Windows Programming

Lecture 15

Z-order

- The Z order of a window indicates the window's position in a stack of overlapping windows. This window stack is oriented along an imaginary axis, the z-axis, extending outward from the screen.
- The window at the top of the Z order overlaps all other windows.
- The window at the bottom of the Z order is overlapped by all other windows.

CreateWindow()

```
HWND CreateWindow
(
   LPCTSTR lpClassName,      // registered class name
   LPCTSTR lpWindowName,     // window name
   DWORD dwStyle,            // window style
   int x,                    // horizontal position of window
   int y,                    // vertical position of window
   int nWidth,               // window width
   int nHeight,              // window height
   HWND hWndParent,          // handle to parent or owner window
   HMENU hMenu,              // menu handle or child identifier
   HINSTANCE hInstance,      // handle to application instance
   LPVOID lpParam            // window-creation data
);
```

Child Windows

- A child window always appears within the client area of its parent window.
- Child windows are most often as controls.
- A child window sends **`WM_COMMAND`** notification messages to its parent window.
- When a child window is created a unique identifier for that window is specified in **`hMenu`** parameter of **`CreateWindow()`**

Prototype of window procedure

```
LRESULT CALLBACK WindowProc
(
   HWND hwnd,              // handle to window
   UINT uMsg,              // WM_COMMAND
   WPARAM wParam,          // notification code and identifier
   LPARAM lParam           // handle to control (HWND)
);
```

WM_COMMAND

- This message is sent when the user selects a command item from a menu, when a control sends a message to its parent window, or when an accelerator keystroke is translated.

Notification Codes

- Common controls are child windows that send notification messages to the parent window when events, such as input from the user, occur in the control. The application relies on these notification messages to determine what action the user wants it to take. Except for trackbars, which use the WM_HSCROLL and WM_VSCROLL messages to notify their parent of changes, common controls send notification messages as WM_NOTIFY messages.

WM_COMMAND Notifications

The **`wParam`** parameter of Window Procedure contains the notification code and control identifier.

low word: ID of the control

high word: notification code

`BUTTON` → **`BN_CLICKED`**

`EDIT` → **`EN_CHANGE etc`**

Child windows application

Three push buttons

"RECTANGLE"

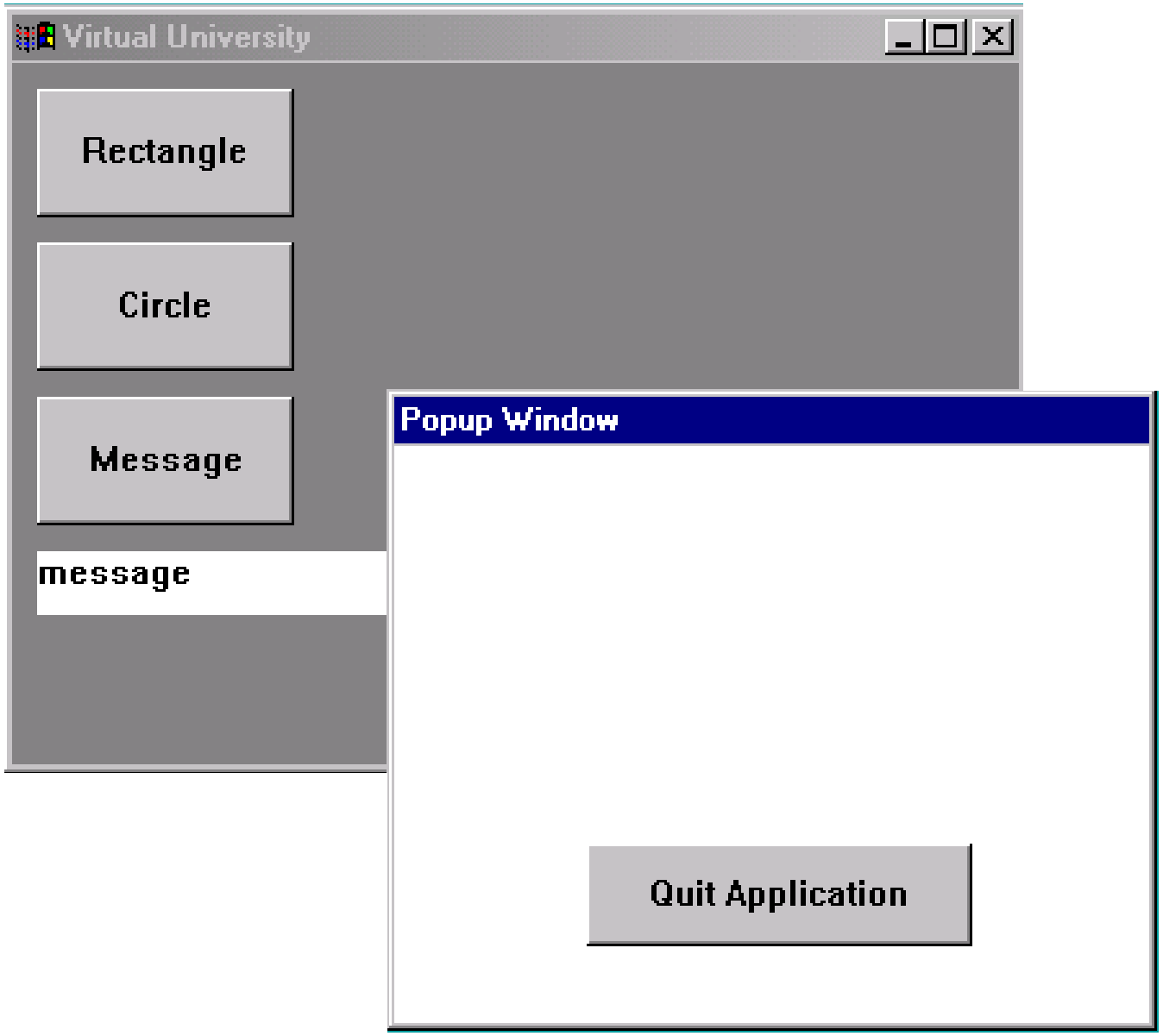
"CIRCLE"

"MESSAGE"

One edit child window or edit control in its client area

One floating popup window with caption with 1 push button
"QUIT APPLICATION"

Child windows application



Objectives of Child windows application

- Parent-child communication
- Message Routing
- Usage of GDI function calls

Window management functions-I

The GetParent function retrieves a handle to the specified window's parent or owner.

```
HWND GetParent
(
    HWND hWnd      // handle to child window
);
```

Window management functions-II

```
HWND GetDlgItem
(
  HWND hDlg,              // handle to dialog box
  int nIDDlgItem          // control identifier
);
HWND FindWindow
(
  LPCTSTR lpClassName,    // class name
  LPCTSTR lpWindowName    // window name
);
```

Application analysis

Window classes in the child windows application:

- mainWindowClass
- popupWindowClass
- System Window Classes

MianWindowClass
Important Parameters

```
wc.lpfnWndProc   = mainWindowProc;

wc.hInstance     = hAppInstance = hInstance;

wc.hCursor       = LoadCursor(NULL,
                              IDC_UPARROW);

wc.hbrBackground= (HBRUSH)GetStockObject
                  (GRAY_BRUSH);

wc.lpszClassName= "MainWindowClass";
```

PopupWindowClass

```
wc.lpfnWndProc   = popupWindowProc;
wc.hbrBackground = (HBRUSH)(COLOR_WINDOW+1);
wc.hCursor       = LoadCursor(NULL,IDC_HELP);
wc.lpszClassName = "PopupWindowClass";
```

Creating windows

```
hWndMain = CreateWindow("MainWindowClass",
                        "Virtual University",
           WS_OVERLAPPEDWINDOW | WS_VISIBLE,
                        100, 100, 400, 300,
                    NULL, NULL, hInstance, NULL);


hWndPopup = CreateWindow("PopupWindowClass",
                                "Popup Window",
           WS_POPUP | WS_CAPTION | WS_VISIBLE,
                                250, 250, 300, 250,
            hWndMain, NULL, hInstance, NULL);
```

Creating child windows

```
CreateWindow("BUTTON", "Rectangle",
             WS_CHILD | WS_VISIBLE,
             10, 10, 100, 50,
             hWndMain, 5,
             hInstance, NULL);

CreateWindow("EDIT", "Message",
             WS_CHILD | WS_VISIBLE
             | ES_LOWERCASE,
             10, 190, 200, 25,
             hWndMain, 8,
             hInstance, NULL);
```

Main window's **WndProc**

```
case WM_COMMAND:
      wControlID = LOWORD(wParam);
      wNotificationCode = HIWORD(wParam);
      if(wNotificationCode == BN_CLICKED)
      {
            switch(wControlID)
            {
            case 5:
               SendMessage(hWndPopup, ?,
                           wparam, ???);
            break;
            ... ... ...
```

User defined messages

WINUSER.H contains Window messages

```
#define WM_LBUTTONDOWN      0x0201 (513)
#define WM_DESTROY          0x0002 (2)
#define WM_QUIT             0x0012 (18)
#define WM_USER             0x0400 (1024)

#define WM_DRAW_FIGURE      WM_USER+786
```

User defined messages

```
case WM_COMMAND:
... ... ...
switch(wControlID)
{
   case 5:
      SendMessage(hWndPopup, WM_DRAW_FIGURE,
                  RECTANGLE, 0);    break;
   case 6:
      SendMessage(hWndPopup, WM_DRAW_FIGURE,
                  CIRCLE, 0);  break;
   case 7:
      SendMessage(hWndPopup, WM_DRAW_FIGURE,
                  TEXT_MESSAGE, 0); break;
}
```

Using Enumerations

Objects of type **enum** are **int** types, and their size is system-dependent. By default, objects of **enum** types are treated as 16-bit objects of type **unsigned short** when transmitted over a network.

The keyword **enum** to identifies an enumerated type.

```
enum FigureType
{
   RECTANGLE, CIRCLE, TEXT_MESSAGE
};
```

Drawing in Popup window-I

```
case WM_DRAW_FIGURE:
   hDC = GetDC(hWndPopup);
   switch(wParam)
   {
       case RECTANGLE:
            SelectObject(hDC,GetStockObject(
                         LTGRAY_BRUSH));
            Rectangle(hDC, 50, 10, 230, 150);
            break;
   }
```

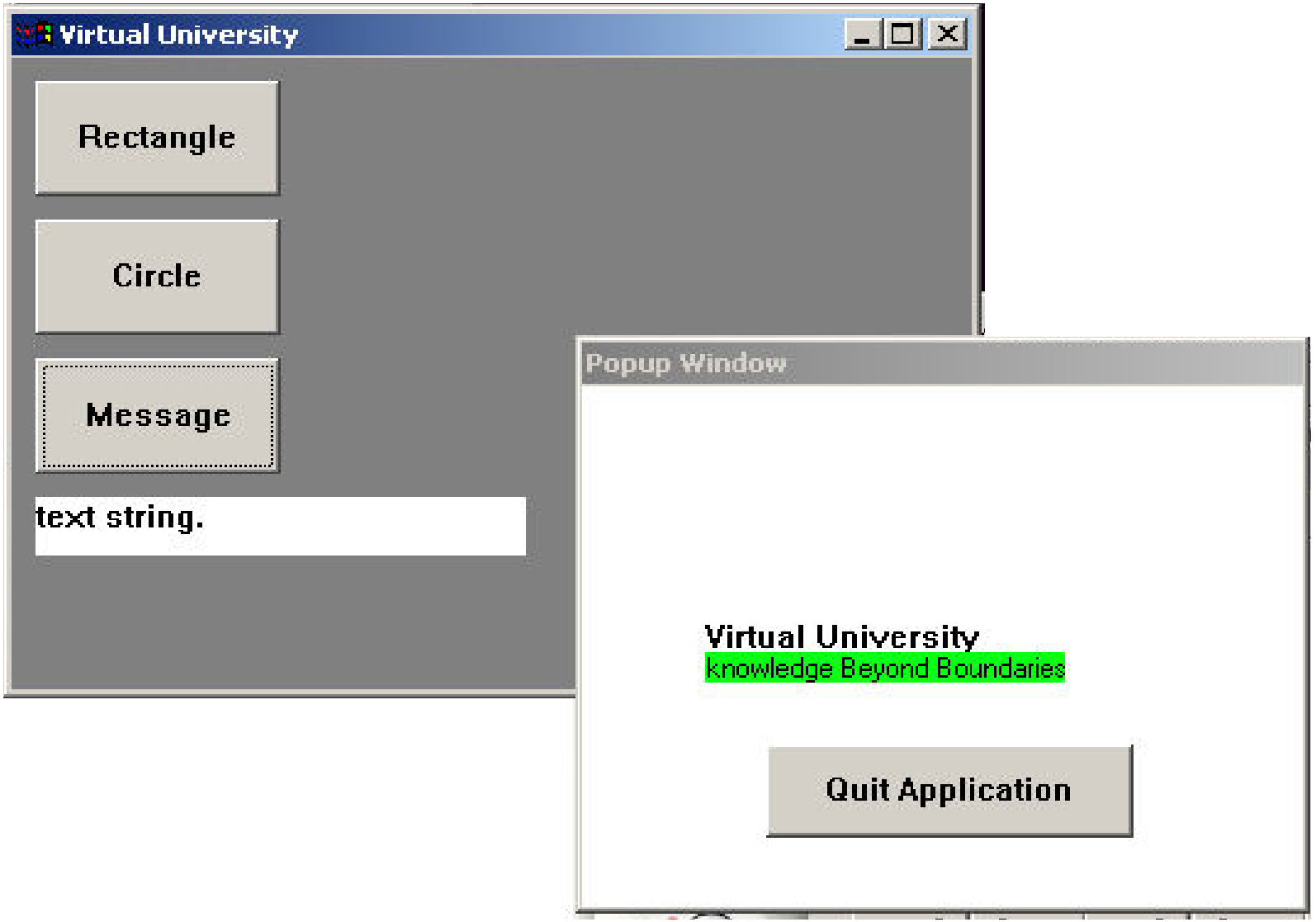
Drawing in Popup window-II

```
case CIRCLE:
 hBrush = CreateHatchBrush(HS_DIAGCROSS,
                           RGB(170, 150, 180));
 SelectObject(hDC, hBrush);
 Ellipse(hDC, 70, 10, 210, 150);
 DeleteObject(hBrush);
break;
```

Drawing in Popup window-III

```
{... ... ...
case TEXT_MESSAGE:
  TextOut(hDC, 50, 100, "Virtual
     University", 18);
  SelectObject(hDC, GetStockObject(
                  ANSI_VAR_FONT));
  SetBkColor(hDC, RGB(10, 255, 20));
  TextOut(hDC, 50, 115, "knowledge Beyond
           Boundaries", 27);
break;
}
ReleaseDC(hWndPopup, hDC); break;
```

Application Output



Informing back to main window

```
case WM_DRAW_FIGURE:
   hDC = GetDC(hWndPopup);
   hwndEdit = GetDlgItem(GetParent(hWnd), 8);
   switch(wParam)
   {
       case RECTANGLE:
       SelectObject(hDC,GetStockObject(LTGRAY_BRUSH));
       Rectangle(hDC, 50, 10, 230, 150);
       SendMessage(hwndEdit, WM_SETTEXT, 0, "Rectangle
                   DrAwN!");
       break;
   }
```

Quit application via a control in the popup window

```
WM_CREATE:
      CreateWindow("BUTTON",
      "Quit Application",
      WS_CHILD | WS_VISIBLE,
      75, 155, 150, 40, hWnd, 1234,
     hAppInstance, NULL);
     break;
```

Quit app. via a ctrl in the popup window

```
case WM_COMMAND:
    wControlID = LOWORD(wParam);
    wNotificationCode = HIWORD(wParam);
if(wNotificationCode == BN_CLICKED)
{
    switch(wControlID)
    {
        case 1234:
        DestroyWindow(GetParent(hWnd));
        break;
    }
}
```

Erase it clean, then draw!

```
    case TEXT_MESSAGE:

SendMessage(hWndPopup,
WM_ERASEBKGND, (WPARAM)hDC, 0);

TextOut(hDC, 50, 100, "Virtual
University", 18);

SelectObject(hDC,
GetStockObject(ANSI_VAR_FONT));

SetBkColor(hDC, RGB(10, 255, 20));

TextOut(...)
```

..... other code .....