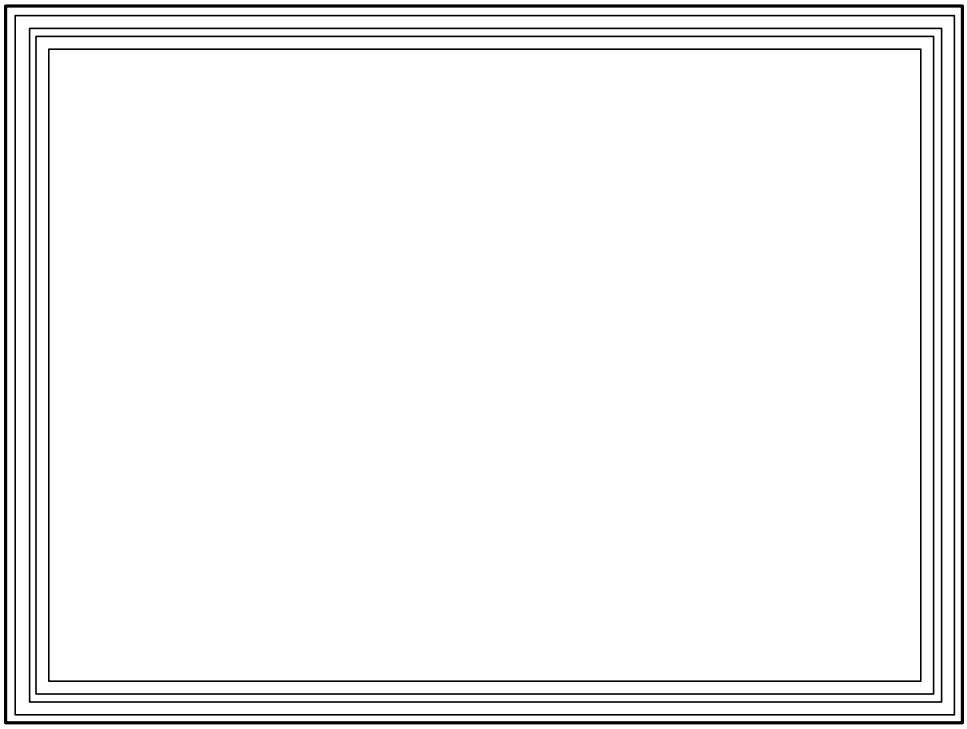


Binghamton University

EngiNet™

State University of New York

Thomas J. Watson

School of Engineering
and Applied Science

EngiNet™

WARNING

CS 560

Computer Graphics

Professor Richard Eckert

Lecture # 2

January 24, 2001

Lecture 2

Wednesday, 1-24-01

- **A. Using Microsoft Visual C++ Developer Studio 97**
- **B. An Introduction to Windows Programming**

A. Microsoft Visual C++ Developer Studio IDE

See notes & example programs from CS-360

URLs:

http://www.cs.binghamton.edu/~reckert/360/360notes.html
http://www.cs.binghamton.edu/~reckert/360/360pgms.html

Especially MFC notes and examples

Using Microsoft Visual C++ Developer Studio

- Self-contained environment for Windows program development:
  - creating
  - compiling
  - linking
  - testing/debugging
- IDE that accompanies Visual C++

Visual Studio Components

- **The Editors**

  C or C++ source program text editor
  - cut/paste color cues, indentation,
  - generates text files

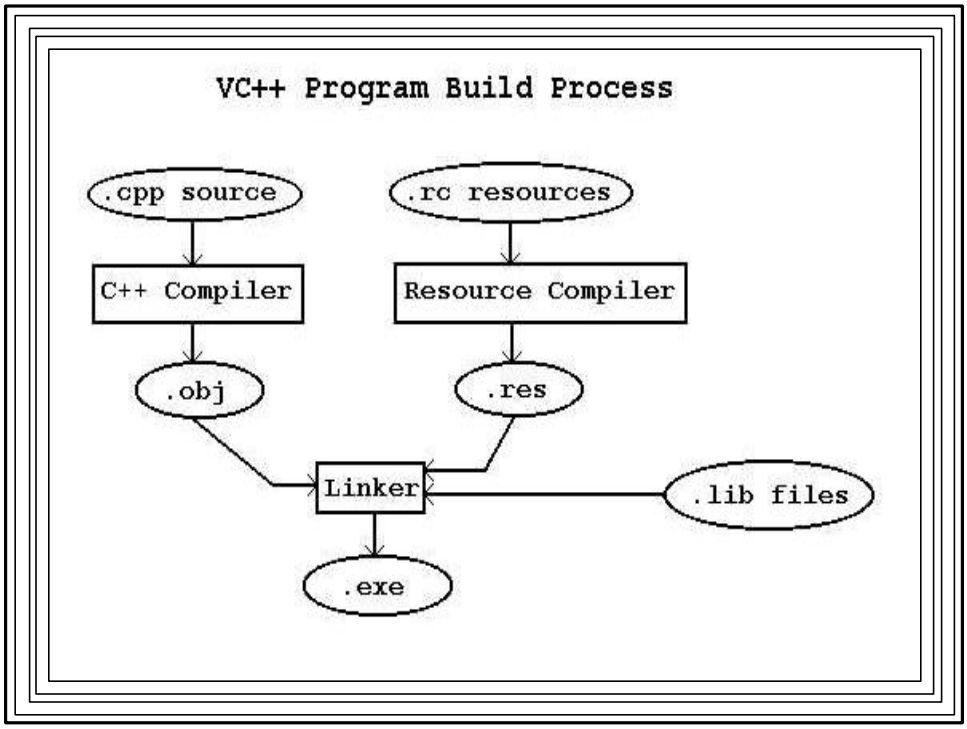
  Resource Editor
  - icons, bitmaps, cursors, menus, dialog boxes, etc.
  - graphical, WYSIWYG, Integrated
  - generates resource script (.rc) files
  - integrated with text editor

The Compilers

- **C/C++ Compiler**
  - translates source programs to machine language
  - detects and reports errors
  - generates object (.obj) files for linker
- **Resource Compiler**
  - Reads .rc file
  - Generates binary resource (.res) file for linker

The Linker

- reads compiler .obj/.res files
- accesses C/C++/Windows libraries
- generates executable (.exe or .dll)



The Debugger

- powerful source code debugger
- integrated with all parts of Dev Studio
- Features
  - breakpoints
  - tracing through/over functions
  - variable watch windows
  - much more

The Wizards

- **AppWizard**
  - Windows code generator for MFC apps
  - automatically creates working program skeletons
- **ClassWizard**
  - facilitates easy extension of AppWizard-generated classes
  - used to tailor AppWizard-generated MFC skeletons

Online Help

- Can be accessed by:
  - InfoViewer book/chapter
  - Topic (keyword search-->relevant topics/articles)
  - F1 help (help on item under mouse cursor)
  - The Web: MSDN (Microsoft Developer Network)

InfoViewer Online Help (Win32 API Programming)

Platform, SDK, and DDK Documentation
- Platform SDK
  - Reference
    - Functions
      - Win32 Functions (Alphabetical listing)
    - Messages
      - Win32 Messages (Alphabetical listing)
    - Structures
      - Win32 Structures (Alphabetical listing)

InfoViewer Online Help (MFC Programming)

Developer Products
- Visual C++
  - Microsoft Foundation Class Reference
    - Class Library Reference
      - Select Desired Class

MSDN Library

http://msdn.microsoft.com/siteguide/sitemap.asp

Go to MSDN Library:

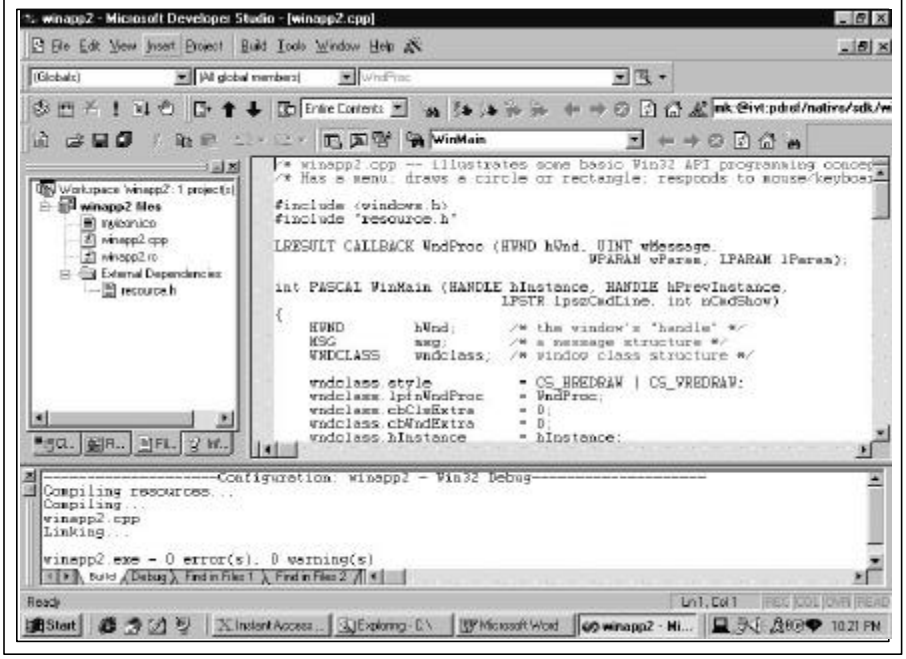
- Visual Studio 6.0 Documentation
  - Visual C++ Documentation
    - Reference
      - Microsoft Foundation Class Library & Templates
        - Microsoft Foundation Class Library
          - Class Library Reference
- Platform SDK
  - Graphics and Multimedia
  - Win32 API
    - Reference
      - Win32 Functions in Alphabetical Order

Other Advanced Tools

- SPY++
- PVIEW
- ActiveX utilities, a gallery of software components
- More

Using Developer Studio 97 and Visual C++ 5.0/6.0

- To prepare many kinds of applications
  - Win32 Console Applications (DOS programs)
  - Win32 API Apps
  - Win32 MFC apps
  - DLLs
  - Lots of others

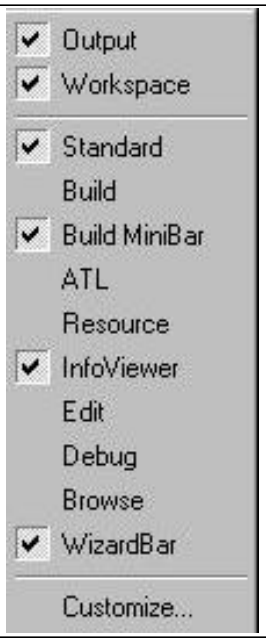


Components

- Menu bar
- Several tool bars
- Project Workspace Window (left)
  - InfoView, FileView, ClassView, ResourceView
- Editor Window (right)
  - Use Text Editor to Enter C/C++ code
  - Use Resource Editors
- Output Window (bottom).
  - System messages (errors)

Toolbars

- Contain icons--instant routes to main menu functions
- Output, Workspace, Standard, Build, Edit, InfoViewer, Resource, etc.
- 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



Projects and Workspaces

- Project
  - collection of interrelated source files
  - compiled/linked to generate a Windows executable
- Project Workspace
  - folder with all information relating to a project or combinations of projects
  - also used to refer to Visual Studio desktop window
- Project info stored in .dsw and .dsp text files

Important Dev Studio Generated Files

- `.dsp` Project file
- `.c or .cpp` C/C++ source
- `.h` C/C++ header
- `.dsw` Workspace file
- `.rc` Resource script
- `.res` Compiled resource
- `.ico` Icon
- `.bmp` Bitmap image
- `.exe` Executable program
- `.dll` Dynamic Link Library (if used)
- `.obj` Machine code translations

Temporary Dev Studio generated files

- **Many are huge and can (should) be removed!**
- `.ilk` Incremental link file
- `.pch` Precompiled header
- `.pdb` Precompiled debugging info
- `.idb` Incremental debug info
- `.ncb` Supports viewing classes
- `.aps` Supports viewing resources
- `.bsc` Browser information file
- `.clw` Supports ClassWizard
- `.opt` Workspace configuration
- `.plg` Build log file

Windows Program Configurations

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

Setting the Configuration

- Click "Build" on Main Menu
- Choose "Select Active Configuration”
- Choose configuration (“Debug” or “Release”)
- Default is “Debug”

Create a Win32 App w/ Dev Studio

- **Startup**
  - click ‘Start’ on Task Bar
  - ‘Programs | Microsoft Visual Studio | Microsoft Visual C++’
- **Creating Project**
  - ‘File | New’ from menu
  - ‘Projects’ Tab (if not chosen)
  - ‘Win32 Application’
  - Name the project (e.g. winapp1)
  - ‘OK’

- **“Win32 Application, Step 1 of 1” Window**
  - Select “An Empty Project”
    - Click “Finish”
- **“New Project Information” Window**
    - Click “OK”

- **Inserting source files into project:**
  - Open new C++ file, type or copy/paste the code into the program
    - "File | New | Files tab | C++ Source"
    - Make sure "Add to Project is checked"
    - Enter a file name (e.g., winapp1)
    - Type or paste in the resulting Edit window
  - To see/modify a file added to project:
    - click FileView tab in Workspace Window
    - click on file name in FileView window

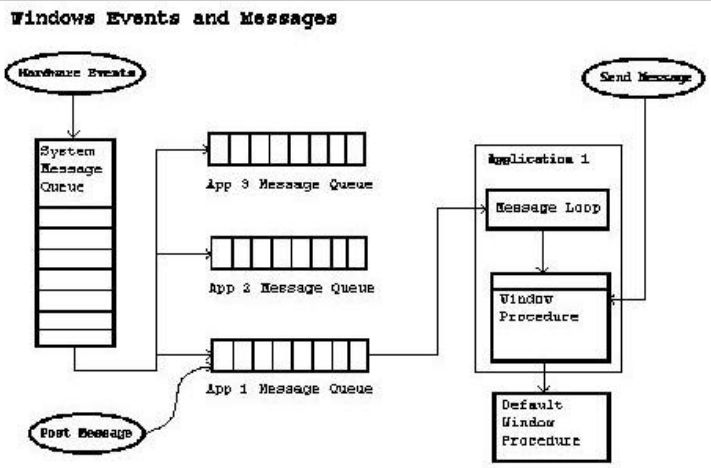
- **Building Project:**
  - 'Build | Build winapp1" from menu
    - Shortcut key: F7
  - Project will be compiled/linked
  - Messages will appear in Output Window
- **Running Program:**
  - 'Build | Execute winapp1"
    - Shortcut key: Ctrl-F5, or click exclamation point

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

B. An Introduction to Windows Programming

Windows Programming

- Event-driven, graphics oriented
- Example: User clicks mouse over a program's window area--
  - Windows decodes HW signals from mouse
  - figures out which window user has selected
  - sends a message to that window's pgm:
    - "User has clicked over (X,Y)"
    - "Do something and return control to me"
  - Program reads message data, does what's needed, returns control to Windows



Overview of Win32 API Program Structure--2 main tasks:

- Initial activities
- Process messages from Windows (the message loop)

Pseudocode

- Initialize variables, memory space
- Create & show program's Window
- Loop
  - Fetch any msg sent from Windows to this pgm
  - If message is QUIT
    - terminate program, return control to Windows
  - If message is something else
    - take actions based on msg & parameters
    - return control to Windows
- End Loop

Essential Parts of a Windows Program

- I. The source pgm (.c/.cpp file):
  - A. WinMain() function
    - 0. declarations, initialization, etc.
    - 1. register window class
    - 2. create a window based on a registered class
    - 3. show window, make it update its client area
    - 4. the message loop (get messages from Windows, dispatch back to Windows for forwarding to correct callback message-processing function)

- B. WndProc(): the message-processing function
    - a big switch/case statement
  - Under Win32 API, programmer must write WinMain() and the WndProc()
  - Under MFC, Wizards do most of the work
  - WinMain() and WndProc() are buried in the framework
  - Write “message mapped handler functions” instead

- II. The resource script (.rc file):
  - contains resource (Windows static) data
  - separate from code and dynamic data
  - compiled by a separate "Resource Compiler"
  - Examples:
    - Keyboard Accelerators, Bitmaps, Cursors, Dialog Box, Fonts, Icons, Menus, String Tables
  - Separation of resources and program code==>
    - separates tasks of programmer & designer
    - can change user interface w/o touching code

Some Important Messages

- WM_DESTROY-- User does action to kill window
- WM_COMMAND--User clicked on menu item (menu item ID provided)
- WM_*BUTTONDOWN--left/right mouse button pressed (* = L or R; x,y coordinates provided)
- WM_MOUSEMOVE--mouse moved
- WM_CHAR--User pressed valid ANSI code character keyboard key combination (ANSI code provided)
- WM_PAINT--Part of window was exposed & should be redrawn
- WM_KEYDOWN--keyboard key pressed (virtual key code provided)

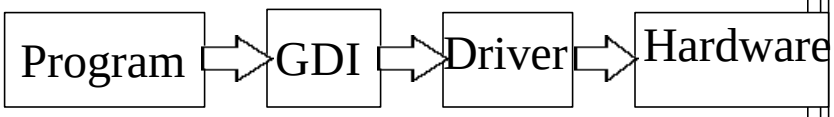
TEXT AND GRAPHICS OUTPUT

- Displaying something in a window
- Text & graphics done one pixel at a time
- Any size/shape/position possible
- Design goal: Device Independence

Device Independent Graphics Interface

- Windows programs don't access hardware devices directly
- Make calls to generic drawing functions within the Windows 'Graphics Device Interface' (GDI) -- a DLL
- The GDI translates these into HW commands

Device Independent Graphics Interface

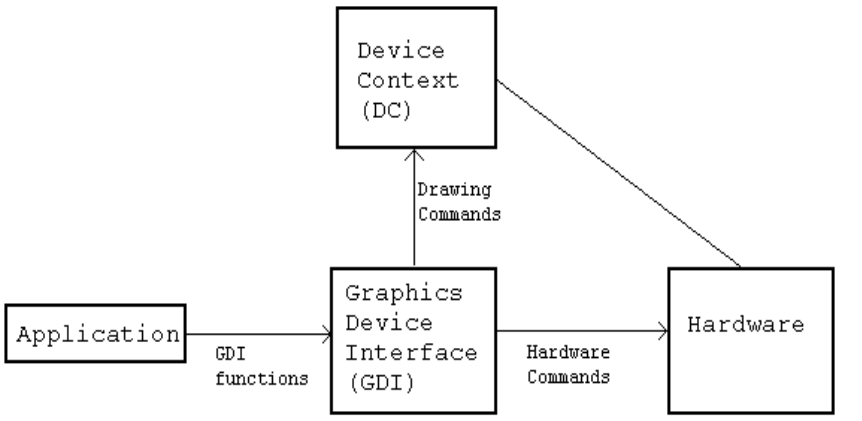


- May use device drivers (HW control programs)
- Thus graphics I/O done in a "standard" way
- Programs will run unaltered on other HW platforms

Device Context

- **Windows programs don't draw directly on HW**
- **Draw on "Device Context" (DC)**
  - **Abstracts the device it represents**
  - **Like a painter's canvas**
  - **Specifies drawing attribute settings**
    - **e.g., text color**
  - **Contains drawing objects**
    - **e.g., pens, brushes, bitmaps, fonts**

The DC and the GDI



Some GDI Attribute Settings

ATTRIBUTE	DEFAULT	FUNCTION
Background color	white	SetBkColor()
Background mode	OPAQUE	SetBkMode()
Clipping Region	whole surf.	SelectClipRgn()
Current Position	(0,0)	MoveToEx()
Drawing Mode	R2COPYPEN	SetROP2()
Mapping Mode	MM_TEXT	SetMapMode()
Text Color	Black	SetTextColor()

Some GDI Drawing Objects

Object	Default	What it is
Bitmap	none	image object
Brush	WHITE_BRUSH	area fill object
Font	SYSTEM_FONT	text font object
Pen	BLACK_PEN	line-drawing object
Color Palette	DEFAULT_PALETTE	color combinations

- Can be created with GDI functions
- Must be “selected” into a DC to be used