Toolbars

You can further enhance your application's menu interface with toolbars. Toolbars contain toolbar buttons, which provide quick access to the most frequently used commands in an application. For example, the Visual Basic toolbar contains toolbar buttons to perform commonly used commands, such as opening existing projects or saving the current project.

- Creating a Toolbar Techniques for creating your own toolbars.
- Negotiating Menu and Toolbar Appearance Working with toolbars and menus for insertable objects.

Creating a Toolbar

The *toolbar* (also called a ribbon or control bar) has become a standard feature in many Windows-based applications. A toolbar provides quick access to the most frequently used menu commands in an application. Creating a toolbar is easy and convenient using the toolbar control, which is available with the Professional and Enterprise editions of Visual Basic. If you are using the Learning Edition of Visual Basic, you can create toolbars manually as described in "Negotiating Menu and Toolbar Appearance" later in this chapter.

The following example demonstrates creating a toolbar for an MDI application; the procedure for creating a toolbar on a standard form is basically the same.

To manually create a toolbar

1. Place a picture box on the MDI form.

   The width of the picture box automatically stretches to fill the width of the MDI form's workspace. The workspace is the area inside a form's borders, not including the title bar, menu bar, or any toolbars, status bars, or scroll bars that may be on the form.

   Note You can place only those controls that support the Align property directly on an MDI form (the picture box is the only standard control that supports this property).

2. Inside the picture box, place any controls you want to display on the toolbar.

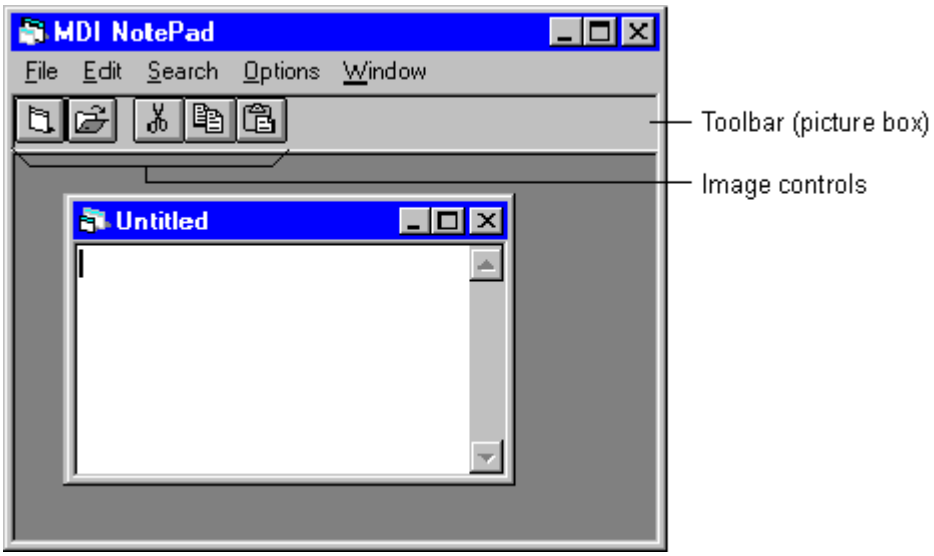
   Typically, you create buttons for the toolbar using command buttons or image controls. Figure 6.16 shows a toolbar containing image controls.

   To add a control inside a picture box, click the control button in the toolbox, and then draw it inside the picture box.

   Note When an MDI form contains a picture box, the internal area of the MDI form

does not include the area of the picture box. For example, the ScaleHeight property of the MDI form returns the internal height of the MDI form, which does not include the height of the picture box.

Figure 6.16 You can create buttons for the toolbar using image controls



3. Set design-time properties.

   One advantage of using a toolbar is that you can present the user with a graphical representation of a command. The image control is a good choice as a toolbar button because you can use it to display a bitmap. Set its Picture property at design time to display a bitmap; this provides the user with a visual cue of the command performed when the button is clicked. You can also use *ToolTips*, which display the name of the toolbar button when a user rests the mouse pointer over a button, by setting the ToolTipText property for the button.

4. Write code.

   Because toolbar buttons are frequently used to provide easy access to other commands, most of the time you call other procedures, such as a corresponding menu command, from within each button's Click event.

Tip You can use controls that are invisible at run time (such as the timer control) with an MDI form without displaying a toolbar. To do this, place a picture box on the MDI form, place the control in the picture box, and set the picture box's Visible property to False.

Writing Code for Toolbars

Toolbars are used to provide the user with a quick way to access some of the application's commands. For example, the first button on the toolbar in Figure 6.16 is a shortcut for the File New command. There are now three places in the MDI NotePad sample application where the user can request a new file:

- On the MDI form (New on the MDI form File menu)
- On the child form (New on the child form File menu)
- On the toolbar (File New button)

Rather than duplicate this code three times, you can take the original code from the child form's mnuFileNew_Click event and place it in a public procedure in the child form. You can call this procedure from any of the preceding event procedures. Here's an example:

```
' This module is in a public procedure.
Public Sub FileNew ()
   Dim frmNewPad As New frmNotePad
   frmNewPad.Show
End Sub

' The user chooses New on the child form File menu.
Private Sub mnuchildFileNew_Click ()
   FileNew
End Sub

' The user chooses New on the MDI form File menu.
Private Sub mnumdiFileNew_Click ()
   frmNotePad.FileNew
End Sub

' The user clicks the File New button on the toolbar.
Private Sub btnFileNew_Click ()
   frmNotePad.FileNew
End Sub
```

Negotiating Menu and Toolbar Appearance

When an object supplied by another application is activated on a form, there are a number of ways that object's menus and toolbars may appear on the container form; therefore, you need to specify how they will be displayed. This process is called *user-interface negotiation* because Visual Basic and the object you have linked or embedded must negotiate for space in the container form.

Controlling Menu Appearance

You can determine whether a linked or embedded object's menu will appear in the container form by setting a form's NegotiateMenus property. If the child form's NegotiateMenus property is set to True (default) and the container has a menu bar defined, the object's menus are placed on the container's menu bar when the object is activated. If the container has no menu bar, or the NegotiateMenus property is set to False, the object's menus will not appear when it is activated.

Note The NegotiateMenus property does *not* apply to MDI Forms.

Controlling Toolbar Appearance

The MDI form's NegotiateToolbars property determines whether the linked or embedded object's toolbars will be floating palettes or placed on the parent form. This behavior does not

require toolbars to be present on the MDI parent form. If the MDI form's NegotiateToolbars property is True, the object's toolbar appears on the MDI parent form. If NegotiateToolbars is False, the object's toolbar will be a floating palette.

Note The NegotiateToolbars property applies *only* to MDI forms.

If an MDI form includes a toolbar, it is usually contained in a picture box control on the parent form. The picture box's Negotiate property determines whether the container's toolbar is still displayed or is replaced by the object's toolbar when activated. If Negotiate is True, the object's toolbar is displayed in addition to the container's toolbar. If Negotiate is False, the object's toolbar replaces the container's toolbar.

Note Menu and toolbar negotiation will occur only for insertable objects that support in-place activation. For more information on in-place activation, see "Programming with ActiveX Components."

You can see how these three properties interact by using the following procedure.

To perform menu and toolbar negotiation

1. Add a toolbar to an MDI form. This is described in "Creating a Toolbar" earlier in this chapter.
2. Place an insertable object on a child form.
3. Set the NegotiateMenus, NegotiateToolbars, and Negotiate properties.
4. Run the application, and double-click the object.
