MFC Programming

- The Microsoft Foundation Class (MFC) Library--
  - A Hierarchy of C++ classes designed to facilitate Windows programming
  - An alternative to using Win32 API functions
  - A Visual C++ Windows app can use either Win32 API, MFC, or both

Some Characteristics of MFC

- Offers convenience of REUSABLE CODE
  - Many tasks common to all Windows apps are provided by MFC
  - Programs can inherit and modify this functionality as needed
  - We don’t need to recreate these tasks
  - MFC handles many clerical details in Windows programs
  - Functionality encapsulated in MFC Classes

Some important Classes

- **CObject**: "Mother of all classes"; Provides Serialization
  - CDC: Encapsulates the device context (Graphical Drawing)
  - CGdiObject: Base class for various drawing objects (bitmaps, pens, brushes, etc.)
  - CCmdTarget: Encapsulates message passing process & is parent of:

- **CWnd**: Base class all windows derived from:
  - CFrameWindow: Can contain other windows
  - CView: Encapsulates process of displaying data in a window
  - CDialog: Encapsulates dialog boxes
- **CWinThread**: Defines a thread of execution & is parent of:
  - CWinApp: Encapsulates an MFC application; Controls Startup, initialization, execution (msg loop), shutdown; When instantiated application runs
  - CDocument: Encapsulates the data associated with a program
  - All MFC apps must have a CWnd-derived class and a CWinApp-derived class
Primary task in writing an MFC program
- To create/modify classes
- Most will be derived from MFC library classes

MFC Class Member Functions
- Most functions called by an app will be members of an MFC class
- Examples:
  - `ShowWindow()`—a member of CWnd class
  - `TextOut()`—a member of CDC
  - `LoadBitmap()`—a member of CBitmap
- Applications can also call API functions directly
  - Use “global scope resolution” operator ::
    - Example: `::UpdateWindow(hWnd);`
  - More convenient to use MFC member functions

MFC Global Functions
- Not members of any MFC class
- Usually begin with Afx prefix (Application Frameworks)
- Independent of or span MFC class hierarchy
- Example:
  - `AfxMessageBox()`—
    - Message boxes are predefined windows
    - Can be activated independently from rest of an application

Message Processing under MFC
- Like API programs, MFC programs must handle messages from Windows
- API mechanism: big switch/case statement
- MFC mechanism: "message maps" (lookup tables):
  - Table entries:
    - Message number
    - Pointer to a derived class member message-processing function

MFC Windows Programming (Document/View Approach)
- Frequently need to have different views of same data
- Doc/View approach achieves this separation:
  - Encapsulates data in a CDocument class object
  - Encapsulates data display mechanism & user interaction in a CView class object
Document Interfaces

- **Single Document interface (SDI) app**
  - Program deals with one document at a time

- **Multiple Document Interface (MDI) app**
  - Program organized to handle multiple documents simultaneously
  - Multiple open documents can be of same or different types
  - Example of an MDI application: Microsoft Word

Frame Window

- **Window in which a view of a document is displayed**
- A document can have multiple views associated with it
  - different ways of looking at the same data
  - But a view has only one document associated with it

Document/View Programs

- Almost always have at least four classes derived from:
  - `CFrameWnd`
  - `CDocument`
  - `CView`
  - `CWinApp`
- Usually put into separate declaration and implementation files
- Lots of initialization code
- Could be done by hand, but nobody does it that way

Documents, Views, & Frames

Microsoft Developer Studio AppWizard and ClassWizard Tools

AppWizard

- Tool that generates a Doc/View MFC program framework automatically
- Can be built on and customized by programmer
- Fast, efficient way of producing Windows Apps
- Performs required initialization automatically
- Creates functional `CFrameWnd`, `CView`, `CDocument`, `CWinApp` classes
- After AppWizard does it's thing:
  - Application can be built and run
  - Full-fledged window with all common menu items, tools, etc.
ClassWizard
- Facilitates message handling in a framework-based MFC application
- A tool that connects resources & user-generated events to program response code
- Writes C++ skeleton routines to handle messages
- Inserts code into appropriate places in program
- Code then can then be customized by hand
- Can be used to create new classes or derive classes from MFC base classes
- Add new member variables/functions to classes
- In .NET many “class wizards” available through ‘Properties window’

SKETCH Application
- Example of Using AppWizard and ClassWizard
- User can use mouse as a drawing pencil
  - Left mouse button down:
    - lines in window follow mouse motion
  - Left mouse button up:
    - sketching stops
  - User clicks “Clear” menu item
    - window client area is erased

Sketch data (points) won’t be saved
- So leave document (CSketchDoc) class created by AppWizard alone
- Base functionality of application (CSketchApp) and frame window (CMainFrame) classes are adequate
- Leave them alone
- Use ClassWizard to add sketching to CView class

Sketching Requirements
- Each time mouse moves:
  - If left mouse button is down:
    - Get a DC
    - Create a pen of drawing color
    - Select pen into DC
    - Move to old point
    - Draw a line to the new point
    - Make current point the old point
    - Select pen out of DC

Variables
- BOOLEAN m_butdn
- CPoint m_pt, m_ptold
- COLORREF m_color
- CDC* pDC

Steps in Preparing SKETCH
1. “File / New / Project”
   - Project Type: “Visual C++ Projects”
   - Template: “MFC Application”
   - Enter name: Sketch
2. In “Welcome to MFC Application Wizard”
   - Under “Application Type” Choose “Single Document”
3. Build App --> Full-fledged SDI App with empty window and no functionality
4. Add member variables to CSketchView
   - Can do manually in .h file
4. Easier to:
   - Select Class View pane in Properties window
   - Select and expand (+) SketchView class
     • Note member functions & variables
   - Right click on CSketchView class
     • Choose “Add / Variable”
       • Launches “Member Variable Wizard” Dialog Box
     • Variable Type: enter CPoint
     • Name: m_pt
     • Access: Public (default)
   - Repeat for other variables:
     • CPoint   m_ptold
     • bool   m_butdn
     • COLORREF m_color
     • CDC* pDC

4. Add message handler functions:
   - Select CSketchView in Class View
   - Select “Messages” icon in Properties window
     • Results in a list of WM_ messages
   - Scroll to WM_LBUTTONDOWN & select it
   - Add the handler by clicking on “<Add> OnLButtonDown” in resulting combo box
     • Note that the function is added in the edit window and the cursor is positioned over it:
       - After “TODO…” enter following code:
         m_butdn = TRUE;
         m_ptold = point;

Repeat process for WM_LBUTTONUP handler
   - Scroll to WM_LBUTTONUP
   - Click: “<Add> OnLButtonUp”,
   - Edit Code by adding:
     m_butdn = FALSE;

Repeat for WM_MOUSEMOVE
   - Scroll to WM_MOUSEMOVE
   - Click: “<Add> OnMouseMove”
   - Edit by adding code:
     if (m_butdn)
     {
       pDC = GetDC();
       m_pt = point;
       CPen newPen (PS_SOLID, 1, m_color);
       CPen* pPenOld = pDC->SelectObject (&newPen);
       pDC->MoveTo(m_ptold);
       pDC->LineTo (m_pt);
       m_ptold = m_pt;
       pDC->SelectObject (pPenOld);
     }
5. Initialize variables in CSketchView constructor
   - Double click on CSketchView constructor (in Class View)
   - After “TODO…”, Add code:
     m_butch = FALSE;
     m_pt = m_ptold = CPoint(0,0);
     m_color = RGB(0,0,0);

6. Build Project and Run

Menus and Command Messages
- User clicks on menu item
- WM_COMMAND message sent
- IDM_XXX identifies which menu item
- No predefined handlers
  - So message mapping macro is different
  - ON_COMMAND(IDM_XXX, OnXXX)
    - OnXXX() is the handler function
    - Must be declared in .h file and defined in .cpp file

Adding Color and Clear Menu Items to SKETCH App

Resource View (sketch.rc folder)
   - Double click Menu folder
   - Double click IDR_MAINFRAME menu
   - Add: “Drawing Color” popup menu item with items:
     • “Red” (ID_DRAWINGCOLOR_RED)
     • “Blue” (ID_DRAWINGCOLOR_BLUE)
     • “Green” (ID_DRAWINGCOLOR_GREEN)
     • “Black” (ID_DRAWINGCOLOR_BLACK)
   - Add another main menu item:
     • “Clear Screen” (ID_CLEAR)

Add Menu Item Command Handler Function
- One way: Use “Event Handler Wizard”
- In “Resource View” bring up menu editor
- Right click on “Red” menu item
- Select “Add Event Handler” in “Event Handler Wizard” dialog box
  - Class list: CSketchView
  - Message type: COMMAND
  - Function handler name: accept default
    - OnDrawingColorRed
  - Click on “Add and edit”
  - After “TODO…” in editor enter following code:
    m_color = RGB(255,0,0);

Another Method of Adding a Menu Item Command Handler
- In Class View Select CSketchView
- In Properties window select Events (lightning bolt) icon
- Scroll down to: ID_DRAWINGCOLOR_RED
- Select “Command”
- Accept “OnDrawingColorRed” handler
- Edit code by adding:
  m_color = RGB(255,0,0);
Repeat for ID_DRAWINGCOLOR_BLUE
– Code:
  m_color = RGB(0,0,255);

Repeat for ID_DRAWINGCOLOR_GREEN
– Code:
  m_color = RGB(0,255,0);

Repeat for ID_DRAWINGCOLOR_BLACK
– Code:
  m_color = RGB(0,0,0);

Repeat for ID_CLEAR
– Code:
  Invalidate();

Build and Run the Application