Menus and Printing

Menus

• The focal point of most Windows applications
  – Almost all applications have a Main Menu bar
  – Main Menu Bar resides under the title bar
  – Main Menu contains menu items
    • Short words/phrases representing actions that can be selected
    • Many of these items are themselves menus
      – “Popup menus” (“drop-down menus”, “submenus”)
    – Main Menu contains “top-level” items
      • Always visible
      • Contains an array of Menu Items
    – Menus can be nested – form a hierarchy
      • Each Menu Item can contain an array of other Menu Items
  – Menu classes – all derived from abstract Menu class
    • Subclasses: MainMenu, MenuItem, ContextMenu classes
    • Not derived from Control class so properties like BackColor, ForeColor, and Font are not available
      – To change these and/or draw images, set DrawMode property to true
        > Then you must install handlers for MeasureItem & DrawItem events

MainMenu Class

• Constructors:
  – MainMenu()
    • If this variant is used, MenuItems must be added to it in code
  – MainMenu(MenuItem[] ami)
    • ami is an array of MenuItem objects to be included in the main menu
    – Attach a MainMenu to a form by assigning it to the form’s Menu property, e.g.:
      this.Menu = new MainMenu(new MenuItem[] {mi_1, mi_2, ...});
    • mi_1, mi_2, etc. are instances of the MenuItem class

MenuItem Class

• Several constructors to create a single menu item:
  – MenuItem();
  – MenuItem(string strText); // strText is the text that appears
  – MenuItem(string strText, EventHandler(ehClick));
    • EventHandler is the Delegate
  – MenuItem(string strText, MenuItem[] ami)
    • ami is an array of MenuItem objects
      – the items to be included in this menu item’s submenu

MenuItem Properties

• Important ones:
  – string Text
  – string Shortcut
  – bool ShowShortcut
  – bool Visible
  – bool Enabled
  – bool Break
  – bool BarBreak
  – bool Checked
  – bool RadioCheck

Manual Coding of a Menu

• Do it “bottom up”
  – Define low-level Menu Items first
  – Then their parents
  – Finally the Main Menu
  – In each case, attach menu items to their parent
• See Menu-Drawing-Manual example program
Using VS Designer to Prepare Menus

- Just drag a “MainMenu” from the tool box to the form
  - It will appear in the component tray below the form
  - Brings up the menu editor/designer
- Where it says “Type Here”, type in menu items and change their Text and other properties in their property boxes
  - In the Text property, prefixing a character with ‘&’ causes an <Alt> keyboard shortcut
  - Submenu items go below, menu items at the same level in the hierarchy to the right
  - Double click on a menu item to add a skeleton Click event handler
- Then just type in the desired handler code
- Set the form’s Menu property to the new main menu
- Menu-Drawing-Designer example program

Context Menus

- A menu that appears at the position of the mouse when mouse is right-clicked on a form or a control
  - Can have different context menus for different controls on a form
- Usually simpler than a main menu
  - Usually don’t contain submenus
- Instantiate a ContextMenu object, set its properties, its menu item click event handlers, etc.
  - Just like for a main menu
- Attach it to the control or form by setting the control’s or form’s ContextMenu property to the context menu
- Or use VS Designer to drag a ContextMenu from the tool box to the control it is to be associated with
  - set its menu items and properties
  - double click to add click handlers

Context Menu Example Programs

- Context -Menu-Manual (Coded manually)
  - Context menu is set to background color when user right clicks on form
  - A new ContextMenu is instantiated, filled with 8 color menu items, and attached to the form
  - this.ContextMenu = new ContextMenu(ami); //ami an array of menu items
  - Menu items have radio buttons – code sets the Checked property of the radio item selected
  - Note use of one handler for all context menu items – can’t do this with VS Designer
- Context -Menu-CDiagBox (VS Designer)
  - Uses a context menu to choose the form’s foreground color and a font for some text in a label
    - Color menu item starts a common color dialog box
    - Font menu item starts a common font dialog box
  - Use VS Designer to drag a context menu, a common color dialog box, and a common font dialog box onto form
  - Set form’s ContextMenu property to the name of Context Menu (property box)
  - Double click on context menu items to add handlers that invoke and use the common dialog boxes

Print in Windows

- Win32 API Printing is complex
  - In some ways like displaying on a screen form
  - But there are many unique printer issues:
    - Is printer on line?
    - Does printer have paper?
    - Is there color support?
    - How much graphics support is there?
    - Wide variety of printer types
    - Printer options
      - Trays, bins, paper sizes, etc.
      - Printers are slower than video displays
      - Programs reuse video display surface
      - Printer must eject completed pages and go on to others
    - Printers can jam
    - Lots of others

Printing in Windows

- Printing
  - Win32 API Printing is complex
  - In some ways like displaying on a screen form
  - But there are many unique printer issues:
    - Is printer on line?
    - Does printer have paper?
    - Is there color support?
    - How much graphics support is there?
    - Wide variety of printer types
    - Printer options
      - Trays, bins, paper sizes, etc.
      - Printers are slower than video displays
      - Programs reuse video display surface
      - Printer must eject completed pages and go on to others
    - Printers can jam
    - Lots of others


- System.Drawing.Printing namespace contains printing classes
- PrintDocument class is the key
  - Printer output set up with its methods, properties, and events
    - Its Print() method starts the printing output
      - Does not return until program is done printing the document
    - Usually invoked in response to user choosing a “Print” menu item or button
  - The Print() method fires a PrintPage event for each page to be printed
    - OnPrintPage event handler must contain code to do the printing
      - First “Object” parameter is the PrintDocument object that triggered the event
      - Second “PrintPageEventArgs” parameter provides data about the printer
        - Most important property is Graphics
        - Provides a Graphics object compatible with default printer
        - Use that Graphics object to display text/graphics on printer page
      - Also contains properties that allow determining page margins, e.g.
        - paper.MarginBounds.Left, also Top, Right, Bottoms or ppas.Graphics.VisibleClipBounds
        - a RectangleF that provides the size of the printable area
Print-Simple: A First Printing Example
(Mostly Manual)
• At top, code should include:
  using System.Drawing.Printing;
• Add “Click” and “Paint” event handlers to the form
  (easiest using VS Designer)
• Form’s “Paint” event handler displays a string that
  says to click the form to print some stuff
• Form’s “Click” event handler:
  – Whenever user clicks on the main form:
    • Instantiates a new PrintDocument object
    • Adds a PrintPage event handler (PrintDocument.OnPrintPage)
      using PrintPageEventHandler delegate
    • Then calls its Print() method to start the printing
• PrintPage handler gets the printer’s Graphics object
  and draws the stuff on the printer page

Printing using the VS Designer
• Drag a PrintDocument control from the toolbox to the
  form and select it
• Add a PrintPage event handler from its properties
  window (lightning bolt)
  – Produces a skeleton PrintPage handler
  – Type in code to specify what needs to be printed
• Print-Simple-Designer Example
  – Prints the same stuff as Print-Simple
  – Uses a “Print” menu item to start the printing

Print Preview Common Dialog Box
• Allows user to view printer’s output on the screen
• Derived from class PrintPreviewDialog
  – If using VS Designer, just drag a PrintPreviewDialog onto the form
  – Set its Document property to the PrintDocument to be printed/previewed
  – Then start the Print Preview dialog box with its ShowDialog() method
• Usually done in the event handler for a menu item or button
  – Same PrintPage event handler executes as for the PrintDocument
  – Several documents can be previewed with one PrintPreviewDialog box
  – Assign desired PrintDocument to PrintPreviewDialog’s Document property
• Print-Preview-Simple example program
  – Add a Print Preview menu item to Print-Simple-Designer example
  – Preview displayed when user clicks a menu item

Displaying Same Output on a Form’s Client Area
and a Printer Page: Subclassing
• Create a “PrintableForm” Class
• Put all code that outputs to either the window or to the printer
  in a separate method in that class
  – e.g., DoPage() method of that class
    • Parameters: the Graphics object (screen or printer), color, Rectangular bounds
• Call DoPage() from Paint handler and PrintPage handler
• Make DoPage() protected and virtual (overridable) so that
  other classes derived from PrintableForm can use it...
  – So that if you want to write a program that displays and prints a single screen of
    graphics, derive your form from PrintableForm instead of from Form
  – This is subclassing
  – Override its DoPage() method to draw what you want
  – Printing will be built into the program automatically
• PrintableForm Example Program

Printing and Previewing Contents
of a List Box
• Listbox-Simple-Print example program
  – Adds printing and print previewing to Listbox-Simple example program
  – User clicks on menu items to initiate actions

Using the PrintableForm class -- Printing a Sketch (Sketch-Print Example)
• Modify our Sketch-dotNet-Bitmap example program so the
  sketch can be printed in response to a ‘Print’ menu item
  – Copy the PrintableForm.cs file into the Sketch-dotNet-Bitmap directory and add
    it to the project (Project | Add Existing Item)
  – Change Namespace name so both .cs files are in same namespace
  – Derive the Form1 class from PrintableForm instead of from Form
    • i.e., change class declaration to: public class Form1 : PrintableForm;
    – Type in an override of the DoPage() method that does the same thing as the
      original Sketch-dotNet-Bitmap form’s Paint handler
        protected override void DoPage(Graphics g, Color clr, RectangleF vcb)
          { g.DrawImage(bmShadow , 0, 0, bmShadow.Width, bmShadow.Height); }
      – Be sure to specify that the Form1 class main() method is the entry point (or
        remove or comment out main() in the other class)
• Note how all of PrintableForm is inherited, including the
  menu, event handlers, and PrintDocument