Advanced Controls

Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

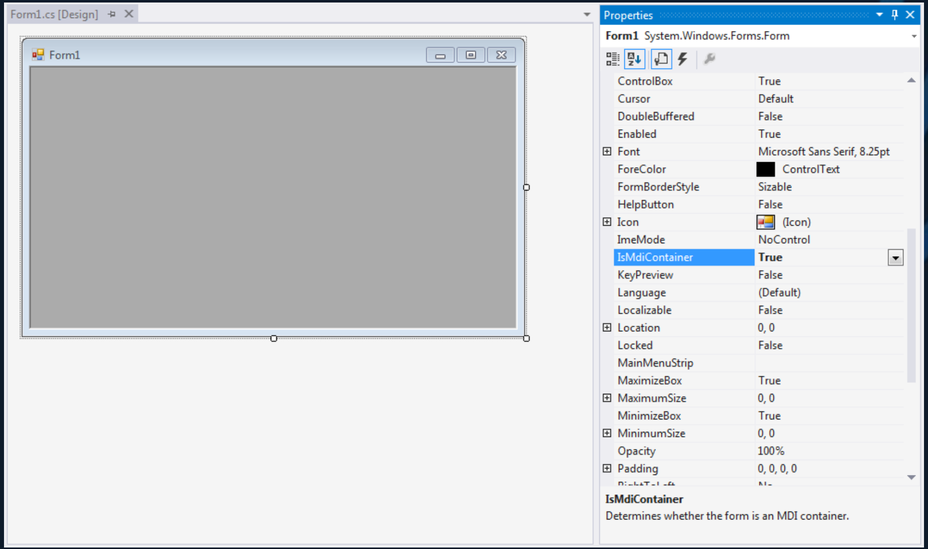
Contents

Multi-Document Interface (MDI) Applications

- Enables display of multiple documents concurrently
- Each document displays in its own window
- Often have a Window menu for switching between windows/documents

Multi-Document Interface (MDI) Applications

- Create a new Windows Form Application
- Set the *IsMdiContainer* property to *true*
- Set *WindowState* property to *maximized*



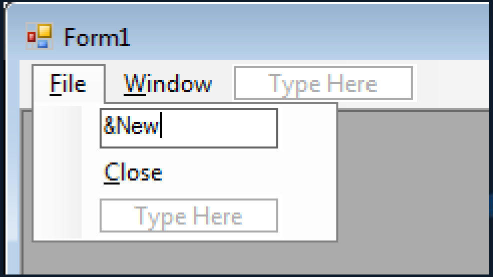
Form1 System.Windows.Forms.Form

ControlBox	True
Cursor	Default
DoubleBuffered	False
Enabled	True
⊞ Font	Microsoft Sans Serif, 8.25pt
ForeColor	ControlText
FormBorderStyle	Sizable
HelpButton	False
⊞ Icon	(Icon)
ImeMode	NoControl
IsMdiContainer	True
KeyPreview	False
Language	(Default)
Localizable	False
⊞ Location	0, 0
Locked	False
MainMenuStrip	
MaximizeBox	True
⊞ MaximumSize	0, 0
MinimizeBox	True
⊞ MinimumSize	0, 0
Opacity	100%
⊞ Padding	0, 0, 0, 0

IsMdiContainer
Determines whether the form is an MDI container.

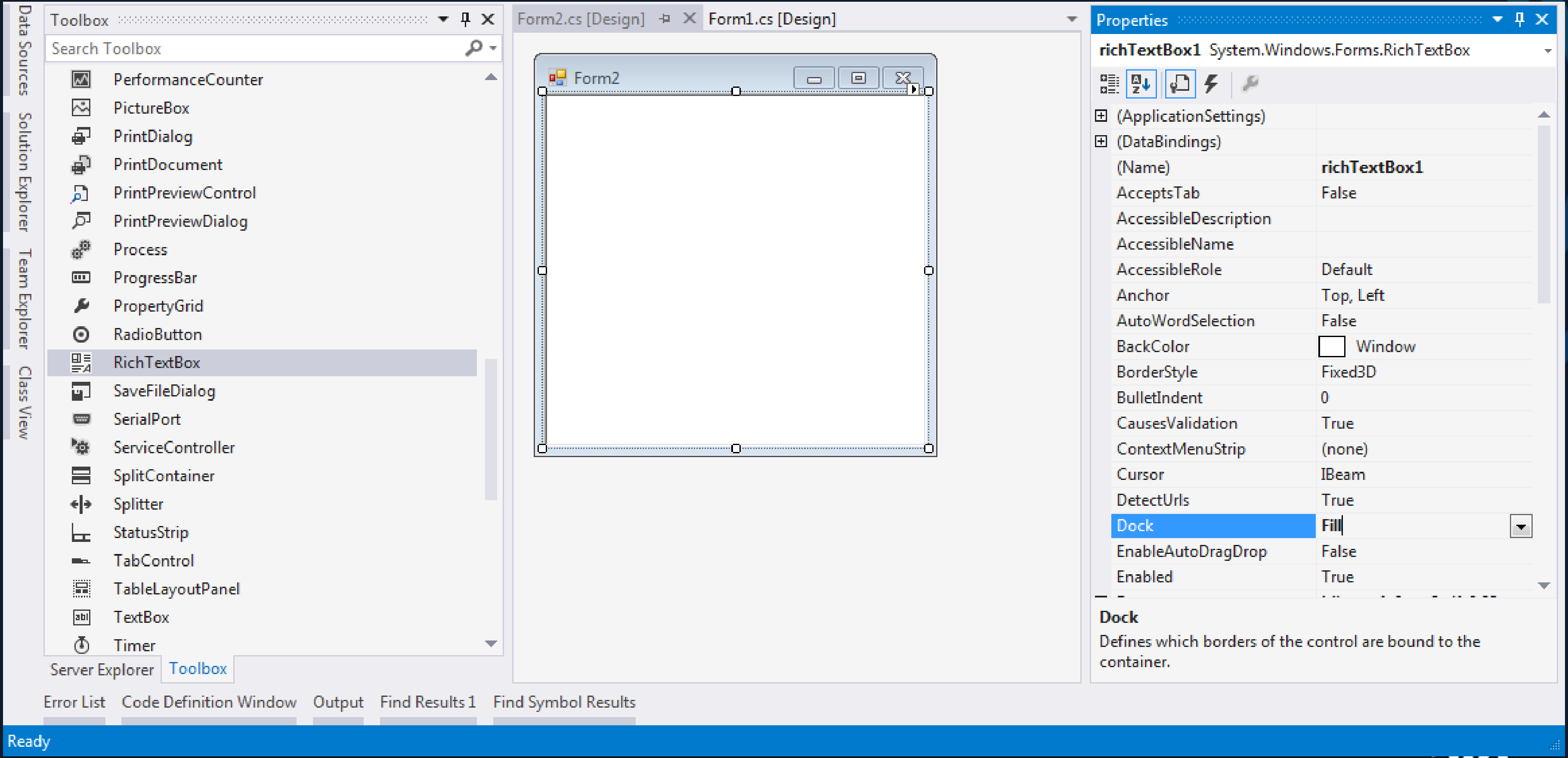
Multi-Document Interface (MDI) Applications

- Drag a MenuStrip onto the form
- Create a tip-level menu item with the text &File
- Create sub-menu items &New and &Close
- Create a top-level menu item called &Window



Multi-Document Interface (MDI) Applications

- In *Solution Explorer*, right-click the project -> *Add* -> *Add New Item*
- Select *Windows Form*
- Name the form *Form2*
- Drag a *RichTextBox* control onto the form
- Set the *Anchor* property to *Top, Left*
- Set the *Dock* property to *Fill*



richTextBox1 System.Windows.Forms.RichTextBox

(ApplicationSettings)	
(DataBindings)	
(Name)	richTextBox1
AcceptsTab	False
AccessibleDescription	
AccessibleName	
AccessibleRole	Default
Anchor	Top, Left
AutoWordSelection	False
BackColor	Window
BorderStyle	Fixed3D
BulletIndent	0
CausesValidation	True
ContextMenuStrip	(none)
Cursor	IBeam
DetectUrls	True
Dock	Fill
EnableAutoDragDrop	False
Enabled	True

Dock
Defines which borders of the control are bound to the container.

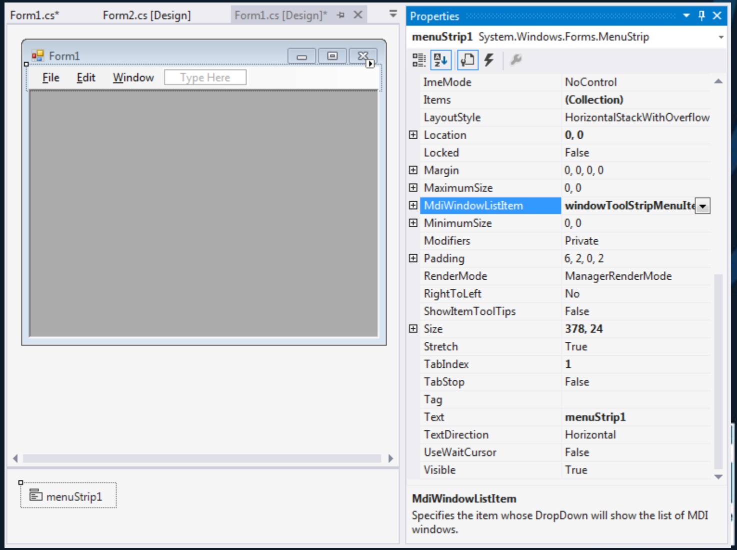
Multi-Document Interface (MDI) Applications

- Double-click on the *New* menu item
- Insert this code in the event handler

```
private void NewMenuItem_Click(object sender, EventArgs e)
{
    Form2 newMDIChild = new Form2();
    // Set the Parent Form of the Child window.
    newMDIChild.MdiParent = this;
    // Display the new form.
    newMDIChild.Show();
}
```

Multi-Document Interface (MDI) Applications

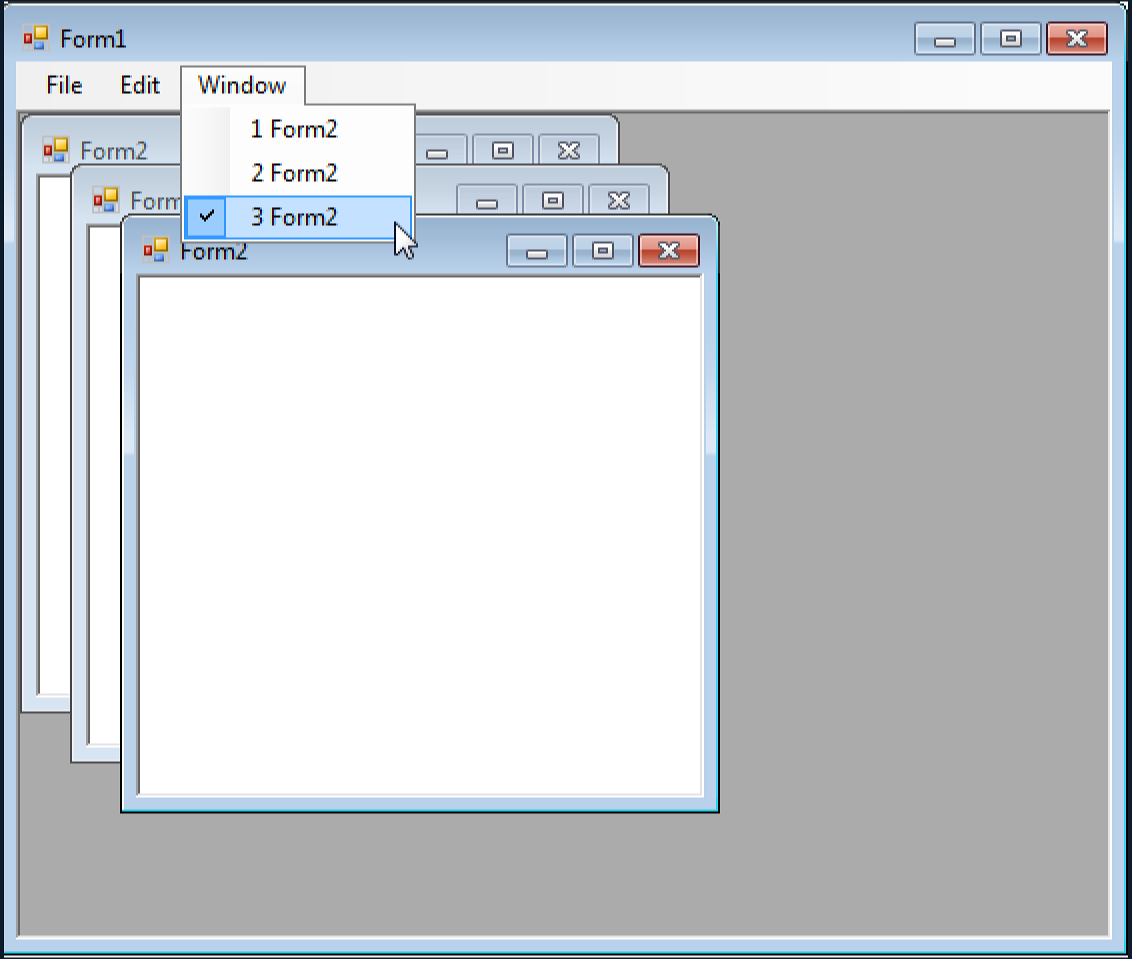
- On *Form1*, select the *MenuStrip*
- Set the *MdiWindowListenItem* property to *windowToolStripMenuItem*
- The *Window* menu will now maintain a list of open MDI children



menuStrip1 System.Windows.Forms.MenuStrip

ImeMode	NoControl
Items	(Collection)
LayoutStyle	HorizontalStackWithOverflow
⊞ Location	0, 0
Locked	False
⊞ Margin	0, 0, 0, 0
⊞ MaximumSize	0, 0
⊞ MdiWindowListItem	windowToolStripMenuIte
⊞ MinimumSize	0, 0
Modifiers	Private
⊞ Padding	6, 2, 0, 2
RenderMode	ManagerRenderMode
RightToLeft	No
ShowItemToolTips	False
⊞ Size	378, 24
Stretch	True
TabIndex	1
TabStop	False
Tag	
Text	menuStrip1
TextDirection	Horizontal
UseWaitCursor	False
Visible	True

MdiWindowListItem
Specifies the item whose DropDown will show the list of MDI windows.


Form1
File
Edit
Window
1 Form2
2 Form2
✓ 3 Form2
Form2
Form2
Form2

Cut and Paste

- With an MDI application, we need to determine the active MDI child
  - We can use the ActiveMdiChild property
- If the form has several controls, we must also specify which control is active
  - We can use the ActiveControl property

Cut and Paste

```
private void CopyMenuItem_Click(object sender, EventArgs e)
{
    // Determine the active child form.
    Form activeChild = this.ActiveMdiChild;

    // If there is an active child form, find the active control, which
    // in this example should be a RichTextBox.
    if (activeChild != null)
    {
        try
        {
            RichTextBox theBox = (RichTextBox)activeChild.ActiveControl;
            if (theBox != null)
            {
                // Put the selected text on the Clipboard.
                Clipboard.SetDataObject(theBox.SelectedText);
            }
        }
        catch
        {
            MessageBox.Show("You need to select a RichTextBox.");
        }
    }
}
```

Cut and Paste

- The Clipboard object exposes functions to write to, or read from the clipboard
- Clipboard.SetDataObject
- Clipboard.GetDataObject
  - Returns an IDataObject
  - Ensure the data is text, then retrieve the text data

Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Cut and Paste

```
private void PasteMenuItem_Click(object sender, EventArgs e) {
    // Determine the active child form.
    Form activeChild = this.ActiveMdiChild;

    // If there is an active child form, find the active control, which
    // in this example should be a RichTextBox.
    if (activeChild != null) {
        try  {
            RichTextBox theBox = (RichTextBox)activeChild.ActiveControl;
            if (theBox != null) {
                // Create a new instance of the DataObject interface.
                IDataObject data = Clipboard.GetDataObject();
                // If the data is text, then set the text of the
                // RichTextBox to the text in the clipboard.
                if (data.GetDataPresent(DataFormats.Text)) {
                    theBox.SelectedText = data.GetData(DataFormats.Text).ToString();
                }
            }
        }
        catch {
            MessageBox.Show("You need to select a RichTextBox.");
        }
    }
}
```

Drag and Drop Operations

- In *Form2* replace the *RichTextBox* with a standard *TextBox*
  - *RichTextBox* offers drag-and-drop via the *EnableAutoDragDrop* property
- For the *TextBox*, set *AllowDrop* property to *True*
- Set *Multiline* property to *True*
- Add the *MouseDown*, *DragDrop* and *DragEnter* event handlers

Drag Operations

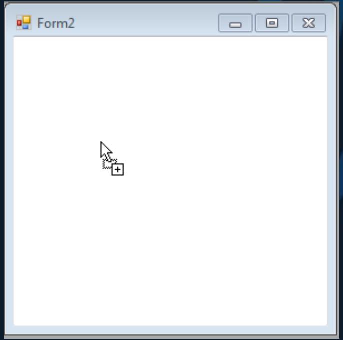
```
private void textBox1_MouseDown(object sender, MouseEventArgs e)
{
    textBox1.DoDragDrop(textBox1.Text, DragDropEffects.Copy |
        DragDropEffects.Move);
}
```

- textBox1.SelectedText will get the currently selected text in the textbox
  - In this instance that won’t work well because dragging to select will instead initiate the drag-and-drop operation

Drop Operations

```
private void textBox1_DragEnter(object sender, DragEventArgs e)
{
    if (e.Data.GetDataPresent(DataFormats.Text))
        e.Effect = DragDropEffects.Copy;
    else
        e.Effect = DragDropEffects.None;
}

private void textBox1_DragDrop(object sender, DragEventArgs e)
{
    textBox1.Text = e.Data.GetData(DataFormats.Text).ToString();
}
```



Integrating User Help

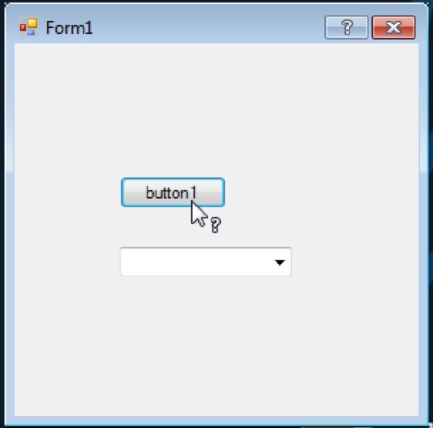
- Often overlooked, but important
- Windows Forms supports 2 different types:
  - Pointing to the Help file
    - HTML or HTML Help 1.x or greater
  - “What’s This” on individual controls
- Both types can be used on the same form
- Tool tips can also be used on controls

Integrating User Help

- Drag the HelpProvider component onto the form
- Create an HTML file containing the help
- Set the HelpProvider's *HelpNamespace* property to this file
- Set the form's *MinimizeBox* and *MaximizeBox* properties to *false*, and *HelpButton* to *true*
- A help icon will appear at the top-right of the form

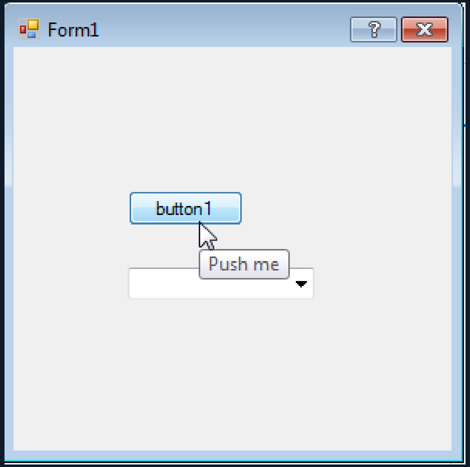
Integrating User Help

- On the control you want to add help to, set the *HelpKeyword* property
- The *HelpNavigator* property determines the way the *HelpKeyword* is passed to the help system
- More information at: https://msdn.microsoft.com/en-us/library/wxdbf1a7(v=vs.110).aspx
- Use the *?* icon to activate the ‘What’s This?’ cursor, or press F1 for help



Tool Tips

- Add the ToolTip control to the form
- Set the *ToolTip* property for any control on the form



Summary

- Multi-Document Interface Applications can contain many child form windows
- Cut and Paste by specifying the correct form, via the ActiveMdiChild property
- Drag and Drop functionality added via the DragEnter and DragDrop events
- Integrate User Help for greater usability