L e c t u r e

21-22

Review of Last Lecture

- Modal dialogs
- DialogBox() and EndDialog()
- DLGPROC: The dialog procedure
- Modeless dialogs
- CreateDialog() and DestroyWindow()
- Child control notification messages
- Manipulating child controls on a dialog by sending messages
- Common dialogs and common controls

Today's Lecture Goals

- Common dialog: Choose colour
- Tab order, tab stop, groups
- Modeless and modal dialog
- Radio Button, push button, edit etc.
- Listbox control
- 2-D Arrays of strings
- Communication between and manipulation of all these

Windows Common Dialogs

- Choose colour: ChooseColor(&CHOOSCOLOR)
- Find: FindText(&FINDREPLACE)
- Choose font:ChooseFont(&CHOOSEFONT)
- Open File:GetOpenFilename(&OPENFILENAM)
- Page setup
- Print
- Replace
- Save As: GetSaveFilename(&OPENFILENAM)

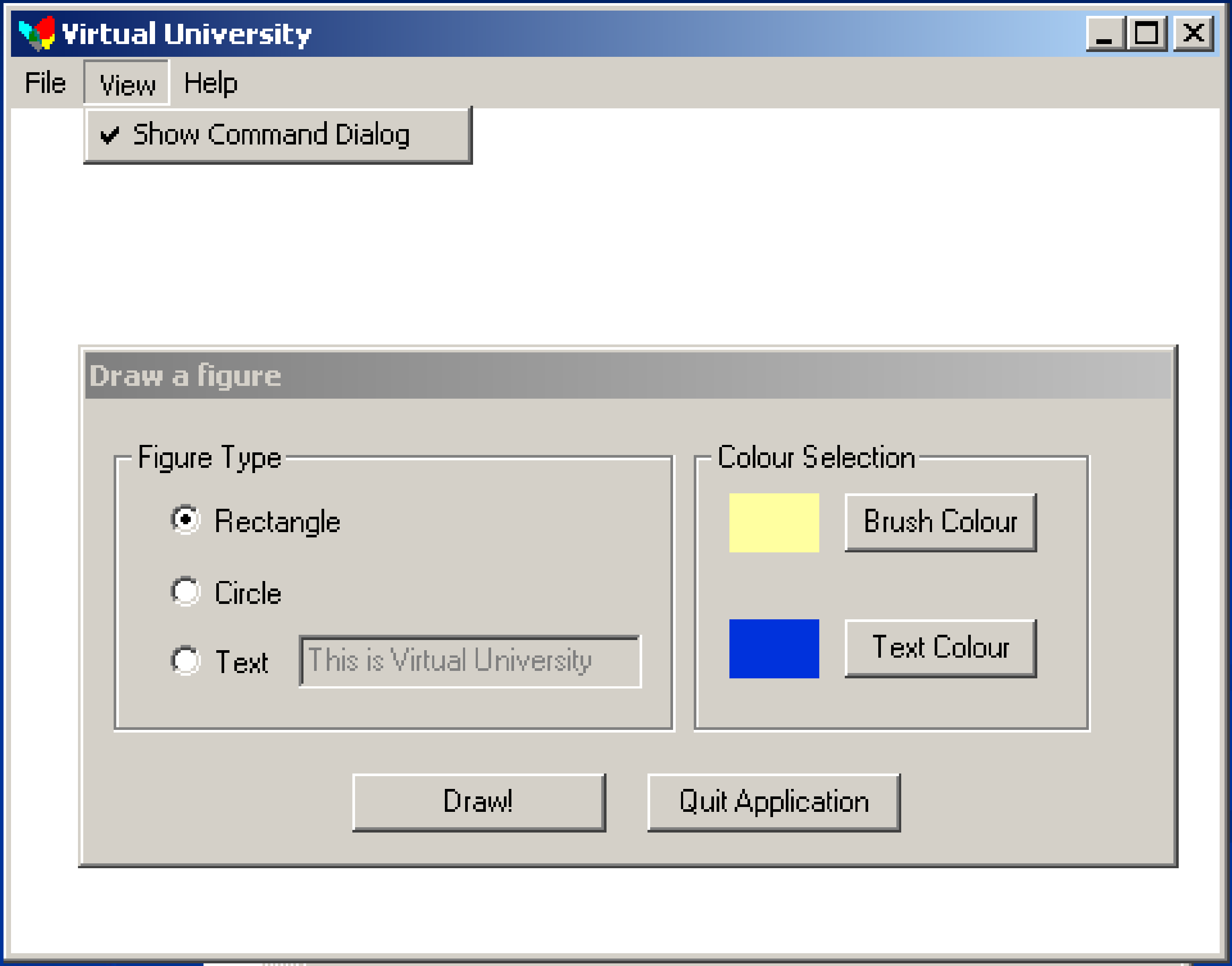
DLU

- **dialog unit (DLU)**
  - A unit of horizontal or vertical distance within a dialog box. A horizontal DLU is the average width of the current dialog box font divided by 4. A vertical DLU is the average height of the current dialog-box font divided by 8.

Tab stops, tab order, Groups

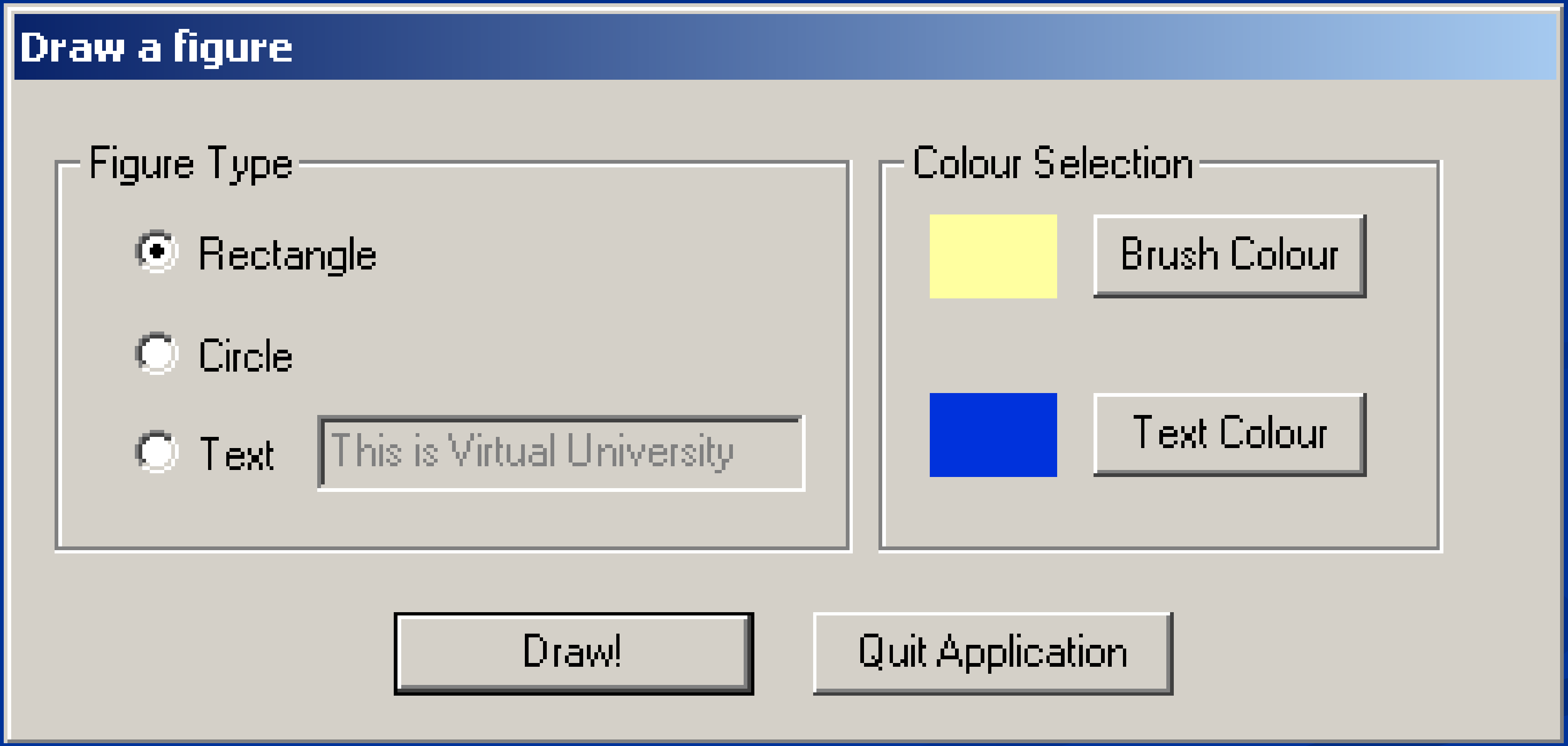
- **WS_TABSTOP** will cause focus to move to that control also when Tab is pressed
- Groups of controls
  WS_GROUP Specifies the first control of a group of controls in which the user can move from one control to the next with the arrow keys. All controls defined with the **WS_GROUP** style **FALSE** after the first control belong to the same group. The next control with the **WS_GROUP** style starts the next group (that is, one group ends where the next begins)
- Focus: SetFocus, GetFocus

Application Description



[✓ Rectangle] Circle Text: This is Virtual University

"Draw a figure" dialog



Draw a figure

Figure Type

[✓ Rectangle]

Circle

Text: This is Virtual University

Colour Selection

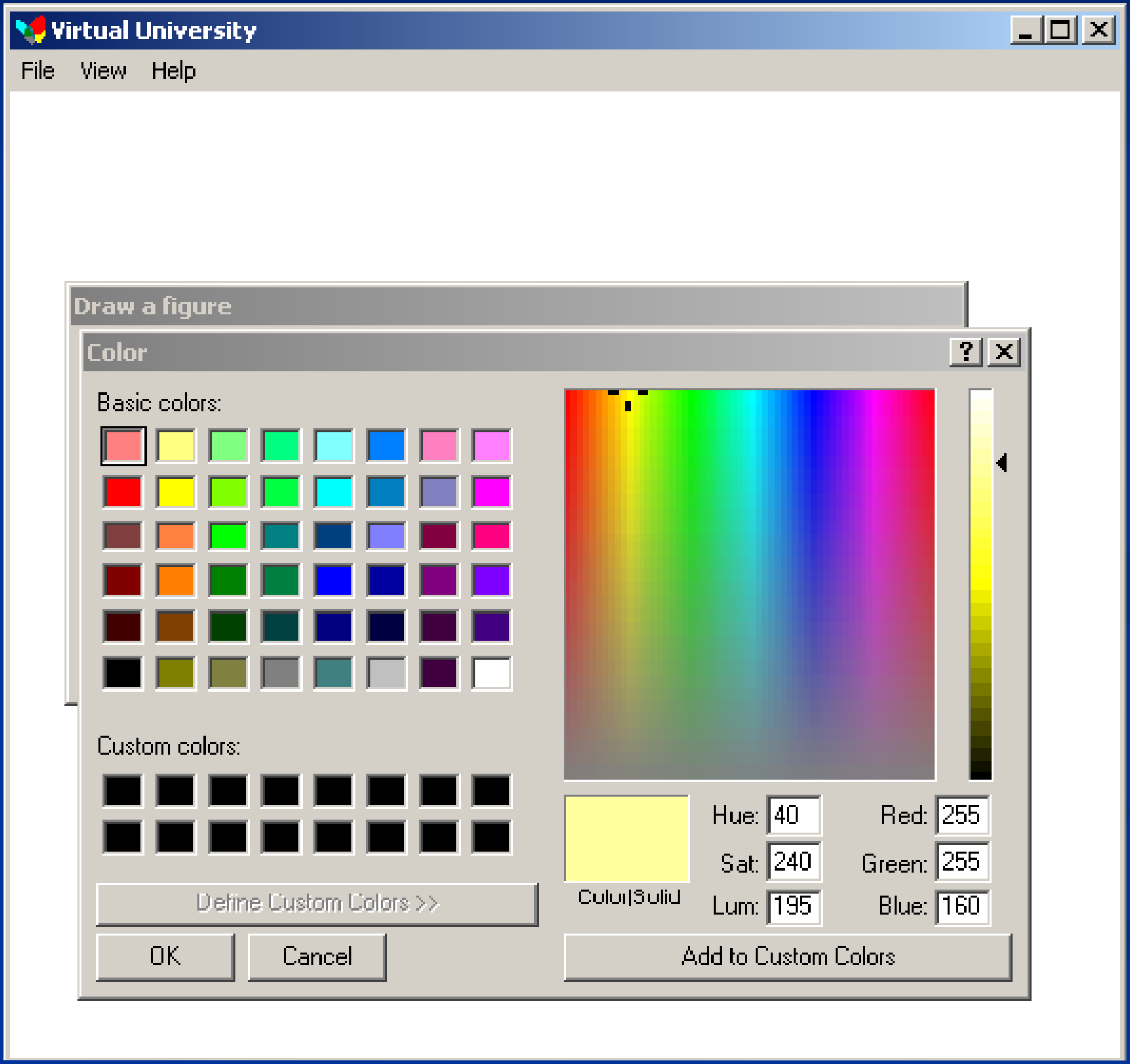
Brush Colour

Text Colour

Draw!

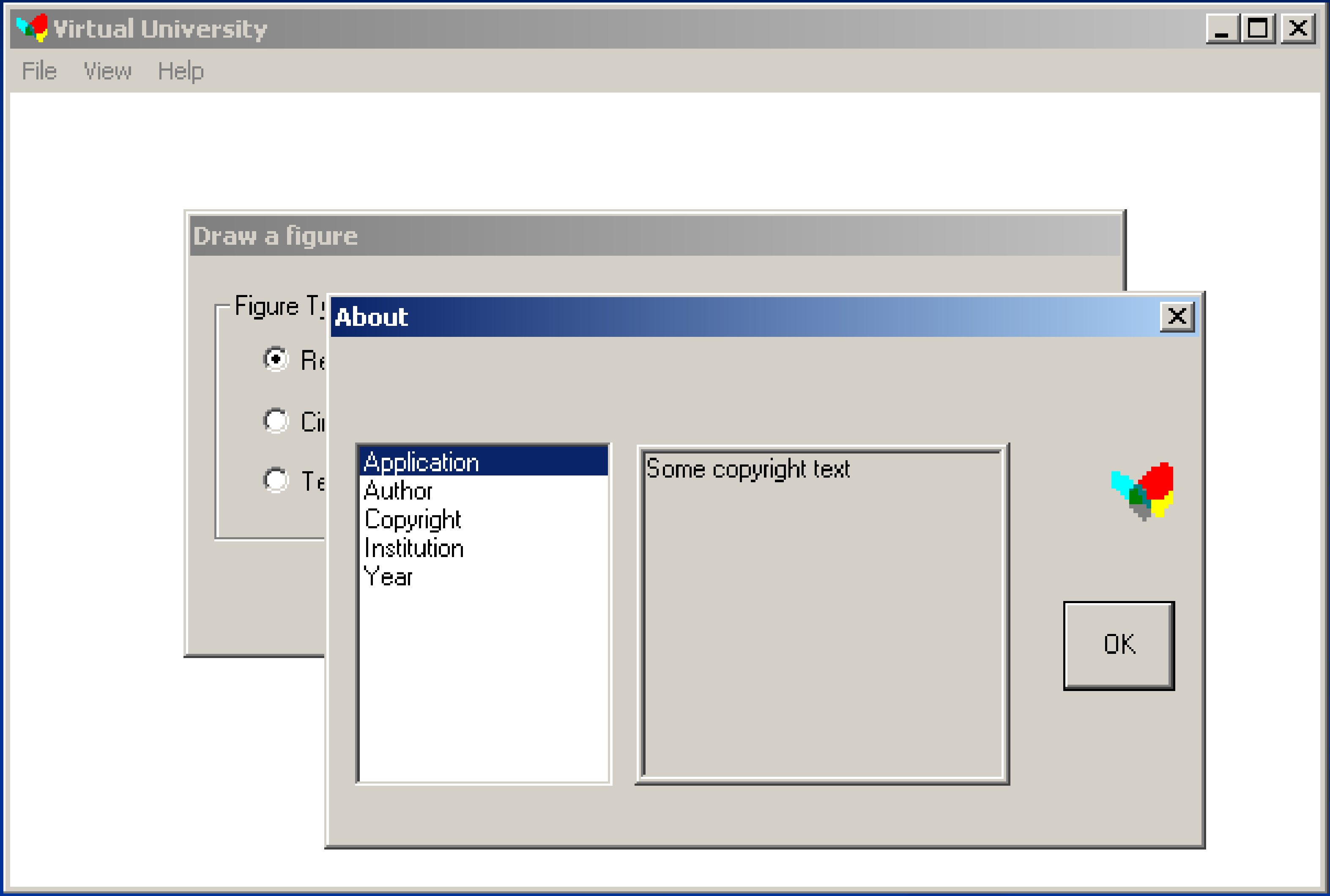
Quit Application

Choose brush colour



Hue: 40
Sat: 240
Lum: 195
Red: 255
Green: 255
Blue: 160

The “About…” dialog



Some copyright text

Resource statements

- Resource editor of Visual Studio 6.0 is used to create all these resource script files.
- It is quite difficult to create so many complex resource scripts manually without the help of a WYSIWYG editor

Resource statements

- General control definition in resource file

```
CONTROL text, id, class, style, x, y, width,
   height [, extended-style]
```

- Example control definition

-
  ```
  CONTROL              "Caption",
  IDC_RADIO_RECTANGLE, "Button",
  BS_AUTORADIOBUTTON  |  WS_GROUP  |
  WS_TABSTOP,21,24,69,10
  ```

- There are 6 control classes:
  Button, combo, edit, list box, scroll bar, static

draw dialog (Modeless)

```
CONTROL          "Rectangle",IDC_RADIO_RECTANGLE,"Button",
                      BS_AUTORADIOBUTTON | WS_GROUP |
  WS_TABSTOP,21,24,69,10


    CONTROL
  "Circle",IDC_RADIO_CIRCLE,"Button",BS_AUTORADIOBUTTON,21
  ,41,69,10


    CONTROL
  "Text",IDC_RADIO_TEXT,"Button",BS_AUTORADIOBUTTON,21,58,
  30,10


    DEFPUSHBUTTON
  "&Draw!",IDC_BUTTON_DRAW,69,89,66,14,WS_GROUP
    PUSHBUTTON        "&Quit
  Application",ID_QUIT_APP,146,89,66,14
    GROUPBOX          "Figure Type",IDC_STATIC,7,9,146,70
```

draw dialog contd.

```
PUSHBUTTON      "Brush Colour",IDC_BUTTON_BRUSH_COLOR,197,22,50,14,WS_GROUP

PUSHBUTTON      "Text Colour",IDC_BUTTON_TEXT_COLOR,197,52,50,14

GROUPBOX        "Colour Selection",IDC_STATIC,158,9,103,70,WS_GROUP
EDITTEXT        IDC_EDIT_TEXT,55,56,89,13,ES_AUTOHSCROLL
    LTEXT       "",IDC_STATIC_BRUSH_COLOR,167,22,23,14
    LTEXT       "",IDC_STATIC_TEXT_COLOR,167,52,23,14
```

About Dialog resource definition

```
IDD_DIALOG_ABOUT DIALOGEX 0, 0, 263, 141
STYLE DS_MODALFRAME | DS_CENTER | WS_POPUP | WS_CAPTION |
   WS_SYSMENU
CAPTION "About"
FONT 8, "MS Sans Serif"
BEGIN
    DEFPUSHBUTTON   "OK",IDOK,222,73,34,25
    LTEXT           "Some copyright
   text",IDC_STATIC_ABOUT,92,29,114,96,0,
                    WS_EX_DLGMODALFRAME | WS_EX_CLIENTEDGE
    ICON            IDI_ICON_VU,IDC_STATIC,235,33,20,20
    LISTBOX         IDC_LIST_ABOUT,7,29,78,96,LBS_SORT |
                    LBS_NOINTEGRALHEIGHT | WS_VSCROLL |
   WS_TABSTOP
END
```

The Application before message loop

```
hWndMain = CreateWindow(windowClassName, windowName,
WS_OVERLAPPEDWINDOW | WS_VISIBLE,
    CW_USEDEFAULT, 1, CW_USEDEFAULT, 1, NULL, NULL,
hInstance, NULL);

if(!hWndMain)     return 1;

hCommandDialog = CreateDialog(hInstance,
MAKEINTRESOURCE(IDD_DIALOG_DRAW), hWndMain,
commandDialogProc);

if(!hCommandDialog)     return 1;

ShowWindow(hCommandDialog, SW_SHOW);

commandDialogShown = TRUE;
CheckMenuItem(GetMenu(hWndMain),
ID_VIEW_SHOWCOMMANDDIALOG, MF_CHECKED | MF_BYCOMMAND);
```

Message Loop

```
while(GetMessage(&msg, NULL, 0, 0) > 0)
{
    if(!IsDialogMessage(hCommandDialog, &msg))
    {
            TranslateMessage(&msg);
            DispatchMessage(&msg);
    }
}
```

Menu cmds

```
case ID_VIEW_SHOWCOMMANDDIALOG:
if(commandDialogShown)      // already visible?
{
ShowWindow(hCommandDialog, SW_HIDE);     // hide it

CheckMenuItem(GetMenu(hWnd), ID_VIEW_SHOWCOMMANDDIALOG,
   MF_UNCHECKED | MF_BYCOMMAND); // uncheck
commandDialogShown = FALSE;
}
Else
{
}
```

commandDialogProc

```
static COLORREF textColour, brushColour;

    case WM_INITDIALOG:
        CheckDlgButton(hDlg, IDC_RADIO_RECTANGLE, BST_CHECKED);
        // BM_SETCHECK message: check rectangle button

        EnableWindow(GetDlgItem(hDlg, IDC_EDIT_TEXT), FALSE);
        // disable edit control

        SendDlgItemMessage(hDlg, IDC_EDIT_TEXT, EM_LIMITTEXT,
    TEXT_LIMIT, 0);     // set text limit

        SetWindowText(GetDlgItem(hDlg, IDC_EDIT_TEXT), "This is
    Virtual University");

        brushColour = RGB_BRUSH_COLOR;  //RGB(255, 255, 160)
        textColour = RGB_TEXT_COLOR;    //RGB(0, 50, 220)
        return TRUE;    // system should set focus
```

Messages to controls

- **`BM_SETCHECK:`**
  wParam: check-state (BST_CHECKED or BST_UNCHECKED)
  CheckDlgButton() sends this message
- **`EM_LIMITTEXT:`**
  wParam: text length
- **`EM_SETSEL:`**
  wParam: starting position
  lParam: ending pos.

  If starting position is 0 and ending position is -1, all the text in the edit control is selected
  If starting position is -1, current selection is deselected

commandDialogProc contd.

```
wNotificationCode = HIWORD(wParam);
wID = LOWORD(wParam);
if(wNotificationCode == BN_CLICKED)
{
        switch(wID)
        {
        case IDC_RADIO_RECTANGLE:
    EnableWindow(GetDlgItem(hDlg, IDC_EDIT_TEXT), FALSE);
        // disable edit control similarly in
    IDC_RADIO_CIRCLE

case IDC_RADIO_TEXT:
EnableWindow(GetDlgItem(hDlg, IDC_EDIT_TEXT), TRUE);  //
    enable edit control
SendDlgItemMessage(hDlg, IDC_EDIT_TEXT, EM_SETSEL, 0, -1);
        // set text limit
SetFocus(GetDlgItem(hDlg, IDC_EDIT_TEXT));
```

WM_CTLCOLORSTATIC

- The trick to show colour
- wParam: hdc (Handle to Device Context)
  lParam: handle to control
- If a dialog box procedure handles this message, it should cast the desired return value to a **BOOL** and return the value directly.
- If the dialog box procedure returns FALSE, then default message handling is performed

commandDialogProc contd.

```
case WM_CTLCOLORSTATIC:
        switch(GetDlgCtrlID((HWND)lParam))
        {
        case IDC_STATIC_TEXT_COLOR:
           if(hBrush)      // if some brush was created before
                   DeleteObject(hBrush);
        hBrush = CreateSolidBrush(textColour);  // create a brush
                   return (BOOL)hBrush;
                   break;

        case IDC_STATIC_BRUSH_COLOR:
           if(hBrush)      // if some brush was created before
                   DeleteObject(hBrush);
           hBrush = CreateSolidBrush(brushColour); // create a brush
                   return (BOOL)hBrush;
                   break;

           default:
                   return FALSE;  // perform default message handling
```

Choosecolor(&choosecolr)

```
typedef struct {
   DWORD          lStructSize;
   HWND           hwndOwner;
   HWND           hInstance;
   COLORREF       rgbResult;
   COLORREF     * lpCustColors;
   DWORD          Flags; CC_RGBINIT | CC_FULLOPEN
    | CC_ANYCOLOR
   LPARAM         lCustData;
   LPCCHOOKPROC lpfnHook;
   LPCTSTR        lpTemplateName;
} CHOOSECOLOR, *LPCHOOSECOLOR;
```

commandDialogProc contd.

```
case IDC_BUTTON_BRUSH_COLOR:
    if(ShowChooseColorDialog(hDlg, brushColour,
        &brushColour))
    {
        GetClientRect(GetDlgItem(hDlg,
        IDC_STATIC_BRUSH_COLOR), &rect);

        InvalidateRect(GetDlgItem(hDlg,
        IDC_STATIC_BRUSH_COLOR), &rect, TRUE);
    }
break;
```

```
BOOL ShowChooseColorDialog(HWND Owner,
COLORREF initClr, LPCOLORREF chosenClr)

CHOOSECOLOR cc;
static COLORREF customColors[16];

memset(&cc, 0, sizeof(cc));

cc.lStructSize = sizeof(CHOOSECOLOR);
cc.hwndOwner = hwndOwner;
cc.rgbResult = initialColor;
cc.lpCustColors = customColors;
cc.Flags = CC_RGBINIT | CC_FULLOPEN | CC_ANYCOLOR;

if(ChooseColor(&cc))   // OK pressed in the dialog
{
   *chosenColor = cc.rgbResult;
   return TRUE;
}
return FALSE;
```

commandDialogProc contd.

```
case IDC_BUTTON_BRUSH_COLOR:
    if(ShowChooseColorDialog(hDlg, brushColour,
        &brushColour))
    {
    // REPAINT CONTROL:  send WM_CTLCOLORSTATIC
        during repainting

        GetClientRect(GetDlgItem(hDlg,
        IDC_STATIC_BRUSH_COLOR), &rect);

        InvalidateRect(GetDlgItem(hDlg,
        IDC_STATIC_BRUSH_COLOR), &rect, TRUE);
    }
break;
```

commandDialogProc (drawing)

```
case IDC_BUTTON_DRAW:
    hDC = GetDC(GetParent(hDlg));
    if(IsDlgButtonChecked(hDlg,
    IDC_RADIO_RECTANGLE) == BST_CHECKED)
    {
    hOwnerBrush = CreateHatchBrush(HS_BDIAGONAL,
    brushColour);

    hOldBrush = SelectObject(hDC, hOwnerBrush);
    Rectangle(hDC, 10, 10, 200, 200);

    SelectObject(hDC, hOldBrush); // restore old
    selection
    DeleteObject(hOwnerBrush);
}
```

The about box (main window proc)

```
case ID_HELP_ABOUT:
    DialogBox(hAppInstance,
              MAKEINTRESOURCE(IDD_DIALOG_ABOUT),
              hWnd,
              aboutDialogProc);
```

ListBox messages

- Relationship between Strings, String Index and Item Data
- **LB_ADDSTRING:**
  lParam: string to add (LPCTSTR)
  Returns zero-based index of the string in the list box
- **LB_SETITEMDATA:**
  wParam: index where to set,
  lParam: value to associate.
  LB_GETITEMDATA is used to retrieve item data
- **LB_SETCURSEL:**
  wParam: item index where to set the current selection
- **LBN_SELCHANGE** notification is received in WM_COMMAND message whenever a selection changes in a ListBox

The about box (aboutDialogProc)

```
static LPTSTR strings[5][2] = {{"Application", "Lecture
     21"},
{"Author", "Sajid Ali Khan Sajidi"},
{"Institution", "Virtual University"},
{"Year", "2002"},
{"Copyright", "2002 Virtual University"}};
```

2-D array of strings

```
case WM_INITDIALOG:
for(i=0; i<5; ++i){
index = SendDlgItemMessage(hDlg,IDC_LIST_ABOUT,
     LB_ADDSTRING, 0, (LPARAM)strings[i][0]);
SendDlgItemMessage(hDlg, IDC_LIST_ABOUT,
     LB_SETITEMDATA,index,(LPARAM)strings[i][1])}
// set current selection to 0
SendDlgItemMessage(hDlg, IDC_LIST_ABOUT,
     LB_SETCURSEL, 0, 0);
```

The about box (aboutDialogProc)

```
LPTSTR str;

case WM_COMMAND:
wNotificationCode = HIWORD(wParam);
wID = LOWORD(wParam);

switch(wID)
{
case IDC_LIST_ABOUT:
    if(wNotificationCode == LBN_SELCHANGE)
    {
index = SendDlgItemMessage(hDlg, wID, LB_GETCURSEL,
    0, 0);
str = (LPTSTR)SendDlgItemMessage(hDlg,
    IDC_LIST_ABOUT, LB_GETITEMDATA, index, 0);
SetDlgItemText(hDlg, IDC_STATIC_ABOUT, str);
```

Something to Do! ☺ → ☹

- Drawing gets erased in this application. Try to perform drawing in WM_PAINT in main window

- Use sub-classing, CHOOSECOLOR() and CHOOSEFONT() to change main window’s background brush and display font at runtime