CS-360/580H
GUI & Windows Programming

Dr. Richard R. Eckert
Computer Science Department
SUNY Binghamton
Fall, 2009

CS-360: MWF, 1:10-2:10 P.M., SL-210
CS-580H: TR, 8:30-9:55 A.M., SW-325

Course Information

- Office: EB-N6
- Phone: 777-4365
- Office Hours: W, R 1:30-2:30 P.M.
- Email: reckert@binghamton.edu
- http://www.cs.binghamton.edu/~reckert/
  - CS-360 link for syllabus, notes, programs, assignments, etc.
- Class Listserv:
  - CS360-L@listserv.binghamton.edu
- CS-360 TA: Elif Dede
- CS-580H TA: Yibo Sun

Course Prerequisites

- CS-240, Data Structures
- Some knowledge of C or C++

Text Book Information

- Required:
  - Deitel, et.al., "Visual C# 2005: How to Program", 2nd Edition, PH/Pearson, 2005, ISBN 0-13-152523-9
- Recommended:
  - Kate Gregory, "Special Edition Visual C++ .NET", Que, 2002, ISBN 0-7887-2466-9
- Many Books on Reserve
  - See Reserve List in Course Syllabus

Software

- Microsoft Visual Studio 2005 or 2008 Professional Edition
  - 2008 available at most University public computer facilities
  - Get your own copy of either
    - From Microsoft Academic Alliance
      - Available now to all registered BU students
      - https://msdn04.e-academy.com/elms/Security/PasswordReminder.aspx?campus=binghamton_watson
- Smaller .NET 2005 or 2008 “Express Editions” free from Microsoft:
  - Visual C++ 2005/08, Visual C# 2005/08, SQL Server 2005/08 Visual Web Developer 2005/08 Express Editions
  - http://msdn.microsoft.com/vstudio/express/

Evaluation

- Programming Assignments 40%
- Term Examinations (2) 40%
- Quizzes (360)/Paper(580H) 10%
- Final Project 10%

Policies

- **Assignments**
  - Individual
  - Due on due date, but can be turned in to CS-360/CS-580H drop drawer in filing cabinets outside CS Department any time that day or night
  - 5% off for every day late
    - Weekends and holidays not included
  - No assignments accepted more than one week late
- **Originality**
  - **Any non-original work (work found to be copied) will be grounds for an F in the course**
  - Individual assignments
    - **Students do NOT work in teams**

Course Schedule (weekly)

1. Intro to GUIs & Windows Programming, Using Visual Studio
2. Win32 API Programming
3. MFC Programming: App/Window & Doc/View Approaches
4. Visual Studio .NET & C#, Classes, Windows Forms, Events, Essential Structures
5. Graphics, Animation, Timers, DateTime
6. Mouse, Images, Bitmaps
7. Text, Fonts, Keyboard, Printing
8. Pages & Transformations, Menus

Course Schedule (continued)

9. Controls: Buttons, Labels, TextBoxes, Scrollbars, Listboxes, etc.
10. Dialog Boxes, Common Dialog Boxes, File/Stream I/O
11. Clipboard, Multimedia
12. Network Programming, TCP/IP Sockets
13. Data Bases and ADO.NET, LINQ
14. XML, Web Forms, Web Controls, ASP.NET; WPF & WCF
15. ASP.NET Web Services
16. Other Windowing Systems: X Windows, Java AWT/Swing

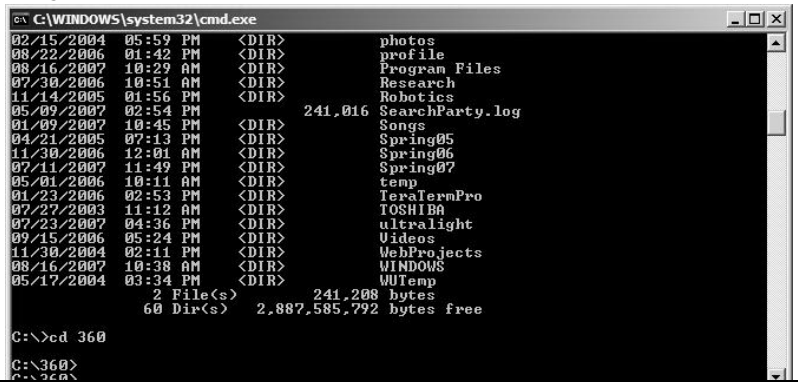
Introduction To GUIs and Windows Programming

User Interface

- Connection between the computer and the user
- Two types:
  - Command Line
  - GUI: Graphical (Visual)

Command Line Interfaces

- User types commands, must remember valid commands
- Results Scroll by
- Text-based
- “Interactive” but hard to use
- Only kind of interface available until 1970s



Visual (Graphical) Interfaces

- Show Graphical Objects on screen
  - e.g., images, icons, buttons, scroll bars
- User interacts using pointing device
- Intuitive
  - Objects can be dragged, buttons pushed, etc....
- Better way of using screen space
  - Panes can overlap
  - Underlying panes can be brought to forefront
  - Desktop metaphor (like papers on a desk)

Graphical Interfaces, Continued

- Use graphics to organize user workspace
  - Visually rich way of conveying information
- Environment allows many tasks to be performed simultaneously
- Different tasks share screen space

Main Feature of GUIs

- The Window
  - Rectangular area of screen onto which a program draws text and graphics
  - User interacts with program that created the window using a pointer device to select objects inside
  - Some window components:
    - border, title bar, client area, menu bar, tool bars, scroll bars, max/min/close buttons, etc.

History of GUIs

- DARPA SRI (late 60s)
- Xerox PARC Alto (early 70s)
- Microcomputers (late 70s to present)
  - PC (DOS command line)
  - Apple Lisa, Macintosh
    - First real microcomputer GUI
  - Microsoft Windows
    - Many versions
    - We'll emphasize GUI Programming for Microsoft Windows in this course

Other GUI-Windowing Systems

- Sun Microsystems: Java
  - AWT
  - Swing
  - Platform independent
  - JDK is free
- The X Window System
  - Developed at MIT, late 1980s
  - Networked graphics programming interface
  - Independent of machine architecture/OS (but mostly used under UNIX/LINUX)

Windowing Systems Features

- Consistent user interface
  - Information displayed within a window
  - Menus to initiate program functions
  - Make use of child window "controls"
  - Point and click user interaction with window
- All programs have same look and feel
- Same built-in logic to:
  - draw text/graphics
  - display menus
  - receive user input
    - controls, dialog boxes, use of mouse

Multitasking

- Many programs run “simultaneously”
- Each program creates/controls its own window
- User interacts with program via its window
- User can switch between programs by switching between windows

Windows Multitasking Features

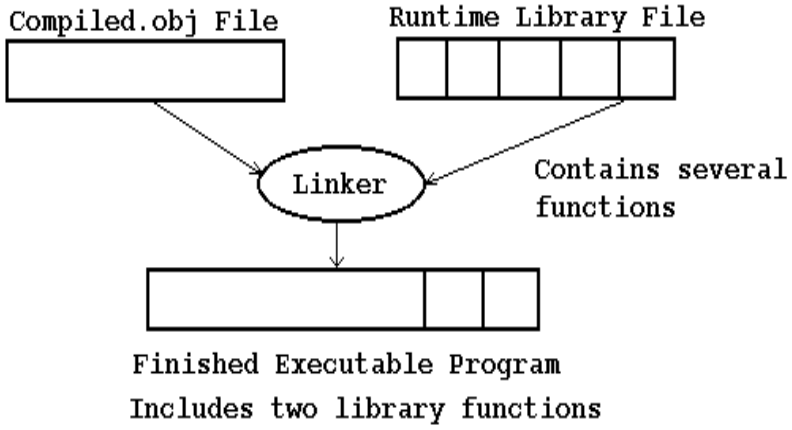
- Cooperative (Windows 3.xx)
  - Programs must give up control so others can run
  - Programs coexist with other programs
- Preemptive (Windows NT, 95, 98, XP, 2000, 2003, Vista)
  - Thread-based: System timer allocates time slices to running program threads
- Under both systems, code is moved or swapped into and out of memory as needed

Windows Memory Management

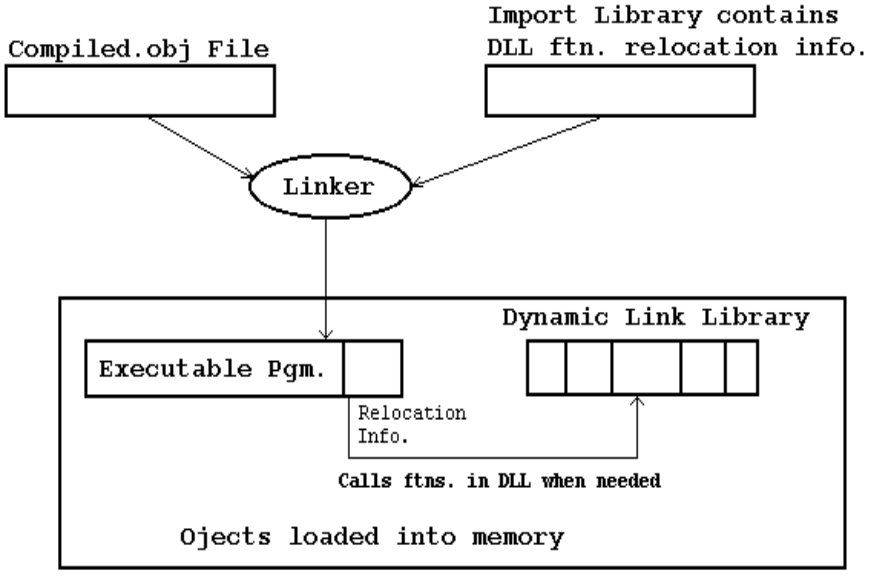
- Older versions: 16-bit, segmented memory
  - Dictated by processor architecture
  - Hard to program
    - 64 kilobyte memory segment limitation
- Newer versions: 32/64-bit, flat memory model
  - Easier to program
  - Each process sees 4 Gigabytes of virtual memory
- As old programs terminate, new ones start
  - Code swapped into and out of memory
  - Windows OS does this automatically
- Programs can share code located in other files (Dynamic Linking)

Static vs. Dynamic Linking

- Static Linking
  - Code incorporated into executable at link time
- Dynamic Linking
  - Code is put into separate modules (DLLs)
  - These are loaded at run time as needed



Static Linking



Dynamic Linking

Pros/Cons of Dynamic Linking

- Smaller programs (code is not in program)
- DLL can be used by many programs with no memory penalty
  - Only loaded once!
- Disadvantages:
  - DLL must be present at run time ==> no standalone programs
  - "DLL Hell" when new DLL versions come out
- Most of the Windows OS is implemented as DLLs

Device Independent Graphics

- Windows programs don't access hardware devices directly
- Make calls to generic functions within the Windows 'Graphics Device Interface' (GDI, GDI+, or WPF)
- The GDI/GDI+/WPF translates these into HW commands

Windows API

- Application Program Interface
- The programmer's interface between an application and the Windows OS
- A library of functions Windows programs can call
- Several versions
  - Win32 API most fundamental
    - (32 bit apps for Windows NT/95/98/XP/2000/2003/Vista)

Classical Win32 API Windows Programming

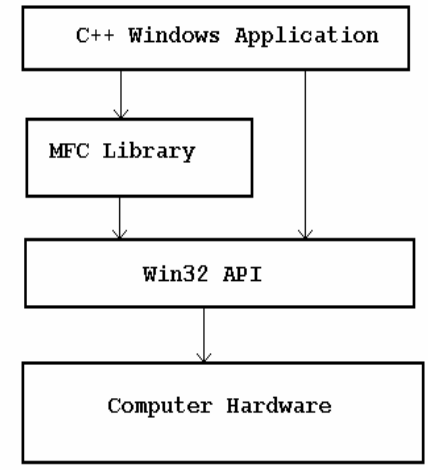
- Use C to access raw API functions directly
- No C++ class library wrappers to hide API
  - But C++ compiler can be used
- Hard way to go, but most basic
- Faster executables
- Provides understanding of how Windows OS and application program interact
- Establishes a firm foundation for MFC and .NET programming

Class-based Windows Programming

- "Microsoft Foundation Class" Library (MFC)
- Microsoft .NET "Framework Class Library" (FCL)
- Borland's "Object Window Library" (OWL)
- Characteristics:
  - Encapsulate the API functions into classes
  - Provide a logical framework for building Windows applications
  - Object Orientation means reusable code

MFC Library

- Microsoft's first C++ Interface to Win32 API
- Most basic object oriented approach to Windows programming
- Some 200 classes
- API functions encapsulated in the MFC
- Classes derived from MFC do grunt work
- Just add data/functions to customize
  - Or derive your own classes from MFC classes
- Provides a uniform application framework
- Fast executables



The Relationship between Windows MFC and Win32 API Programming

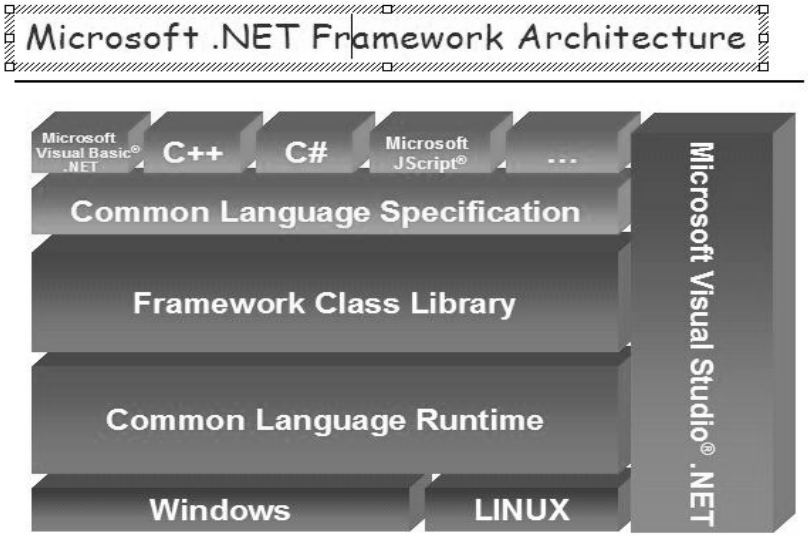
Microsoft .NET Framework

- A software system that addresses new SW requirements
  - 1. Windows Forms: standalone Windows applications
  - 2. Windows distributed applications over the Internet
    - ASP.NET
    - ADO.NET
    - Multi-tier applications
  - Language Independent (programs can be written in multiple languages)
  - Platform Independent Architecture
  - New program development process
    - Object oriented
    - Provides increased productivity
    - New vision for using the Internet in software development
  - New security and reliability features

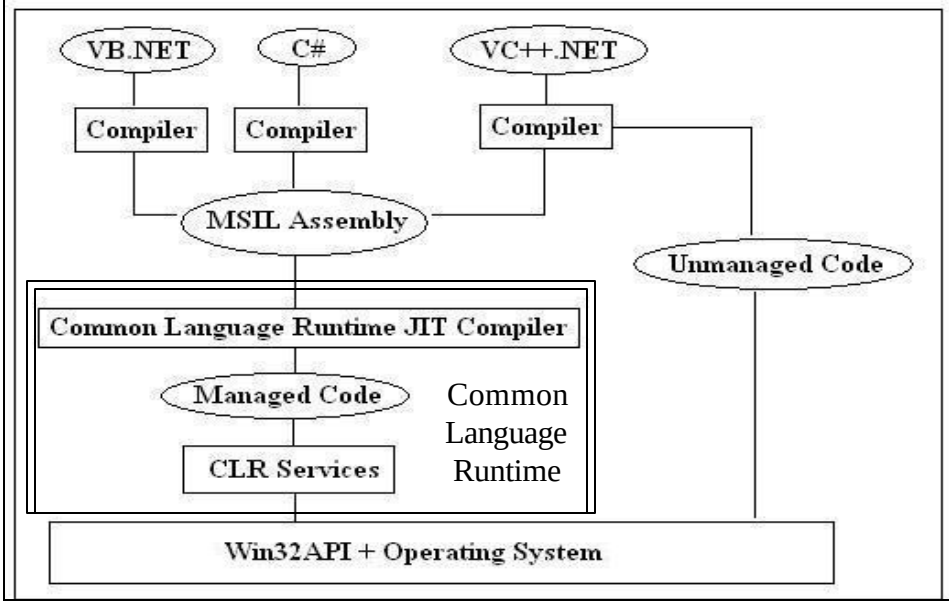
Components of .NET Framework

- Language compilers
- The .NET Framework Class Library (FCL)
  - Organized into "namespaces"
    - like packages in Java
  - Handle things like: I/O (simple & file), Windows Forms, Web Forms, Windows Controls, User Interfaces, Drawing, Threading, Exceptions, Networking, Web Services, Data Bases (ADO), XML, ASP, Security, Collections, … lots of others
- Common Type System (CTS)
- Common Language Specification (CLS)
- Common Language Runtime (CLR)

.NET Architecture



Compilation in the .NET Framework



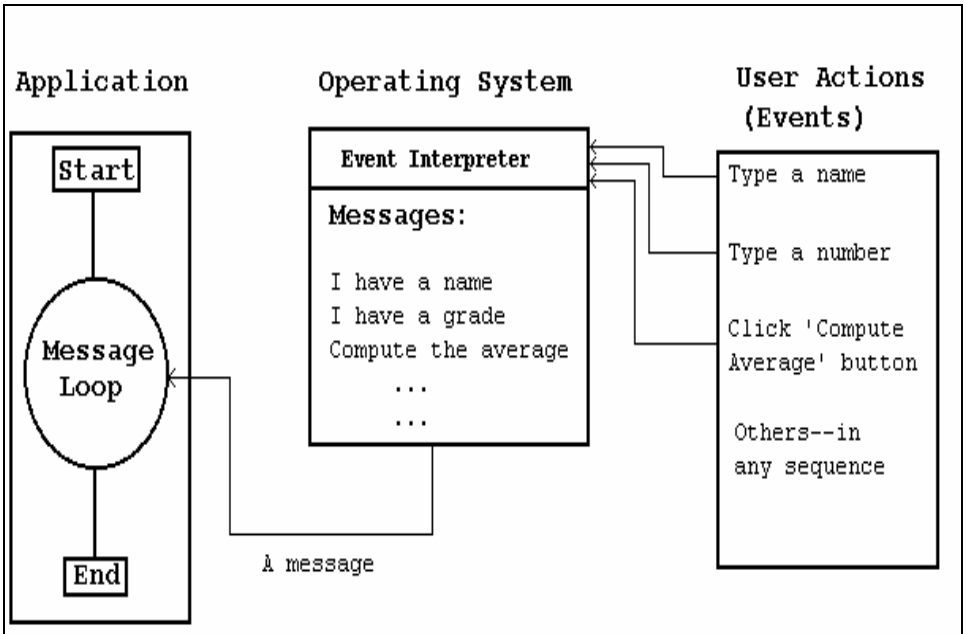
Sequential Programming versus Event-driven Programming

Sequential Programming (Console Applications)

- Standard programming--program solicits input (polling loop)
- Approach follows a structured sequence of events
- Example--averaging grades:
  - Input name
  - Input first grade
  - Input second grade
  - Input third grade, etc.
  - Calculate average
  - Output average

Event-Driven Programming

- Designed to avoid limitations of sequential, procedure-driven methodologies
- OS processes user actions (events) as they happen: non-sequential
- Program doesn't solicit input
- OS detects an event has happened (e.g.., there's input) and sends a message to the program
- Program then acts on the message
- Messages can occur in any order



The Event-Driven Programming Paradigm

Sequential vs. Event-Driven Programming

- Standard Sequential programming:
  - Program does something & user responds
  - Program controls user
    - the tail wags the dog
- Event-Driven Programming:
  - User does something and program responds
  - User can act at any time
  - User controls program
    - the dog wags the tail
  - OS really is in control (coordinates message flow to different applications)
  - Good for apps with lots of user intervention