

**Bio 595**

**Computers in**

**Biomedical Research**

**Class notes set 6**

**Fall 2003**

# Option Buttons

**Like the Mac's radio buttons: only one can be selected at a time (within a frame).**

**A “frame” is a rectangular area on a form where controls are placed.**

**The frame itself is a control upon which other controls can be grouped and placed. Frames then group controls for placement on the larger form.**

# Colors

**Eight common colors can be identified by name, although millions of different shades can be programmed.**

**The literals (with the color they code for represented by the name preceded by the prefix “vb” (designating a built-in visual Basic literal) are:**

**vbBlack, vbRed, vbGreen, vbYellow, vbBlue, vbMagenta, vbCyan, vbWhite**

# Check Boxes

**Like radio buttons, but not mutually exclusive  
(multiple selections can be checked)**

**Box can be grayed out if unavailable**

**Available style values include checked,  
unchecked, grayed, and unchecked and  
checked graphical.**

# ScrollBars

**Properties are set to control relative position within range of values. Properties include:**

**LargeChange** Amount scrollbar changes with click in scrollbar's shaft area

**Max** Maximum number of units at highest setting (default = 32767)

**Min** Minimum number of units at lowest setting (default = 1)

**SmallChange** Amount scrollbar changes with click at either end of scrollbar

**Value** Unit of measurement represented by scrollbar

# The VB Clock

**Timer control is shown on form at design time, but not at run time.**

**Set Interval property from 1 to 65535, time in milliseconds; generates a timer event.**

**tmrClock\_Timer() event**

**The timer is accurate to few tenths of a %.**

# Common Dialog Box

**These are not displayed on form until programmed to appear by code, and depending on what the code “needs” at the time.**

**The File Open dialog box is similar to that seen in applications like Excel or Word, allowing pathname selection, selecting another drive, etc.**

**There is also a File Save dialog box.**

# Common Dialog Box

**Using a common dialog box instead of File or Directory List boxes insures consistent appearance with commercial applications software, giving user immediate familiarity with what's expected.**

**Note however that these displays are the human interface only; you must still provide the code that senses input to the dialog box and takes appropriate action.**

# **Other Common Dialog Boxes**

**Color, with about fifty colors displayed, and access to custom colors**

**Font Dialog box gives user choice of what's installed on the computer in use**

**Print Dialog box, allows selection of a printer**

**Help Dialog box, which links to help files that you must generate**

# Common Dialog Box

**Adding the common dialog box control:**

- 1. select Project | Components to display Components dialog box**
- 2. scroll to Microsoft Common Dialog box, select**
- 3. control will appear**

**Double-click common dialog box to add control to form**

# Adding the Common Dialog Box

**Control on your form doesn't look as it will at runtime, because the properties you set determine this.**

**An additional Properties dialog box appears also: the tabs display choices (Open/Save As, Color, Font, Print, Help).**

**A Filter property allows selection of extension, so only selected types of files are displayed (like \*.asc, using wild card, for any ASCII files).**

# Common Dialog Box Methods

**Specify either ShowColor, ShowFont, ShowHelp, ShowOpen, ShowPrinter, or ShowSave**

**For a Common Dialog Box named cdbFile, the statement would be**

**cdbFile.ShowSave.**

# Common Dialog Box Methods

**Example: displaying a File Open dialog box:**

**cdbDialog.Title = "File Open"**

**cdbDialog.Filter = "\*.txt" 'show only text files**

**cdbDialog.FileName = "\*.txt" 'default**

**cdbDialog.ShowOpen 'now display dialog box**

# Menus

**These are control objects like buttons or text boxes**

**Menu bar properties can be set from the Properties window, like any object**

**But, the Menu Editor is easier to use than manual control through the Properties window To invoke the Menu Editor, select Tools/Menu Editor.**

**The editor creates a menu but you must write the event procedures which link menu selections (by a mouse click on the menu item) to particular actions.**

**The editor lets you add a menu bar, pull-down menu commands, separator bars (grouping menu options), submenus and checked items to your application.**

# Menu Bars

**To add a standard menu bar to a new application:**

**Open the menu editor.**

**Each menu bar command must be given a Caption and a Name.**

**Add options using the tab key to move from Caption to Name property:**

**File (caption = &File, name = mnuFile)**

**Edit (&Edit, mnuEdit), View (&View, mnuView)**

**Help (&Help, mnuHelp).**

# Menu Bar

**The ampersand prefixes assign accelerator keystrokes for the commands.**

**The Menu Editor's lower section displays the menu and options as they are added.**

**Run program after the menu bar is created (but nothing will actually happen without further code if you try to click on a menu command).**

# **Add Menu Options**

**Follow standard naming conventions by using the 'mnu' prefix, then the name of the menu command, followed by the pull-down menu item.**

**Return to the Menu Editor and insert a pull-down command to appear under File.**

**Select the &Edit line in the lower portion of the Editor box, then click on 'insert'; a blank line will appear.**

**Use the right arrow button to indent (four dots - an ellipsis - appears when this is done).**

**The left arrow removes a level of indentation, and the up-down arrows allow scrolling to different menu items.**

# Add the Menu Options

Enter the caption **&New**, then tab to enter the name **mnuFileNew**.

Note the naming convention: the prefix **mnu**, and the Menu command under which this pulldown command is to appear.

Repeat inserts to add **Open (&Open, mnuFileOpen)**, **Close (&Close, mnuFileClose)** and **Exit (&Exit, mnuFileExit)** to the menu.

Finally, add a separator bar above the **Exit** command under **File**.

Insert and indent, using a hyphen (-) as the caption property and **mnuFileBar1** as the name.

# More Menu Options

**Add a checked object.**

**Add an indented option (caption = Highlighted, name = mnuViewHighlighted) under View.**

**Add a submenu off File/Open to give two choices: Binary (&Binary, mnuFileOpenBinary) and Text (&Text, mnuFileOpenText).**

# **Link Menu Selections to Event Procedures**

**Remember: menu objects must be linked to code which will respond to the button click.**

**In design time with the Forms window displayed, double click on the pull-down Exit selection under the File menu to open a code window showing the 'wrapper' lines Private Sub mnuFileExit\_Click() and End Sub.**

**Enter Unload Me as one line, then the keyword End.**

**Run this program to verify that selection of Exit from the File pull-down menu will terminate program execution.**

# Replace() Function

Find and replace or remove the found substring within a string

**Syntax:** `strResult=Replace(strA,strFind,strRplWith,  
intStart,intCnt,intCompare)` where

**strResult** is the string returned by the function

**strA** is the string to be changed

**strFind** is the string to find in strA

**strRplWith** is the string to use to replace strFind

`strResult=Replace(strA,strFind,strRplWith,  
intStart,intCnt,intCompare)`

**Optional arguments:**

**intStart:** position within strA at which to begin looking for strFind

**intCnt:** maximum number of times to replace strFind in strA

**intCompare:** case sensitivity: 0 means case-sensitive, 1 means case-insensitive

If strRplWith is zero length, Replace() returns with all occurrences of strFind removed.

# Reverse() Function

Completely reverse the order of a string

Syntax:

```
strResult=StrReverse(strA)
```

Where

**strResult** is the returned string

**strA** is the source string to be reversed

# Split() Function

Takes a list of words or numeric values stored in a string and splits them into individual elements of an array of strings.

Requires knowing the delimiter (a space or a comma)

Syntax:

```
varResult=Split(strList,strDelimiter,intElemCnt, intCompare)
```

```
varResult=Split(strList,strDelimiter,intElemCnt, intCompare)
```

where

**varResult** is variant variable containing a one-dimensional array of strings

**strList** is the string list to be parsed **strDelimiter** is character (or string of characters) used to separate elements

**intElemCnt** is the number of string array elements to create from the **strList**

**intCompare** indicates case sensitivity: 0= case-sensitive, 1= case-insensitive

# Filter() Function

After Filter() has created an array of strings, creates a new list, a filtered version of the original list. Syntax:

**varResult=Filter(varList,strFind,bolInclude,intCompare)**

**varResult** contains the 1-dimensional array returned from the function;

**varList** is the variant variable containing the array to be filtered;

**strFind** is character string used to identify elements to include in the new array;

**bolInclude** is a Boolean flag: are elements containing **strFind** are to be included (default) or excluded?

**intCompare** indicates case sensitivity (0=yes, 1=no)

# Join()

**Counterpart of Split() function – combines elements of an array into a single string.**

**Syntax:**

**strResult=Join(varList,strLimiter) where strResult is string containing concatenated array**

**varList contains one-dimensional string array to be combined**

**strDelimiter is character or string of characters to be placed between array elements (default is a space); if a zero-length string, no delimiter**