CS-360
GUI & Windows Programming

Dr. Richard R. Eckert
Computer Science Department
SUNY Binghamton
Fall, 2001

MWF, 2:20-3:20 P.M.
AA-G07

Course Information

- Office: EB-N6
- Phone: 777-4365
- Office Hours: TBA
- 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
- TA Information: TBA

Course Prerequisites

- CS-220, Computer Organization and Assembling Language Programming
- CS-240, Data Structures
- Some knowledge of C or C++ helpful
  - Not essential

Text Book Information

- Required:
  - Ivor Horton, "Beginning Visual C++ 6 (A Complete Visual C++ Package)," Wrox Press, 1998, ISBN 1-86100-196-7. (For MFC)
- Recommended:
  - Charles Petzold, "Programming Windows," Fifth Edition, Microsoft Press, 1999, ISBN 1-57231-995-X. (For Win32 API)
  - Paul E. Kimball, "The X Tookit Cookbook," Prentice Hall PTR, 1995, ISBN 0-13-973132-6. (For X-Windows Programming)
- Many Books on Reserve

Evaluation

- Programming Assignments 45%
- Term Examinations (2) 40%
- Final Project 15%

Policies

- Assignments
  - Individual
  - Due on due date, but can be turned in to CS-360 drop drawer 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 work found to be copied will be grounds for an F in the course

Course Schedule (weekly)

1. Intro to GUIs & Windows Programming
2. Using Visual Studio,Win32 API Programming
3. MFC Programming: App/Window Approach
4. MFC Programming: Doc/View Approach
5. Graphics, Animation, Bitmaps, Timers
6. Windows Controls, Dialog Boxes
7. Printing, Mapping modes, Serialization, File I/O
8. Clipboard, DLLs
9. Multimedia, Data Bases

Course Schedule (continued)

10. ActiveX Controls
11. Multitasking & Multithreading
12. Network & Web Programming
13. Introduction To Visual Basic
14. The X Window System
15. X Toolkit Intrinsics, OSF/Motif Toolkit

Introduction To GUIs and Windows Programming

User Interfaces

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

Command Line Interfaces

- User types commands ==> must remember
- Results Scroll by
- Text-based
- "Interactive" but hard to use
- No direct interaction between user and screen

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)
    - Well, not exactly!

Graphical Interfaces, Continued

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

Main Feature of GUIs:

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

Brief History of GUIs

- 1968: ARPA-funded Stanford Research Center (Doug Engelbart)
- First windows (screen sliced up into overlapping panes)
- Only textual info
- Underlying windows could be popped to the top
- Selection done with light pen
- Invented the mouse

Xerox PARC--Alto Computer

- 1970
- First GUI
- Cursor tracked position of mouse
- WYSIWYG
- Windows with precise text
- Displayed more than just text
- First interactive painting program
- Technology “acquired” by Apple

Recent History (PCs)

- 1983: Apple Lisa (failure)
- 1984: Apple Macintosh--standard for GUIs
- 1985: Microsoft releases Windows 1.0
  - Difficult to program
  - Prone to crashing
  - Needed hardware not yet available
- 1987: Windows 2.0 (still real mode only)
- 1988: Windows/386 (Virtual 86 mode on 386==>multiple DOS sessions in windows)

Recent History (Microsoft)

- 1990: Windows 3.0
  - 80x86 protected mode, up to 16 Meg memory, cooperative multitasking
- 1992: Windows 3.1, Windows for Workgroups 3.11
  - TrueType fonts, multimedia, protected mode only; Networking
- 1993: Windows NT
  - 32-bit flat memory space, 16 MB, thread-based pre-emptive multitasking, separate from DOS, multi-platform, networking, secure)

Recent History (Microsoft)

- 1995: Windows 95
  - Runs on 4 Meg, long file names, plug and play, new controls, new desktop/window style
  - Hybrid 16/32 bit OS, depends on DOS, lacks security of NT, no portability to RISC
- 1998: Windows 98
  - Integrated Web functionality
- 2000: Windows 2000
  - Like 98, more stable, independent of DOS

Other GUI-Windowing Systems

- IBM OS/2: Presentation Manager
- Sun Microsystems: Java
  - AWT
  - Swing
  - Platform independent
  - JDK is free
- The X Window System
  - Developed at MIT
  - Networked graphics programming interface
  - Independent of machine architecture/OS (but most used under UNIX)

Course Content

- Microsoft Windows Visual C++
  - Using Microsoft Developer Studio (Visual Studio 97)
  - Win32 API Programming
  - MFC Programming
  - Visual Basic
  - X-Windows Programming
  - Example programs and notes online at:
    - http://www.cs.binghamton.edu/~reckert/
    - "CS-360" link

Win32 API Programming

- Event-Driven Programming (Messages)
- Menus and other Resources
- Text and Graphics
- Mouse and Keyboard
- Bitmaps, Animation, Timers
- Child Window Controls
- Child and Popup Windows
- Dialog Boxes
- The Clipboard

MFC Programming

- The MFC Class Hierarchy
- The Application/Window Approach
- The Document/View Approach
- Using "AppWizard" & "ClassWizard"
- Drawing, Menus, & Dialog Boxes with MFC
- File Handling and Printing
- Dialog-Based MFC Applications & Common Dialog Boxes
- DLLs; Windows Multimedia
- Working with data bases (ODBC)
- Multitasking and Multithreading
- OLE, ActiveX Controls
- Network Programming (TCP/IP)
- HTML-based Applications with MFC

Introduction to Windows Programming in Visual Basic

- A quick introduction

X-Windows Programming

- Client/Server Model
  - X Display Servers
- XLIB Programming
- Toolkits and Widgets
  - Xt Intrinsics
  - OSF/Motif

Windowing Systems Features

- Consistent user interface
  - Display within a window
  - Menus to initiate program functions
  - Make use of child window "controls":
    - predefined windows used with main program window
    - examples: buttons, scroll bars, edit controls, list boxes, drop-down list boxes, combo boxes
    - Dialog box--popup window containing several controls

Consistent User Interface, continued

- 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

- Every program acts like a RAM-resident popup
- Programs run "simultaneously"
- Each program occupies its own window
  - User interacts with program in its window
- User can switch between programs

Windows Multitasking Features

- Cooperative (Windows 3.xx)
  - Programs give up control so others can run
  - Programs coexist with other programs
- Preemptive (Windows NT, 95, 98)
  - 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 Object Orientation

- A window is handled like a C++ object
  - Has a user-defined type (Windows class)
  - Instances of class created at run time
  - Messages sent to windows affect their behavior

Windows Memory Management

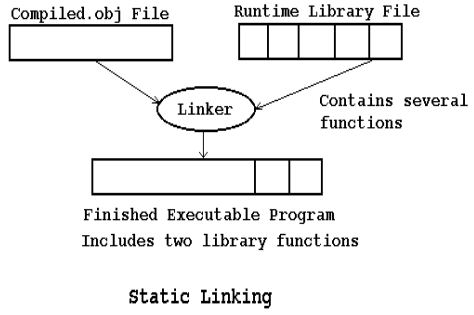
- Older versions: 16-bit, segmented memory
  - Dictated by processor architecture
  - Hard to program
- Newer versions: 32-bit, flat memory model
  - Easier to program
- As old programs terminate, new ones start
  - Code swapped into and out of memory
- Fragmentation can occur
- Windows must consolidate memory space
- Moves blocks of code/data continually

Memory Management, continued

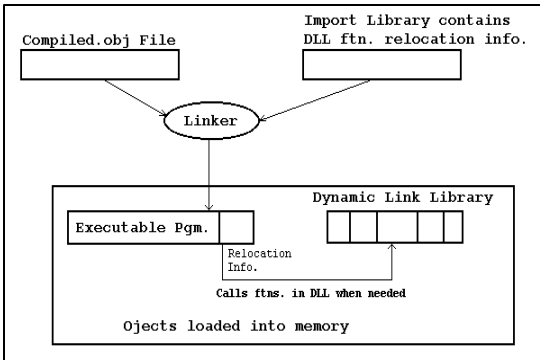
- Programs can share code located in other files (Dynamic linking)

Static vs. Dynamic Linking

- Static Linking
  - code incorporated into executable at link time
- Dynamic Linking
  - Code is put into separate modules
    - These are loaded at run time
  - Linker generates relocation information
    - Only that is put into executable
    - Smaller programs
  - DLL loaded when needed
  - Relocation info used to get DLL function code as needed



Static Linking



Dynamic Linking

Pros/Cons of Dynamic Linking

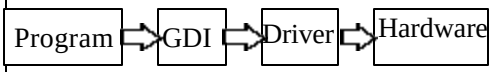
- Smaller programs (code is not in program)
- DLL can be used by many programs with no memory penalty
  - Only loaded once!
- Updates to DLLs don't require recompilation of programs using them
- Disadvantage--DLL must be present at run time ==> no standalone programs

Device Independent Graphics Interface

- Windows programs don't access hardware devices directly
- Make calls to generic functions within the Windows 'Graphics Device Interface' (GDI)
- 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

Windows API

- The interface between an application and Windows
- A library of functions Windows programs can call
- Several versions
  - Win16 (16 bit apps for Windows 3.xx)
  - Win32 (32 bit apps for Windows NT/95)
  - Win32s (patches Win16 to create 32 bit apps that run under Windows 3.xx)

Classical Win32 API Windows programming

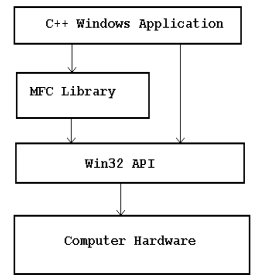
- Use C to access raw API functions directly
- No C++ class library wrappers to hide API
- Hard way to go, but most basic
- Faster executables
- Provides understanding of how Windows and application program interact
- Establishes a firm foundation for MFC programming
- We will try to do both

Class-based MFC Windows Programming

- Microsoft's MFC Library
- Borland's OWL Library
- Characteristics:
  - Encapsulate the API functions into classes
  - Provide a logical framework for building Windows applications

MFC Library

- Microsoft's C++ Interface to Windows API
- O-O 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 app
- Provides a uniform application framework



The Relationship between Windows MFC and Win32 API Programming

Microsoft Visual C++

- Developer Studio IDE
- 2 Windows application development systems
  - C programs using Win32 API
  - C++ programs using MFC
- Some Developer Studio IDE Components
  - Text/Resource Editors
  - C/C++, Resource Compilers
  - Linker
  - Debugger
  - Wizards
  - On-line Help

Some MFC Characteristics

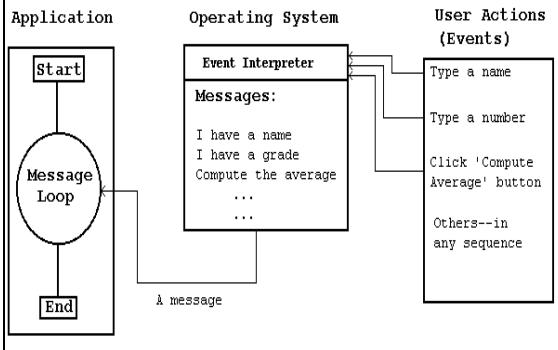
- Reusable code
- Smaller executables
- Faster program development
  - But a steep learning curve is required
  - And there is less flexibility
- Programs must be written in C++
- Require the use of classes==>
  - Programmer must know OOP

Sequential Programming (Console Apps)

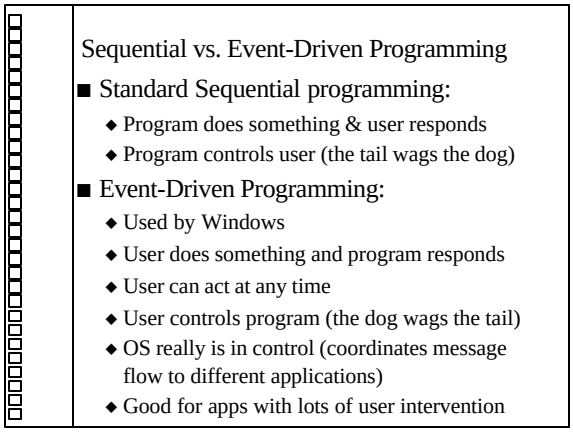
- Standard programming--program solicits input (polling loop)
- Approach follows a structured sequence of events
- Example--averaging grades:
  - Input name
  - Input first grade
  - Input second grade
  - Input third grade, etc.
  - Calculate average
  - Output average

Event-Driven Programming

- Designed to avoid limitations of sequential, procedure-driven methodologies
- Process 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:
  - Used by Windows
  - 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