

Programming Techniques: Win32 vs. .NET

CTEC1831/2010F
Windows Programming II

Main Window

Win32

- `WNDCLASSEX` structure
- `CreateWindow` function:
2nd argument sets window title
- `HWND` window handle

.Net

- `Form1` class
- `Text` property is window title
- **C#:** `"this"` reference (*not strictly required*)
- **VB:** `"Me"` reference

Main Window: Resizing

Win32

- `CreateWindow` function
- `MoveWindow` function
- `SetWindowPos` function
- `GetClientRect`,
`GetWindowRect` functions

.Net

- `ClientSize` property (client area)
- `Size` property (window)
- `MinimumSize`,
`MaximumSize` properties (window)

Main Window: Handling Events

Win32

- `WindowProc` function
- `WM_XXX` constants (defined in `windows.h`)
- Each message has a `case`
- You can write a separate function to handle each message

.Net

- You decide which events to handle when designing the form
- Each event is handled by a separate method

Main Window: Message Loop

Win32

- `MSG` structure
- `GetMessage`, `DispatchMessage` functions

.Net

- Built into application framework
- Because Win32 is still the *native* API, you can tap into the message queue and intercept or ignore `WM_` messages

Events: Main Window Created

Win32

- `WM_CREATE` message

.Net

- Form `Load` event

Global Variables

Win32

- You can use true global variables (dangerous) ... naming convention is recommended or encapsulate them in a structure and/or function

.Net

- Data members at the Form class level
- If you have multiple Form objects on the screen, you can make the data members **static** (C#) or **Shared** (VB)

Resources

Win32

- Stored in `.rc` file
- Accessed using string identifiers or integer IDs defined in `resource.h` file with `MAKEINTRESOURCE` macro
- Various functions to load them, e.g., `LoadBitmap`

.Net

- Defined and stored as part of project (`My Project` in VB; `Resources.resx` in C#)
- Accessed as data member: `My.Resources.xxx` (VB) or `Properties.Resources.xxx` (C#)

Menu & Accelerators

Win32

- Stored in **.rc** file
- Attached to main window by **WNDCLASSEX** structure
- Accelerators are defined separately, and use **LoadAccelerators** and **TranslateAccelerator** functions

.Net

- **MenuStrip** control
- Designed into form, and becomes one of its class members ... so does each menu item
- Accelerators are built in to each menu item as **ShortcutKeys** property

Events: Menu Items & Accelerators

Win32

- Each menu item/accelerator is identified by an integer ID, represented by a constant, defined in the `resource.h` file
- `WM_COMMAND` message; menu item is `LOWORD(wParam)`

.Net

- Each menu item/accelerator has its own event handler method

Events: Menu Items & Accelerators (2)

Win32

- To simulate a menu item being clicked in code, call the `SendMessage` function and send a `WM_COMMAND` message with `MAKELPARAM(menuItemID, 0)`

.Net

- Call the `menuItemName.Click()` method

Events: Window Closing

Win32

- `WM_CLOSE` message
- Trigger it by calling the `SendMessage` function

.Net

- Form `FormClosing` event
- Trigger it by calling the form's `Close` method

Message Boxes

Win32

- `MessageBox` functions
- `MB_XXX` button styles and icons; `ID_XXX` return values (constants in `windows.h`)

.Net

- **VB:** `MsgBox` function; `MsgBoxStyle` and `MsgBoxResult` enumerated (`Enum`) types
- **VB/C#:** `MessageBox` class and `Show` method; `MessageBoxButtons`, `MessageBoxIcon`, `DialogResult` enum types

Exit Confirmation: Win32

```
case WM_CLOSE :
{
    INT rc = MessageBox( hWnd,
                        "Are you sure you want to exit?",
                        "Quit Program",
                        MB_YESNO | MB_ICONQUESTION ) ;

    if ( rc == IDYES )
    {
        DestroyWindow( hWnd ) ;
    }
}
break ;
```

Exit Confirmation: VB

```
Private Sub Form1_FormClosing(ByVal sender As System.Object, _  
    ByVal e As System.Windows.Forms.FormClosingEventArgs) _  
    Handles MyBase.FormClosing  
  
    Dim res As MsgBoxResult  
  
    res = MsgBox("Are you sure you want to close?", _  
        MsgBoxStyle.YesNo + MsgBoxStyle.Question, "Close")  
  
    If res <> MsgBoxResult.Yes Then  
        e.Cancel = True  
    End If  
  
End Sub
```

Exit Confirmation: C#

```
private void Form1_FormClosing(object sender,
    FormClosingEventArgs e)
{
    DialogResult res = MessageBox.Show(
        "Are you sure?",
        "Exit",
        MessageBoxButtons.YesNo,
        MessageBoxIcon.Question
    );

    if (res.Equals(DialogResult.No))
    {
        e.Cancel = true;
    }
}
```

Events: Painting

Win32

- `WM_PAINT` message
- `PAINTSTRUCT` structure
- `BeginPaint`, `EndPaint` functions
- `HDC` handle to device context

.Net

- Form `Paint` event
- `PaintEventArgs` class
- `Graphics` class

Painting Bitmaps

Win32

- `WM_CREATE` message
- `HBITMAP` type
- `LoadBitmap` function

.Net

- Automatically loaded as resources
- Assign to `Bitmap` variables at class level or subroutine level

Painting Bitmaps (2)

Win32

- `WM_PAINT` message
- `HDC` memory device context
- `CreateCompatibleDC`,
`SelectObject`,
`BitBlt`, `DeleteDC`
functions

.Net

- `DrawImage` methods
(LOTS!) in the `Graphics`
class

Painting Text

Win32

- TextOut, DrawText, GetTextExtentPoint32 and GetTextMetrics functions
- TEXTMETRIC, SIZE structures

.Net

- Graphics class methods: DrawString, MeasureString,
- System.Drawing.Text Namespace
- TextRenderer class

Repainting

Win32

- InvalidateRect,
UpdateWindow
functions

.Net

- Form class methods:
Invalidate,
Update

Mouse Clicks

Win32

- `WM_LBUTTONDOWN` message
- (x, y) are in `LOWORD(lParam)` and `HIWORD(lParam)`, respectively

.Net

- Form `Click` event
- `MouseEventArgs` class (cast it from the `EventArgs` parameter in the event handler method)
- (x, y) are `MouseEventArgs` members `X` and `Y`, respectively

Key Presses

Win32

- `WM_KEYDOWN`, `WM_CHAR` messages
- For `WM_CHAR`, ASCII code is in `wParam`

.Net

- Form `KeyPress` event
- Character is in `EventArgs` member `KeyChar`

Dialog Boxes

Win32

- Design dialog in resource editor with dialog ID and control IDs in `resource.h`
- `DialogBox/DialogBoxParam, DLGPROC` functions

.Net

- Design a second form in the form editor; it generates another class and source code file(s)
- Display the form by creating an object using the default constructor and then calling the `ShowDialog` method

Dialog Boxes (2)

Win32

- Write your own callback function and handle messages like `WM_INITDIALOG`, `WM_COMMAND`, and `WM_CLOSE`
- Use global variables

.Net

- Handle form events for the dialog box just as you would for the main form
- Pass the main form object to the dialog box constructor to be able to set main form data members

Dialog Boxes (3)

Win32

- An “About” dialog box is generated by the Win32 App Wizard; you can customize it

.Net

- An “About Box” windows form is one of the common items that you can add to your project
- The layout is not as easily customized
- The content of the “About Box” is taken from the Assembly Information in the Project settings

More Information

- Language Equivalents:

<http://msdn.microsoft.com/en-us/library/czz35az4%28v=VS.71%29.aspx>