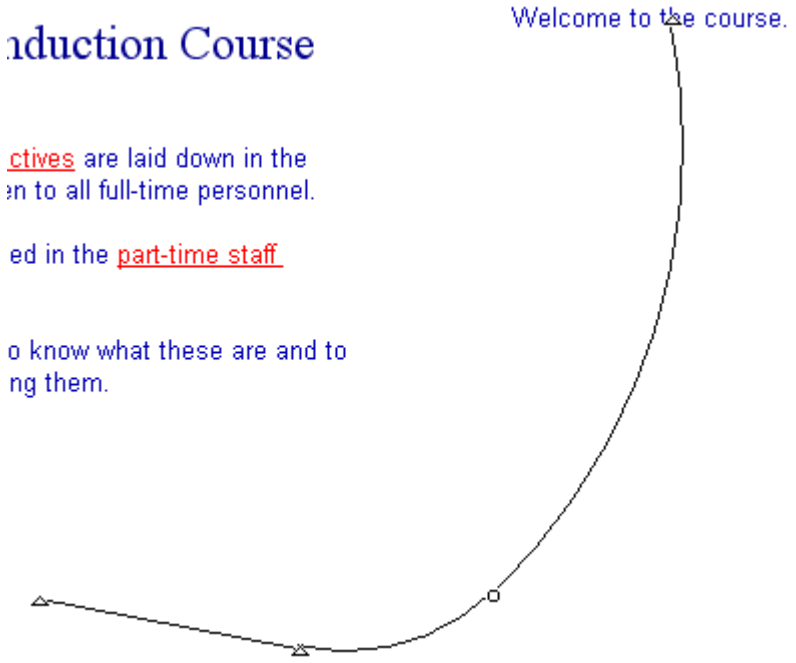
program after all). You may need to experiment on screen to see the true effect depending on screen resolution and size.



Concurrency has two settings, 'Wait Until Done' which stops all other program execution until the motion is complete, and 'Concurrent' which allows the program to continue executing while the motion is still carrying on.

It is possible to change the type of motion from 'Direct to Point' to a number of other settings by selecting them from the 'Type' list box.

The most useful of these is the 'Path to End', which allows you to define a path along which the object will move by dragging it from one location to the next. Each time you release the object a triangular marker will be displayed and you can then make a further movement by click and dragging the object again. Double clicking on the markers will convert them from an abrupt change of direction to a smooth one and back again.

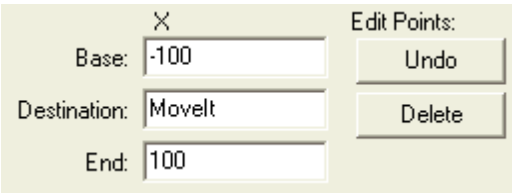


Individual markers can be selected by clicking on them, they can then be moved or if you wish to delete them, just press the delete key on the keyboard.

Once you have chosen a Movement Type the appropriate X and Y co-ordinates will become available. You can if you wish, type in numeric values and these will control how far along a path or to what screen co-ordinates the object will move. It is far more useful instead to enter the name of a variable and then to control the value of the variable with a little coding in a calc icon elsewhere.

As an example consider the above path and the corresponding movement values, this motion type is set to be 'Path to Point'. Two entirely arbitrary values are used for the 'Base' or start of the line and the 'End', you can use any values you wish, even

negative ones. A variable called 'MoveIt' is then used to show the position on the line where the object will stop.

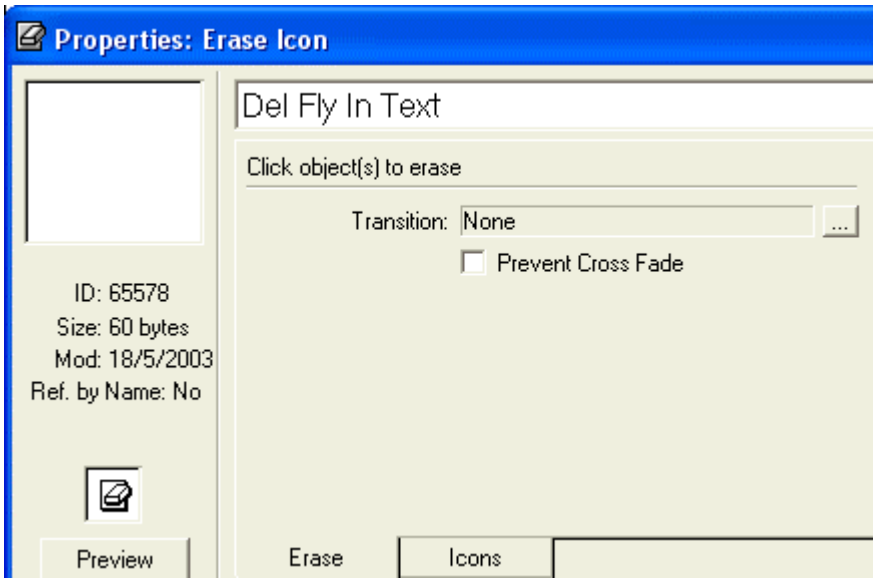


By making the value of MoveIt equal to something between the Base and End values the object will be moved a corresponding distance along the line, for example if MoveIt is set to be 0 the object will move half way along the line.

Similar techniques can be used for the other Motion Types in this icon.

Erase

The erase icon will remove any previously shown display icon. It is important to note that unless an icon is explicitly erased it will remain on the screen throughout the remainder of the program. The only exception to this is that display icons that are contained within interaction, framework, and decision icons are usually erased automatically when the program flow leaves the containing icon.



Placing an empty erase icon on the flow line and running it in authoring mode will cause the program to stop when the icon is encountered and you will be prompted to select the icons to be erased. You select the icons by clicking on them on the display area. You can erase as many displayed icons as you wish with a single erase icon. Remember that all of the contents of a display icon will be erased at the same time. If you want a part of a display to remain on screen then you must ensure that it is displayed in a separate icon.

The second tab on the dialogue window 'Icons' will show you a list of the icons to be affected by this erase icons.

You can either select 'Icons to Erase' or 'Icons to Preserve'. The first is fairly self-explanatory. If you choose the second then all icons visible at this point will be erased and only those selected will be left displayed on screen – this is not often used.

To remove an icon from the list, simply click on its name and then on the 'Remove' button. This is quite difficult to do if your icons do not have unique names!

The Preview button will show the icons being erased using whatever transition effect you have selected.

Erase icons have access to the same sort of transitions as display icons and the use of these should be considered subject to the same general advice.

Pause

The pause or wait icon will interrupt the program flow when it is encountered. By default a button is displayed, usually in the top left corner of the screen with the label 'Continue' displayed on it.

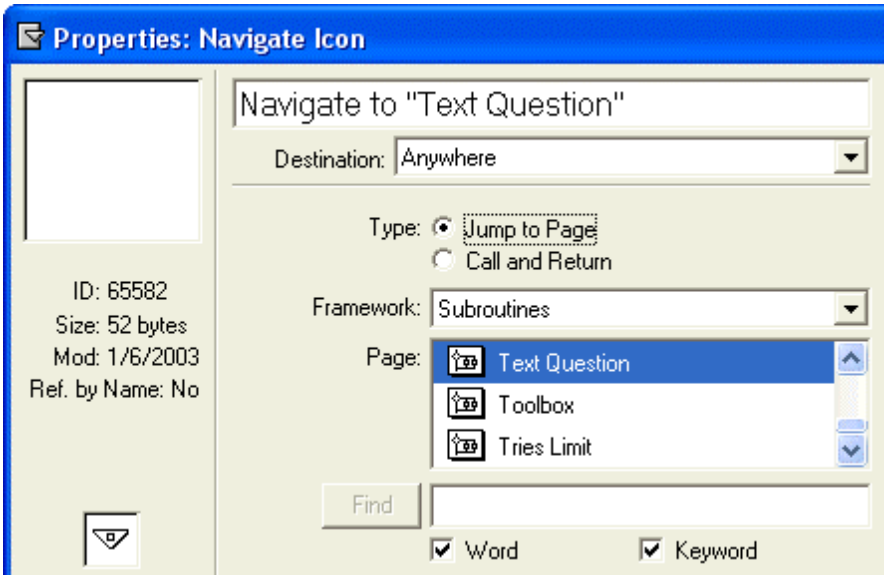
The position of the button can be changed by holding down the 'Ctrl' key and pressing 'P', which will pause the whole program. Then click and drag the 'Continue' button to the preferred position and finally press 'Ctrl' 'P' again to resume the program flow.

Double clicking on the pause icon will open a dialogue box which will allow the pause to be defined as being for a specified period of time, until a key is pressed or until a mouse button clicked. Any combination of these is allowed. You should always have at least one set.

If a time is specified then you can display a small countdown clock so that the student can be showed the time remaining.

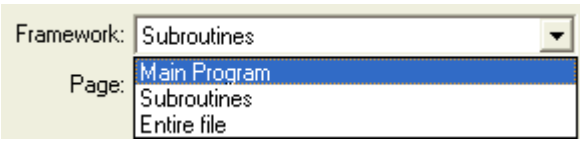
Navigation

The navigation icon is most often used within a framework (see below), where it is inserted automatically. To use it manually you drag it to the flow line and double click on it to specify where the flow should go. There are a large number of variations available for this icon and overuse can make it difficult if not impossible to follow your program flow later. The safest use of the navigation icon is to use it to go to named icons within a framework and this in fact is the default effect.



When Authorware encounters an unlinked navigation icon, or when you double click on one the Properties palette is displayed.

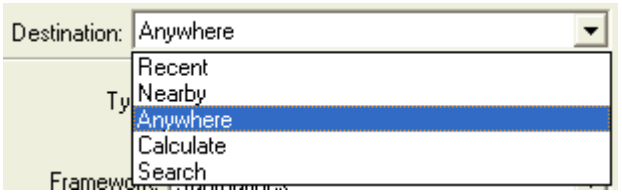
In this example the navigation icon will jump to a destination that can be anywhere in the file, the only restriction is that the destination must be a named icon attached to (hanging off) a framework icon. The program will go directly to the chosen icon and will not expect to return.



If you have a lot of frameworks and you want to narrow your choice down then you should select a specific framework from the drop down list in the Framework field. In this example there are two

frameworks, each of which could have many icons attached. If you choose 'Entire file' then you will see all of the possible destinations in the whole file.

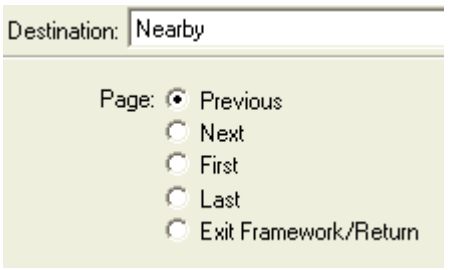
This is one of the most common uses of the navigation icon alone. It is almost always used inside a framework icon in one form or another.



In addition to using the navigation icon to go to 'anywhere' there are a number of other alternatives. The most useful of these is 'Nearby'.

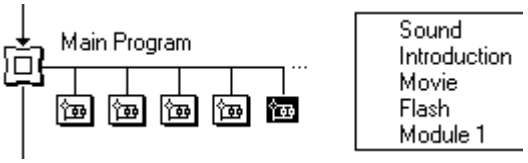
Selecting the option Nearby will change the appearance of the dialogue box and show a number of options, these all actually relate to what might happen inside a framework icon and in fact both the framework and navigation icons were introduced at the same time in version 3 of Authorware.

These options are fairly self-explanatory, they allow you to page backwards and forwards through the icons attached to a framework. The final option of the five will actually exit from the framework or if it has been entered with a 'Call and Return' option will take the user back to where they were before they entered it. This is particularly useful in a sub-routine (see below).



Framework

The framework icon is potentially one of the most useful icons and can be very simple to use. Drag the icon to the flow line and then drag either display or map (see below) icons alongside the framework icon. These will form separate pages within the framework and Authorware will automatically add buttons to allow the student to move from page to page, carry out searches and see a history of the pages viewed. If there are more than five items attached to a framework then Authorware will automatically add scroll bars to the list of names.



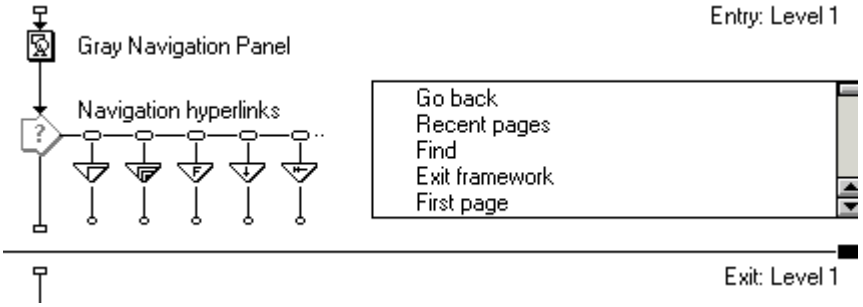
To select an icon just scroll until its name is visible and then click on the name to highlight the icon and double click on the icon.

Double clicking on the framework icon itself will open it up to reveal the structure within the framework, that will contain a display and various buttons on an interaction. You can treat these items as objects within the flow line and add, delete or change any of them in the same way as any other item within Authorware.

Note that the navigation icons on the interaction have been given specific functions such as 'Next', 'Previous' and so on as outlined in the Navigation section above.

The effect of the framework is as follows:

The top part, above the black line called 'Entry Level 1' contains icons that will either appear in or affect every icon that is attached to the framework, so for example 'Gray



Navigation Panel' and 'Navigation hyperlinks' and their associated buttons will appear on every display icon attached to the framework. The beauty of this is that you only need to construct a navigation function once and it will appear on however many icons are attached to the framework (there might be fifty or more).

The bottom part, below the black line called 'Exit Level 1' contains icons that will affect the program as it leaves the framework. In this particular case there are none, but you could include a calc icon that fr example recorded the fact that the user had visited this framework, which pages they had seen and how long they had spent there.

Remember that the default settings for the framework can be freely edited, removed or replaced by whatever is suitable for your application.

The default 'Gray Navigation Panel' is just a small graphic panel. The buttons attached to the 'Navigation hyperlinks' interaction are designed to sit on the panel and the whole looks like this.

These buttons and their associated navigations give you the immediate functionality to go to the next, previous, first or last pages, or to list recently viewed pages, to go back within the list of recently viewed pages, to execute a 'Find' function or to leave the framework altogether.

Once you become proficient in authoring you may well find that you will always customise the framework con, and the associated buttons and functions to match your specific needs, nevertheless the default values give you an excellent starting point.

A particularly useful facility within a framework icon is the ability to import text in the Rich Text Format (RTF). The text must have been saved as an 'RTF' file.

- Add a single display icon to the framework, double click on it and select the text tool.
- Place the text tool where you want the text to appear and if you think it necessary, select 'Text' and 'Scrolling Text' from the menu.
- Leave the text box open and available for text and select 'File' and 'Import' from the menu.
- Use the navigation dialogue box to find your text and select it.

The text will be imported into the display icon and every forced page break that is in your text will automatically create a new display icon within the framework. By setting up the text box first, all of the new display icons will have the text in exactly the same place.

RTF text can be created from within most word processing packages including Word 6, and Word 7, by using the 'Save As' feature and selecting 'Rich Text Format' from

the list of file types offered. (Forced page breaks are created in Word by holding down the 'Ctrl' key when pressing the 'Return' key).

You should be aware that Microsoft Word does not necessarily produce RTF files that are 100% compatible with the accepted standard. It is considered a useful exercise to actually open the file in WordPad (also from Microsoft and installed automatically with Windows) first and then save as RTF from that application to obtain the most accurate RTF format.

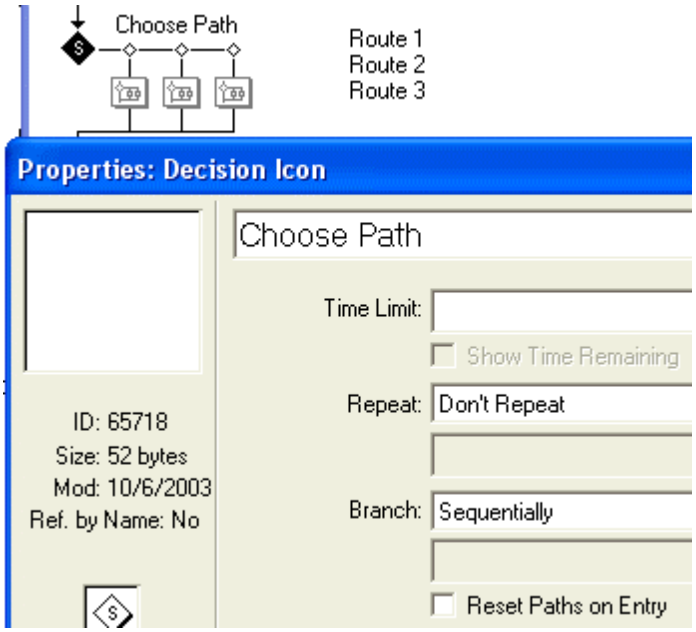
Sub-routines

This is probably a good point in the notes to mention sub-routines. These are powerful and highly recommended to save development time and to reduce the number of icons in your application. They are created by using a combination of navigation and framework icons.

Basically a sub-routine is a section of the application that contains one or more pieces of functionality that you need to repeat many times in the rest of the program. For example, you might want the user to type some text and evaluate what they have typed against a correct answer. This typing and evaluating might need to be repeated many times during the course and rather than repeat the necessary icon and code structure you can create it just once, place it inside a sub-routine and then use a navigation icon to call it when needed.

Decisions

Decision icons have a number of uses, they can allow the program flow to change depending on the value of some variable, they can cause the program flow to loop continuously and they can cause the program flow to take each of several paths consecutively.

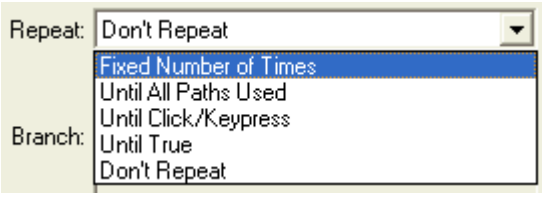


Drag the decision icon to the flow line and then add display or map icons alongside it. Double clicking on the decision icon will open it and allow you to make decisions about how or why the flow should change direction.

A typical situation might be after a question has been asked and marked, a correct answer will require one type of feedback and an incorrect one another. The decision icon can be set to go down the appropriate path as required.

A further example could be when there are several icons to be displayed at the same time, You could then set up this type of Decision loop, where every path will be taken until they are all used and none of them are erased

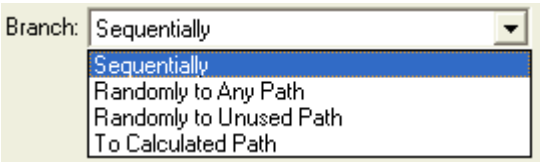
There are several options for repeating, ranging from 'Don't Repeat' to 'Until True'. These all have quite different effects and are used in different circumstances. Lets take a look at the difference between them:



- 'Fixed Number of Times' might be if you want to play a sound and display an error message, perhaps causing it to flash on and off. You can then enter a value in the field below the 'Repeat' field to indicate the number of times to repeat. This field can contain either a number or a variable.
- 'Until All Paths Used' would simply cause the program to pass down every leg in the decision structure, in turn until the last one had been completed. This might be to display the contents, or, you might have a set of questions in each leg and this would cause the user to be shown each question in turn.
- 'Until Click/Keypress' would allow the user to see something on screen and the program would then repeat until they clicked the mouse or pressed a key.
- 'Until True' would cause the program to loop in the decision structure until some condition becomes true. For example if you set a variable say 'Completed' to be false before entering the decision structure the user might then be detained within the loop until they have completed that part of the application, when you would change the value of Completed to be true and they could then move on. Beware of using this in circumstances where the condition might never become true. You could spend quite a time trying to figure out why the application doesn't just continue.
  The condition to be tested would be entered in the blank field below the 'repeat' field.
- 'Don't Repeat' means that the decision structure would be executed just once and the program flow would then continue. This is most often used when you want to display a message or feedback to the user depending on whether they answered a question correctly or not.

Note that the available paths are numbered internally by the program, the left most one is '1' the next is '2' and so on. By using variables with these values you can easily control the program flow.

The default route that the Decision icon takes is 'Sequentially' from left to right, depending on the 'Repeat' settings. 'Randomly to Any Path' means that you cannot predict which route will be taken next.



More useful is 'Randomly to Unused Path' which means that eventually all of the paths might be used but that no path will be repeated until they have all been followed. This is particularly useful if you want a series of questions to be asked in a random order.

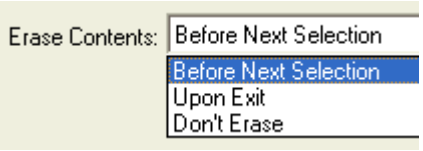
Another useful 'Branch' value is 'To Calculated Path'. This is commonly used in conjunction with 'Don't Repeat' in the repeat' field. By using a variable in the blank field below this, you can have a different set of feedback functions depending on how

the user has responded to a question, one for completely correct, one for partially correct and a third for wrong for example.

Remember that you can place a map icon alongside the Decision and that this can contain absolutely anything that you want, even effectively a whole module or course.

Each of the Decision Legs has its own properties, although these are primarily to do with whether or not the contents of that leg are erased when the leg is exited.

The options for this are similar to those for Interaction/Responses and have the same effect. If you choose the 'Pause Before Branching' a standard 'Continue' button is displayed before the leg is exited.



 Interaction

This is easily the most important icon in the program. By interacting with the student the whole learning experience is increased and expanded. There are a number of types of interactions and you are able to mix and match them as you wish. The ones that will be used most often are buttons and text, you can also use hot spots and hot objects and single key presses. The following list gives a brief explanation of the different types.

NB By double clicking on the interaction icon itself, you can use it in exactly the same way as a display icon. All of the information about display icons shown above applies equally to the contents of the interaction icon.

NNB All interaction response can be made 'Active If' some condition is met. This is extremely useful and as it applies to all responses is dealt with at the end of this section.

A brief introduction to the response types is given here and a more detailed explanation of the more common ones is given later.

Buttons Basic buttons can be used to give the student the option to select an item from a menu, or one answer from several choices.

There are however several different types of buttons, including multiple choice ones so that the student can make several choices.

See below for an explanation of how to customise buttons.

Hot Spot A hot spot is an invisible area on the screen, defined by you. When the student clicks on the hot spot some event is triggered.

Several hotspots might be defined in various parts of a picture and depending on where the student clicks different responses can be given.

Hot spots can only be rectangular in shape, but can be of any size that fits on the screen.

Hot Object A hot object is the visible part of something that has been displayed in a display icon. Every thing from the selected display icon is considered to be part of the hot object. For this to take effect you must select the object that you want to make 'hot'.

Target Area The target area allows you to define a specific part of the screen as a target and then allow the student to drag an object to it. When this has been done a response can be given.

Pull Down Menu A pull down menu is only available if you have defined the menu bar as used in the 'Modify', 'File', 'Properties' option when you set up the program. The title which you have given to the interaction will appear on the normal Windows style menu bar, and the title of each pull down menu will then

appear as an option on the menu when the student clicks on it.

This might give you the option to allow the students to print or save some information as they wish.

Conditional This is similar in effect to the decision icon in that depending on some condition a response will be generated. Perhaps if the value of a variable matches a condition then the option is selected.

Text Entry When you want the student to enter some text into the program this will allow you to require it. What the student enters will be compared with the title of the text entry icon and if it matches then the response will be given.

As an example you might ask the student to enter the name of their department and have several text entry icons to respond with a map for whichever one they enter.

If you want to allow the student to enter anything at all the use the asterisk '*' as the title of the last text entry icon in the interaction.

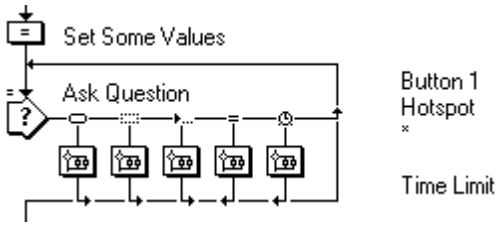
Key Press If you want the program to respond after just one key press then use this option. Each key that you wish the program to respond to must have its own separate icon, with that key as its title. If you want to respond to any key at all then use the question mark '?' as the last key press icon.

Tries Limit If you want to ask a question and respond after the student has had a certain number of tries then include a tries limit option within the icon and Authorware will automatically activate it if the student exceeds the permitted number of attempts.

Time Limit You may allow the student a restricted amount of time to respond to the interaction by including a time limit choice.

Event One of the more advanced features of Authorware is the ability to use Microsoft ActiveX controls. These are external to the main part of the program and can trigger responses automatically if you include an event option in the interaction.

Here is an admittedly unlikely combination of several types of response to an interaction. The different type of response are indicated by the different mini-icons on the horizontal section of flow line and the corresponding response names on the right side of the flow loop.

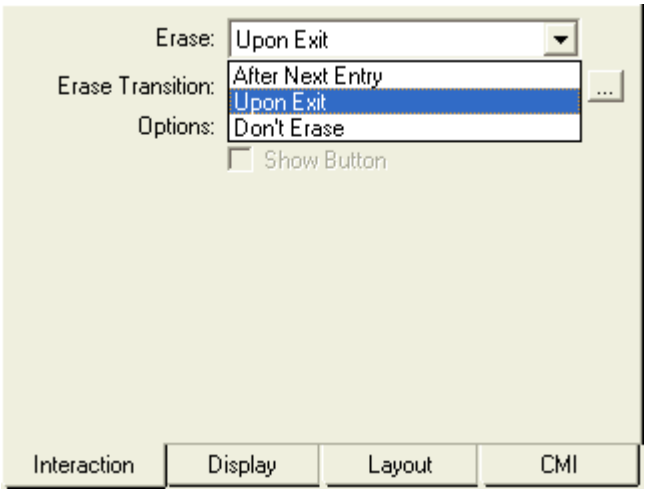


All interaction options can have several results, these can range from allowing the program flow to exit the interaction to repeating the interaction or continuing along the other options.

A quick way of changing the response result is to hold down the Ctrl key and double-clicking on the small arrow below the response.

Remember that Authorware does not by default delete the contents of the screen when a new icon is displayed. This holds true most of the time, but not necessarily when a display is made within either the interaction icon or within a display that forms part of the response to an interaction.

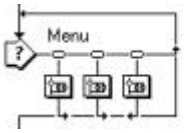
There are a number of ways in which you can control what is deleted and at what point within this part of the program the deletion takes effect. The default is for all of the items to be deleted when the interaction is left and for items displayed from within a response to be deleted when the next response to the interaction is made. Generally, these default values will cover most simple needs, however as soon as you start to carry out anything slightly more complex you will need to make some changes.



If you display the properties window for the interaction icon itself you will see that there are three basic alternatives to the Erase option.

'Upon Exit' is the default and will suffice to remove the contents of the interaction and any responses to it when the program flow moves on.

'After Next Entry' will erase the contents of the interaction icon itself as soon as the user matches one of the responses. This is particularly useful if you are using the interaction as a menu and want the contents of the menu to be removed when the user has made a choice. If this option is set then the contents of the interaction will be re-displayed when all of the functions and icons held within the response have been used, in other words, when the user has finished with the menu choice and returns back to the menu.



Introduction
Basic Concepts
Case Study

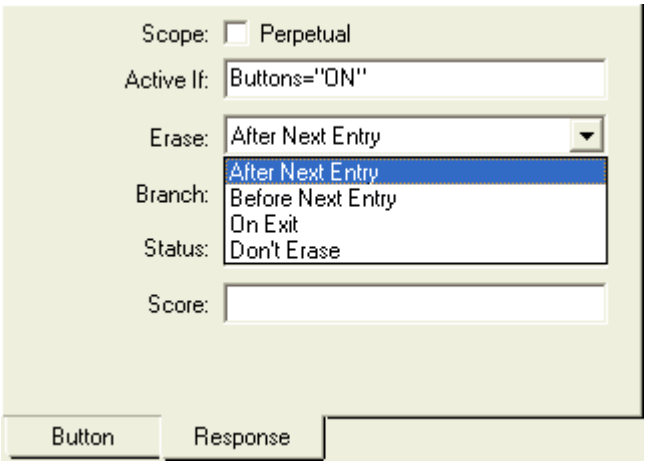
The final option is 'Don't Erase'. This will cause the contents of the interaction icon to remain on screen until you explicitly erase them with the erase icon. This option should be used with caution as you can easily forget that you have set it and then spend a lot of time trying to understand why something is displayed when you do not want it to.

You can also control if items are automatically deleted within the interaction. There is actually one more choice for the individual responses than there are for the whole interaction icon itself.

The reason for this is the need to be able to deal with the situation when a user selects one response after another for the same interaction.

The default is 'After Next Entry'. This option will cause the contents of the current response to remain on screen until the user selects a new option. This might for example occur when the user has chosen an incorrect option from a multiple-choice question and you wish the feedback to be visible until the user makes a further choice.

The second option is ‘Before Next Entry’ this is the opposite of the previous option. As soon as the user completes whatever is in one response its contents are erased as the user is taken back to the interaction. This is the option that you would usually choose when using the interaction icon as a menu. Once the user has finished dealing with one choice from the menu you would want all of that information to be removed before the menu is re-displayed.



The third option is the ‘On Exit’ option. This is used relatively rarely; the default option on the interaction icon would have the same effect. When the program flow leaves the interaction section the contents of the whole interaction including responses are, by default, erased.

The fourth option is again ‘Don’t Erase’. The caution about the use of this option is the same as that for the interaction icon, use it only if there is no alternative and be aware that if there is content on screen that you do not expect you will need to look for this setting.

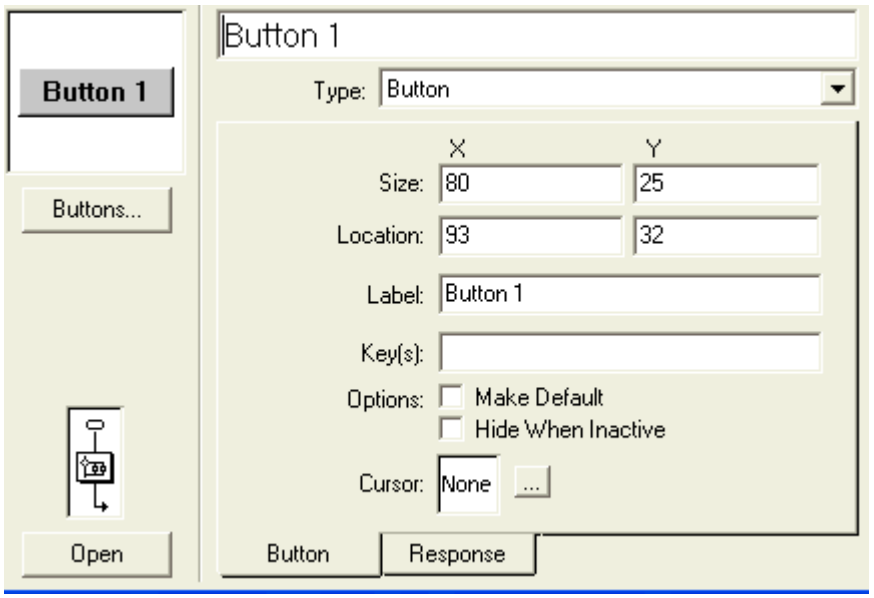
Lets now spend a little time examining some of the detail involved in the more common of the response items. Most users will quickly want to get to grips with Buttons, Text Entry, Hot Spots and Hot Objects and possibly also Target Area.

Most of the information to each of these is similar although there are some clear distinctions between them.

Buttons

A button is the easiest and quickest way to obtain some interaction with the user. A selection of standard Windows type buttons is included with Authorware and you are also encouraged to add your own buttons as needed (see below).

Certain buttons have specific rules that people expect you to follow, for example Radio Buttons should be mutually exclusive, while Selector Buttons should allow multiple choices to be made. It is usual to change the cursor to be the familiar 'hand' shape when it is over an active button.



By double clicking on the button symbol alongside the interaction icon you can display the properties dialogue window for that response. (Each response can have its own, completely different set of properties.).

Notice the field 'Label', this is the same as the name given to the response on the flow line and will also contain the same value as the text that will be displayed on the face of the button. The only exception is if you use the button editor and choose not to use text labels. (Version 6.5 of Authorware lets you use variables as button labels).

You can also use this properties window to change the type of response. This is currently 'Button', but from the Type drop-down list you can choose any of the available response types.

The size and location are given in pixels. If you are using the buttons provided with Authorware then the width will always automatically expand to fit the label inside the button face. You can replace these values with you own and you will find as you become more experienced with Authorware you will often replace these values with variables so that you can position or even move buttons on the screen as you need.

Note that Authorware considers the top left corner of the screen to be at co-ordinate 0,0, rather than the more usual bottom left corner.

The second 'Label' field matches the contents of the top field, you can amend either field and it will have the same effect.

The 'Keys' field is where you can allow the user to press a key on the keyboard rather than click on the button with a mouse. This is very important as you make your piece suitable for people who may have difficulty using a mouse, or be visually impaired.

You can allow the user to press one of several keys by specifying the keys separated by a vertical bar, (on the same key as the backslash symbol '\'). In Authorware this

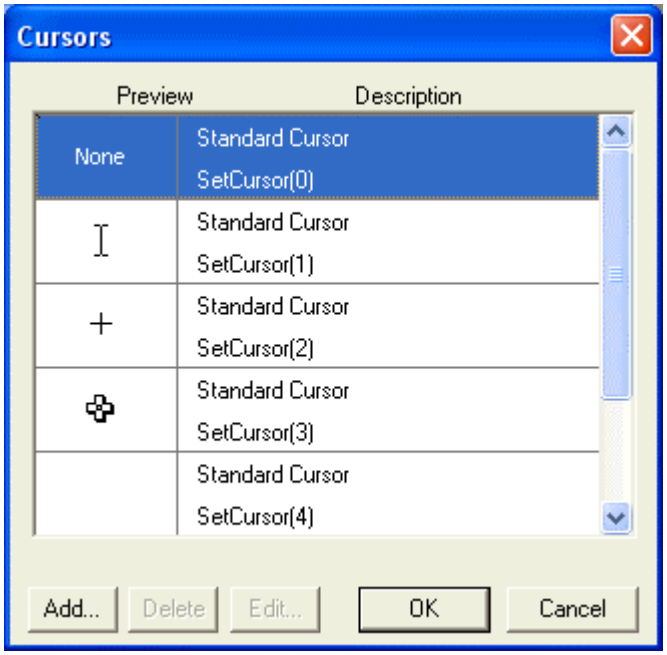
vertical bar actually means ‘or’ so in the illustration above the user can either click on the ‘Basic Concepts’ button or press ‘b’ or ‘B’ to enter that response structure.

The next two options, ‘Make Default’ and ‘Hide When Inactive’ just require a click to toggle them on or off.

‘Make Default’ has the effect that when you have several button responses on the same interaction the one that is set to be the default will be highlighted when the user enters the interaction and this option can be selected just by pressing the ‘Return’ key on the keyboard. It doesn’t make any sense to have more than one button set as the default on the same interaction and Authorware will only recognise the first one that it encounters when looking alongside the interaction icon (it looks from left to right).

The ‘Hide When Inactive’ option allows you to make a button completely invisible when you have chosen to prevent the user from selecting it.

The final choice on this screen is the cursor that will be displayed when the user points to the button with the mouse.



By default there is no cursor selected so that the mouse pointer remains the same. Most users now expect the cursor to change to the ‘Hand’ symbol when it points to something that is significant or ‘active’ on screen. You can choose which of several standard cursor shapes to display for each button, or you can design your own and import them into the piece.

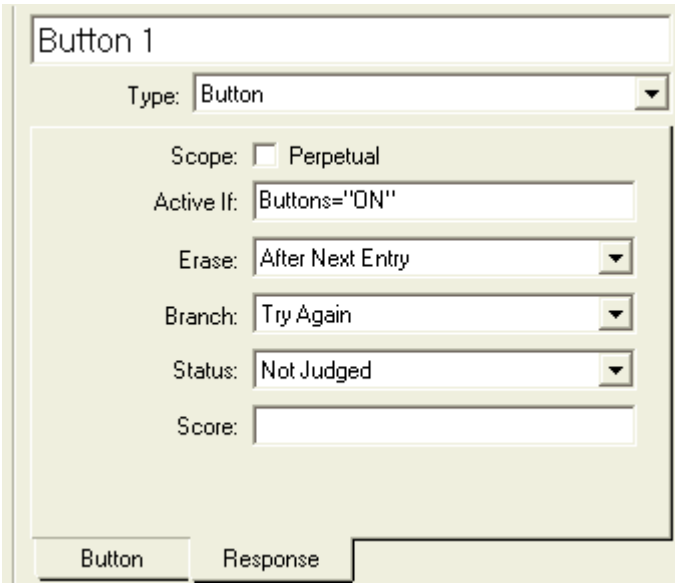
Clicking on the button alongside the cursor field will display the cursor dialogue window; you can then scroll down through the choices available (there are seven standard options) to choose the one that you want. The ‘Hand’ is the seventh on the list.

If you want and are able to design your own cursors (MicroAngelo is a shareware application that is highly recommended for this purpose). Cursors must be two colours only and a maximum of 32x32 pixels in size.

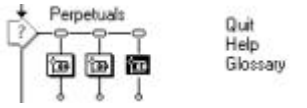
In addition to the properties shown on the ‘Button’ tab of the properties window there is a second tab marked ‘Response’ this allows us to set further properties for the button.

We have seen earlier what the 'Erase' options are. Lets now consider the other options on this window.

'Scope:' 'Perpetual' looks a little mysterious; it is actually a commonly used option. It allows you to ensure that the button is available even after one of the other options on the interaction has been chosen, or even in some circumstances after the interaction has been left behind altogether.

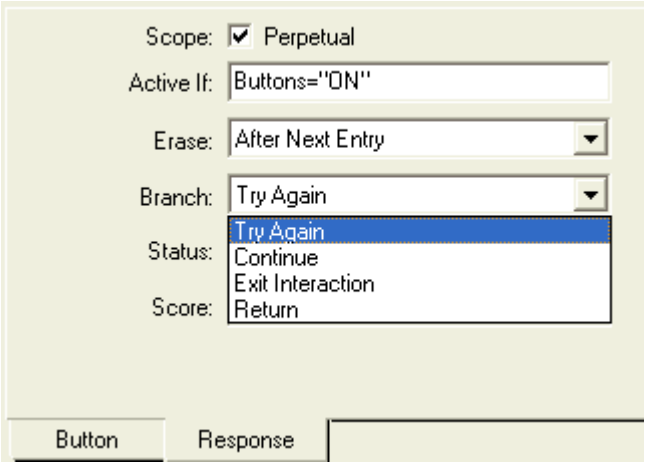


Why would you want to do this? Well consider how you are going to allow the user to leave the program. Generally you will provide a 'Quit' button. To be more 'friendly' to the user you would perhaps want this button to be available at any time within the piece. One way would be to have a quit routine attached to every single interaction throughout the whole piece, which would be inefficient and a nightmare to manage if something in the quit routine changed.

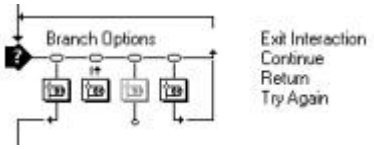


The solution to this problem would be to have an interaction set at the top of the flow line with a 'Quit' button attached to it, to set this response to be perpetual and then change the 'Branch' setting to 'Return' to allow the program to flow through the interaction without pausing.

The other setting is the 'Branch' option. You have already seen how you can change the branch setting by Ctrl and Double-clicking on the arrow below the response option, you can also change it within the properties window by clicking on the drop-down 'Branch' option.



Each of these options has a different effect on how the flow line looks alongside the interaction icon and therefore how the program moves through the available options. In this example you can see an example of how a flow line would look when all of the available branching options have been used on the same interaction.

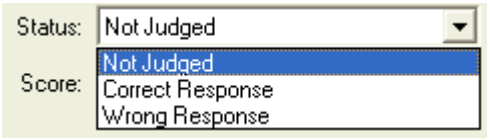


The 'Exit Interaction' should be fairly self-explanatory, the interaction is exited and program execution continues along the remainder of the flow line.

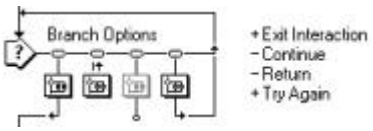
'Continue' causes whatever is in the map icon to be executed and when finished the program returns to the interaction to see if there are any other items that might have been matched.

'Return' is the option that allows the program to execute whatever is within the map icon and then return to whatever the program was doing when the user clicked on the option. Note that the 'Return' option **ONLY BECOMES AVAILABLE** when the 'Scope – Perpetual' option is set. It does not appear unless you first click on this option.

'Try Again' is the remaining option; this effectively bypasses any other options alongside the interaction icon and returns to the beginning. (Contrast this with the 'Continue' option).



The Status field is another drop-down list. Status in this context refers to whether a response to the interaction is classed as 'Not Judged', 'Correct Response' or 'Wrong Response'. This provides a beginner with a quick way of keeping a score of how many questions a user has answered correctly or incorrectly. The default status is 'Not Judged'



You can see the status of a response in the design window as the correct responses have a '+' sign next to their name, an incorrect response has a '-' sign and a none-judged response has neither.

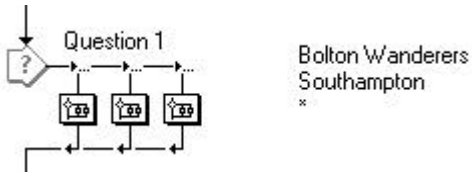
Just as Ctrl double-clicking below the response changes the branch effect of the response, you can also change the Status of each response by clicking to the left of the response title. The first click will mark the response correct, the second incorrect and the third as not judged.

The final option in this window is 'Score'. You can enter a number, a calculation or a variable in this window to indicate what score the user should get for making this choice. You could for example give a score of 1 for each correct choice and a score of –1 for each incorrect choice.

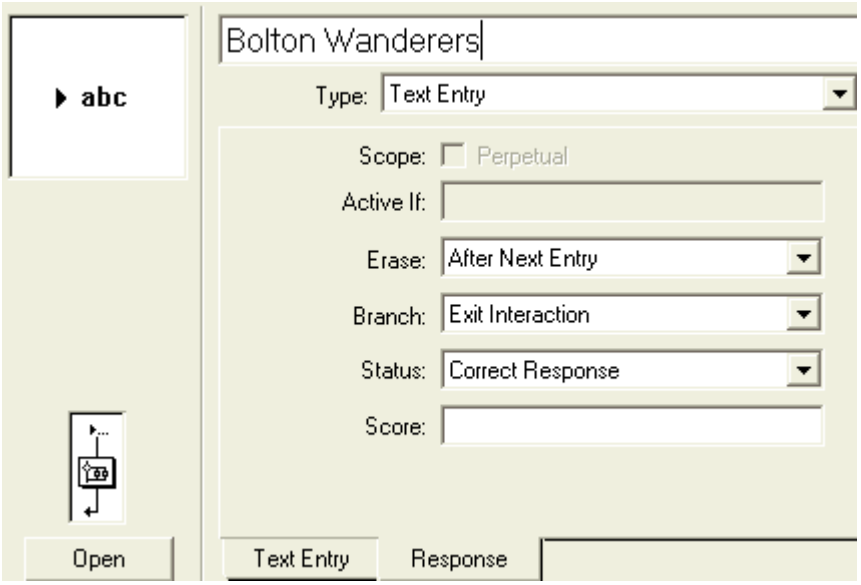
System variables called respectively 'TotalScore' and 'TotalWrong' will then keep a track of the score of the user. There are also system variables called 'PercentCorrect' and 'PercentWrong' that will also keep track of the percentage of judged interactions that the user has chosen. Although the system keeps track of these for you once you become more experienced with Authorware you will seldom use these and you will find that using your own variables is quite easy and much more flexible.

Text Entry

After buttons the most common interaction with users is through getting them to enter text into the computer. This might be in response to a question or it might be to allow them to enter their name or perhaps a password.



Text entry can also be marked as right or wrong just like buttons and indeed like all other responses. We will only consider the differences from this point on.



By double clicking on the text entry marker above each response you can open the properties window for that response. Remember that each text entry can have its own set of properties.

The 'Text Entry' tab of the dialogue window shows items that are specific to text entry. In particular the 'Pattern', which is what the user must type in order to match this specific response. So if the question is 'Which is the best team in the world?' the user might well type in 'Bolton Wanderers' as a valid response. The entry would then be evaluated, in this case as correct and the program would allocate a correct response and give the appropriate number of points to the user.

Generally the checking of text by a computer can be quite difficult as there are so many possible entries but for simple discrete answers Authorware copes quite well.

You can specify how many words from the correct answer the user must match, so for example if this value was set to 1 then 'Bolton' would be a valid answer.

'Incremental Matching' allows the user to get part of the answer right at the first attempt and then 'add words' to their answer in subsequent attempts.

You can ask Authorware to ignore capitalisation, extra spaces and punctuation as well as the order in which the words are typed and still accept the input from the user.

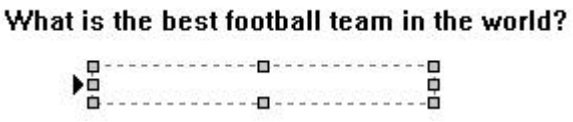
The items on the 'response' tab are exactly the same as for Buttons except that Text Entry cannot be made into a perpetual response.

If there are several possible or likely text entries that a user might make then you will need a response for each. Remember that only entries that match a response title will be accepted.

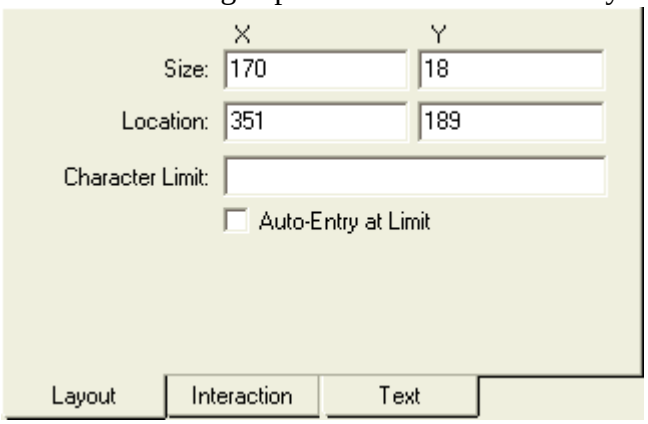
This of course would then cause a problem if a user entered some text that did not match any of the responses. To deal with this eventuality we always have as the last of the possible responses a separate response that has as its title an asterisk '*'. This acts as a 'catch-all', and you can then put something in that response that will deal with any other entry from the user.

No matter how many text responses you have for an interaction there will only be one text entry field shown on screen, they will all share the same input area.

To format and position the text entry field you should allow the program to run up to the interaction icon and press Ctrl and P to pause execution.



You will then see the text entry field and be able to edit its properties. As a start you could click and drag to position it where on screen you want it to be.



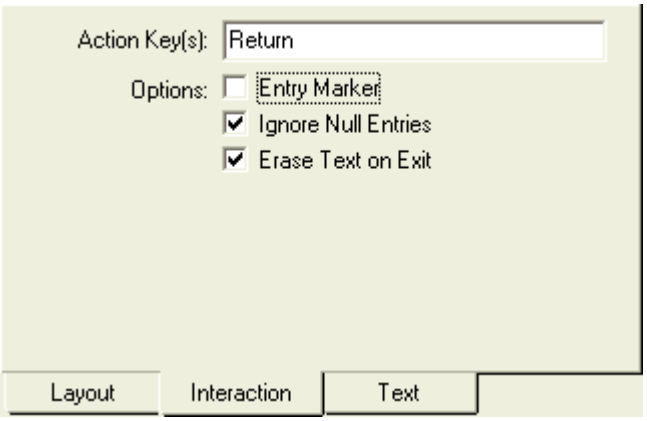
The amount of text that the user will be able to enter is basically controlled by the size of the entry field, so click on it to select it and then you can change the width and height of the entry field and make it as large as you need.

If you double click on the entry field you will be able to change a number of other properties of the field.

For example you can position it exactly by putting values into the X and Y fields for Size and Location.

If you only want the user to enter a certain number of characters then you can restrict it by entering a value into 'Character Limit'.

If you also click on the 'Auto-Entry at Limit' field then when the user enters as many characters as you have set in the 'Character Limit' field the entry will be evaluated as though they had pressed the Return key.



In the 'Interaction' tab there is a further set of properties that can be changed.

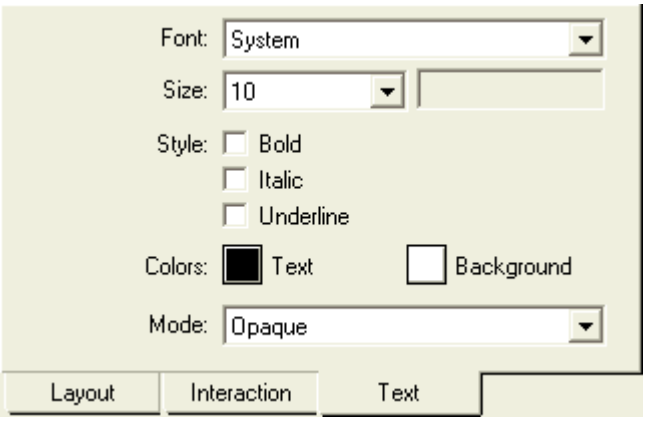
The most important of these is the 'Action Key(s):'. By default this is set to be 'Return', so that when the user presses the Return key the text will be evaluated against all of the possible text entries. You could also add to this field a vertical bar '|' and type in 'Enter'. Note that Authorware considers the Return and Enter keys to be different. This is also useful for those occasions when you are

simulating another program that for example might use the 'Tab' key to move between fields. A complete list of the key 'names' is available in the Authorware help system.

The 'Entry Marker' is the rather ugly triangle that Authorware puts by default alongside the text entry field. You might want to make a habit of removing this.

'Ignore Null Entries' will if checked cause Authorware to refuse to accept the user just pressing the Return key without making any text entry.

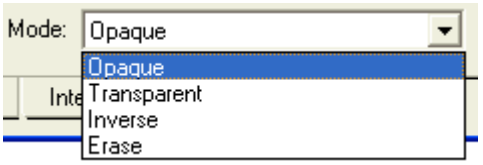
The final option on this window is the 'Erase Text on Exit', which will cause Authorware to delete the text after the program leaves this particular interaction. Switching this off will enable your text to remain on screen.



The third tab in the 'Text Field Properties' window is 'Text'. This is where you can specify in what format the text the user enters is displayed:

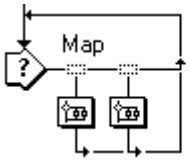
Note that you cannot use text style to format this text. You will need to specify the font, size, colour and style for each instance of text entry individually.

A useful option is the Mode. This allows you to set the mode in which the text is displayed. The most common of these is 'Transparent' in which the normal background of the display or interaction icon is allowed to 'show through' the area where the user enters the text.

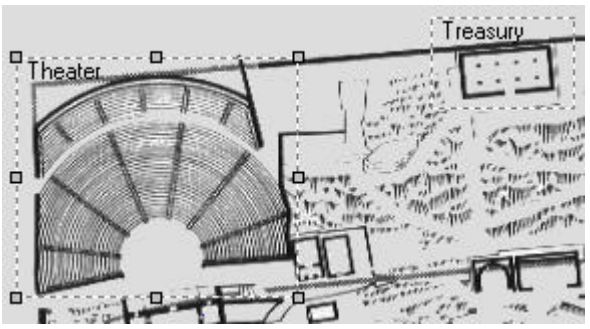


Hot Spots

You will often need to allow the user to click on a specific place on the screen, maybe a location on a map, or a place on a photograph. This sort of place is called a 'hot Spot'.



Hot spots are simply responses set after an interaction icon. They conform to all of the same functionality as button responses – they have a location, and dimensions and can cause a change in the cursor shape. Their biggest difference is that they are effectively invisible on screen in the normal running of an application.

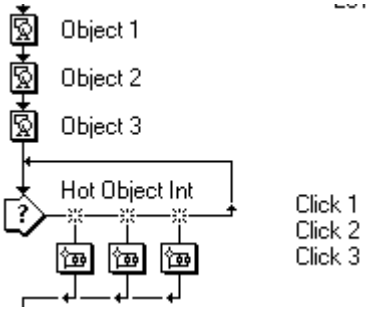


The easiest way to edit the details of the hot spot response is to pause the program when one is encountered (Ctrl – P) and the hot spots effective at that point in your application will be displayed, on screen as dotted areas. They will have names as defined in the flow line. You can select the hot spot by clicking on it, in this example 'Theater' is selected while 'Treasury' is not. You can then use the grab handles to move, or re-size the hot spot as needed.

Hot spots are always rectangular in shape. If you need a non-rectangular shape then you will need to use a hot object.

Hot Objects

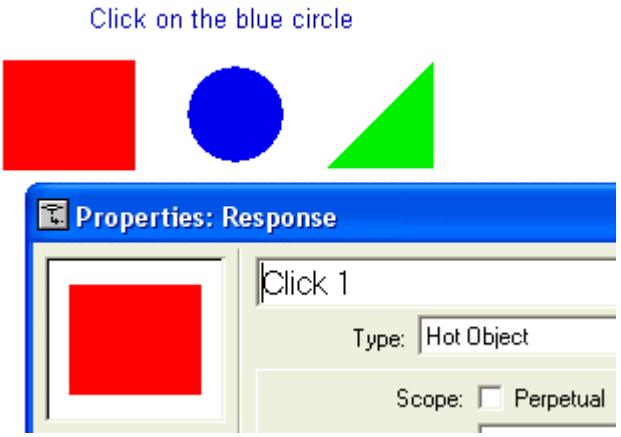
Sometimes confused with hot spots, hot objects do have some similarities but should not be mistaken with them.



For the purposes of hot objects, an object is ALL of the contents of a display icon. You can have as many hot objects as you wish as a response to an interaction icon, but remember that if the display icon contains more than one thing, e.g. text as well as an image then both the text and the image will be considered 'hot'.

In this example we have three display icons, each of which is active within the interaction.

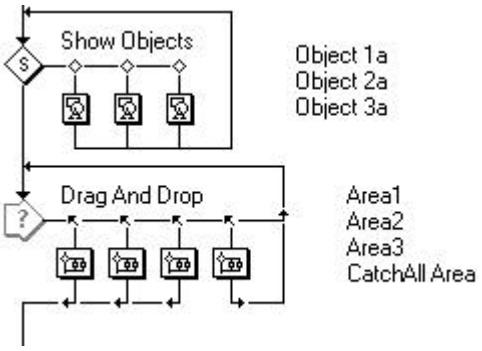
When Authorware encounters an interaction with unassigned hot object responses it will pause and allow you to indicate which specific icons should match which response. You make the connection simply by clicking on the object on screen that you want to associate with the response.



The effect of this response is then almost exactly the same as that of a button. You can define a cursor and also associate some code or other feedback to happen when the user clicks on the hot object. The one thing that you cannot do is define the size and location of the object within the response – this is obtained from the display icon itself.

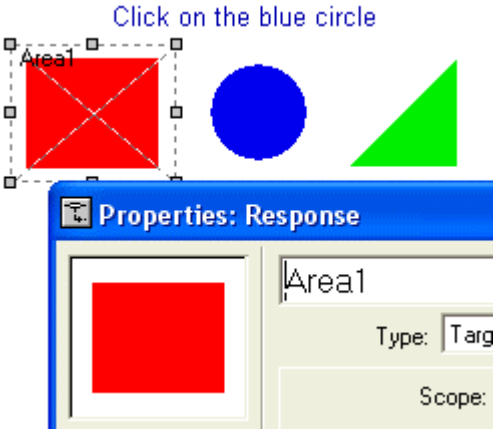
Target Area

This allows you to set up a 'drag and drop' type interaction. Similar in a way to 'Hot Objects' in that the response will involve all of the contents of a display icon, the difference is that you can allow the user to move the contents of the object on screen and respond when they drop it in a specific location.



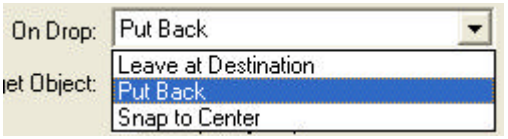
In this example there are three objects that could be moved, but there are four target areas defined.

When Authorware reaches this interaction it will stop and allow you to associate each target area with one display icon. You do this in a similar way to defining a Hot Object, simply click on the contents of the display icon that you want to associate with each area. Because there are four areas Authorware will ask the same question four times.



In response to each question, click on the object on screen that you wish be associated with the target area. The target area will then move on screen so that it is centred over the object and a small representation of the object will be displayed in the dialogue window. You can then move the target area itself to the correct location on the screen. This will be the spot into which the user will be expected to drop the object.

You can also set the action that Authorware will take after the object is dropped. Generally this would be 'Leave at Destination' or 'Snap to Center' depending on how you want to deal with the response.



Each object can normally only be associated with a single target area. The exception to this is the 'Catch All Area'. You don't have to have one of these, but it is useful if you want to automatically move an object back to its original location when it is dropped in the wrong place.

To set up the catch all area, make sure that you click in the 'Accept Any Object' field, and also that this response is located as the last response on the interaction (so that the user has an opportunity to make a correct choice before the catch all takes effect).

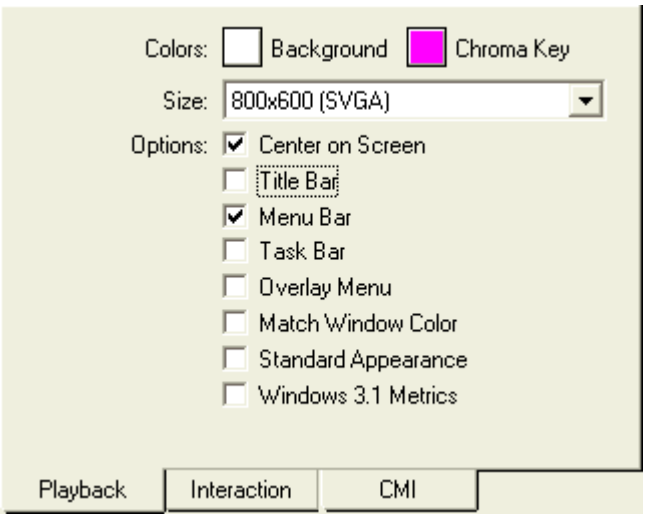
Then enlarge the area on the screen (the dotted lines can be moved and re-sized with the corner grab handles) so that it covers the entire screen.

The effect of this is that when any object is dropped on screen in anywhere other than the correct area the catch all area will take effect. If you then change the 'On Drop' field to contain 'Put Back' Authorware will move the object back to its last place on

screen when it is dropped – useful if you want to ensure that the user drops the object only in a correct place.

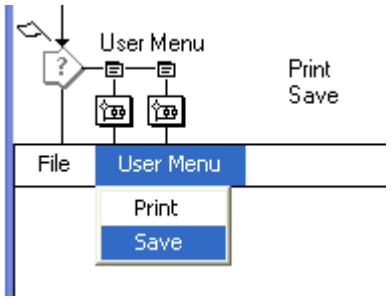
Pull Down Menu

Many user are now quite familiar with the use of menus and it can be reassuring to them if they see a menu that contains simple commands, such as ‘Save’, ‘Print’ or ‘Quit’



Setting up a pull down menu in Authorware is quite easy provided that you remember to set the file properties to include ‘Menu Bar’ (Modify – File – Properties). You do not have to include the Title Bar, but you must include a tick in the ‘Menu Bar’ field, otherwise no menu will ever appear.

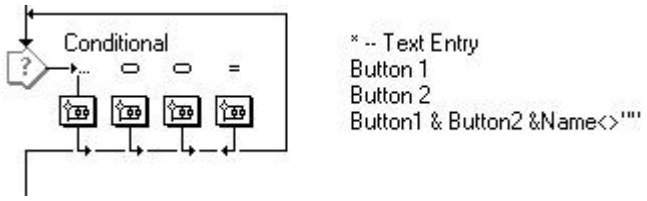
Pull down menus can contain whatever interactions you want; they are in effect just another response. However it is usual to have them contain the code to carry out utility like tasks, such as load or save a file, print some information or quit from the program. For these to work as people expect them and for them to be available throughout most of the program you might want to put the interaction at the top of the flow line and set the responses to be ‘Perpetual’ with a Branch option of ‘Return’



The user will then have access to the option for all of the time that the program is running as they would normally expect.

Conditional

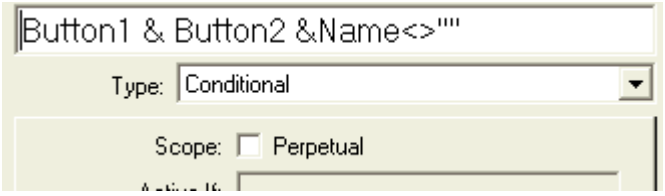
This response is much more a developers response rather than one over which the user has any direct control. Imagine a situation in which a user is required to do two or more things in the same interaction, for example type in some text and then click on two buttons.



You would need to know that all three things had been carried out before allowing the user to leave the interaction. Generally the condition would be set to consider the value of one or more variables, so for example you could check that a counter had been reached, or a flag set indicating that all three conditions had been met.

Notice in the example above that the condition, 'Button1 & Button2 & Name<>"" ' appears in the title of the response. This also appears in the dialogue window. You may edit the condition in either place.

This example would work when the user clicks on Button 1 and a variable called 'Button1' is set to be true by some code in the response area (MyButton1:=True) and a similar thing happens when the second button is clicked. Finally, when the user enters some text in the text response and presses the Enter key, a variable called 'Name' is set to be equal to whatever has been typed (Name:=EntryText) and the condition is then set and the program flow will follow the condition line to exit from the interaction.



Note that in this example it does not matter in what order the buttons and text are activated.

You might also want to choose at what point the condition is tested. There are three options, for this.

The condition can be 'Off', i.e. it will not take effect, 'When True' it will trigger at any stage that the conditions are met, even if some conditions are met as the user enters the interaction, and finally 'On False to True' the condition will only trigger at the time that the condition changes from being false to being true – so that if the conditions have been set before entering the interaction, the condition will not trigger.

It is your responsibility to ensure that the appropriate conditions have been set up before the program reaches the interaction, depending on what you want to do.

The conditions that you might consider using are basically simple, but you can make them more and more complex as required by extending them. Here are some examples:

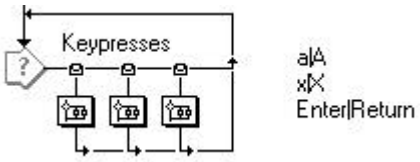
Condition	Meaning
MyVar = 27	The value of MyVar is exactly 27
MyVar >= 27	MyVar is 27 or more
MyVar <= 27	MyVar is 27 or less

Condition	Meaning
MyVar <> 27	MyVar is not equal to 27
MyVar	MyVar is True (contains any value other than 0)
MyVar & My2ndVar	Both MyVar and My2ndVar are True
MyVar | My2ndVar	Either MyVar or My2ndVar are True
MyVar = "Agrees"	MyVar has the value "Agrees"

There are many more complex ways of expressing conditions, but these will form the basis of most requirements in the early days of using conditional responses.

Key Press

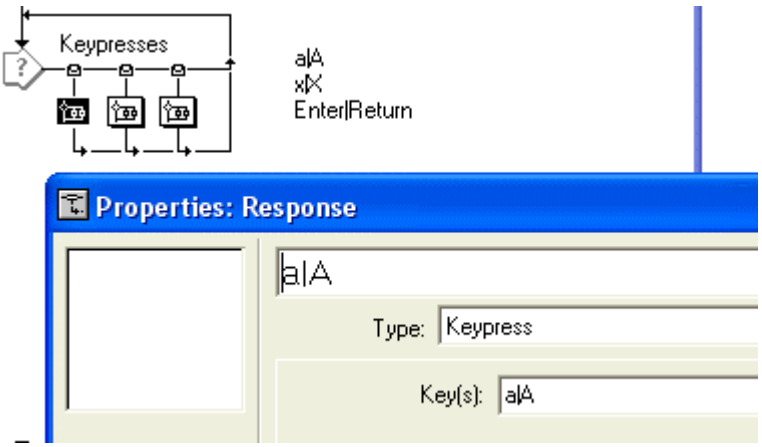
Sometimes you will want the user to simply press a key on the keyboard and have Authorware respond in some way that depends on what key they have pressed. An example might be a menu where the user has to choose by pressing 'H' to view the Help screen.



There are a number of points to note about this particular response.

Firstly it cannot be made perpetual; if you want a key press to be perpetual you must use a button and assign a key press to the button. The button can then be located somewhere off the visible screen so that the user will never actually see it but the key associated with the button will still be available. Note that this technique can cause some problems if you also have a text input response as the key press will take precedence over the text – you will need to make the perpetual inactive using the 'Active If' technique described below.

Secondly, you should take account of the user's likelihood of pressing either the upper or lower case letter. Authorware correctly considers these to be different keys, you can therefore allow the user to press either by separating them with a vertical bar '|'. This is usually found on the backslash key on your keyboard '\'.



You can use the same technique to allow the user to press alternative keys for the same response, so for example 'a|A|b|B|c|C' would trigger the same response if the user pressed a, b or c in upper or lower case.

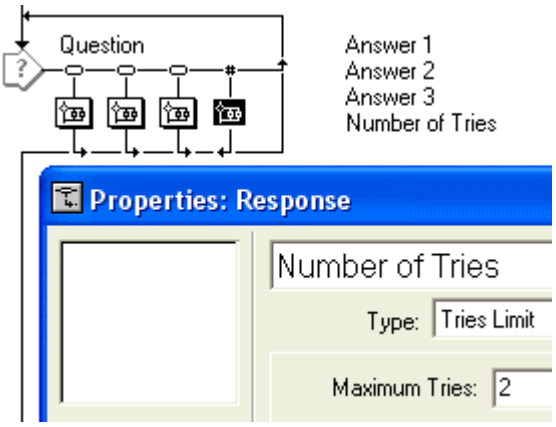
Thirdly certain keys actually have names, for example 'Return' and 'Enter' are identified with the full words – Authorware considers these to be different keys. Other named keys are 'LeftArrow', 'RightArrow', 'UpArrow' and 'DownArrow' are examples, as are 'Tab', 'Home' and 'End'. (These names are not case sensitive).

To accept any key press, use the question mark '?'.

Tries Limit

This response is very useful if you want to allow a user to have a number of goes at answering an interaction but easily limit just how many.

You can add this response to your interaction in the same way as any other. The only property that you would be likely to want to set for this response is the maximum number of tries that the user is allowed.



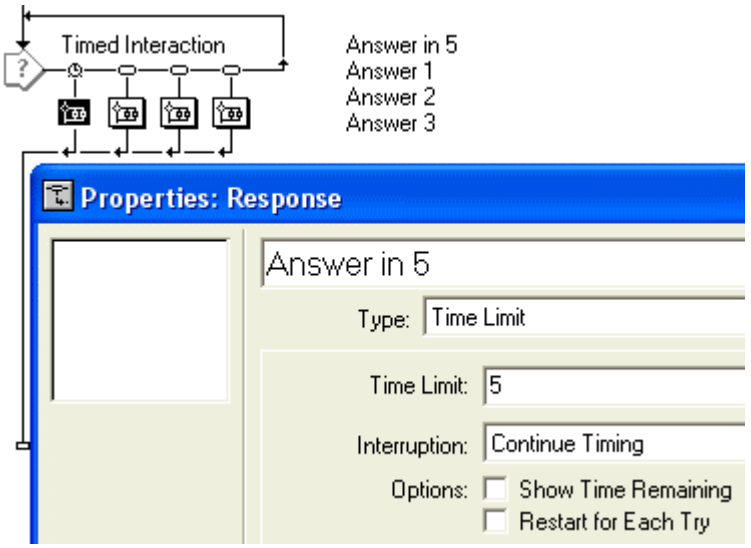
You could if you wished use a variable in the 'Maximum Tries' field, for example if you wanted to change the number of attempts allowed depending on the previous experience of the user.

This response cannot be set to be perpetual.

Time Limit

This response is very useful if you want to limit the amount of time that a user is allowed to respond to an interaction

You can add this response to your interaction in the same way as any other. The main property that you would be likely to want to set for this response is the amount of time allowed, in seconds.



You could if you wished use a variable in the 'Time Limit' field, for example if you wanted to change the time allowed depending on the previous experience of the user.

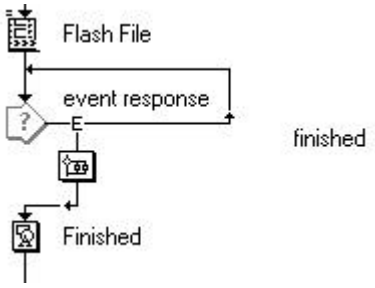
By setting the 'Show Time Remaining' you can cause the rather ugly little alarm clock to be displayed.

If you set 'Restart For Each Try' to be on then the clock will re-set itself back to the start each time that the user makes a choice from the options.

This response cannot be set to be perpetual.

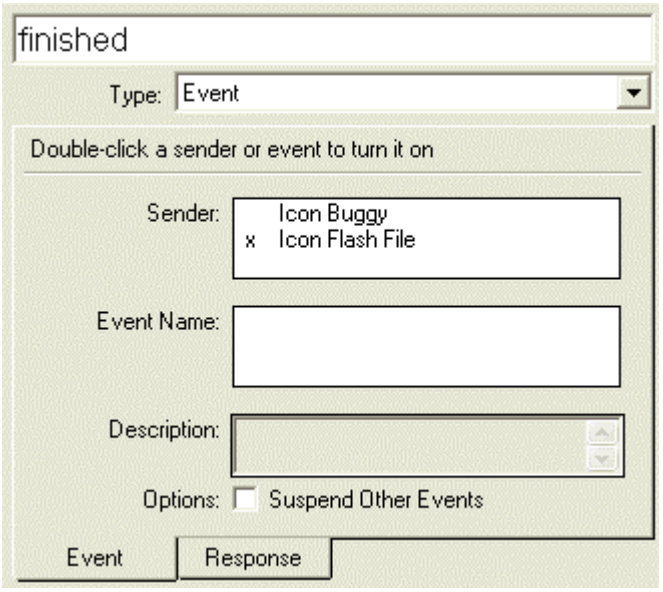
Event

The event response is basically designed to deal with non-Authorware applications that are running within your piece. Such things as ActiveX or Flash movies will carry out some internal action that will cause an 'event' to be passed to the Authorware piece which will then trigger the action within the piece. These are usually rather advanced topics and while they are somewhat beyond the scope of a basic course a simple structure is shown here.



A Flash movie has been inserted into the flow line using 'Insert Media – Flash Movie' it has been given the name 'Flash File'. This simple movie places an object on screen, moves it over 50 frames and then passes a message using the Flash function 'GetURL'. Authorware picks this up and the event is triggered.

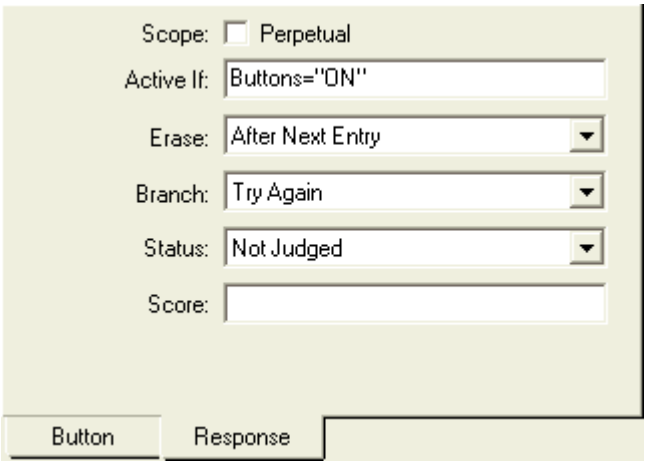
There are a number of properties that can be amended relating to the event, its name, type and timing. They however depend on the originating object sending a message in a recognised form. Basically you need to know just what is being passed and when to use this response.



Active If

On a number of occasions above reference has been made to making responses active at certain times. In fact the default value is that all responses are active when their parent interaction icon is active. There will however also be many occasions when for one reason or another you want to disable a response, for example a 'Continue' button may be disabled until a user has completed some task.

Unless you specifically set an active condition therefore, you can take it that the response is active. As soon as you set a condition the response will be inactive unless the condition is met.



Lets consider a simple example. You have a button that you want to switch on or off depending on what is happening on screen, maybe a movie is playing and at this time you want to make sure that the user cannot move elsewhere in the program until the movie has completed.

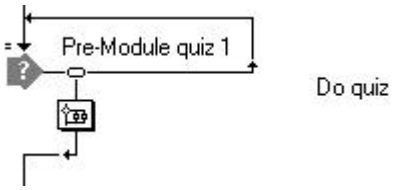
Define a variable, lets call it 'Buttons'. Before the movie starts, set the variable to be 'Off' (Buttons:="Off"), then play the movie and once it is completed, set the variable to be 'On' (Buttons:="On").

If you now set the 'Active If' field in the response properties to be 'Buttons="On" ' the button will be active or not dependant on the value of the variable 'Button'.

You can use this technique on any response. Even one that is set to be 'Perpetual', so for example a 'Help' option could be available throughout the majority of the program but be disabled during a quiz.

Calculation

The calculation icon (usually called 'calc' icon) is where most of the programming is done. There are apparently two different types of calculation, the standard calculation icon that stands alone and is visible on the flow line and one that is attached to any other icon and is indicated by an equals sign '=' displayed alongside the other icon. In practice there is no difference between an entry in a standalone calc icon and the same entry in an attached calc. You can use either completely interchangeably.



It is within the calc icon that you can control variables and use functions. These are explained in more detail later.

 Map

The map icon is essential in your programs, but it does not actually do anything itself. It is a container icon that contains any of the other icons. It often contains other map icons and can be 'nested' in this way to any depth.

Double clicking on a map icon will open it to show its contents.

A group of other icons can be inserted into a map icon by highlighting them all at the same time (click and drag across them as a group) and then hold down the 'Ctrl' key and press 'G' (for Group).

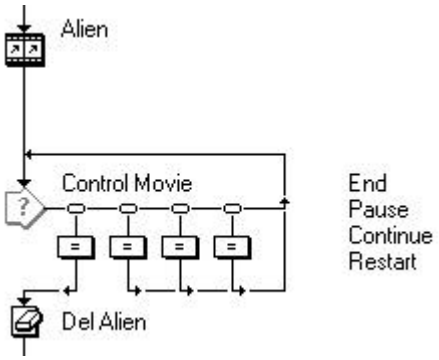
To 'ungroup' a map icon, select it and then hold down 'Shift', 'Ctrl' and press the 'G' key.

Be aware that deleting a map icon will also delete all of the icons that it contains.

Movies

The movie icon allows you to import video clips in a number of standard formats, these include AVI, MOV, FLC and FLI. DIB sequential files can also be treated as movies for these purposes. In each case you will need to identify the file you wish to include. You can control such features as the start and end frame of the movie and its playback speed.

Movies are displayed in front of anything else on the screen and while they must be explicitly deleted you will find that transitions for display and deletion have no effect.



There are a number of controls that you can impose on a movie, including pausing it, re-starting and causing it to go to a particular frame. These could be controlled within an interaction arrangement as above. For your information the contents of the four calc icons shown above are listed here:

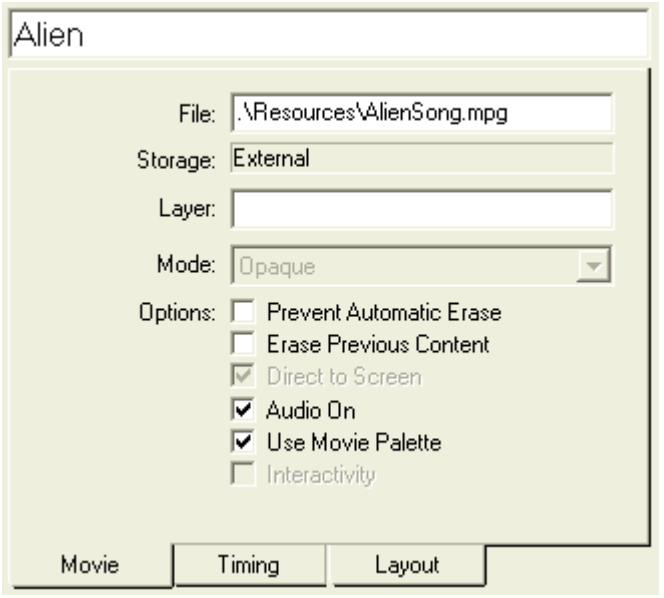
End MediaSeek(IconID@"Alien", MediaLength@"Alien")

Pause MediaPause(IconID@"Alien", TRUE)
MyPause:=TRUE

Continue MediaPause(IconID@"Alien", FALSE)
MyPause:=FALSE

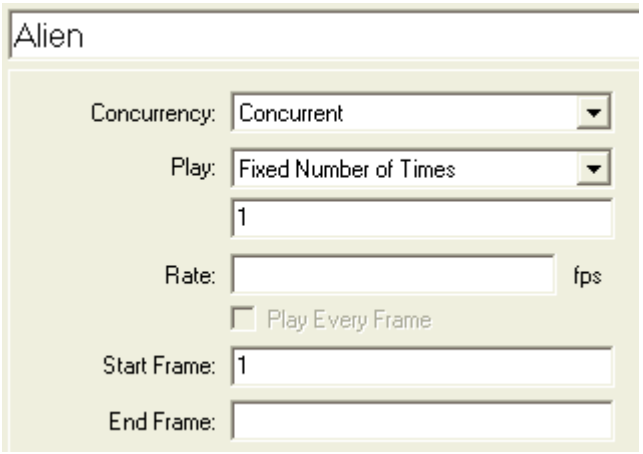
Restart MediaSeek(IconID@"Alien", 0)
MediaPlay(IconID@"Alien")
MyPause:=FALSE

Note that the Pause and Continue buttons are physically located exactly on top of each other, that their properties include a 'Hide When Inactive' and that they are set to be inactive when MyPause is False and True respectively.



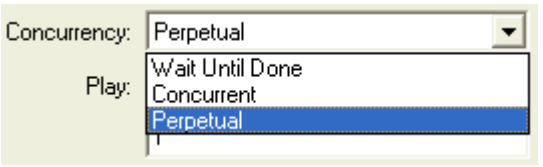
Other properties of a Movie icon that you should consider are, how the movie is to be played; its concurrency and how many times it is to be played.

The drop down menus controls these options.

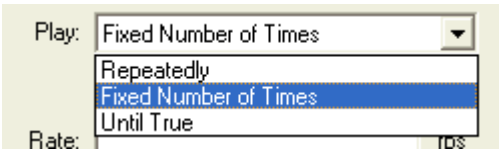


Concurrent means that the program flow continues, Authorware can do other things while the movie is being played, in particular to utilise the control framework shown above the movie must be set as Concurrent, otherwise the program would pause when the movie starts and wait for it to finish before reaching the controls.

'Wait Until Done' means what it says; Authorware will stop all other tasks until the movie has reached the last frame. Perpetual means that the movie will be available even after Authorware has left that part of the application. You can then use variables to control the movie from elsewhere in the program. It is not available when display or map icons are 'attached' to the movie icon (see 'Sound' below for an example of this).



To control how many times a movie plays use the 'Play' field, generally this will be 'Fixed Number of Times', with the number in the blank field below set to the number 1. You can set the value to be anything you wish, or you could place a variable in the field and set the variable somewhere within the program to be the required number.



'Repeatedly' will cause the movie to keep playing until it is erased.

'Until True' will cause the movie to repeat until the condition that you would enter into the blank field was true, this might be something like 'Credits=True' where you would keep the movie playing until the credits for your piece have been shown.

If you leave the 'Rate' field blank then the movie's original speed will be used, otherwise you can show the movie at whatever frame rate you wish, by entering a figure, or a variable.

'Start Frame' would normally be set to 1 to cause the movie to start at the beginning, leaving the 'End Frame' field blank would cause the movie to play until its last frame.

Note that all of these fields can contain variables and also that by using the functions as shown in the control structure above you can also control the movie from within the program or even give control over to the user if necessary.

If you wish to use bitmap files as a movie, this is a very effective way to utilise a video effect, the way to do it is to create a sequence of bitmaps, preferably all using the same size and same palette, each following on from the previous one. They must share the same four letter start to their name and have a four digit number at the end of the file name, similar to this – vide0001.bmp, vide0002.bmp, vide0003.bmp…

You can then import the first of these as a movie and Authorware will automatically include the following files until it reaches the last one in sequence. Each image will

then be considered as a frame in the movie and can be controlled just as the normal movie.

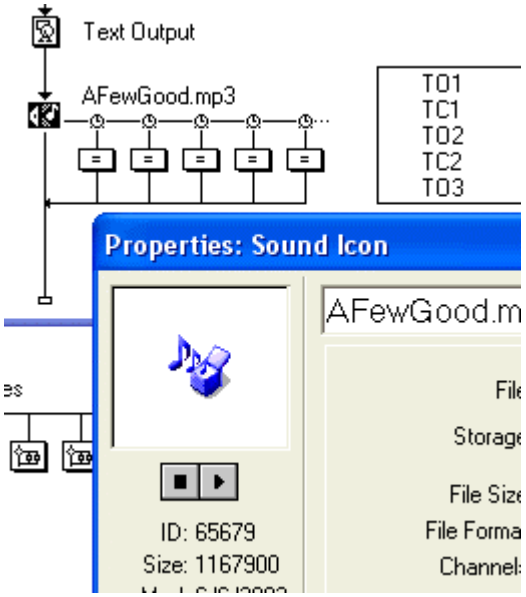
If you wish to show a movie and have displays occur at specific points along the movie then you can attach displays (or maps with several displays) alongside the movie icon and have these take effect at pre-determined points in the movie. These pre-determined points can be set to either time or frames within the movie. See the next section on Sound icons for an example.

IMPORTANT NOTES

- Movies by their nature are physically large files; their use should be given careful consideration and they should only be used when they actually add to the application that you are developing. In particular if you are developing a piece for distribution over the Internet think very carefully about file sizes and download times.
- AVI, MPEG QuickTime and Flash (see below) are always stored externally to the Authorware file. If you use one of these types of movie you will also need to distribute it along with the finished Authorware application.
- DIB, BMP FLC/FLI movies can be stored internally within the Authorware piece, but they will considerably add to the size of the finished, as well as being restricted in other ways – no audio is available with these file types.

Sound

Sound icons are very similar to Movie icons and can be placed in your application just like any other icon.

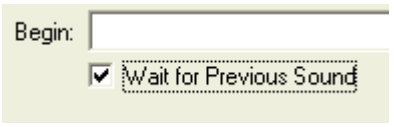


Like many forms of media you can import a sound file (usually a *.wav, or a *.mp3 file) directly into Authorware, or you can leave it as an external file. The choice is yours, usually if there are many sound files you would leave them as external files and create a link. This is done during the import process, simply by clicking in the 'Link to File' option box.

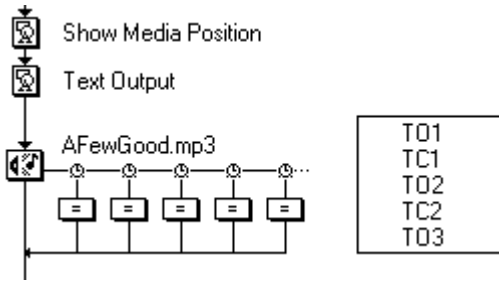
Remember that if you choose to leave the file as an external file then you must be sure to distribute it when the piece is finished and sent out to your users.

Once you have imported the file, or created the link to it you can then set the properties as to how it is played, in much the same way as with a movie. The playing rate – its speed, and concurrency can all be set.

One particularly important difference from Movie icons is the field, 'Wait for Previous Sound'. This will cause the program to pause if, when Authorware reaches this icon a sound from a previous icon is still playing. This helps to ensure that Authorware does not interrupt one sound with another. Authorware is unable to play two *.wav files at the same time unless you purchase an extension from one of the third party suppliers.



New to Authorware in version 6 is the ability to attach icons to both Sound and Movie icons. This is extremely useful in the synchronising of Display icons with an extended voice over, other sounds and even movies.



This attachment works like this. You place the media icon on the flow line as normal and then alongside it place Display, or even Map icons containing other icons. These can then be synchronised with the media as it plays. The synchronising can either be to time, or to the position within the file that is playing. (Position is probably more accurate as a lower performing computer playing the media at a slower speed does not compromise it).

In the example shown here, Calc icons have been used. A single Display Icon called 'Text Output' is positioned before the sound icon. This has a couple of variables displayed on screen in it. The property of the Display icon has been set to 'Update Displayed Variables'. Calc icons have been

attached to the sound icon and these are used to change the value of the variables at particular positions while the sound is playing.

When you are using Display icons attached to a media icon you have the choice as to when the contents of the Display icon are erased. This is also contained in the synchronise icon; it is very similar in extent to the erase options available with an Interact or Decision icon.

Video

The final icon is the video control icon. This is almost redundant and is intended to allow the program to control a video or videodisk player attached to the computer and showing its output on a standard TV screen.

This icon has been replaced in Version 7 of Authorware by the DVD icon.

Start and Stop

The white and black flags shown below the icons have no effect in the finished program but allow you to indicate where in the development environment the program should start and stop when you run it. These flags are very useful to test just a small section of the program without running through long sections that you know are correct.

Drag the flags to the appropriate points on the flow line where you need them. By clicking on the empty spaces in the icon bar the flags will automatically be returned to their 'home' positions.

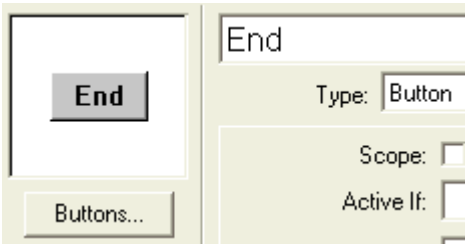
Colours

Although the colour palette displayed at the bottom of the tool bar does not change any of the properties of the application that the end-user will see, it can be very useful for the developer. You can decide for yourself how these colours are going to be used. A couple of examples might be:

- Each member of a development team could have their own colour assigned, so that they each know who is working on a particular section of the application.
- When an icon contains work that has been quality assured, it could be coloured green. Maybe keeping red for new work and blue for items under construction with orange for icons that have been QA'd but require corrections.
- Tutorials could be coloured brown and interactions in grey.
- And so on…

Button Editor

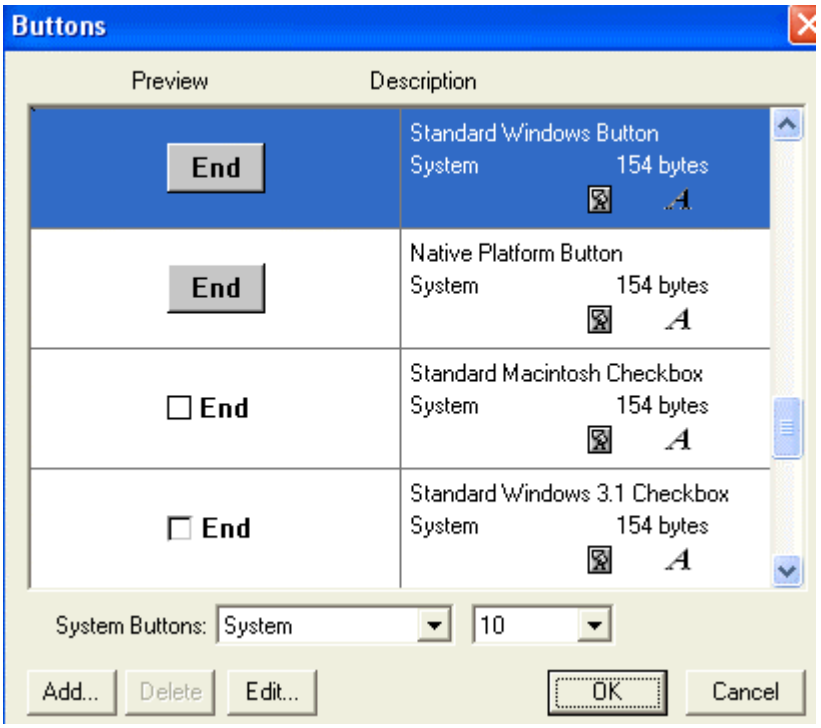
To call this facility a button editor implies that you can actually edit the buttons themselves within Authorware. This is not the case, what you can actually do is to import graphics created outside Authorware into the piece that you are creating and use them as though they were standard Windows type buttons.



You can access the button editor in two ways, either directly by clicking on 'Window – Buttons' from the menu bar, or from within the properties window of a button response to an Interaction icon.

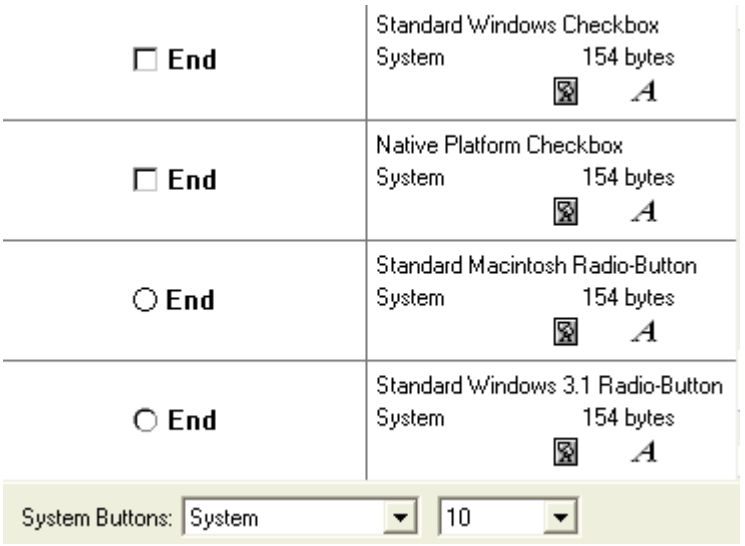
In either case you will see the same dialogue window.

This will show all of the available buttons that your piece knows about. By default these will be the standard Windows type buttons that you are familiar with from normal Windows operations. If any non-standard buttons have been defined as in this example you will see them in the list as well.



At this stage you can define which particular button you wish the current response to use. Just scroll up and down through the list until you see the one that you need, click on it and then click on the 'OK' button. Your choice will then be applied to the response that you were editing.

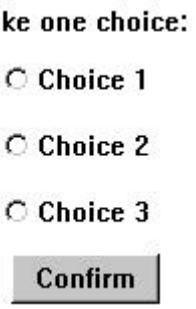
Certain button types should be treated as a special case. Generally most of the buttons that you use are of the 'Click once and forget' variety. You click on them, some function is carried out and then you move on to the next task. However this is not the case with a class of button called 'Radio Buttons'. You will have used these somewhere in your use of computers, perhaps without realising it. These buttons are designed to allow you to choose *one* from a range of choices, before confirming your choice. In this situation the choices are mutually exclusive – clicking on a different choice will de-select the previous choice.

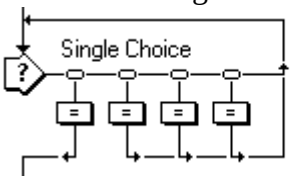


In the examples shown on the left, the *circular buttons* are classed as radio Buttons. The square ones are also choice buttons, but by convention you can make several choices when presented with the square buttons and only one when presented with the circular ones.

Authorware allows you to use all buttons in exactly the same way. To match your piece to the accepted Radio Button standard you will have to do a little work.

First build the Interaction with the required responses. In this example we have a simple question, three possible choices and a confirmation button. On screen the user will see the example shown on the right.





What is happening in the background is that a structure like this shown on the left has been created.

The actual work to make the buttons behave in the correct way is done in the Calc icons below the three responses. In the 'Choice One' response we have this simple code:

```
Checked@"Choice Two":=FALSE
Checked@"Choice Three":=FALSE
```

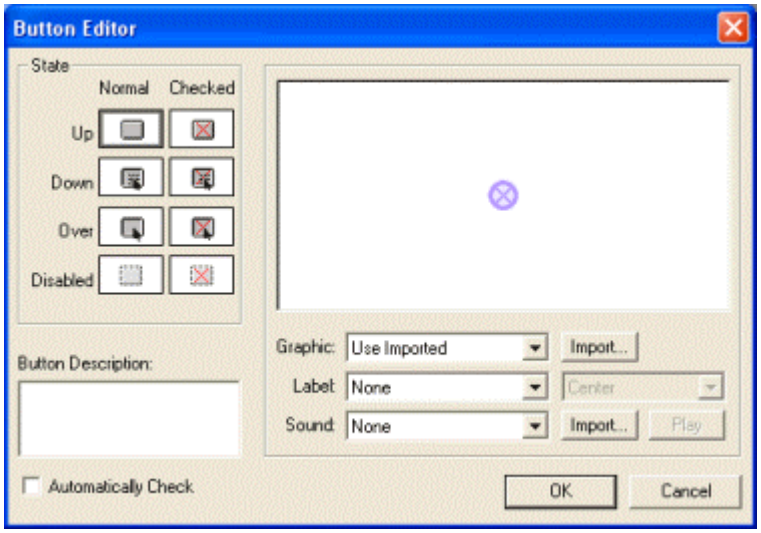
The system variable 'Checked' contains either True or False for the button named immediately after it. In this case what happens when the user clicks on 'Choice One' is that the code in the Calc icon gives Checked for the other buttons on that Interaction a value of False, which removes the marker for them.

A similar process is used to 'unset' 'Choice One' when the user clicks on either of the other two buttons.

Finally when the user clicks on 'Confirm' the Interaction is exited and the Calc icon under 'Confirm' can then carry out whatever further processing you require.

Adding Your Own Buttons

It is easily possible to add in your own design of button to your piece, using the Button Editor.



When the editor window shown above is on screen, click on the 'Add' button and you will be shown the button state window.

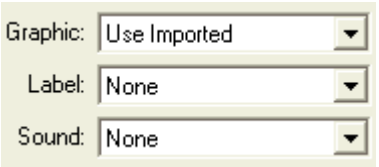
Notice that there are eight possible 'States' for a button. Generally you only need to use three or four. The buttons should generally be designed in a separate graphics editor; Fireworks, Photoshop or whatever you normally use.

Design your button to have an 'Up' state, which is its normal appearance, a similar but slightly depressed appearance for the 'Down' state, and you may want a third appearance for the button when the cursor is pointing at it, perhaps a different colour so that the user knows something will happen when they click.

Save the graphics in a convenient location.

You can then select which of the states you want to import by clicking on the appropriate graphic in the eight button window.

Use the 'Import' button and then navigate to the graphic that you designed to match the choice that you have just made. The graphic will then be imported into the application.

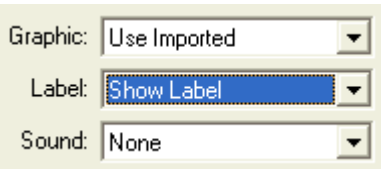


Notice that the default values for all button states are 'Same As Up' so if you do not specifically import another graphic the 'Up' graphic will be used.

Repeat the process for all of the states for which you have designed a graphic, when you have completed these states click on the 'OK' button and the button is then ready for selection.

Three other items should be mentioned in this area, the fist is the use of a sound file attached to the button. If you decide to use this option then generally you should attach the sound to the 'Down' state, so that it will play when the user clicks on the button.

To attach a sound, simply select the state from the eight graphic window and click on the 'Import' button next to the Sound field, navigate to the sound file and select it.



The second item to be considered is the font and text colour on those buttons that you decide should have text displayed on them.

To change the font, or the colour of the text, you must ensure that the button is allowed to have text displayed on it. To do this the 'Label' field should display 'Show Label'.

Now click on the text actually displayed in the button example graphic and grab handles will be shown around the text to show that it selected.

You can now click 'Text – Font' from the menu to select the font and size of text that you want. If you press 'Ctrl – K' the colour palette will be displayed so that you can choose the colour that you want.

The third item is the importation of a button graphic in a slightly different form. Normally when a button graphic is imported, it is displayed in a 'matted' format. In this format any pure white colour on the button that is not itself surrounded by another colour is made transparent, this makes the button appear to sit directly on the background and generally gives the best appearance.

Occasionally you will want to import a graphic that does not do this. Where the white should actually appear as white on the button. To cause this to happen, you import the button as normal, but you must hold down the 'Ctrl' key during the whole importation process.

Functions & Variables

Introduction

There are two icons on the tool bar at the top of the screen in Authorware. One will display the functions dialogue box and the other the variables dialogue box. Both of these work in a similar way, all of the functions and variables are split into different categories and you can choose the category that you want by clicking on the down arrow next to the category list box.

Once you have selected the category that you are interested in using you can scroll up and down in the function or variable box until you find the specific one that you want to use.

There are two categories that need special mention. These are 'All' and one that will either be 'Untitled' or will have the name of your program. The 'All' category is useful if you cannot find the function or variable you want as it shows all of the available internal functions and variables.

The other category will contain different items for every program you create. Functions, which are not available internally, are provided externally. These can be loaded from the functions dialogue box. There are a number provided on the Authorware CD, the most common one to use is the 'Cover' function. This will provide a black background around a program that does not occupy all of the available screen.

Variables can be created at any time in an Authorware program, they are completely untyped, that is they can be either numeric or string variables and can be changed from one type to another as required. It is possible to define a variable as a particular type if there is some reason to do this. The first time that Authorware encounters a variable or when you have first created it, you will be asked to confirm the variable name, give it an initial value and offer the opportunity to attach a comment to it. If you assign the initial value as "", that is a null string, then the variable will be treated as a string for display purposes.

A value can be assigned to a variable by using the ':=' function. It is worth noting that the '=' sign is used within Authorware as a comparison operator. In particular:-

A:=B means set the variable A to be the same as variable B

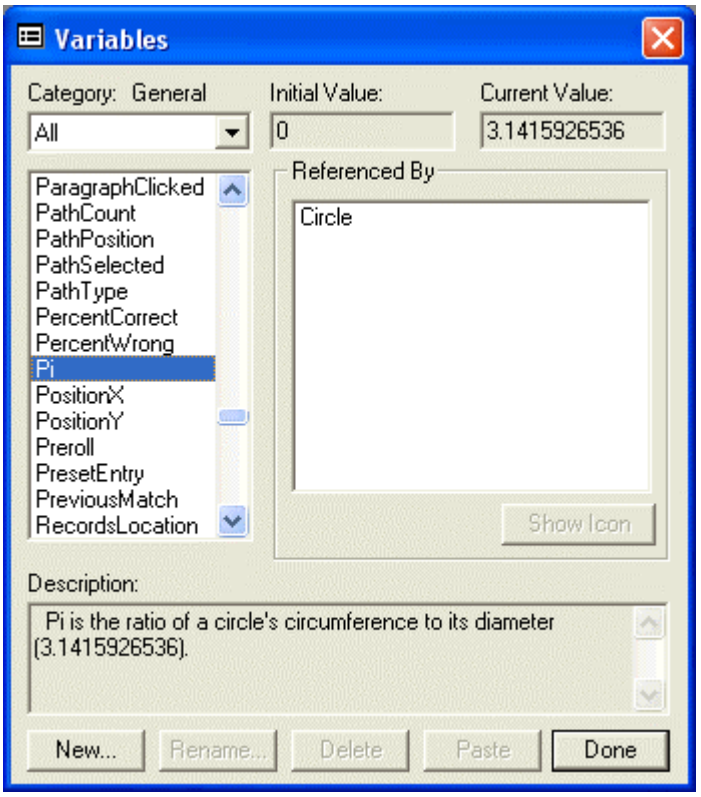
and

A=B means compare the variable A with variable B (the result can be either True or False).

It is possible to show any variable, either system user in a display screen by inserting a text block with the text tool and typing in the variable name surrounded by curly brackets, '{' and '}'.

If the value of the variable will change and you wish to always show the current value of the variable then while the display icon is open, select 'Modify', 'Icon', 'Properties' and click the 'Update displayed variable' option on the Display Page.

System Variables



All the system variables are listed in the Variables dialogue box. These are broken down into twelve Categories plus you can see 'All' variables in the list.

Some of the variables can have values assigned to them in exactly the same way as custom variables, i.e. BranchPath:=2 is a valid thing to do. Such variables will also have values assigned to them by the program.

Any attempt to assign a value to a variable that cannot have a value assigned will simply generate an error message during development. So you should not worry about this aspect of variables. Any error will be trapped before the system is packaged.

The system variables all have a description of their purpose and current value shown in the dialogue window. To see the precise use of these variables simply click on the variable name in the list.

If any variable has been used in the application then the 'Referenced By' field will automatically contain the name of the icons using it. If you are looking for a particular variable in your piece then this is a very useful feature. It means that you can look up the variable in the dialogue window and then by clicking on it and clicking on the 'Show Icon' button you will be taken straight to that icon.

Probably the most useful system variable is FileLocation. This variable always contains the full path name to the folder in which the application was when it started running. Holding this value as a variable means that relative paths to other folders can easily be found, for example:-

For an application that is started in C:\Maths

The variable FileLocation holds "C:\\Maths\\"

Note that in Authorware to use a backslash within a character string you must repeat it. This is because Authorware uses a single backslash as a special marker called an 'Escape Character'.

So if you wish to save records to a file C:\Maths\Records, you could just use a custom variable say MyRecords and set this to be FileLocation^"Records\\RecordFile.txt".

Even if the application were moved to a different disk or folder the RecordFile.txt would still be accessible as the folder just under the current one.

Custom Variables

There are two basic types of custom variable, character or numeric. Character strings are also often called 'Strings'. It is generally considered good practice to define a variable as one type or the other and then stick to this type. Authorware however will allow you to change the type of a variable as you wish.

A variable is can be thought of as a box into which we put a value. This is done with a statement like this:-

StudentName:= "Isaac Newton"	This is a character variable.
CurrentScore:= 23	This is a number variable.

Every variable has a name, there are very few rules governing variable names, but they should start with a letter and not contain punctuation marks or spaces.

It is a good idea to use descriptive names for your variables, and when necessary to use more than one word for a variable name then the first letter of each word should be capitalised to make it easier to read.

StudentName	Is a good variable name, you know what is held by this variable
J67	Is a poor variable name, what does it refer to?
2ndDate	Is a prohibited name. It begins with a number. It would be better to use SecondDate.
First Question	Is a prohibited name. It contains a space. It would be better to use FirstQuestion
scorefromthirdquestion	Is okay to use but would be more easily read if you had used capitals ScoreFromThirdQuestion

You can assign values to your variables whenever you need to do so. On occasions you will want to add or subtract a value to an existing variable. You can do this like this:-

CurrentScore:=NewScore+OldScore	Combines old and new to give current.
CurrentScore:= CurrentScore+1	Takes the value of CurrentScore and adds 1 to it.
CurrentScore:=CurrentScore-Penalty	Reduces the value of CurrentScore by whatever penalty you assign.

Character or string variables can be handled in a way that is similar to numbers, you can add two or more strings of characters together to make a new string, so for example:-

FullName:= FirstName^ " "^SecondName	Adds the first name and a space and the second name to give a persons full name.

The '^' or carat character adds two pieces of text together. This process is correctly called concatenation.

If the persons first name is held as Michael and their second name as Faraday, then some Authorware program might look like this:-

FirstName:= "Michael"

SecondName:= "Faraday"

FullName:=FirstName^" "^SecondName

The effect of this is that when you display {FullName} Authorware will replace it with Michael Faraday.

Lists

A special form of variable within Authorware is called a list (other programming languages call them arrays). A list is simply a series of variables, related by a common name and identified with a number, like this:-

First we tell Authorware that MYUsers is a list by defining it as such. There are a couple of ways of doing this and the easiest is shown here:

MyUser:=[]

MyUsers[1]:= "James Lister"

MyUsers[2]:= "Florence Nightingale"

MyUsers[3]:= "Jane Austen"

And so on. The benefits of this are that we can recover any particular value by using a variable as the index, like this:-

i:= 3	Set the value of the index:
CurrentUser:=MyUser[i]	CurrentUser is Jane Austen

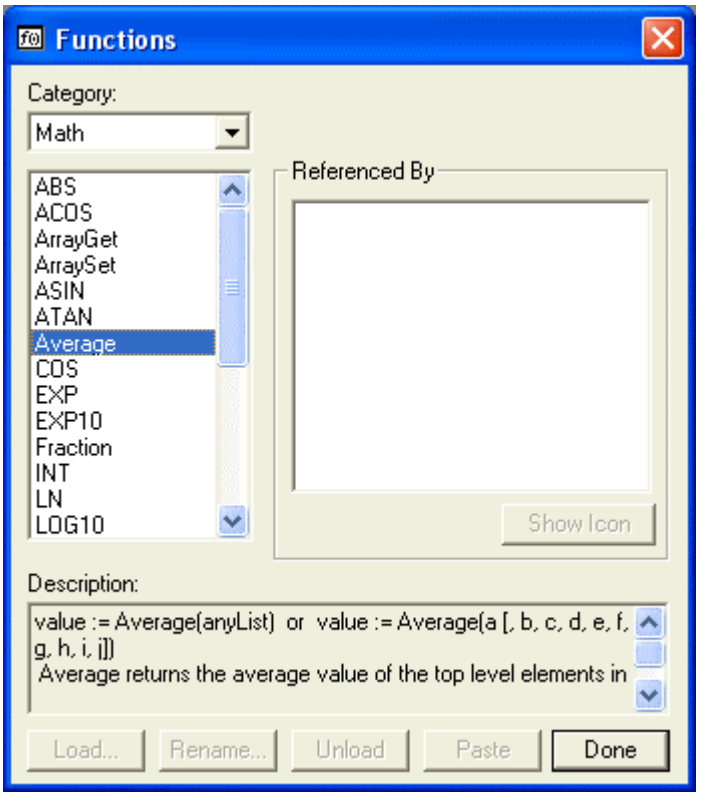
An alternative way of defining a variable as a list is by using the Array function, which works like this:-

MySubjects:=Array("",50)	This creates a variable called MySubjects, tells Authorware that it is for characters and reserves 50 spaces for it.
MyScores:=Array(100,25)	This creates a variable called MyScores, tells Authorware that it is for numbers, creates 50 spaces and fills each of those spaces with the value 100.

There are many uses of this type of variable and practice is recommended.

Functions

There are a great many functions built into Authorware, most are beyond the scope of an introductory course, but several are essential and will be covered briefly here. Note that many functions require an argument, which is the name given to the things that must follow the function name. These arguments are always surrounded by brackets. When arguments are not required by a function you MUST include empty brackets '()'. Many functions that are not built into Authorware are available in external files. These add considerably to the functionality of Authorware and some are considered essential.



Internal Functions

Quit()

This function is essential as it will close the Authorware application and return control to Windows.

The possible arguments are numeric and work like this:-

Quit() or Quit(0)	No space between the brackets. Closes the application. Returns to another Authorware application if the current application was called by it, otherwise returns to Windows. This is the most common usage.
Quit(1)	Closes the application. Returns to Windows.
Quit(2)	Closes the application and restarts the computer.
Quit(3)	Closes the application and then closes Windows (except for Windows 3.1).

UpperCase
LowerCase

These two functions will set all the characters in a string to be upper or lower case respectively.

MyName:=UpperCase("Fred Bloggs") MyName will contain FRED BLOGGS

MyName:=LowerCase(“Fred BLOGGS”) MyName will contain fred bloggs

Capitalize (Note the spelling)

This function will set the first character of every word in a string to be a capital.

MySlogan:=Capitalize("bolton wanderers for ever")	MySlogan will contain Bolton Wanderers For Ever

These functions and indeed any that can operate on character strings work just as well with variables as they do with littorals enclosed in quotes so that: -

MySlogan:="bolton wanderers for ever"
MySlogan:=Capitalize(MySlogan)
MySlogan will then contain the sentence 'Bolton Wanderers For Ever'.

There is another optional argument to the Capitalize function, that is if you use it with the number 1 after the character string only the first word will be capitalised, like this:-

MySentence:="reebok is a world class stadium."
MySentence:=Capitalize(MySentence,1)
MySentence will now contain, Reebok is a world class stadium.

SubStr

Often you will want to get just part of a string of characters, for example to get the first five letters of someone's surname to generate an ID number. The SubStr function will do this:

Surname:="Wilkinson"

PartSurname:=SubStr(Surname,1,5) PartSurname will contain Wilki

You could then concatenate this with a course code to give an identifier like this

CourseCode:= "AH1243"
Surname:= "Wilkinson"
DateOfBirth:= "12/03/83"
PartSurname:=SubStr(Surname,1,5)
Identifier:=CourseCode^PartSurname^DateOfBirth
Identifier would now contain AH1243Wilki12/03/83.

There are a number of other functions that deal with characters and a little experimentation will reveal that these are both versatile and powerful.

If…Then…Else…End If

This is a programming function but don't let that put you off if you are not a programmer. All it does is check to see if something is true and does one thing if it is and something else if it is not. For example:

If it is raining then take a raincoat, else take a jacket.

If the answer is right then add one to the score, else don't add one to the score.

In Authorware you would use this function like this:-

```
If ResponseToQuestion = “Alfred the Great” then
score:=score+1
else
score:=score-1
end if
```

The effect of this is to see if the user typed Alfred the Great in response to a question, if they did then we add one to their score or because this is a particularly important question if they get it wrong we subtract one.

You will find that you will use this function over and over again because it is so useful.

Sometimes the result has only one possible action so we can drop the else part, the following are all valid uses of this function:-

```
If Score>7 then
Result:= “Pass”
Else
Result:= “Fail”
End if
```

The > symbol means greater than.
If the score is more than 7 then the person has passed the test.

```
If Password= “assassin” then
User:= “Teacher”
Score:=100
ShowHiddenButton:=True
Else
User:= “Student”
Score:=0
ShowHiddenButton:=False
end if
```

We can put several or many lines of functions and code between the various parts of the if then else function.

False and True are called Boolean values and can be used for many purposes within Authorware.

```
If MyTest= “running” then MyMessage:=”Running”
```

If there is no other option to do then you can fit everything on one line and you don’t need the else or the end if.

Repeat with

Closely linked to the use of list variables is the Repeat with function. This seems to be a complex function that has little use until you start to repeat things and then you will see that it is quite straightforward and very useful.

You will always need to do something else with this function as all it does is repeat something a certain number of times.

Here is a simple example:-

```
Repeat with i:=1 to 50
UserNames:=Usernames^MyUsers[i]^Return
end Repeat
```

Can you see what UserNames will contain after fifty repeats?
‘Return’ is the carriage return character.

WriteExtFile
AppendExtFile
ReadExtFile

These three functions are closely linked and are likely to be very useful for keeping scores, personal details like passwords and when a person has used an application.

WriteExtFile writes some information to a file on the disk. It automatically creates a new file if one does not already exist and overwrites one if it does.

AppendExtFile writes some information to a file on the disk. It automatically creates a new file if one does not already exist and if the file already exists then it adds the new information to the end of the old.

ReadExtFile reads the information back from the file into Authorware where you can process it as you like. It does not create nor change any information already held on the disk.

In the case of the first two functions you need to supply two pieces of information, the value that you want to write and the name of the file to which the information is to be written. Here is an example:-

MyUser:= "Charles Dickens"
MySubject:= "English Literature"
MyScore:= 85

MyRecord:=MyUser^Tab^MySubject^Tab^MyScore^Return
MyFileName:= "C:\\TestResults\\Test99.txt"
WriteExtFile(MyFileName,MyRecord)

This would have the effect of creating a file on drive C: in folder TestResults and called Test99.txt. The file would contain Charles Dickens, English Literature and 85 all separated by a tab and with a carriage return at the end of the line.

If I wished to add to the file I would use the AppendExtFile function like this:-

MyUser:= "Charlotte Bronte"
MySubject:= "English Literature"
MyScore:= 91

MyRecord:=MyUser^Tab^MySubject^Tab^MyScore^Return
MyFileName:= "C:\\TestResults\\Test99.txt"
AppendExtFile(MyFileName,MyRecord)

And so on.

To read the information back in I would use a ReadExtFile function like this:-

MyFileName:= "C:\\TestResults\\Test99.txt"
MyInformation:=ReadExtFile(MyFileName)

This would read the ENTIRE file into the variable MyInformation, I would need then to separate it into its various parts with other functions. Without getting too deeply into this here is an example of how this might be done, see if you can figure out what is happening:-

Users:=[]	Create a list for users.
Repeat with i:=1 to LineCount(MyInformation)	LineCount simply returns the number of lines separated by a carriage return
OneRecord:=GetLine(MyInformation,i,i)	This takes the information held in line 'i' and brings it into the variable OneRecord.
Users[i]:=GetLine(OneRecord,1,1,Tab)	User now contains the value in the first part of the variable OneRecord, in other words the name part of the record. In this case this is always the first part so we use the number 1 rather than a variable i. And we indicate to Authorware that these particular lines are separated by Tabs rather than carriage returns.

End Repeat

By combining this technique of repeating with lists we could end up with a set of lists where the complete details of one person were held in a number of lists but all with the same index number:-

Users[1]	Charles Dickens
Subject[1]	English Literature
Score[1]	85

And so on. This would allow us to get all of the details of any user by just referring to the correct index

External Functions

Before you can use an external function it must be loaded into your application. This is done from the functions window. Select the category that has your application name and click on the 'Load' button.

You will be presented with a file find dialogue box and you should use this to locate the function. 32 bit functions have an extension of U32 and 16 bit functions have an extension of UCD. It is usually most convenient to ensure that the function file is located in the same folder as the application executable. The function file *must* be in

the same folder when the application is delivered that it was when authored. The file itself is not loaded but a link between the application and the function file is established.

Once you have located and selected the function file you will be presented with a list of the functions that are available within it. Select the ones that you need by highlighting them and clicking 'Okay'.

The functions will then be available from the functions window under the category of the application name.

Cover.U32

Cover() and uncover()are probably the most widely used of all of the external functions. There is NO argument inside the brackets, although the brackets are required.

The purpose of cover() is to 'blank out' the desktop surrounding the display window on those applications where the display window is smaller than the screen size as defined by its resolution. This enables you to develop at a resolution of say 640x480, and publish the application on a screen set at a resolution of 800x600 without showing any of the desktop icons or background around your application.

Cover() always and only works in black.

Uncover() simply redisplays the background and desktop icons. You should always use uncover() before quitting from the application although by default the normal screen will be displayed when you exit from the application.

ODBC.U32

Note that this function is a rather advanced facility and really requires some knowledge of SQL to be used effectively.

This function file contains three functions, ODBCOpen(), ODBCExecute() and ODBCClose().

These functions allow the developer to use Authorware to communicate with any ODBC compliant database, such as Microsoft Access.

A requirement is that the ODBC 32 bit drivers have been properly installed on the target machine and that the DNS values have been set up properly (see tMsDNS.U32 below).

Before any call to the database can be made a communication link must be established, this is done by opening the database using the function ODBCOpen()

The syntax for the open function is:

ODBCHandle:=ODBCOpen(WindowHandle, ErrorVar, Database, User, Password)

You are required to supply the DNS for the database, this is the name by which the database is known to the ODBC driver and might be something like 'Courses'. You can use a string surrounded by double quotes "Courses" or you can use a variable like this;

MyDataBase:="Courses"

ODBCHandle:=ODBCOpen(WindowHandle, ErrorVar, MyDataBase, User, Password)

You will only need to supply a user name and password if security has been established on the database, in which case the same comments apply to these that applied to the database name. If security has not been imposed then you can either close the brackets after the database name or replace the user and password with two pairs of double quotes.

WindowHandle and ErrorVariable will actually be returned by the function so you need not assign a value to them.

The function will return a value to the variable at the left of the equals sign and this value is then used for all other ODBC calls.

Authorware uses Structured Query Language or SQL to communicate between itself and the database. There is simply no space here to describe the syntax or use of SQL, which is a complex and powerful application in its own right. What is suggested is that any queries that are required are established in the database, and then the SQL structure is copied from the database and pasted into an Authorware calculation icon. Microsoft Access is particularly useful for this.

The syntax is quite long and can appear intimidating but with care excellent results can be obtained.

A command string must be created an example is given here:

ODBCCommand:="SELECT Student.StFName, Student.StSName, Courses.CrsTitle, Courses.CrsDate, Courses.CrsScore "

ODBCCommand:=ODBCCommand^"FROM Student INNER JOIN Courses ON Student.StudentID = Courses.StudentID "

ODBCCommand:=ODBCCommand^"WHERE (((Student.StSName) Like "'^StSurname^'"));"

This string extends over three lines in a calculation icon, there is no particular requirement for it to do so, but by breaking it down in this way and then building it up it is much easier to read.

The first line starts the ODBCCommand variable (you can call it what you wish, it is simply a normal user string variable), and indicates what pieces of information we require from the database. In this example the forename and surname of the student plus the title, date and score that were obtained for each course taken.

The second line then adds to the first a description of how the two tables in the database are related, the student table is related or joined to the courses table by their having a field called StudentID that is common to them both.

The third line then adds to the first two the criteria for the search, in this case we are looking for all cases where the surname of the student matches (or is similar to) the custom variable (in Authorware) StSurname.

Note that in SQL, string variable are surrounded by a single quote (in Access a double quote is used) when reading the above command you need to pay particular notice to

how the Authorware string variable is actually surrounded by both a single and a double quote as well as the concatenation character'^'.

It is very important to note that SQL does not use the asterisk'*' as a wild card indicator, but requires the use of a percent sign '%' so that a search for 'sm%' will find Smith, Smithers and so on.

Once the command has been created you must execute it using the ODBCExecute() function:

ReturnString:=ODBCExecute(ODBCHandle, ODBCCommand)

Note the use of the ODBCHandle variable the value for which was obtained when we opened the database earlier.

The result of the search will be placed in the variable ReturnString.

Remember that there is a system limitation in Authorware on the size of any variable. You are not allowed to exceed 30,000 characters, if the search returns more than this value the least that you can expect is that the string will be truncated to the first 30,000 characters.

What you do with the variable that is returned is a matter for yourself to decide. Using the internal string handling functions you can separate the variable into individual lines, a carriage return is usually used as the separator between records, or you can extract individual fields, a tab is the usual separator between fields.

Once you have finished with the database you should close it down with the ODBCClose() function like this:

ODBCClose(ODBCHandle)

Again we use the ODBCHandle variable discussed above.

tMsDSN.U32

All ODBC connections require that an entry exists in the ODBC driver program that contains the DNS by which the database is known and the file location and name of the database involved. Normally this information must be set up by someone on each machine, this is usually impossible if the application is to be distributed widely.

An external function file tMsDSN.U32 which is available from The Media Shoppe as a free download takes care of this problem for most users. The exception is for any machine that is using Microsoft Windows NT where the user does not have the necessary privileges to write to the system files and registry. In these cases the initial installation and first running of the application should be done by such a person and this should take care of the initialisation of the necessary links.

There is only one function inside this file, tMsDBRegister()

This function has three arguments and should be used like this:

result:=tMsDBRegister(dbReqType, dbType, dbList)

dbReqType :=1.

dbType:="Microsoft Access Driver (*.mdb)"

dbList:="DSN=Courses;Courses;Access;DBQ=C:\\AWCourse\\Courses.mdb;"

These values will establish an ODBC link with a name of Courses, to an Access database that is found in the specified location on the disk.

Note that in the last of the arguments Courses appears twice, and that a semi-colon separates each value in this argument.

This function actually only needs to be executed once in any installation although repeated executions will not cause duplicate entries in the ODBC table.

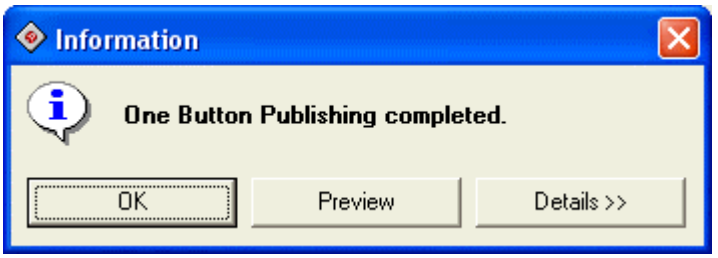
One Button Publishing

Once we have completed the application it must be packaged so that the users can run it. They don't need to have the full Authorware program, we just give them the run time sections and a packaged program that they can run but not change.

New to version 6 of Authorware is the ability to publish a piece, in a form that is suitable both for the Internet and for LAN/CD delivery with the minimum of fuss. This is called 'One Button Publishing' although in practice you will have to press more than one button to set things up in the first place for anything more than the most basic of delivery.

Quick Start

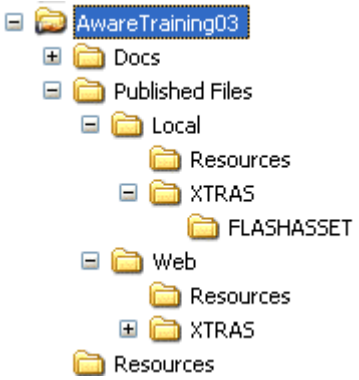
To just accept whatever the defaults are and quickly get your application published, simply select the menu options File – Publish – Publish, or press the function key 'F12' on your keyboard.



There will be a short delay and various messages will appear on your screen before a completion dialogue window will be displayed.

You can if you wish view the details of the process by clicking on the 'Details>>' button. This will allow you to view or save a log of the actions taken, and to view a list of the files that have been created.

During the default publishing process, Authorware creates two separate versions of your piece. One is intended for publication for a CD-ROM or LAN based delivery and the other is intended for delivery via the Internet.



These two versions are located in separate folders and the contents of the folders and their directory structure must be adhered to in the delivery of your piece. In particular any external files that you have used, and any 'Xtra' files that Authorware decides that you need must accompany the delivered application.

One of the more difficult areas for people new to Authorware to understand are the significance of the 'Xtras' that are added to a published piece. An Xtra is a file that holds further functions for Authorware to use. These Xtras might allow the application to import or view different picture formats, or play sounds. When you have packaged your file you must ensure that the Xtras folder is located within the applications folder and that it contains the correct Xtra files. There will *always* be some Xtras, although the precise ones will vary depending upon what your piece does.

The Xtra files can be found in the Xtras folder inside the folder where your installation of Authorware is. It is a bad idea to include all of the Xtras. Authorware

loads all of the available Xtras when it starts and this can slow down execution of the application.

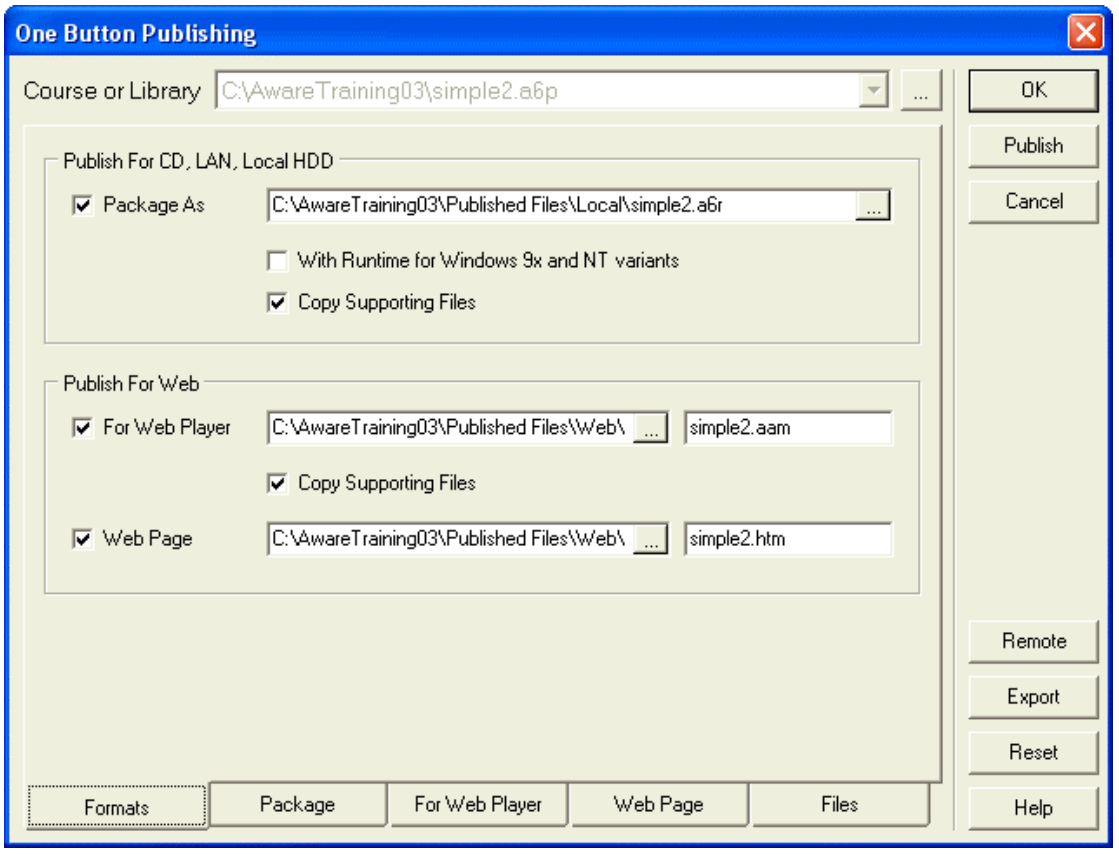
One Button Publishing Set-up

There are a great many options involved in setting up the one button publishing process. It is a highly flexible procedure that you can fine-tune to match your exact needs. Many of these options are beyond the scope of a basic course, but some of the more basic ones are described here.

Click on File – Publish – Publish Settings or press Ctrl F12 on your keyboard and you will be shown the OBP Settings dialogue window. There are five tabs across the bottom of this window.

Formats

You might only want to publish for delivery on a CD-ROM or a LAN, or perhaps only for delivery on the INTERNET. This is the screen where you will make that choice. You can also choose the destination of the published application by using the standard Windows browse functions started by clicking on the '…' buttons alongside the destination fields.



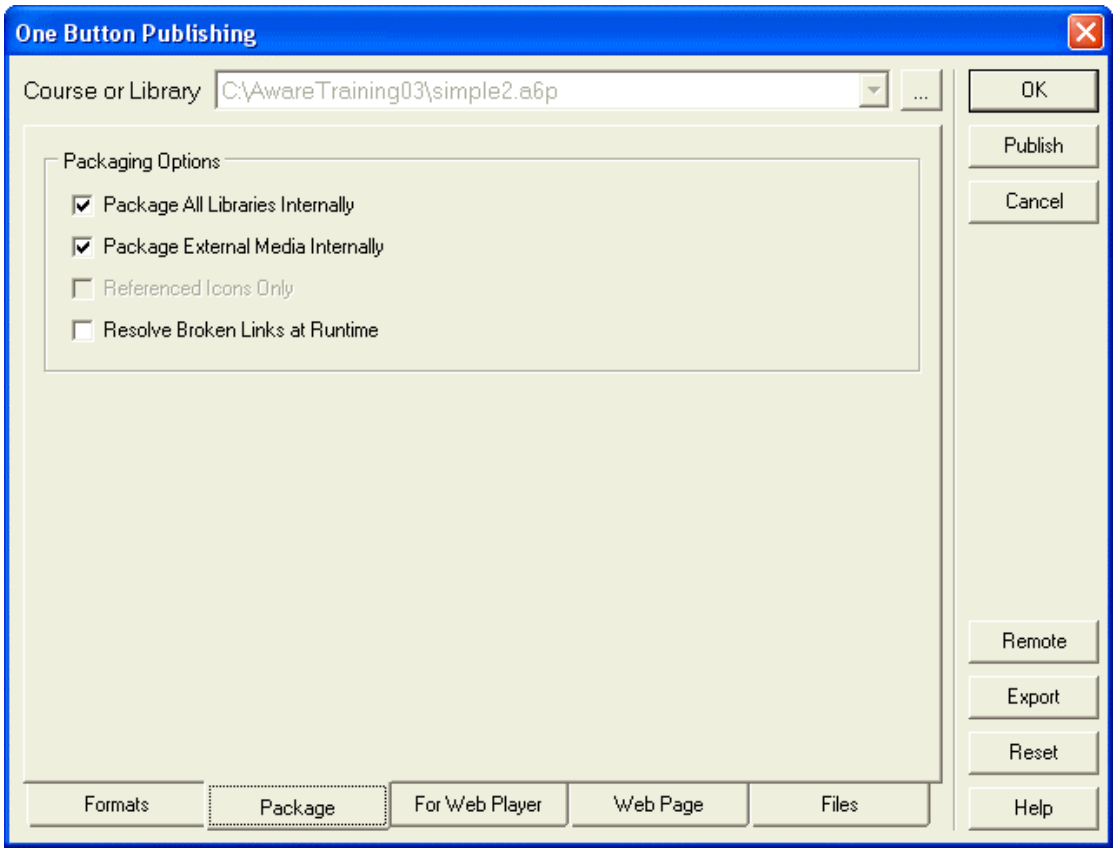
If you decide to deliver the piece on a CD or LAN you can also make the choice as to whether the piece should be 'stand-alone' or require the inclusion of the Authorware run-time files.

If the piece is to be delivered to computers where Authorware is not installed, or where there is no other installation of the Authorware run-time files then the simplest choice is to create a 'stand-alone' executable file. This file will have the usual '.exe'

extension. It will be about 1.2 megabytes larger than a delivered application that needs the Authorware run-time files delivered separately. This is the most common way of delivering the piece *but it is NOT the default way* so if you want to produce stand-alone executable programs you must enter this screen and make the necessary changes.

To create the piece in this way you *MUST* tick the field 'With Runtime for Windows 9x and NT variants' this will cause the creation of the executable file. You will *still* need the Xtras and any other external files that are required to run your piece

On the second tab of the OBP dialogue window you will be given the chance to 'Package All Libraries Internally. Libraries are separate files that can contain screen objects or calc files that you use more than once in your file. This is explained more in the Libraries section. Generally speaking library files remain separate to the main application and can be accessed by different applications that might share some common items such as logos or background images.



To simplify the distribution you can choose for the library files and most external media files, such a sounds and graphics to be included within the main application at this point. ***If you are packaging for the Internet it is highly recommended that you use this option.*** You cannot however choose to package most movie files internally, avi, mpeg, QuickTime and flash files must always remain external.

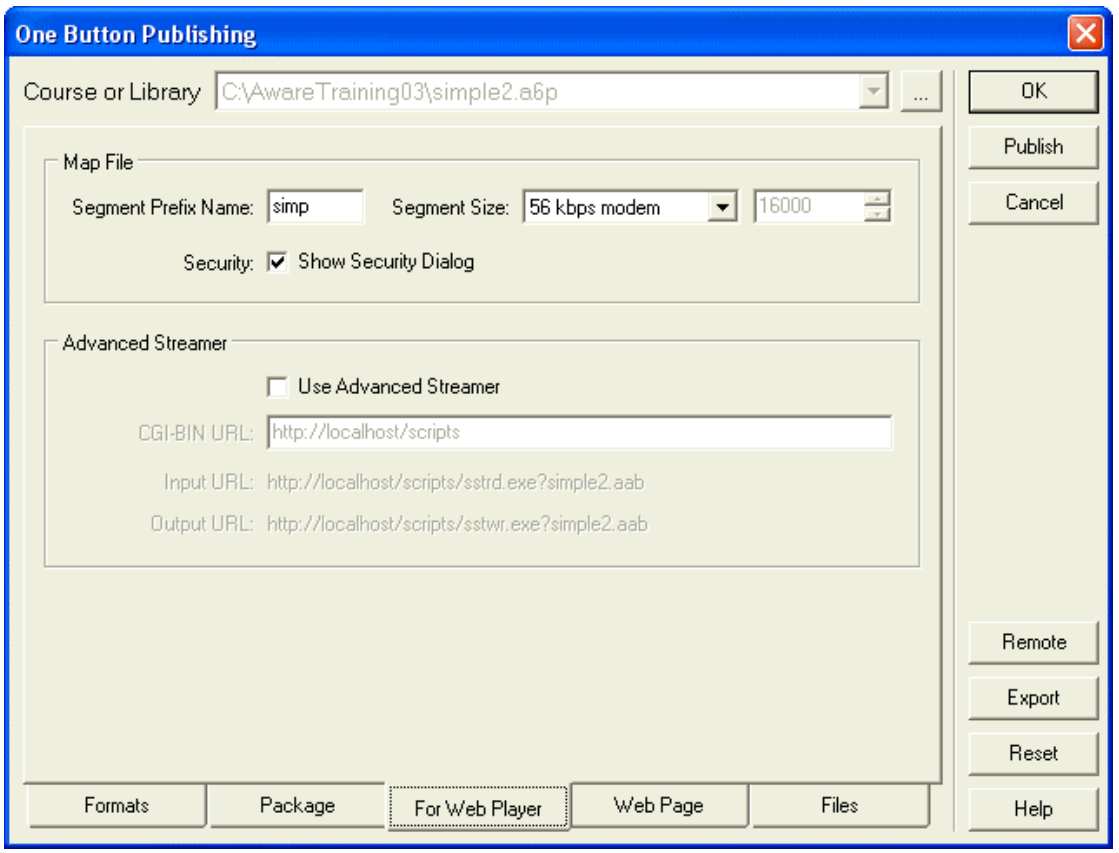
If there is a library file included with your piece, (in the above example there is not) you could choose that only those icons in the library that are actually referred to in the piece are packaged internally with the completed file.

The final option on this screen would allow an end-user to search for any files that are not in the places that Authorware expects them to be, and therefore to resolve the

broken links. A properly developed and packaged piece would not normally require this option.

When you are publishing a piece that will be delivered on the Internet, or an Intranet then you can set the values that relate to the way in which it will be delivered by the web server.

Authorware packages a piece for the Internet by breaking it down into small individual files, each of which will be given a sequential number. These individual files are then downloaded to the user's computer as needed (you can have some control over this process from within the piece – see the NetPreLoad() function).

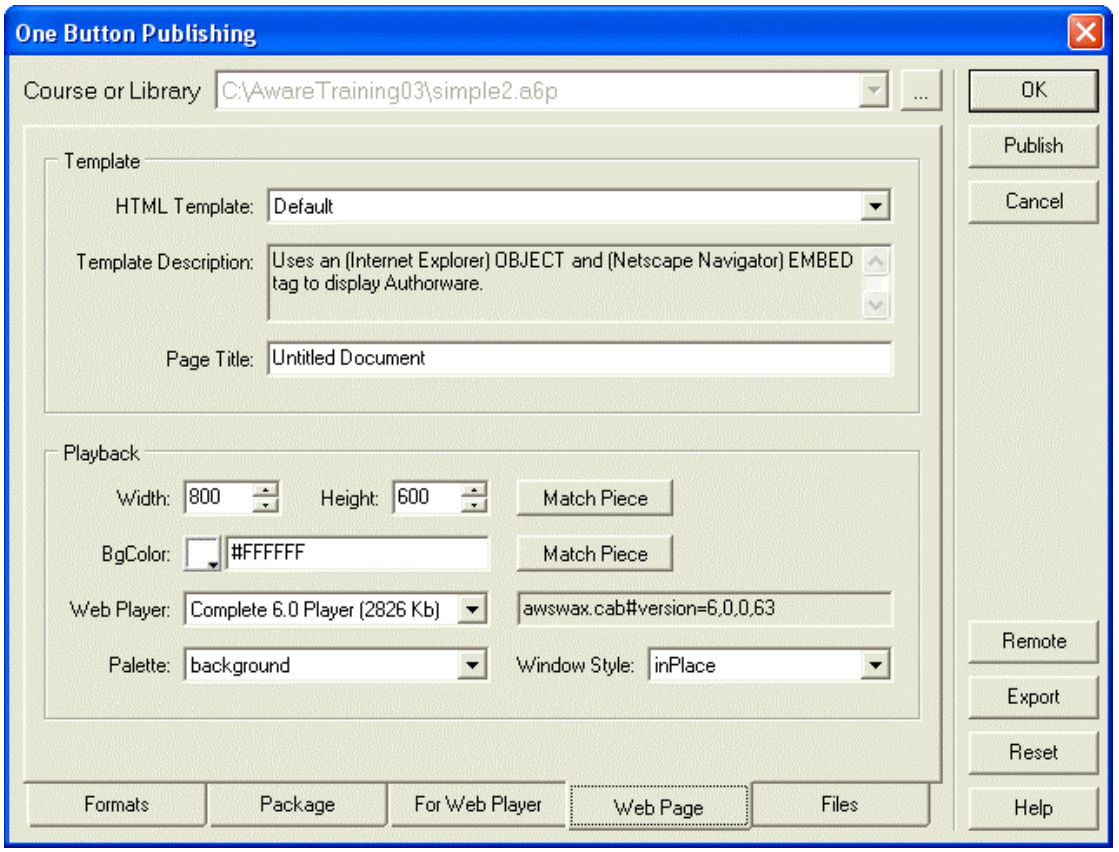


In this page you can define the first part of the name for the individual files, they will all be given the extension 'aas'. The 'Segment Size' field will determine the size of these files. You should indicate the likely speed of the user's connection from the drop down list. If you choose 'Custom' from this list you can then indicate the preferred file size and Authorware will attempt to break the file into the closest it can to the indicated file size. You may want to experiment with these sizes and speeds to achieve the best download performance for your users.

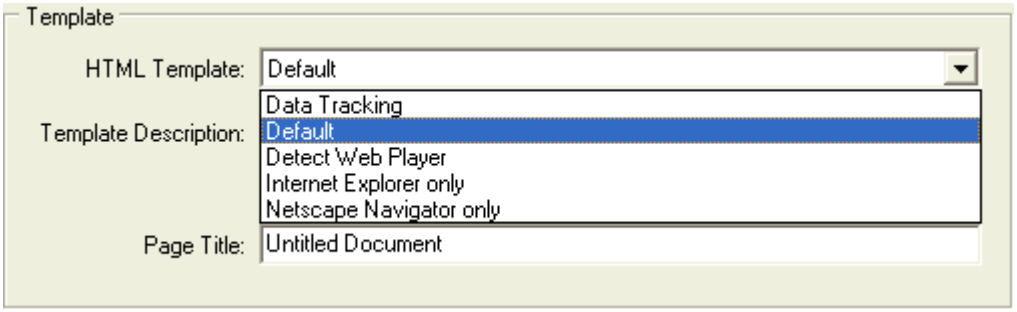
The 'Advanced Streamer' facility will only apply to those installations where the server is running under a Windows NT operating system. If used this will automatically monitor the way in which users run through the application, and require the downloads, it will then attempt over a period of time to automatically pre-download the individual file segments in the most efficient order. You are recommended to consult the manual for advice on the best way to set this up.

The HTML code for the browser in which the piece is played can also be modified by the OBP set up facility.

In the main you will not need to make many changes to this page, although there are a couple of items that should attract your attention.



A default HTML template is installed with Authorware and this will include all of the code necessary for your piece to be viewed via a browser. If you are not sure what plug-ins will be installed on the user's browser then you might want to consider one of the alternatives.

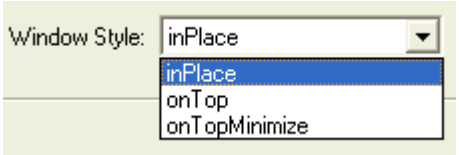


If you choose 'Detect Web Player' the web page will automatically check if the correct plug-in is installed and if not it will download it from the Macromedia web site.

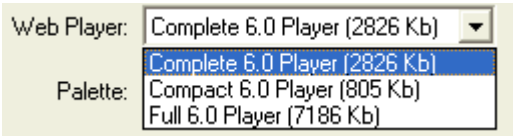
You should set a value for the field 'Page Title' that reflects your piece; this is the text that will appear in the title banner across the top of the browser window (it uses the HTML code <title>…</title>).

The playback size should usually be set to Match the Piece and you can choose the background colour of the browser window if you wish.

The ‘Web Player’ field has three options;



- The ‘Complete 6.0 Player’ is a common choice. This player includes all of the standard Xtra files, so you do not need to worry too much about them. Once installed, these files will not need to be downloaded to the user’s computer, saving time in the initial running process.
- The ‘Compact 6.0 Player’ does not include any of the Xtra files. This player is useful for a public facility; the download of the plug-in is relatively small, taking about 2 minutes on a 56k modem. However you will need to include the correct Xtra files with your piece, making the piece itself slightly longer to download and run on the first occasion that it is run.
- The ‘Full 6.0 Player’ includes everything necessary to run not only version 6 applications, but also pieces that were developed using Authorware versions 5 and 4. You should only use this option if you are also delivering applications developed using these older versions.

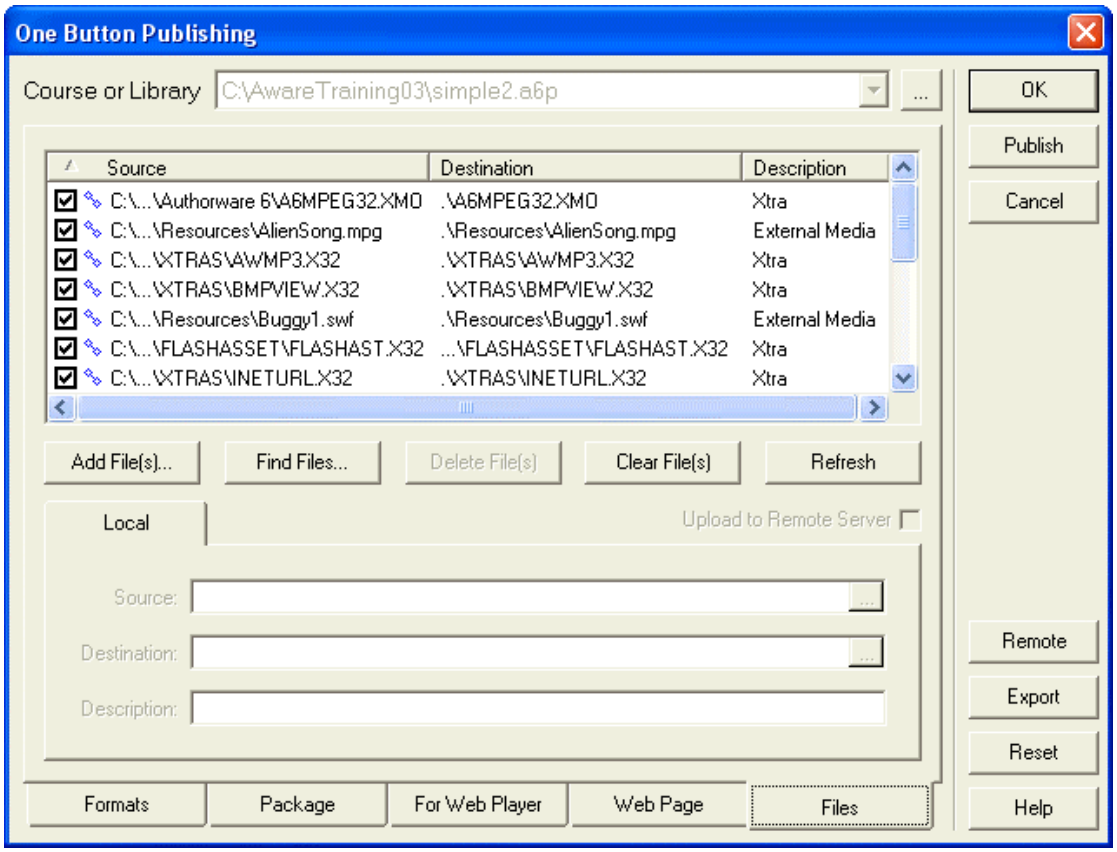


The only other option that you need to consider on this page is whether your piece will be delivered from within the browser window, or on top of it. The safest option is always ‘On Top’ as this will not be affected by the size of the browser window, over which you have no control.

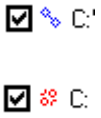
If you choose ‘inPlace’ then the application will actually play completely within the browser window.

The third option ‘onTopMinimize’ will cause the piece to play on top of the browser window, but it will start in a minimised state.

The most complex of the pages of this dialogue window is the Files section. It is however sufficiently well designed that in general you will not need to make many changes for most applications.



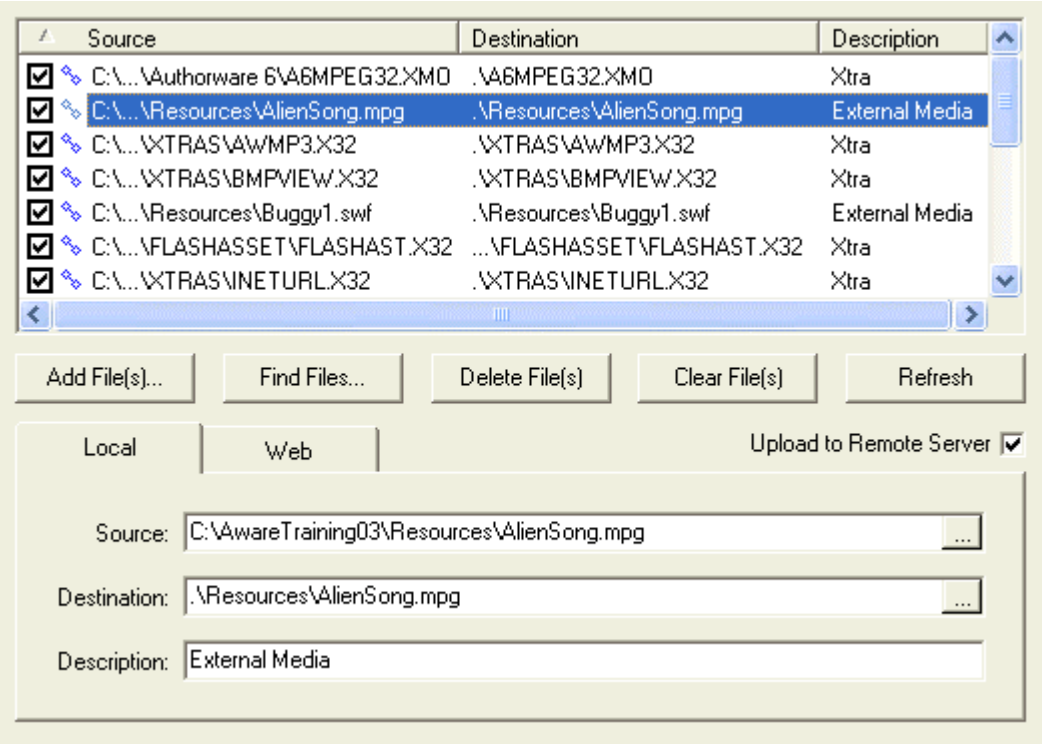
A complete list of all of the files that will be included for the download will be sown in the main part of the window. Notice that a small symbol, like a chain is shown at the left of the name, if the file has been located correctly and can be 'seen' by the OBP window this will be shown in blue. If the file has not been located then the chain will be broken and shown in red – you will need to correct this by making sure that the file is in the correct place and clicking on the 'Refresh' button.



As part of the publishing process, Authorware will always create a small text file that will have the same name as your original piece, but a n extension of 'aam'. This file is referred to as the 'map file'. It contains the names of all of the files referred to in the main part of this window and in fact is the file referred to within the HTML code that the browser displays. Its contents are editable in Notepad once you know what they mean.

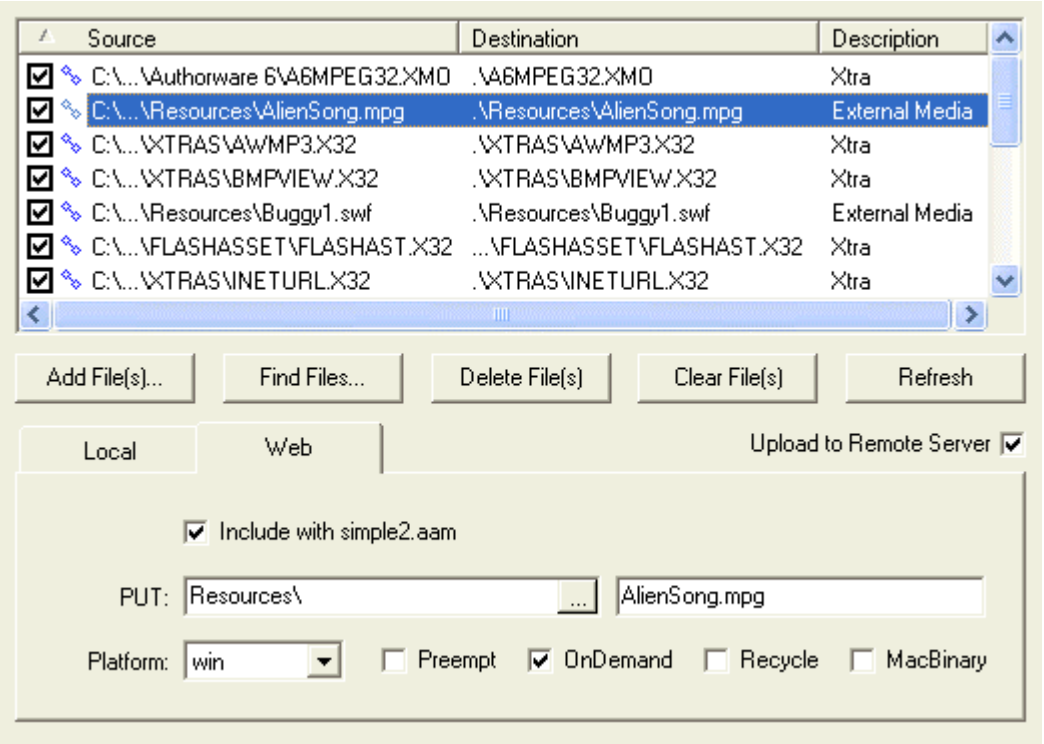
You may occasionally need to 'Add Files' using the button shown. This is most likely if, in your piece you have included external media but have referred to it by means of a variable rather than by explicitly naming it. This process displays a normal Windows 'Find File' dialogue and you can then locate your extra files using this. The file details will be shown in the main part of this page of the dialogue window.

If you select a file in the list, by clicking on it, the bottom part of the page will become active.



You can then change the Source or Destination of the files. You are strongly recommended not to make any unnecessary changes to these fields that have been set automatically until you are certain what you are doing. In particular you need to pay attention to the relative location of the files to your published piece to make sure that the files can be located by your piece when it is running.

The most likely occasion that you will need to make changes here is when you have used the 'Add File(s)' option and need to set the Destination location of the file. The easiest way to make the setting is to ensure that the file is in a similar location to one that has been detected automatically and copy the same location for the extra file.



Similar advice applies to the 'Web' tab for a selected file.

Generally the automatic process correctly sets these values. There are however a couple of tips that might help in making the piece play better over the web.

The first relates to the 'Recycle' option. Generally speaking the browser caches files that are downloaded for playing. It cannot however do this for files that are downloaded by the Authorware plug-in as this is a separate application, the default behaviour is that extra files, including media and Xtra files are actually deleted by the plug-in once the piece is completed. If you select a file and click in the 'Recycle' option the file is downloaded but is not deleted after use, it can then be re-used when the piece is played again, saving possibly considerable download time.

The second relates to the 'Preempt' option. This effectively works in the opposite way to the 'Recycle' option. If this is selected then the file will always be downloaded from the server even if it exists on the users machine. This might be particularly useful if the file is one that is updated regularly at a central location.

The 'OnDemand' option should normally be left selected; this will cause the file to only be downloaded when the piece needs it.

Libraries

A library is a file that is separate from but related to your application, it is used to hold a single instance of an icon that you actually need to use in many places within the application.

There are two main benefits of this. The first is that the file size of the application can be substantially reduced, a bitmap for example can be a megabyte in size and if it is to be used in several places the size of the application would be substantially increased for each use. If the same image is kept in a library then the initial file is saved once and a series of links are established. Each of these links is just a few bytes in size giving a considerable saving each time that the image is used.

The second benefit is that the content of the library can then be re-used for multiple applications. A good example of this might be a splash screen containing a logo and some copyright information that would then be used in every application that you produce. If a calculation is attached to an icon before it is placed in a library then the calculation will also be included in every instance of the use of the icon.

Note that when you use a library icon in your file while you can move the location of the contents you will not be able to add or change anything inside the icon in the application. To make changes you will need to change the icon within the library.

To create a library, first save your application as normal, and then click File – New – Library. A second, small window will be displayed inside the Authorware window. This represents a separate file that you will need to save at some stage, you can give the library file any relevant name that you want and save it in any file location that you wish. Remember that the library file will need to be available to the application when you package and deliver it (unless you use the package libraries internally option).

To add an icon into the library you can just create it as normal within your application and then drag it from the normal flow line into the library. You could also drag an icon directly into the library window and then double click on t to edit and name it as required.

To use an icon from the library just drag it from the library and place it in the required place on the flow line.

Note that the title of any library icon on the flow line will be displayed in italics so that you know that it is a library icon.

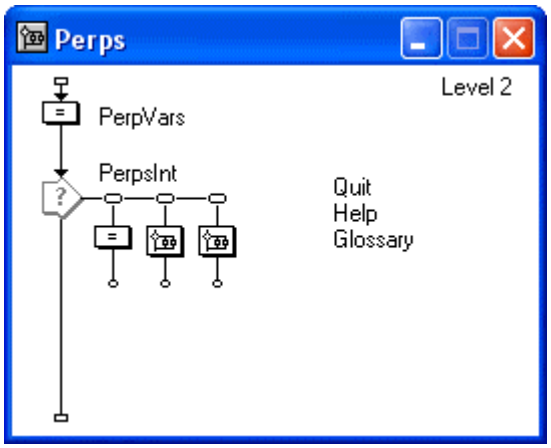
You can have several (up to seven) libraries associated with a single application.

If you want to share a library with other developers across a network then the library must be in a location on the network that is shared and the first user of the library must 'lock' the library so that other users can see it but not make changes to it, this is done by clicking on the small icon of a pencil at the top of the library window, a warning message will be displayed to indicate that the file has been locked.

Models

A model differs from a library in at least three ways:

- While a library can contain only individual icons a model will almost always contain several, (or even many) icons.
- Libraries are really containers for content, images or text while models usually contain structure.
- Crucially, libraries will contain the actual icons and the main file will contain links to them while a model is physically imported into the main piece and forms part of that piece.



To create a model, build the structure within your piece as normal. Make sure that the structure is contained within a Map icon and give the Map icon a name that is indicative of what it contains. In this example we have a map containing some buttons that are perpetual and will be available throughout the whole piece, giving access to a quit, help and glossary structure. Once you have this structure working properly a great deal of time can be saved by re-using it rather than by having to re-build it for every piece that you develop.

Saving the model can be done in two different ways:

- You can then click File – Save As Model and navigate to the Knowledge Objects folder within your Authorware 6 folder (where Authorware is actually installed, not where you keep your working files). If a folder called 'Model Palette' exists, save your model in there with the name that you have chosen. If this folder does not exist you are recommended to create it and save the model there.
- Hold down the 'Ctrl' key and press '3' on the number keys across the top of your keyboard. A new floating palette will appear, this will be empty at the moment.

- To populate this palette with your model, simply drag the map icon containing the model and drop it on the palette. That model will then be available in all of your future developments.

To utilise the model in a piece that you are developing you can just drag and drop the model from the palette onto the flow line within your piece as you want. Holding the mouse pointer over the icon will cause its name to appear so you know what the icon does.

You might have several models that you will use many times in the same piece, for example a particular type of interaction, multiple choice question or text input, you can save all of these on the model palette and insert them into your application as often as you wish.

You can even organise the model palette to contain different types of models. The contents of this palette are actually contained within the folder 'Model Palette' referred to above. The important thing about this folder is that it is contained within the 'Knowledge Objects' folder. This is a significant folder in the Authorware installation.

Each time that Authorware starts it scans this folder and uses the sub-folders that it finds there to build the Knowledge Object dialogue window. If there is a folder called 'Model Palette' then this folder and whatever it contains is used when the developer presses 'Ctrl – 3' to build the model palette.

If you Right-click on the model palette you will see a drop down menu with the names of all of the sub-folders in the Knowledge Objects folder. Clicking on any of those will show the KO's on the palette instead of the model icons that you have created. These KO's can then be used in the same way as though you had used them from the normal KO window.

However, you can create whatever folders you wish within the Knowledge Objects folder and these folders will also become available when you right-click on the Model Palette. Using this it becomes easy to create say a folder called 'Interactions' and another called 'Ebooks' and perhaps a third called 'Utilities' each containing the necessary structure to carry out these common functions.

You can actually have several floating palettes visible at the same time. 'Ctrl-3' will always start you with the 'Model Palette' but you can then right-click, select a different category, and 'Ctrl-3' again.

Appendix A

Control Keys in Authorware

Ctrl A	Select all of the objects in the current window
Ctrl B	Display the current working icon
Ctrl C	Copies selected object or text to the clipboard
Ctrl D	Displays the Fill Palette
Ctrl F	Displays the Find dialogue box
Ctrl G	**'Groups' two or more objects into one**
Ctrl Shift G	**'Ungroups' grouped objects into their constituent parts**
Ctrl I	**Displays the properties box for icon or display**
Ctrl J	**Jumps between the display window and the flow line**
Ctrl K	**Displays the Colour Palette for text and objects**
Ctrl L	Displays the Line Palette for thickness and style of lines
Ctrl M	**Displays the Mode palette for text and images**
Ctrl N	Creates a New Application
Ctrl Alt N	Create a New Library
Ctrl O	Opens an existing application
Ctrl P	Pauses or resumes an application
Ctrl Q	Quits Authorware or stops a running application
Ctrl R	Runs the current application
Ctrl S	Saves the current application
Ctrl T	**Displays the Transition Dialogue box**
Ctrl U	Displays the Video preferences box
Ctrl V	Pastes the contents of the clipboard
Ctrl W	Closes the current application
Ctrl X	Cuts the currently selected object or text to the clipboard
Ctrl Z	**Undoes the last operation**

The most useful and important control keys have been shown in bold, although you may have use for all of them at some time the bold ones are the ones that you should expect to use frequently.

A complete list of all of the available keyboard shortcuts can be found in the Help file under Keyboard Shortcuts.

Note that holding down the 'Ctrl' key and clicking once on the title bar of an icon will display the parents and grandparents of the icon. This is a highly useful function.

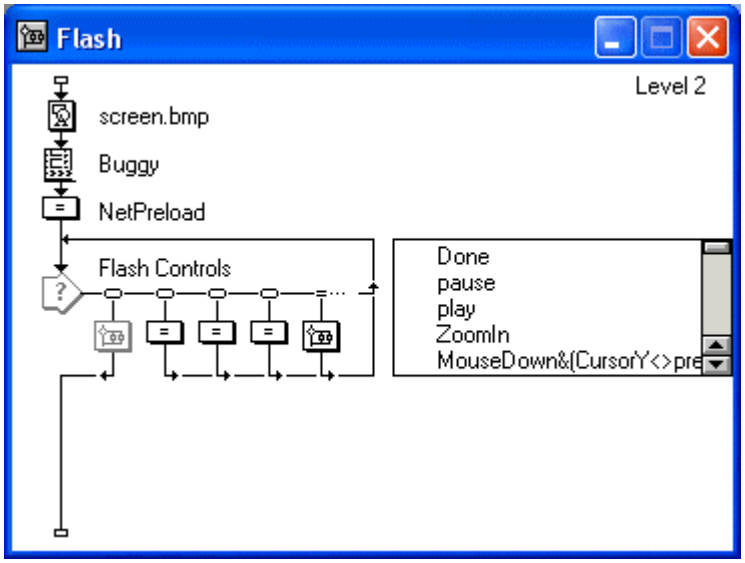
Appendix B

Using Flash Objects in Authorware

These notes do not attempt to explain how to create a Flash movie, however once such a movie has been created it can be inserted into and controlled with an Authorware application.

To simply insert a Flash movie click on 'Insert – Media – Flash' and then browse to find the movie file – within Authorware version 6 the Flash movie must have been saved as Flash version 5 or earlier.

This will then insert on the flow line a Flash icon. If you do nothing else then the movie will be played as it is encountered in the normal running of the piece. You can use a Delete icon to remove it from the flowline, but the normal transitions will have no effect on how the removal is carried out.



To actually control the movie you will need to either have created the necessary controls within the Flash movie itself (which is beyond the scope of this course), or you can exert a number of controls from within the Authorware piece.

Here is an example of an Authorware control structure that can control a Flash movie, displayed in an icon called 'Buggy'.

Notice that you actually place the Flash movie before the control structure on the flowline.

The NetPreload icon is simply there to ensure that the Next part of the piece is downloaded in the background, to the users computer while they are marvelling at the intricacies of the Flash movie that they are watching.

The Flash movie is controlled by changing its properties in response to the user clicking on the buttons attached to the 'Flash Controls' Interaction icon.

There are many of these properties and a full list can be found in the help system that is copied to your computer when Authorware is installed. (Actually this section of help is separate from the normal Authorware help. The files can be found in 'Program Files\Macromedia\Authorware 6\xtras\FlashAsset\Help').

As an example of how these properties are used, lets consider the two controls 'pause' and 'play'. When the user clicks on either of these controls a simple piece of code is executed in the Calc icons below the button responses.

The 'play' icon contains this:

```
if GetSpriteProperty(@"Buggy", #playing)   = FALSE then
   CallSprite(@"Buggy", #play)
end if
```

This first checks to see what the current state of the #playing property is. If it is FALSE, then the CallSprite() function is used to start the movie playing.

The 'pause' icon simply does the reverse:

```
if GetSpriteProperty(@"Buggy", #playing) then
   CallSprite(@"Buggy", #stop)
end if
```

If the status of the #playing property is TRUE (you don't actually need to use the comparison in this situation), then the CallSprite() function is used to stop the movie.

To zoom in on the movie (make it appear larger) the ZoomIn icon uses the following code:

```
percent:=percent+10
SetIconProperty(@"Buggy", #scale, percent)
```

See if you can figure out how to zoom out on the movie – make it appear smaller. Remember that you don't want it to zoom out beyond zero percent.

When the movie is first placed on the flow line it will always appear in the centre of the screen at some default size. You can change both its position and size directly by running the piece up until the Flash movie appears then press 'Ctrl – P' on the keyboard. The piece will pause and by clicking on the movie you can select it, drag it to a different part of the screen or use the standard Windows type grab handles around the movie to alter its initial size.

Note that by using the property controls as outlined above you will not make the on screen size of the movie change. All that you will be doing is making the appearance of the objects within the Flash movie change size. This is quite useful as if you have a graphic frame around the movie, it will remain around the movie even if you zoom in very close.