CS 460

Computer Graphics

Professor Richard Eckert

Lecture # 1

January 27, 2009

CS-460
Computer Graphics

Richard R. Eckert

T,R 10:05-11:30 A.M.
SW-327

Lecture 1 - 1/27/2009

Contacting Me or the TA

- Office: EB-N6
- Office Hours: W 10-11:30 A.M., R 1-2:30 P.M.
- Office Phone: 607-777-4365
- Department phone: 607-777-4802
- email: reckert@binghamton.edu
- My web front page: www.cs.binghamton.edu/~reckert/
  - See link to: CS-460/560 (Computer Graphics)
- Listserv: CS460-L@listserv.binghamton.edu
  - Activated during the first week of classes
- TA: Yibo Sun, sunyibo@gmail.com
- TA's Office hours: TBA

Course Materials

- Text book
  - D. Hearn and M.P. Baker, "Computer Graphics with OpenGL", 3rd Edition, Prentice Hall
- Online notes
  - CS-460/560 link on my home page
    - Lots of information available there
  - CS-360 link on my home page
    - Information on using Visual Studio, VC++, C#, Example Programs
- PowerPoint slides in PDF format
  - Will be online at course notes web site

Software

- Microsoft Visual Studio 2005/2008 Professional Edition
  - In all Pods & Watson School Microlab
  - Available to Watson School students (free)
    - Through Microsoft Academic Alliance
  - Go to:
    - msdn04.e-academy.com/binghamton_watson
  - Search for product
    - To download you will need a password
    - You should have it or it will be emailed to you

Course Prerequisites

- Data Structures (CS-240)
- Basic Knowledge of Linear Algebra
  - Matrix/Vector Manipulation
- C or C++ Programming
  - Visual C++ Ideal
    - But we will do a quick review
    - Extensive notes/examples at CS-360 web pages
- Some Knowledge of Computer Organization
  - e.g., CS-220

Course Evaluation

- 2 Term Exams (20% each)
- Programming Assignments (40%)
- Final Exam (20%)

Course Schedule (by weeks)

- Introduction/Applications, Introduction to Windows and OpenGL Programming
- Computer Graphics Hardware and Software
- Graphics Output Primitives: Scan converting lines, polygons, circles, curves, text
- Display Attributes and Area Fill Algorithms
- 2-Dimensional Geometric Transformations
- 2-D Windows, Viewports, and Clipping

*** Term Examination # 1 ***

Course Schedule (by weeks)

- Interactive 2-D Graphics: Input Devices, GUI Techniques
- Segmentation, Hierarchical Modeling; PHIGS, OpenGL
- Curved lines and surfaces, parametric equations, Bezier and B-spline curves
- Animation, Sprites, Game Development, DirectX
- 3-D Graphics: Modeling & Transformations
- 3-D Graphics: Viewing and Projections

Course Schedule (by weeks)

- Hidden Surface Removal

  *** Term Examination # 2 ***
- Illumination, Reflection, Shading, Texturing, Ray Tracing, Radiosity
- Fractals, Iterated Function Systems, L-Systems, Particle Systems, Escape-time algorithms, Chaos

Introduction to Computer Graphics

Computer Graphics

- Using a computer to generate visual images
- Definition of Computer Graphics:
  - Creation, storage, manipulation, and display of models of scenes using a computer
- Interactive Computer Graphics:
  - User dynamically controls displayed image attributes by means of interactive input devices

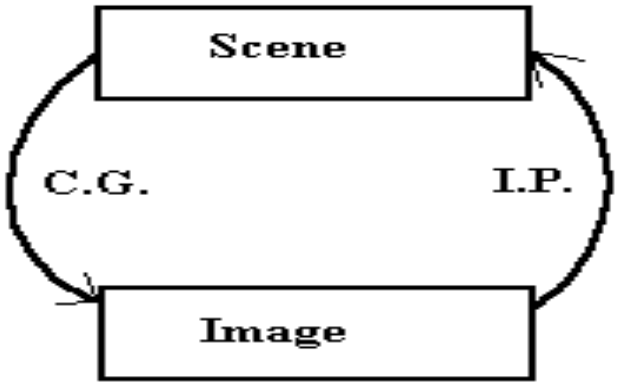
Motivation

- Human visual channel highly developed
- Efficient for communicating complex ideas

Related Field: Image Processing

- Image enhancement/understanding
- Reconstruction of objects from images
- Computer Graphics--Synthesis of images
- Image Processing--Analysis of images
- Image Processing subfields:
  - image enhancement
  - Image understanding
  - computer vision
  - pattern recognition (A.I. important)

Computer Graphics & Image Processing



Three Phases of Computer Graphics

- Modeling
  - Representing objects/scenes mathematically
- Rendering
  - Producing an image from a model
- Animation
  - Making an image move

Features of Computer Graphics Models

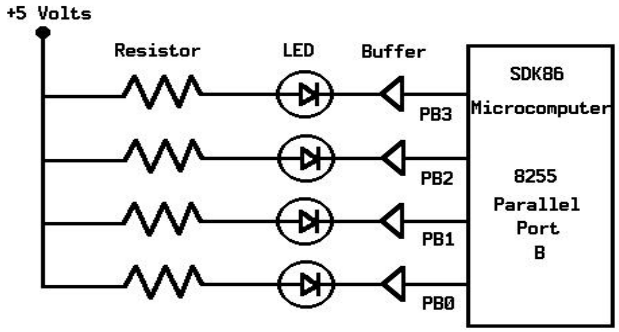
- Output primitives:
  - building blocks
- Data structures:
  - how primitives relate to each other

Levels of Complexity of Computer Graphics

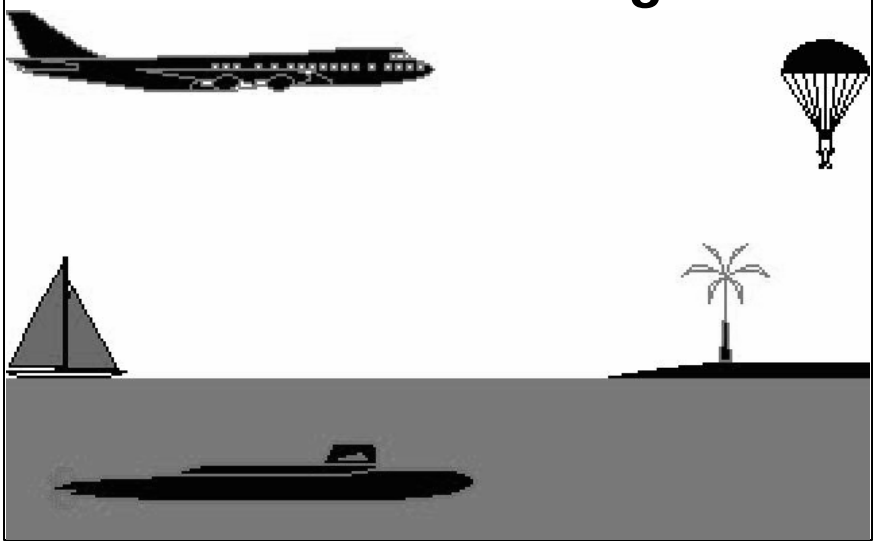
- 2-D line Drawings: Primitives
- 2-D colored images: Area fill
- 3-D line drawings: 3-D to 2-D projection
- 3-D colored images: Hidden surface removal, color, shading
- 3-D photorealistic images: materials properties, lighting, reflection, transparency, shadows (physics), complex object models
- Animation at all levels: Movement

2-D Line Drawing

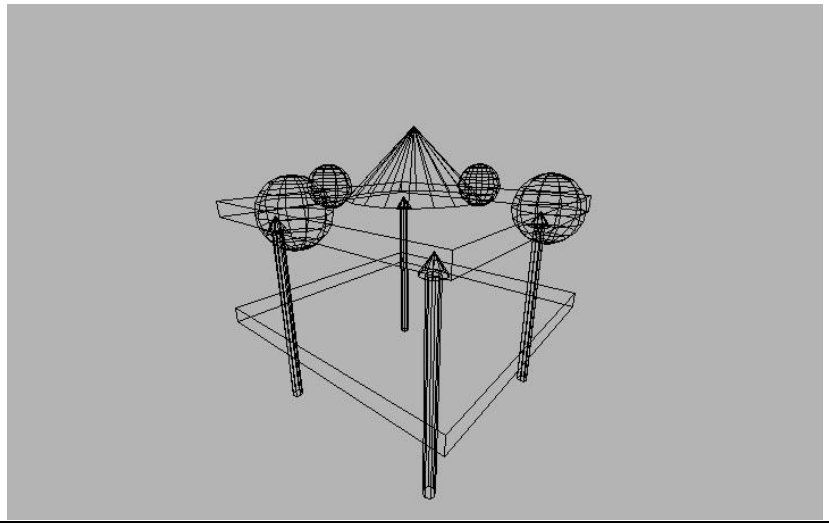
The Hardware



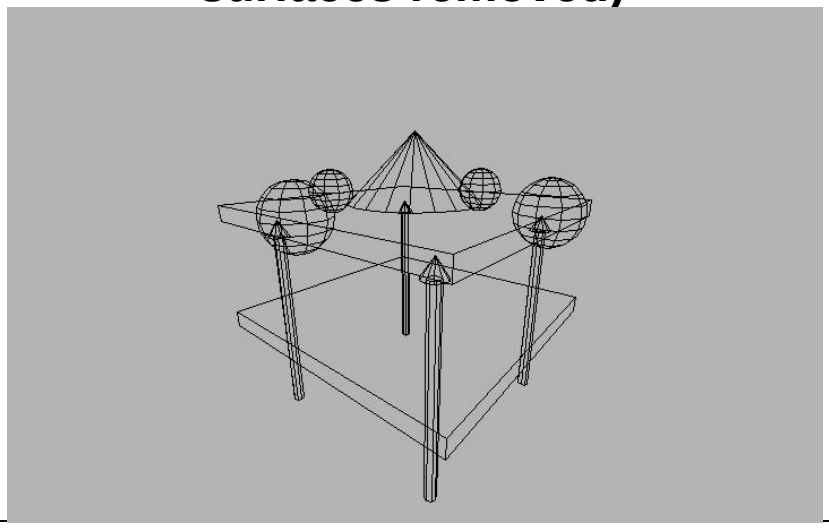
2-D Colored Image



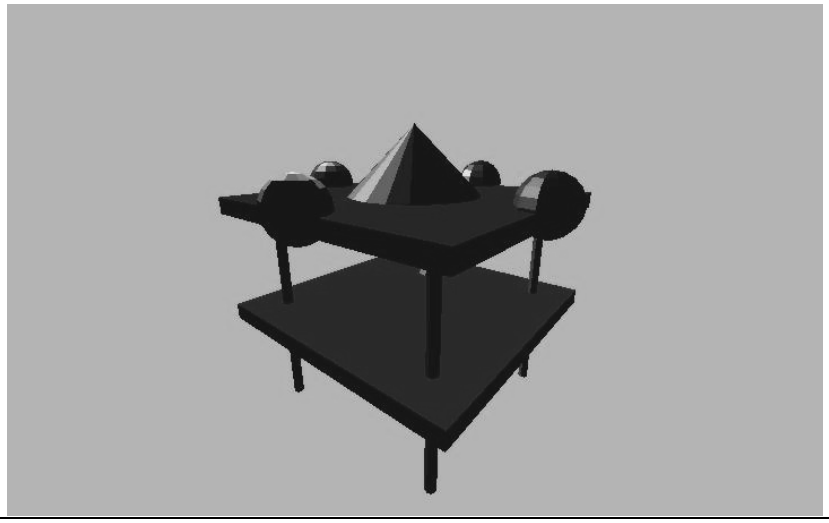
3-D Line Drawing



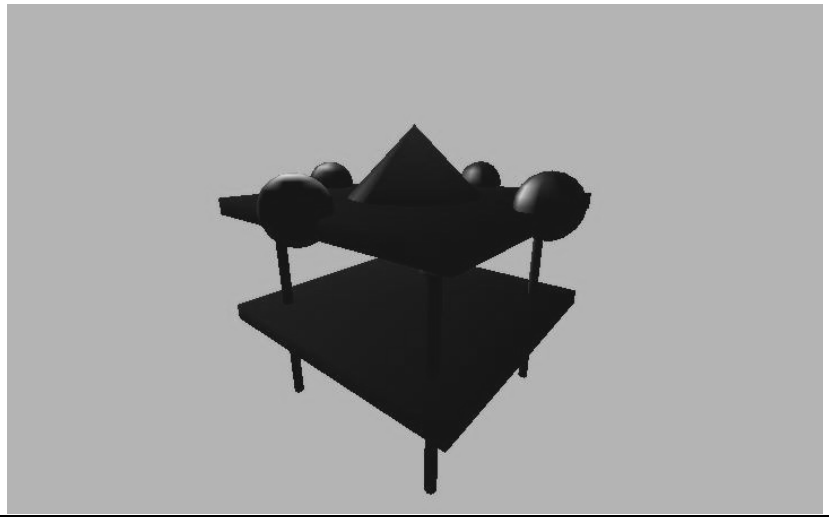
3-D Line Drawing (some hidden surfaces removed)



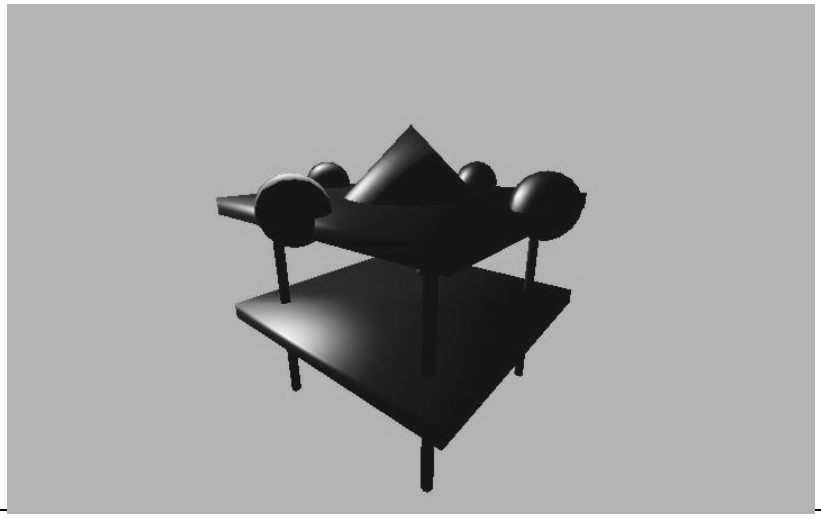
3-D Colored Image (flat shaded)



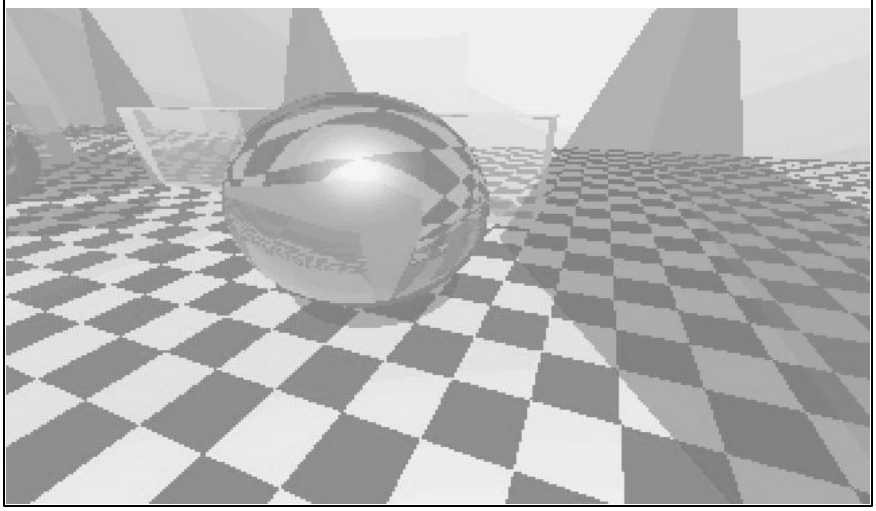
3-D Colored Image (smooth shaded)



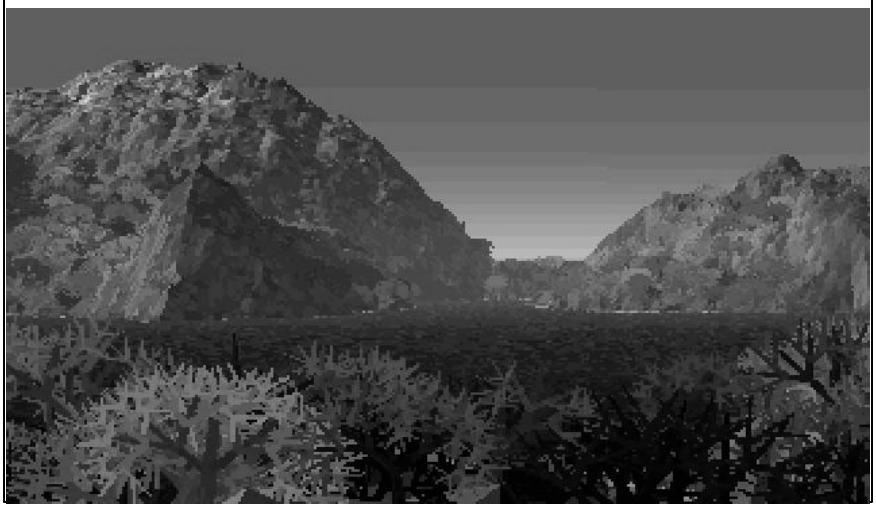
3-D Colored Image Smooth Shaded with Specular Highlights



3-D Photorealistic Image (ray traced image with texture mapping)

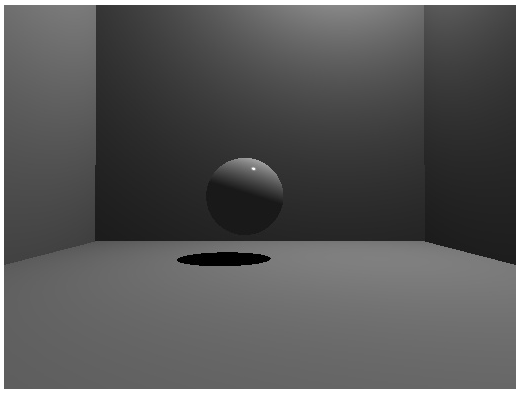


3-D Photorealistic Image (fractal mountains, L-system plants)



An Animation of a 3D Scene

- Frames generated by ray tracing

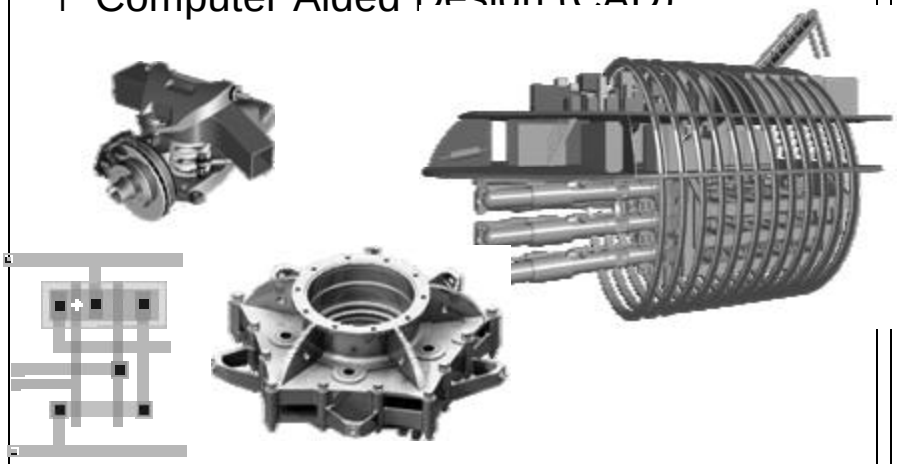


Some Applications of Computer Graphics

- Data Presentation (statistics, business, scientific, demographics...)
- CAD, CAM, CIM
- Painting/Drawing systems
- TV Commercials
- Entertainment
  - Video Games
  - Motion Picture Industry
- Cartography
- Computer Art

Graphics Applications

- Computer Aided Design (CAD)



Graphics Applications

- Entertainment: Cinema

Pixar: Monster's Inc.

Video Games

- Microsoft Xbox 360
- Sony PlayStation 3
- Nintendo Wii
  - Wireless controller – Wii Remote

Video Games - Nintendo Wii

Graphics Applications

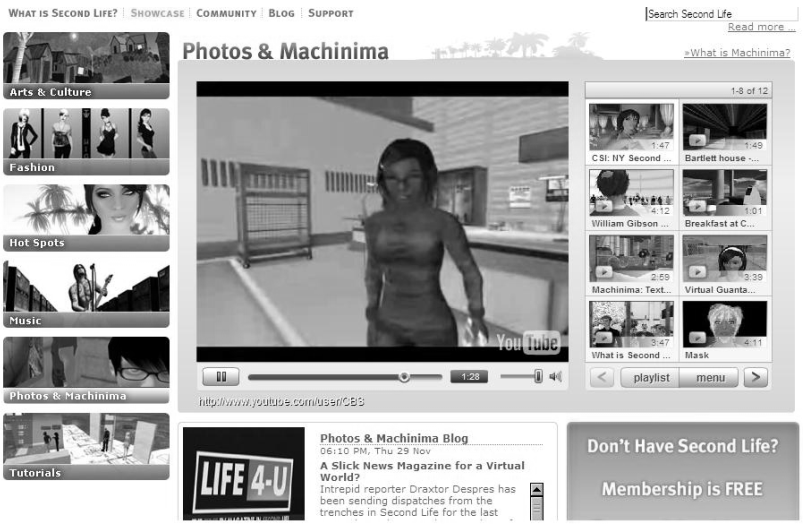
- Desktop Publishing
- Architectural Design
- Simulation of Reality
  - Flight simulators
  - Ground vehicle simulators
  - Arcade games
  - Virtual reality
    - Second Life

Simulation

Driving Simulation
(Evans & Sutherland)

Flight Simulation
(NASA)

Virtual Worlds – Second Life

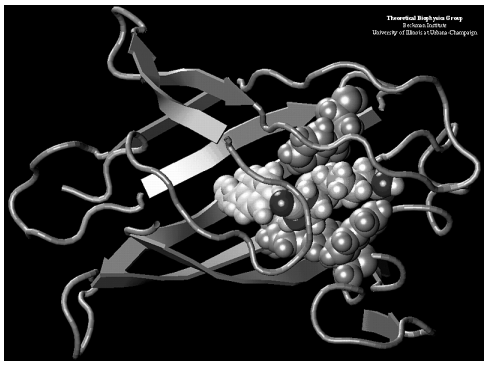


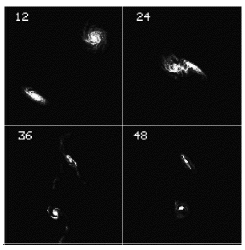
Graphics Applications

- Scientific Simulation/Visualization
  - Use graphics to make sense of vast amounts of scientific data
  - Use when too dangerous/expensive or impossible to do real experiments
- Hypermedia
  - Integrate broadcasting, computing, publishing
- Education and Training
- CASE

Graphics Applications

- Scientific Visualization



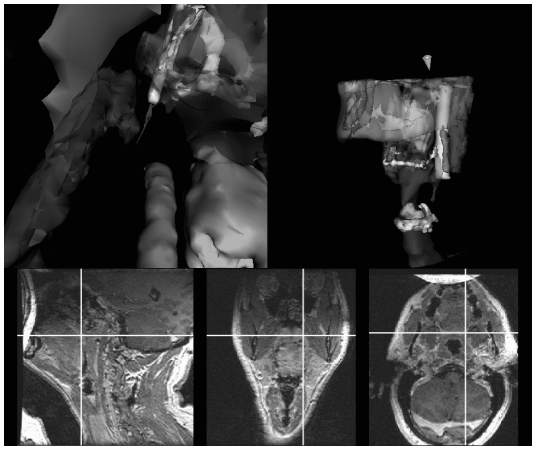


Graphics Applications

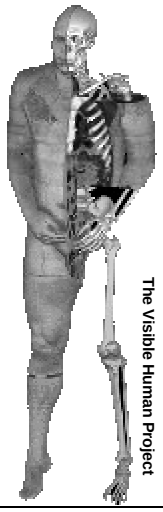
- Image Processing/Enhancement
- Medicine
  - Computed Tomography (CT Scan)
  - X-ray, ultrasound, NMR, PET:
  - All can give 3-D images of human anatomy
  - Computer-aided Surgery
- GUIs
- World Wide Web Development
- New Stuff--can't even be imagined

Graphics Applications

- Medical Visualization



MIT: Image-Guided Surgery Project



The Visible Human Project

Computer Graphics--

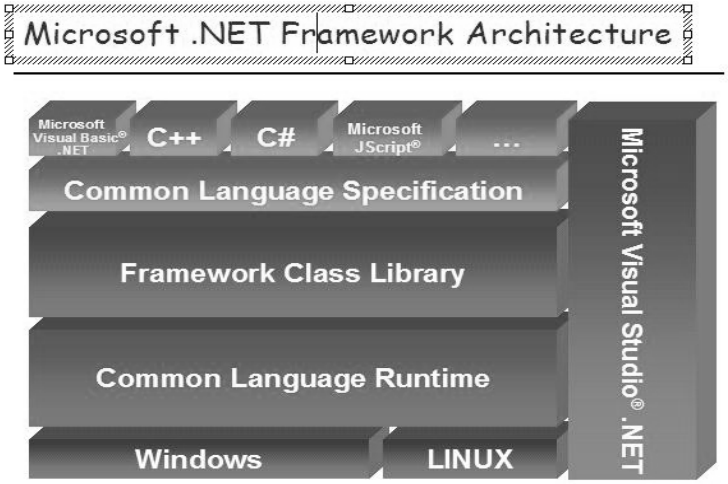
- A huge, fast-moving, exciting field that integrates the best of art and science
- Needs new Renaissance men & women
  - Bright and analytic enough to understand the science & math
  - Sensitive and creative enough to do the art
- Both left and right sides of the brain required!

Microsoft Visual Studio .NET: An Integrated Windows Program Development Environment

Using Microsoft Visual Studio .NET

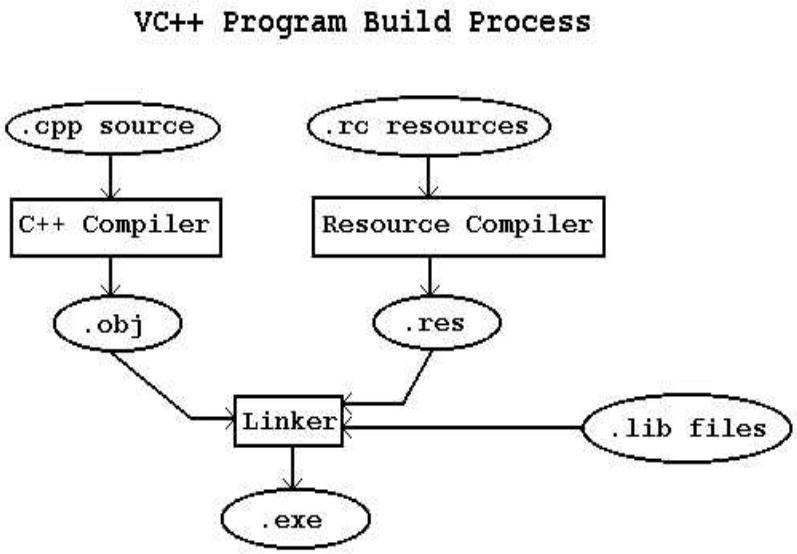
- Self-contained environment for Windows program development:
  - creating
  - compiling
  - linking
  - testing/debugging
- IDE that accompanies Visual C++, Visual Basic, Visual C#, and other Microsoft Windows programming languages

.NET Architecture

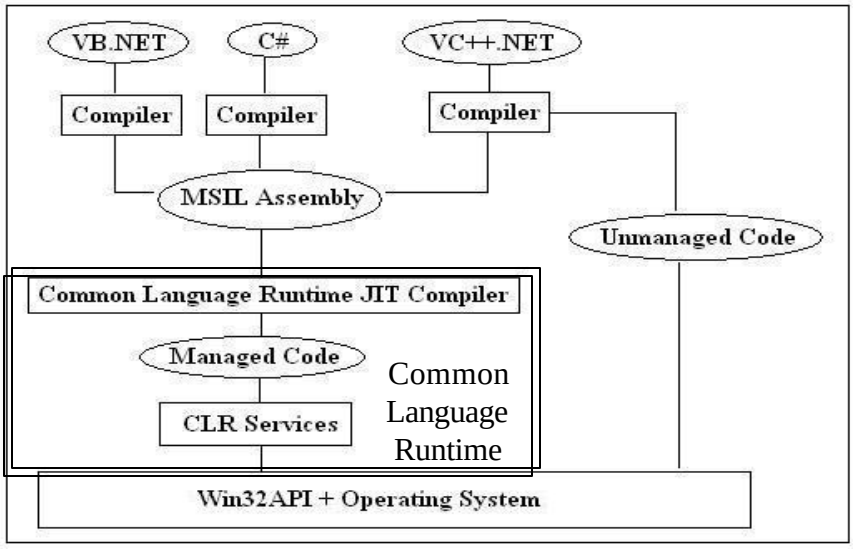


Visual Studio Capabilities

- Generate starter applications without writing code
- View a programming project in many different ways
- Edit source, header, and include files
- Build the application's user interface visually
- Build (compile and link) an application
- Debug an application while it runs
- Obtain online help
- Lots of others (Wizards)

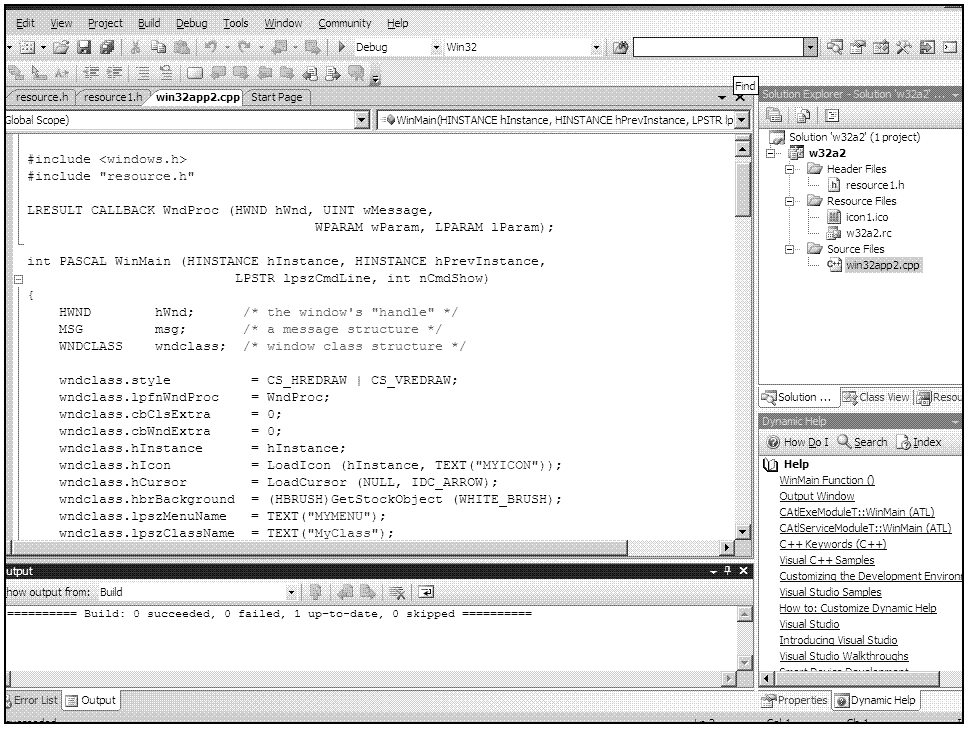


Compilation in the .NET Framework



Using Visual Studio .NET

- 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, C++, and other languages
  - ASP.NET Web Apps and Services
  - ADO.NET Data Base Apps
  - Others including OpenGL



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

An Introduction to Windows Programming Using VC++

- Two approaches:
  - Win32 API
    - Most basic
  - MFC
    - Encapsulates API functions into classes
    - For most apps, easiest to use

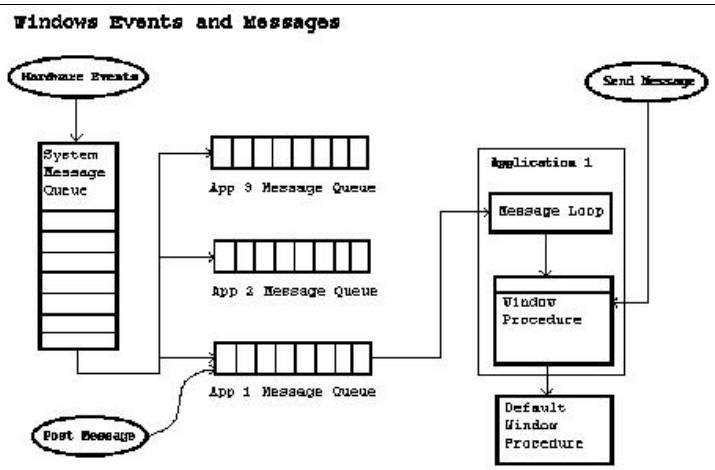
Win32 API Programming

- Additional notes at:

  http://www.cs.binghamton.edu/~reckert/360/class2a.htm
  http://www.cs.binghamton.edu/~reckert/360/class3a.htm

Windows Programming

- Event-driven paradigm
- Example: User clicks mouse over a program's window area (a mouse event)--
  - Windows decodes HW signals from mouse
  - figures out which window user has selected
  - sends a message to that window's program:
    - "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



Essential Parts of a Windows Program

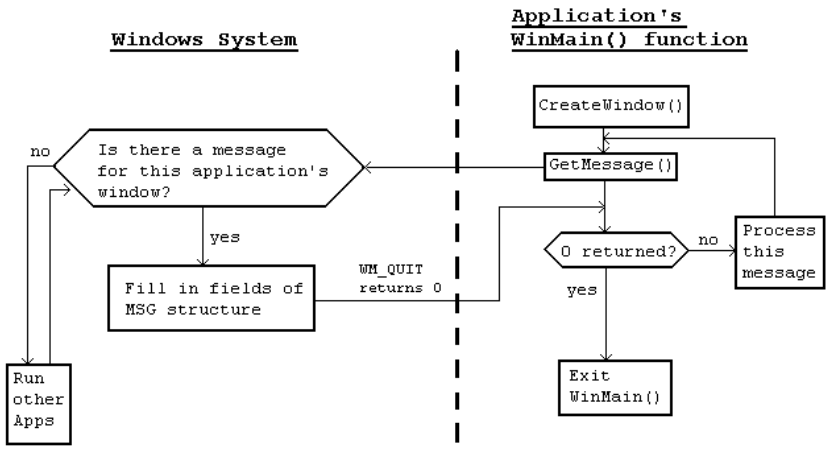
- I. The source program (.c/.cpp file):
  - A. WinMain() function
    - 0. declarations, initialization, etc.
    - 1. register window "class"
    - 2. create a window based on a registered "class"
    - 3. show window, make it update its client area
    - 4. the message loop
      - get messages from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

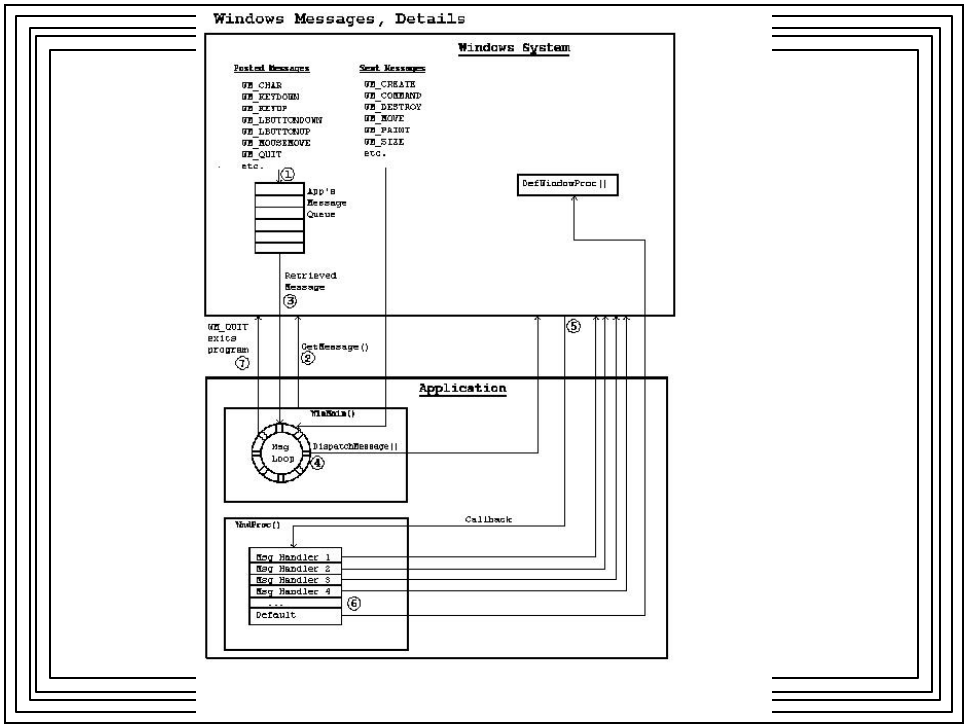
  - B. WndProc(): the message-processing function
    - a big switch/case statement
    - handles messages of interest

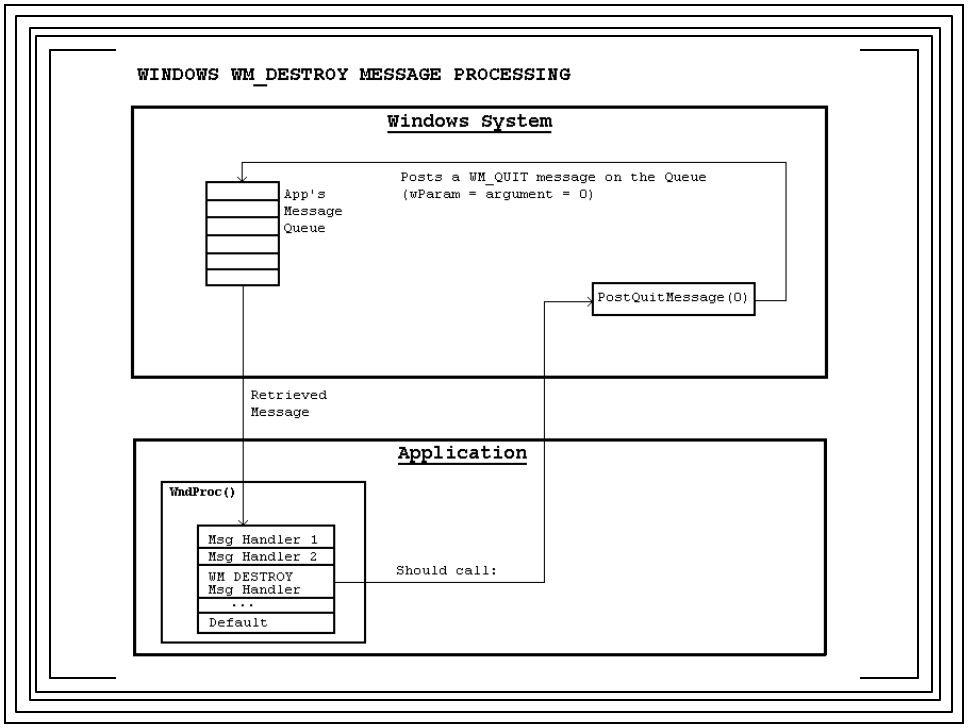
  - Under Win32 API, programmer must write WinMain() and the WndProc()
  - Under MFC, .NET Wizards do most of the work
    - WinMain() and WndProc() are buried in the framework
    - Write "message mapped handler functions" instead

II. The resource script (.rc file)

- Contains resource (Windows static) data
- Determine “look and feel” of the application
- 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
- Visual Studio can generate this file




Windows Messages, Details
Windows System
Posted Messages
WM_CHAR
WM_KEYDOWN
WM_KEYUP
WM_LBUTTONDOWN
WM_LBUTTONUP
WM_MOUSEMOVE
WM_QUIT
etc.
Sent Messages
WM_CREATE
WM_COMMAND
WM_DESTROY
WM_MOVE
WM_PAINT
WM_SIZE
etc.
App's Message Queue
DefWindowProc()
Retrieved Message
WM_QUIT exits program
GetMessage()
Application
WinMain()
Msg Loop
DispatchMessage()
WndProc()
Callback
Msg Handler 1
Msg Handler 2
Msg Handler 3
Msg Handler 4
...
Default


WINDOWS WM_DESTROY MESSAGE PROCESSING
Windows System
Posts a WM_QUIT message on the Queue
(wParam = argument = 0)
App's Message Queue
PostQuitMessage(0)
Retrieved Message
Application
WndProc()
Msg Handler 1
Msg Handler 2
WM_DESTROY Msg Handler
...
Default
Should call:

Some Other important messages

- WM_COMMAND--User clicked on menu item
  - LOWORD(wParam)=menu item ID
- WM_*BUTTONDOWN--left/right mouse button pressed
  - * = L, R, or M
  - lParam=x,y coordinates
- WM_MOUSEMOVE--mouse moved
  - lParam=x,y coordinates
- WM_CHAR--User pressed valid ANSI code character or keyboard key combination
  - wParam=ANSI code
- WM_PAINT--window was exposed, should be redrawn
- WM_KEYDOWN--keyboard key pressed
  - wParam=virtual key code

The Resource Script (.rc file)

- Resources--static data
- Example: a menu
- Defined in a script (.rc) file--

```
#include "resource.h"
  MYMENU MENU
  BEGIN
   MENUITEM "&Circle",      ID_CIRCLE
   MENUITEM "&Quit",        ID_QUIT
  END
```

The Resource header (.h file)

```
// resource.h
#define ID_CIRCLE       40006
#define ID_QUIT         40007
```

- Must #include in .CPP and .RC files
- Can use Visual Studio's resource editors to prepare .rc and .h files visually
  - ID numbers generated automatically

Text and Graphics Output

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

Device Independent Graphics Interface

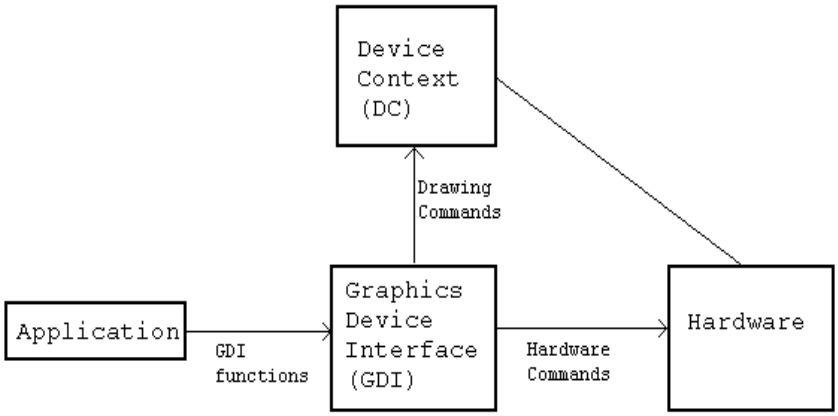
- Windows programs don't access hardware devices directly
- Make calls to generic drawing functions within the Windows 'Graphics Device Interface' (GDI) -- a DLL
- The GDI translates these into HW commands

Device Context

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

The DC and the GDI

Windows Drawing Using the GDI and the DC



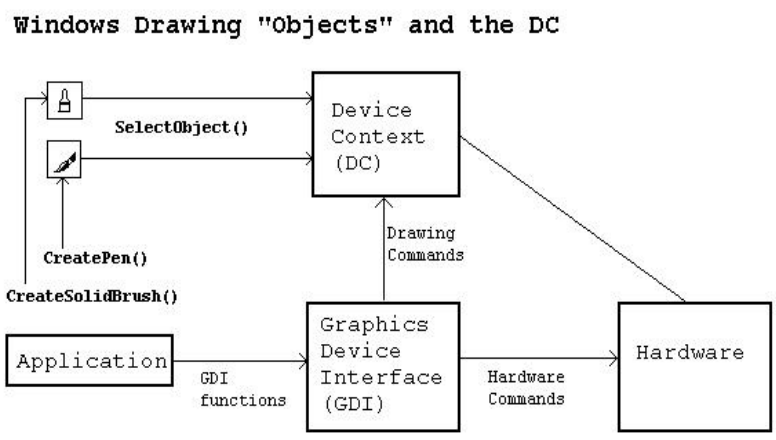
Some GDI Attributes

ATTRIBUTE	DEFAULT	FUNCTION
Background color	white	SetBkColor()
Background mode	OPAQUE	SetBkMode()
Current Position	(0,0)	MoveTo()
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



Colors in Windows

- Uses 4-byte numbers to represent colors
- Simplest method--direct color:
  - typedef DWORD COLORREF;

```
-------------------------------------------------------------
| 0 | Blue (0-255) | Green (0-255) | Red (0-255) |
-------------------------------------------------------------
```

  - MSB=0:
    - ==> RGB direct color used (default)
    - Other bytes specify R, G, B intensities

RGB() Macro

- Specify Red, Green, Blue intensities
- RGB() generates a COLORREF value
- Can be used in color-setting ftns), e.g.

```
COLORREF  cr;
cr = RGB (0,0,255);   /* blue */
```

- Example usage in a program

```
SetTextColor(RGB(255,0,0)); //red text
SetBkColor(RGB(0,0,255)); //blue bkgnd
```

A Typical Sequence With Drawing Objects:

```
HPEN    hOldP, hNewP;
HDC     hDC;
hDC = GetDC(hWnd);
hNewP = CreatePen(PS_SOLID, 3, RGB(0,0,255));
hOldP = (HPEN)SelectObject(hDC, hNewP);
// NOW DO SOME DRAWING WITH THE NEW PEN
SelectObject(hDC,hOldP); //displace pen from DC
DeleteObject(hNewP); //now can be deleted
ReleaseDC(hWnd,hDC);
```

Some GDI Drawing Primitives

- Arc(hDC,x1,y1,x2,y2,xStart,yStart,xEnd,yEnd);
- Ellipse (hDC,x1,y1,x2,y2);
- MovetoEx (hDC,x1,y1,p.Point);
- LineTo (hDC,x1,y1);
- Polygon (hDC,points_array,nCount);
- Polyline (hDC,points_array,nCount);
- Rectangle (hDC,x1,y1,x2,y2);
- SetPixel (hDC,x1,y1,colorref);
- Many more (see on-line help)

An Example Win32 API Program

- Has Menu items to:
  - Draw a circle
  - Quit
- Types an “L” at cursor position when user left clicks the mouse
- Has an icon
- On CS-460 Sample Programs web page
  http://www.cs.binghamton.edu/~reckert/460/api.html

Creating the Example Win32 API Application with Visual Studio

1. Startup

- click ‘Start’ on Task Bar – ‘All Programs’
- ‘Microsoft Visual Studio .NET 2005’ | ‘Microsoft Visual Studio .NET 2005’

2. Create 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. api-ex) & click ‘OK’
  - All solution files will be in a new directory with that name

3. 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’

4. Insert 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., api-ex)
- Add the source file to the project:
  - Choose ‘Project’ | ‘Add Existing Item’ from menu
  - Click on the file you saved (e.g. api-ex.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

5. Create an Icon Resource (and the .rc resource script file)

- Select 'Project | Add Resource | Icon | New'
  - Brings up icon editor
- Draw desired icon
- Click on IDI_ICON1 in "Resource View" to bring up the "Properties" window and change the icon ID to "MYICON"
  - Don't forget the quote marks
- Give a name to .ico file (or leave the default name)

6. Add a Menu

Select ‘Project | Add Resource | Menu | New’

- Brings up the menu editor
  - Type the caption: &Circle in the “Type Here” rectangle
  - In resulting "Properties" box, Select “False” for “Pop-up”
  - Click on the resulting Circle menu item to bring up the “Properties” box again.
  - Note the default ID of ID_CIRCLE
- Click on the next rectangle over in the menu editor
  - Repeat the above steps using caption: &Quit
  - Keep the default IDs
- Click on “IDI_MENU1” in “Resource View” to bring up “Properties” window; change menu ID to “MYMENU”

7. Build the Project

- ‘Build’ | ‘Build Solution’ from menu
- Project will be compiled/linked
- Messages/errors will appear in Output Window

8. Run the Program:

- ‘Debug’ | ‘Start’ from menu
  - Shortcut key: F5
- Or ‘Debug’ | ‘Start Without Debugging’ from menu
- Shortcut key: Ctrl-F5

Copy Project to a CD

- Copy the entire topmost directory to your diskette or CD-ROM
- If using a public computer, delete the workspace directory from the hard disk