Simple Interaction

A More Interesting Page

Interactive Multimedia

Interaction is a very important part of CBT. Lessons with little or no interaction are derisively called "page turners." Such lessons are probably better off in traditional bound paper (book) format. On a computer, you can take advantage of interaction features that let the user explore and learn in different ways. Interactivity helps maintain the user's interest.

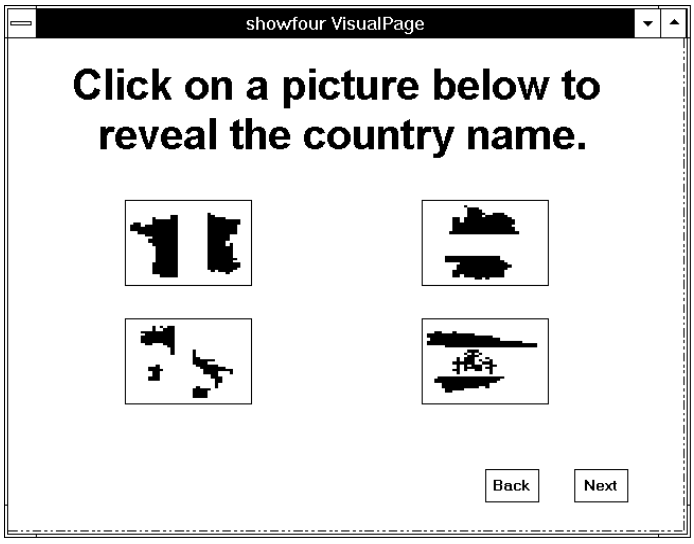
Objectives

After completing this chapter you will be able to

- erase only certain objects from a window
- delete an object from the Book Editor
- use the PicBin object
- embed graphics files in the book
- copy objects quickly
- display graphics on Buttons
- change the value of attributes at run time
- use the A-pex3 Assistant to write programming code
- proportionally resize objects to match the window

Page Appearance

When this page is complete, the VisualPage editor should resemble



Preparing the showfour Page

New Page

From the ToolSet

drag a Page icon

and drop it on the VisualPage editor.

When Everest asks for the name of the new page, enter

```
showfour
```

The showfour page will display in Buttons the flags of four countries, and let the user click on each Button to learn the corresponding country name.

Add a Layout

From the ToolSet

drag a Layout icon

and drop it on the VisualPage editor.

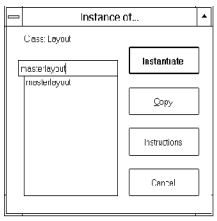
Whoops!

We want to make this Layout an instance of the one from the @start page. Normally, you would have dragged the icon via the right-side mouse button. We intentionally skipped telling you that step so that now you can learn a different way to instantiate.

Recall that in the previous chapter you used the SaveAsObject feature to create a master Layout object. To make this page's Layout an instance of the prior one, in the Attributes window

from the Options pull-down menu, choose Instance of...

The small "Instance of" window should appear.



To load an instance of the prior Layout

double click on masterlayout

Add an Erase

Next

put an Erase object in this page.

Recall the discussion of IDNumbers from the previous chapter, and that the Erase can be told to erase only certain objects. The Next button of the @start page has an IDNumber of 91. Since the showfour page will also have a Next button, let's avoid erasing the one that will still be in the window at run time when @start branches to this page.

So, in the Attributes window, set the Erase object's EraseFromID attribute to

```
1
```

and EraseToID to

```
90
```

If you leave the EraseFromID and EraseToID attributes empty, Everest erases all objects from the window (just as if EraseFromID were 1 and EraseToID were 99).

Add a Textbox

For this page's heading

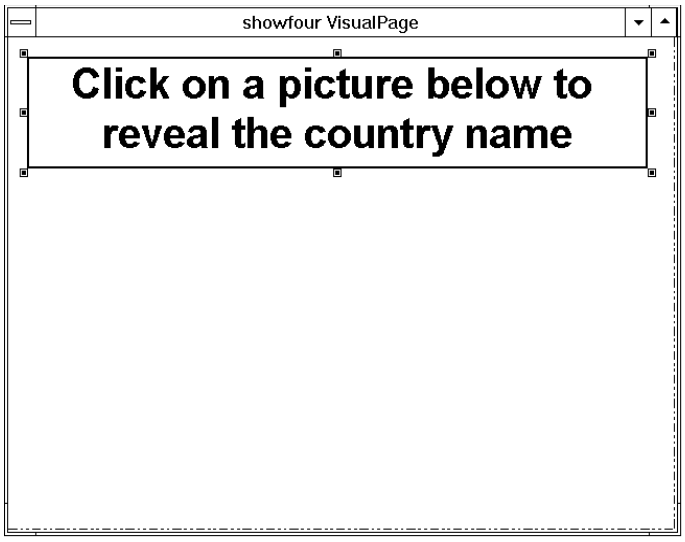
put a Textbox near the top of the VisualPage editor

and enter the following text in it

```
Click on a picture below to reveal the country
name.
```

Adjust the font (we suggest Arial in size 24), color and Alignment as you wish.

Your VisualPage editor should now resemble



The Picture Bin

Images Galore

Often interactive multimedia applications use dozens or even hundreds of small graphics images. This could easily apply to the project you build with Everest. If each image were stored in its own disk file, maintenance of the files would quickly grow tedious. Fortunately, Everest has a feature that lets you group similar images in one disk file, and extract them individually when needed. This feature is accessed via the PicBin object.

The PicBin is an invisible container for multiple images. When you want to display one of the images (say, on a Button, as you will do here in a few minutes) you simply request it by number.

Image Samples

Over 100 icon images are included in the Everest samples...not in individual files, but rather grouped together in a few .BMP files. One of the sample files is named FLAGS.BMP. Before you load it into a PicBin object, you should get an idea of how the images are stored.

Via the Peek button, you could view the FLAGS.BMP right from the Load File window, however, as an exercise, perform the following steps instead.

Add an SPicture Icon

From the ToolSet

drag an SPicture icon

and drop it on the VisualPage editor.

In the Attributes window

double click on SPictureFile

to open the File Load dialog box.

Double click on the flags.bmp file.

Your VisualPage editor should now resemble

Drag the sizing handles of the SPicture object to make it large enough to clearly show the FLAGS.BMP file. Notice how the flags of many different countries are contained within this single .BMP file. There are 10 images across and 7 images top to bottom (actually, only the first 3 of the 7 contain something, the others are empty).

Delete the SPicture Object

The purpose of this exercise was twofold: to try the SPicture object, and to learn how to delete an object. When you are done viewing the arrangement of the flag icons, delete the SPicture object (it is no longer needed). To do so

click on the SPicture object icon in the Book Editor

from the Book Editor's Edit menu, choose Delete.

Add a PicBin Object

Now your page is ready for the PicBin object. The PicBin object must appear in the page before (i.e. above) the objects that use the images. Since you have not yet added the Button objects that need the flag images, this is a good place for the PicBin.

Instead of dragging and dropping, you can double click on the PicBin icon in the ToolSet.

From the ToolSet

drag a PicBin icon

and drop it on the VisualPage editor.

Load FLAGS.BMP

In the Attributes window

double click on the BMPFile attribute

to open the Load File dialog box. Then

double click on flags.bmp

to tell Everest that's the file you want. Notice that the FLAGS.BMP image is NOT visible anywhere in the VisualPage editor. The PicBin holds the image in memory so you can later retrieve the individual icons stored within.

Linking vs. Embedding

Until now, all the graphics files you have used in this Tutorial have been referenced via their file name. For example, right now, the BMPFile attribute should read C:\everest\samples\flags.bmp. A reference like this is known as “linked file.”

Everest's Project Packager can automatically gather together all the linked files your book needs.

A linked file does *not* become part of the book. Instead, at run time, Everest searches on disk for the graphics file whose name you specified. Consequently, when you distribute your book to end users, you must also distribute the linked files. This means the end users have will have a copy of your graphics files that they can modify quite easily.

Perhaps you want more security for your graphics files in order to discourage end users from modifying or reusing them. Fortunately, Everest supports another storage method for graphics files: embedding. Embedded files differ from linked files in they are stored right in the book itself. A secondary benefit is that Everest compresses embedded files prior to storage in the book, thereby reducing their size.

Try Embedding

As a learning exercise, let's try embedding the FLAGS.BMP file into the book. In the Attributes window,

make sure BMPFile is still the attribute being edited (if not, click on it).

Then, from the Options pull-down menu

choose Embed File.

The Embed File window will open; in this window

find the file named flags.bmp

double click on flags.bmp

Upon doing so, you should see a message that resembles:

> FLAGS.BMP (length 143478 bytes) has been stored in C:\EVEREST\SAMPLES\ETUT.ESL, compressed to 9104 bytes.

Not bad...the file has been compressed to just 6% of its original size. Not all files will compress as much (.BMPs happen to compress very well), but the savings can be substantial. The smaller your project, the easier it is to distribute, especially electronically.

What's the | Character?

Notice the BMPFile attribute. It should now read |flags.bmp. Note the | character at the start of the file name. This is the character Everest uses to designate an embedded (rather than linked) file. Everest knows to ignore the disk path (if any) because now FLAGS.BMP is in the book; if you load a file from a different location, it leaves the disk path there simply to remind you of its original source location.

Verify the Columns and Rows

All images in a given PicBin file must have the same width and the same height.

Now, back to the other attributes. The PicBin does not know how many images are contained in the BMPFile...you must tell it. You do so by specifying the number of columns and rows of images. For FLAGS.BMP (as well as all the sample icon image files included with Everest) there are 10 columns and 7 rows.

By default, the PicBin is set for 10 columns and 7 rows. In the Attributes window, check the Columns and Rows attributes, and correct their values if necessary.

If you set Columns and/or Rows incorrectly, when you later display the images, they will appear off center, or you will see more than one at a time. That's a clue to go back to the PicBin object and verify the Columns and Rows settings.

Enable SaveAsObject

Since several pages in this book will need the same icons, in the Attributes window

double click on SaveAsObject to set it to Yes

Finally, change the Name of the PicBin (remember how?) to

masterflags

so it will be easier to reuse later.

The Four Buttons

The Approach

This page is ready for Buttons; a total of four are needed. We're going to start with one Button, set all the attributes as needed, then make three copies.

So, from the ToolSet

drag a Button icon

and drop it on the VisualPage editor beneath the left side of the Textbox.

The Attributes

In the Attributes window, set the following attributes as indicated:

set Width to `120`

set Height to `80`

set FontSize to `12`

set Font3d to `1` (Raised)

And,

double click on the CaptionColor attribute

to open the Color dialog box, and choose the box with the color

Black

which will contrast with the color of the country flags.

Making Copies of the Button

Everest has a feature that lets you quickly copy the current object. The current object is the one displayed in the Attributes window; right now, that should be the Button.

To copy the current object, simply hold down the Shift key while dragging an icon from the ToolSet. Do it now:

hold down the Shift key

drag a Button icon from the Toolset

drop it on the VisualPage editor *to the right of the first Button*.

Except for location, the new Button should look identical to the first.

Repeat for the third Button (again holding the Shift key), except

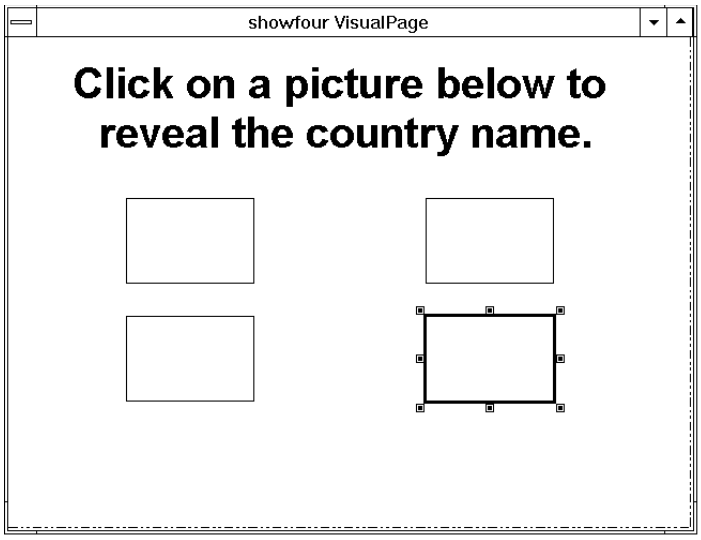
drop it on the VisualPage editor *under* the first Button.

Once more, for the last Button (again holding the Shift key), except

drop it on the VisualPage editor under the *second* Button.

You should now have a total of four Buttons in the VisualPage editor.

The Buttons should be arranged in an imaginary square. If not, adjust them so the first Button is in the upper-left corner of this imaginary square, the second in the upper-right, the third in the lower-left and the last in the lower-right. To move a Button, click the left mouse button on it first, then drag it via the right mouse button. Via the keyboard, you can "nudge" the current object one pixel by pressing Alt+CursorKey. When done, your VisualPage editor should resemble:



Displaying the Icons

The Country Flags

At this point, the Buttons are ready for the country flags. You specify which icon to display via the Pic attribute.

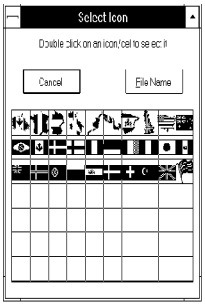
Now, in the VisualPage editor

click on the first Button (the one in the upper left) to highlight it.

Then, in the Attributes window,

double click on Pic

which opens the Select Icon window, shown below.



You want the country outline and flag for France, so

double click on the second icon from the left in the top row (cel 2).

When you do so, the icon should appear in the Button. Everest will set Pic to 2 (since the French flag is in the second cel in the PicBin). The flag image in the Button will appear a bit "grainy". This is because the original size of the image in the FLAGS.BMP is quite small, and the Button is magnifying it a fair amount. For the purposes of this Tutorial, this is acceptable; for actual use, you might want to make the FLAGS.BMP larger and more detailed, or use it on a smaller Button.

In the Attributes window,

from the Edit menu choose Rename

enter `france_button` as the new name

The Second Flag

Next, in the VisualPage editor

click on the second Button (upper right).

Then, in the Attributes window

double click on its Pic attribute.

In the Select Icon window

double click on the icon for Germany

which is in cel 3, immediately to the right of the icon for France.

In the Attributes window, use Rename and

enter `germany_button` as the new name

The Third Flag

Now, in the VisualPage editor

click on the third Button (the lower left one).

Again

double click on the Pic attribute.

In the Select Icon window

double click on the icon for Italy

which is in cel 4, to the right of the icon for Germany.

In the Attributes window, use Rename and

enter `italy_button` as the new name

The Fourth Flag

Finally, in the VisualPage editor

click on the last Button (lower right).

In the Attributes window

double click on the Pic attribute.

In the Select Icon window

double click on the icon for Spain

which is in cel 7.

In the Attributes window, use Rename and

enter `spain_button` as the new name

The Other Picxxx Attributes

You might have noticed that each Button has several Picxxx attributes (PicDown, PicUp, etc.). These other Picxxx attributes let you display different icons depending on the state of the Button (whether it is depressed, etc.). This feature will not be used on this page.

Navigation Buttons

Add a Next Button

This page is ready for navigation buttons. From the ToolSet,

use the *right* side mouse button to drag a Button icon

and drop it on the VisualPage editor.

When the popup menu appears

choose “Instantiate/Copy”

In the Instance of... window

double click on nextbutton

to instantiate it. (That sure beats creating it from scratch again!) Notice how Everest automatically repositions the Button to match that of the master copy of the nextbutton.

Create a Back Button

Since this page can back up to the @start page, it needs a Back button. You did not create such a Button with the @start page, so there's no object to instance...or is there? You certainly could create a Back button from scratch, however, there's an easier way. From the ToolSet

hold down the Shift key

using the left mouse button, drag a Button icon

and drop it on the VisualPage editor. Position it next to the Next Button.

This copies what had been the current object (the nextbutton). Now, all you need to do is rename it. In the Attributes window

from the Edit menu, choose Rename

change the name to `backbutton`

Change More Attributes

You don't want two Buttons on the page that both read "Next." So, in the Attributes window, change the Caption attribute (of the Button you just renamed) from Next to

```
Back
```

Also, change the ClickEvent from 34 to

```
33
```

which is the event code generated by the PgUp key. Finally, change the IDNumber to

```
92
```

Care to speculate about what would happen at run time if you had set the IDNumber to 91 (the same as the nextbutton)? Hint: the IDNumber uniquely identifies the object within its class in a window. The answer: Everest would run the page, display the Next button in the window, then immediately replace it with the Back button (because its IDNumber was the same). You can use this technique intentionally when you want to update an existing object in a window.

Wait for the User, and Alfred (?)

Add a Wait Object

Just as on the @start page, this page needs a Wait object. From the ToolSet

drag a Wait icon

and drop it on the VisualPage editor.

Trap Next and Back

The Wait object needs to handle a click on the Next *and* the Back buttons. In the Attributes window for the Wait object, set NextActivator to the PgDn key event code

```
34
```

and NextAction to

```
BRANCH quizfour
```

Also, set BackActivator to the PgUp event code

```
33
```

Tip: double click on BackAction, and Everest will set it to BRANCH @prev.

and BackAction to

```
BRANCH @prev
```

Branching to the Previous Page

BRANCH @prev? What does that mean?

Recall that the @ symbol in a page name means something special. Specifically, @prev is the special name used in a BRANCH instruction to branch to the previous page viewed by the user at run time.

Compare this with BRANCH @back, which was mentioned in the Quick Start. BRANCH @back branches to the page listed above the current one in the Book Editor at design time.

Everest can also maintain a menu stack; it's handy for projects that are menu driven.

At run time, Everest automatically remembers the user's path through your project by storing the page names on what is called the backup stack. When Everest sees a BRANCH to @prev, it removes the page name at the top of the backup stack, and branches to it. This feature eliminates the need for you to manually program backwards branching, and assures the user will back up to the page from which they had come.

A complete list of these special @ page names can be found in the technical reference/on-line help for the BRANCH command.

Button Captions

So far, you have not entered a Caption for the four flag Buttons. If you did so now, the Caption would appear immediately at run time, rather than wait for the user to click the Button. You want to add the Caption only after a click is detected. To do so, you will use a tiny bit of A-pex3 programming.

If you are not a programmer, don't worry. Alfred will help you.

Who's Alfred? Alfred is our nickname for the A-pex3 Assistant, a wizard that writes programming code for you.

Design and Run Time Changes

By now, you are probably comfortable changing the value of object attributes via the Attributes window. What you have been doing is called "changing the attributes at design time" ("design time" means "while authoring"). Doing so presets the appearance of the object for its *initial display* to the user.

Here is the conceptual leap: most of these very same attributes can also be changed at run time ("run time" means "when the user is viewing your project"). Doing so changes the appearance of an object that is *already displayed* to the user.

Modifying an Attribute at Run Time

To change the value of an attribute at run time, you need the following information:

- the object's name (such as france_button)
- the attribute name (such as Caption)
- the new value

Using the A-pex3 Assistant (Alfred!)

The A-pex3 Assistant helps you write individual lines of programming in Everest's built-in language. The syntax of the A-pex3 language is similar to that of Microsoft's Visual Basic (VB) and Visual Basic for Applications (VBA), so if you are familiar with VB or VBA, the code will look familiar.

When the user clicks a Button at run time, we want to display the name of the country within that Button. This is a job for the ClickEvent attribute.

ClickEvent Accepts Programming Too

Until now, we've only set the ClickEvent of various Buttons to either a character string, like "next", or a number, like 34. Here's what is significant: the ClickEvent also accepts *programming*. In fact, this is true for all the xxxEvent attributes; you may have seen the names of a few of the others: GotFocusEvent, MouseOverEvent, etc.

The French Caption

OK, in the VisualPage editor

click on the Button containing the icon for France

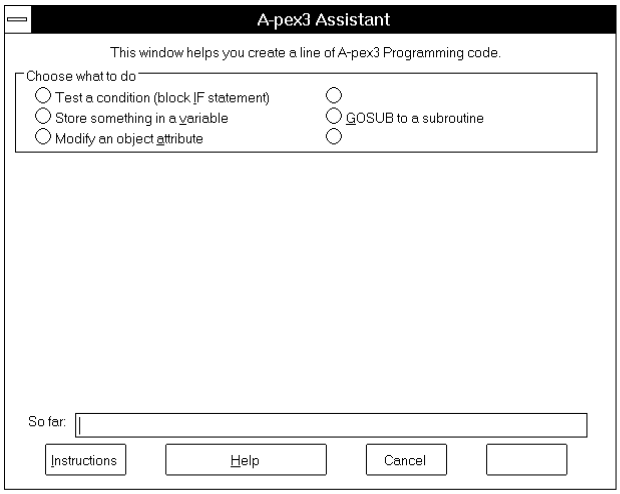
to focus on it. Then, in the Attributes window

click on the ClickEvent attribute.

Finally, in the Attributes window,

from the Help menu, choose Assistant

to open the A-pex3 Assistant window, which resembles:



Choose What to Do

First, tell the Assistant, oops, Alfred, what you want him to do. From the choices provided

select “Modify an object attribute”

and a new frame will appear.

Choose the Object

In this new frame, use the drop down list on the left and select

france_button

which is the name of the first Button.

Choose the Attribute

Use the drop down list on the right and select

Caption

and another new frame appears.

Store What in This Attribute?

Describe what you want to store in the attribute. From the available choices,

select “String constant”

and more frames appear.

Type the Character String

In the field near the bottom (where it says to “type the literal string below”)

type `France`

So Far

If you have made the correct choices, the So far box near the bottom should now display

france_button.Caption = “France”

This is the line of code the Assistant has programmed for you. If the programming is syntactically correct (which it should now be), the Assistant's OK button will be enabled.

It's OK

When done, in the Assistant window

click OK (use it)

and Everest will copy the programming into the Caption attribute in the Attributes window. No text appears in the Button in the VisualPage editor yet; that will happen only at run time.

Repeat for Other Buttons

Now, repeat the Assistant process above for the second Button (Germany). In the VisualPage editor

click on the Germany Button

to highlight it. Then, in the Attributes window

click on ClickEvent

and then choose Assistant from the Help pull-down menu. Follow the same process in the Assistant window, except for the object, choose

germany_button

and for the character string

type `Germany`

instead of France.

Italy and Spain

Repeat for the Italy and Spain Buttons. Be sure to choose italy_button and spain_button respectively for the objects, and enter

```
Italy
```
and `Spain`

as appropriate.

Try It

Save the Page

Before you preview your page, save it. From the Book Editor
in the Page pull-down menu, choose Save.

Preview

Next, in the Author window
from the Run menu, choose Preview.

Click on each of the flags. When you do so, the country name should appear.

Don't Stop Yet

There's something else we'd like to show you how to do. First, try this. Maximize the window in which the preview is now running. To do so,

> click on the arrow button in the extreme upper-right corner of the window that is previewing your page.

The preview window expands to fill the monitor (if you are running Windows in 640 x 480 the appearance will be approximately the same). But, if you are running at a higher Windows resolution, the objects in the preview window will stay put, and an empty area will appear to the right and below the objects.

Dynamic Resizing

Sometimes authors like to proportionally resize the objects so they fit inside the window nicely. With Everest, that's a built-in feature. We'll tell you how in a moment. For now, restore the window to normal size by once again clicking on the button in the extreme upper-right corner.

Stop

Recall that Ctrl+M brings the Author window to the top.

Now, to stop the preview, from the Author window's Run menu, choose

Stop

or, double click on the control box in the upper-left corner of the preview window.

Edit the Layout

The Layout is the first object...you may need to scroll upward in the Book Editor.

After the preview window closes, Everest returns you to the editors. Click on the

Layout

icon (named masterlayout) in the Book Editor. Then, in the Attributes window

double click on the AutoResize attribute

to set it to `Yes`. When AutoResize is enabled, Everest proportionally resizes objects when the size of the window changes.

Preview Again

Again, in the Author window

from the Run menu choose Preview.

After the preview has started

click on the maximize button.

Now Everest adjusts the size and location of the objects to match.

Restore, Stop and Save

At this time, you can restore the Preview window's original size. Then, from the Author window's Run menu

choose Stop.

Finally, in the Book Editor

from the Page menu, choose Save.

Review

Review

In this chapter you learned how to

- erase only certain objects from a window
- delete an object from the Book Editor
- use the PicBin object
- embed graphics files in the book
- copy objects quickly
- display graphics on Buttons

- change the value of attributes at run time
- use the A-pex3 Assistant to write programming code
- proportionally resize objects to match the window