Microsoft Visual Studio: An Integrated Windows Program Development Environment

Microsoft Visual Studio

- Self-contained environment for Windows program development:
  - Creating/editing
  - Compiling/linking (building)
  - Testing/debugging
- IDE that accompanies Visual C++, Visual Basic, Visual C#, and other Microsoft Windows programming languages
- See Chapter 2 & Appendix C of the Deitel text
- Also Appendix C of the Gregory text

Some Visual Studio Components

- **The Editors:**

  C, C++, C#, VB source program text editors

  - cut/paste, color cues, indentation
  - generate source text files

  Resource Editors

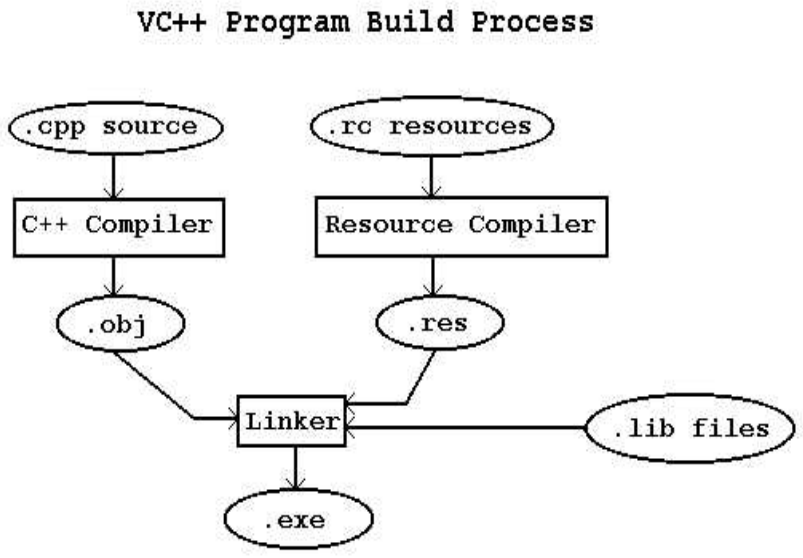
  - Resources: Windows static data
  - Determine look and feel of an application
    - icons, bitmaps, cursors, menus, dialog boxes, etc.
  - graphical
  - generate resource script (.rc) files
  - integrated with text editor
  - created visually

.NET Language Compilers

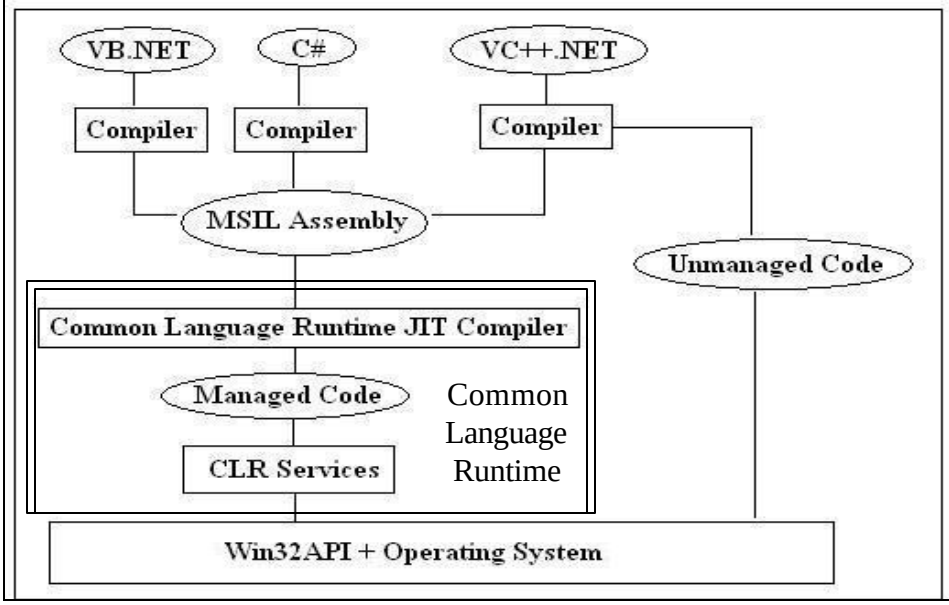
- **Unmanaged Code C/C++ Compiler**
  - translates source programs to machine language
  - generates object (.obj) files for linker
- **Managed Code .NET Language Compilers**
  - Many of them ✍ multi-language interoperability
  - Translate source programs to MSIL
  - Generate a "Portable Executable" that must be translated to target machine language by the CLR
- **Resource Compiler**
  - Reads .rc file
  - Generates binary resource (.res) file for linker

The Linker

- Reads compiler .obj and .res files
- Accesses C/C++/Windows libraries
- Generates executable (.exe or .dll)



Program Build and Run in the .NET Framework



The Debugger

- Powerful source code debugger
- Integrated with all parts of Visual Studio
- Features
  - breakpoints
  - tracing through/over functions
  - variable watch windows
- See Appendix C of Deitel text book

The Wizards

- **AppWizard**
  - Windows code generator for Windows apps
  - automatically creates working program templates & skeleton code
- **ClassWizard**
  - facilitates easy extension of AppWizard-generated classes
  - creation of new classes
  - used to tailor AppWizard-generated MFC & .NET skeletons
  - Accessible in the **Properties Window** in .NET

Help

- Essential when developing Windows apps
- Hover over key words in edit window and a one-line help message appears
- 'Help' Menu Item
  - 'Dynamic Help' – context sensitive
    - Click on text in edit window and corresponding topic appears in help window
    - Click on topic in help window to get help
  - 'Contents': Select a topic
  - 'Search': Enter a topic
  - 'Index': Enter/choose a topic

MSDN Library (on Web)

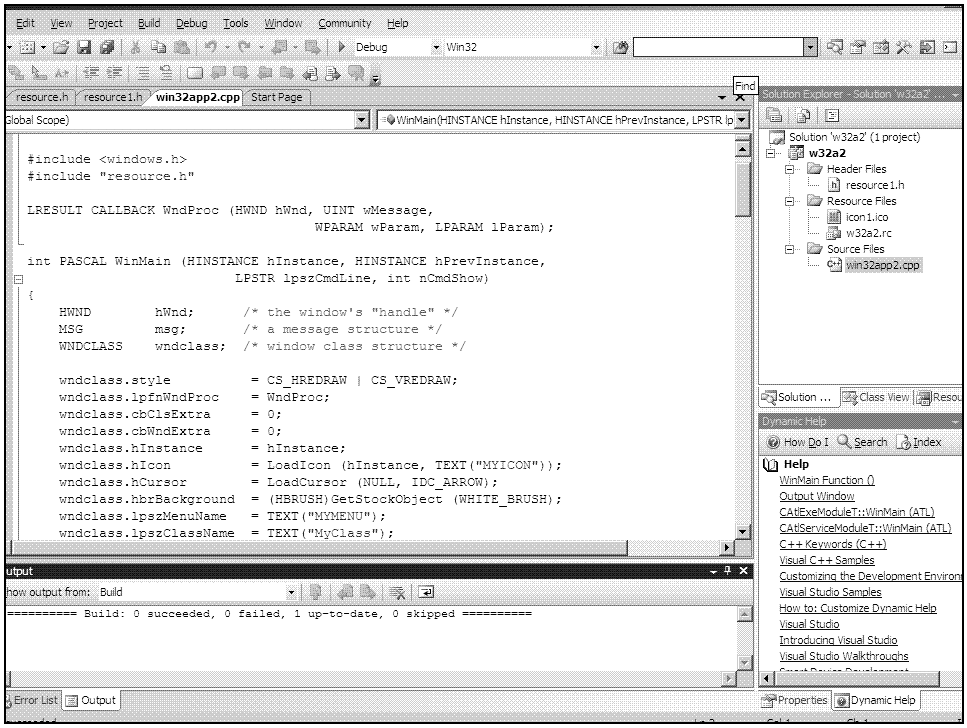
- **Go to:** http://msdn.microsoft.com
  - **Search MSDN for desired topic**
  - **Some examples:**
    - Windows API reference
    - MFC reference
    - Windows forms class library

Using Visual Studio

- To prepare many kinds of applications
  - Win32 Console Applications (DOS programs)
  - Win32 API Apps in C or VC++
  - MFC Apps in VC++
  - DLLs
  - .NET Windows Forms Apps in Managed C#, VB, VC++, and other languages
  - ASP.NET Web Apps and Services
  - ADO.NET Data Base Apps
  - Others

Visual Studio Layout

- Menu Bar
- Several Tool Bars
- View Windows (to the side)
  - Solution Explorer
  - Class View
  - Resource View
  - Properties Window
- Working Area (main window)
  - Text Editor to enter/modify source code
  - Resource Editors
  - Tab between different work areas
- Output Window & Status Bar (bottom).
  - System Messages (errors)
- Windows can be moved around, docked and undocked



Toolbars

- Contain Icons--instant routes to main menu functions
- May not be visible
- If not, right click on any visible toolbar
- Brings up following popup window
- Can activate a toolbar by clicking on its check box

Keyboard Shortcuts

- All Menu/Toolbar selections are available from the keyboard using key combinations
- Can be faster
- More information in Online Help
  - ‘Index’ | ‘Keyboard Shortcuts’ | ‘Predefined’

Solutions and Projects

- Solution
  - A single application
  - Can contain one or more projects
    - In Managed applications, projects can be in different languages
  - Overall solution information stored in a .SLN file
  - Open this when you want to work on a solution
- Project
  - Basic component of an application
  - Collection of files:
    - Source, headers, resources, settings, configuration information, many more

Important Visual Studio Generated Files

.sln	**Solution**
.vcproj	**Project**
.c, .cpp, .cs	**C/C++/C# Windows App source code**
.h	**C/C++ header**
.rc	**Resource script**
.res	**Compiled resource**
.ico	**Icon**
.bmp	**Bitmap image**
.exe	**Executable program**
.dll	**Dynamic Link Library**
.aspx	**ASP.NET Web Form source code**
.asmx	**ASP.NET Web Service source code**

Temporary Visual Studio Generated Files

- **Many are very big and can (should) be removed!**
- **`.obj` Compiler machine code translation**
- **`.ilk` Incremental link file**
- **`.pch` Precompiled header (huge!)**
- **`.pdb` Precompiled debugging info**
- **`.idb` Incremental debug info**
- **`.ncb` intellisense database (huge!)**
- **`.aps` Supports viewing resources**
- **Others**
- **Can be deleted**

Program Configurations

- Debug
  - appends debugging information
  - produces more and larger files
- Release
  - no debugging information
  - optimized for size, performance, & efficiency

Setting the Configuration

- Click ‘Build’ on Main Menu
- Choose ‘Configuration Manager’
- Choose desired configuration (‘Debug’ or ‘Release’) in Configuration Manager’s ‘Active Solution Configuration Box’
- Default is ‘Debug’

Creating a Win32 API Windows Application with Visual Studio

- **Startup**
  - Click ‘Start’ on Task Bar – ‘All Programs’
  - ‘Microsoft Visual Studio 2008’ | ‘Microsoft Visual Studio 2008’
- **Creating a new Win32 API solution**
  - ‘File’ | ‘New’ | ‘Project’ from Menu Bar
  - In ‘New Project’ box, select ‘Visual C++’ ‘Win32’ from ‘Project Types:’ & click on ‘Win32 Project’ in ‘Templates’
  - Set the ‘Location’ to a convenient directory & name the project (e.g. win32app1)
  - ‘OK’

- **Click 'Application Settings' in resulting 'Win32 Application Wizard' Box**
  - Select 'Windows Application' from 'Application Type' radio buttons
  - Select 'Empty Project' from 'Additional Options' check boxes
  - Click 'Finish'

- **Inserting source files into project:**
  - Open a new C++ file & type or copy/paste the code into the program:
    - 'File' | 'New' | 'File' from menu
    - Choose 'Visual C++' from 'Categories', C++ file (.cpp) from 'Installed Templates', & click 'Open'
    - Type or paste source code into the resulting Edit window
    - Save the file in the project's subdirectory as a C++ source file, giving it an appropriate name (e.g., win32app1)
  - Add the source file to the project:
    - Choose 'Project' | 'Add Existing Item' from menu
    - Click on the file you saved (e.g. win32app1.cpp)
    - Confirm that it was added to the project by expanding 'Source Files' in the Solution Explorer Window
      - If Solution Explorer is not visible, select 'View – Solution Explorer' from the menu

- **Alternative Way of Adding a Source File to a Project:**
  - You can also copy an existing source code file into the project's subdirectory
  - Then as before:
    - Choose 'Project' | 'Add Existing Item' from the menu
    - Select the .cpp file & click 'Open'
      - Should appear in Solution Explorer window
      - Open it by double clicking on it

- **Building the Solution:**
  - 'Build' | 'Build Solution' from menu
  - Project will be compiled/linked
  - Messages/errors will appear in Output Window
- **Running the Program:**
  - 'Debug' | 'Start' from menu
    - Shortcut key: F5
  - Or 'Debug' | 'Start Without Debugging' from menu
    - Shortcut key: Ctrl-F5

Compiling from Command Line

- Command Line Compilers:
  - C++: CL.EXE
  - C# : CSC.EXE
  - Visual Basic: VBC.EXE
- All are run from a DOS session, but directory paths must be set correctly
- Easiest to start a Visual Studio 2008 Command Prompt (paths already set)
  - From Task Bar:
    - Start | All Programs | Microsoft Visual Studio 2008 | Visual Studio Tools | Visual Studio 2008 Command Prompt

Command Line Compiling, continued

- To compile our first Visual C++, Win32 API application (win32a1.cpp) from the command line:
  - cl kernel32.lib user32.lib gdi32.lib win32a1.cc
  - Note that any required libraries (DLLs) must be specified
- There are many compiler options:
  - See Online Help:
  - 'Index' | 'cl.exe compiler' | 'building programs'
  - For C#: 'Index' | 'csc.exe'
- We won't be using command line compilers much in this course, but they're there if you need them

- **Cleanup:**
  - Copy solution, project, source, header, resource files to disk
  - Copy .exe file from project's Debug directory
  - Best: Delete all temporary files & copy entire solution (project directory) to floppy or CD
  - Delete project directory from hard drive
- **Exiting Developer Studio:**
  - ‘File’ | ‘Exit’ from menu