CS3283

GUI Programming

Hugh Anderson
Department of Computing Science,
School of Computing
NUS
hugh@comp.nus.edu.sg

Preface

The FOLDOC[1] dictionary of computing defines a GUI as:

> *(GUI) The use of pictures rather than just words to represent the input and output of a program. A program with a GUI runs under some windowing system (e.g. The X Window System, Microsoft Windows, Acorn RISC OS, NEXTSTEP). The program displays certain icons, buttons, dialogue boxes etc. in its windows on the screen and the user controls it mainly by moving a pointer on the screen (typically controlled by a mouse) and selecting certain objects by pressing buttons on the mouse while the pointer is pointing at them.*
>
> *Though Apple Computer would like to claim they invented the GUI with their Macintosh operating system, the concept originated in the early 1970s at Xerox's PARC laboratory.*

The *official* NUS description of CS3283 is:

> *This module aims to teach the nuts and bolts of GUI programming. At the end of the course, students will acquire practical knowledge in Windows programming and techniques of programming interactive systems. Topics include Windows programming, Motif, Tcl/Tk programming.*

An interesting aspect of this course is that there is some emphasis on graphical visualization methods. These notes are just an expanded set of overheads - you are asked to read supporting papers, and maintain an active interest in GUI design and implementation. You may find the latest copy of the notes at

http://www.comp.nus.edu.sg/˜cs3283/ftp/cs3283.pdf

It is my intent in this course to give lots of examples of different GUIs - successful and otherwise, and to show the methods used to develop these interfaces.

[1]The Free-On-Line-Dictionary-Of-Computing is found at http://wombat.doc.ic.ac.uk/foldoc/index.html.

Assessment: The proposed assessment may be modified slightly if the need arises, but currently is as follows:

Assessment			**Weighting**	**Grade**
Assignments				35%
Ass1	Design	Group	10	
Ass2	Design, prototype	Individual	20	
Ass3	GUI implementation - Swing/MFC	Individual	40	
Ass4	GUI/Visualization implementation - Tk/Java3D	Group	30	
Tutorials				5%
Mid-term		Closed book		10%
Final Exam		Open Book		50%
Total marks				**100%**

Textbook: Unfortunately, no single textbook adequately covers the material presented in this course, so I am providing a set of notes, which may be supplemented by readings from:

1. The User Interface Concepts & Design, Lon Barfield, Addison-Wesley (1993)
2. The JFC Swing Tutorial - A Guide to Constructing GUIs, Kathy Walrath & Mary Campione, Addison-Wesley (1999)
3. Tcl and the Tk Toolkit, John K. Ousterhout, Addison-Wesley (1994)

Tools: You may wish to download and install the following toolkits, copies of which are found at the course web site at http://www.comp.nus.edu.sg/˜cs3283/ftp.

1. JDK Version 1.3
2. The Cygwin development system, including Tcl/Tk and debuggers
3. The Netbeans IDE, the glade GUI builder, and so on

Topics to be covered: During the course, we cover

- Fundamental GUI concepts (1 lecture)
- Design and programming techniques (3 lectures)
- Cross platform GUI development using Swing and Tcl/Tk (6 lectures)
- Visualization techniques (2 lectures)

Enjoy the course!

Contents

Chapter 1

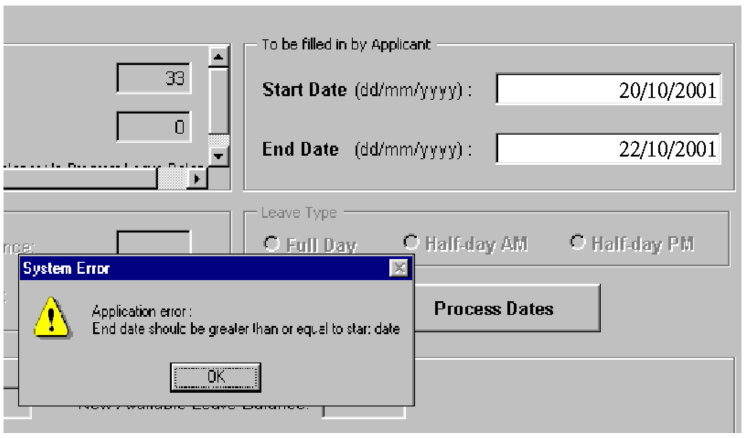
GUI concepts

The user interface for software has changed over the years. Early user interfaces were text based, and normally had a fairly simple interrogative style - prompting the user to provide needed information in a fixed order. The modern GUI provides for complex interaction between the user and the application, and often relies on shared concepts or metaphors.

GUI programming is about the conceptualization, design and implementation of that part of a software application which is concerned with user interaction.

1.1 How not to do GUI

I have decided to start with an example from here at NUS - the leave system for staff in the department. Notice the dates I have entered, and the error message[1]:


To be filled in by Applicant

Start Date (dd/mm/yyyy) : 20/10/2001

End Date (dd/mm/yyyy) : 22/10/2001

System Error

Application error :
End date should be greater than or equal to start date

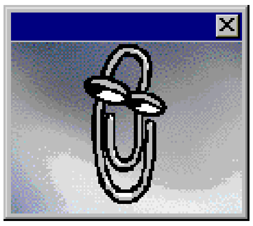
The question is ... what did I do next?

[1]It says: "**End date should be greater than or equal to start date**".

What do we learn from this? I think two points stand out:

1. Try out your applications before delivering them.
2. Ensure that error messages are precise, and indicate the next step.

During this course, I hope to give useful examples of interfaces and techniques - but equally important is to see these poor examples - *how-not-to-do* things. How about this one:



Often a single poor example can remind you of things to avoid - a picture of a poor GUI reminding you of a whole range of things.

1.2 General rules of GUI

A key point to understand as we begin our investigation of GUI development is that effective GUIs owe more to effective psychology than to effective programming. A graphical user interface is not just windows, icons and so on - it also includes an abstract view - not visible, but understood by the user. A successful GUI will have no clash between this view of the user interface and the more concrete one involving icons, buttons and so on. The term ergonomics might be used here - we hope for a correlation between the physical and conceptual ergonomics.

Another key point is that humans are not equipped to handle multiple things at one time, and this leads us to try to keep interfaces simple and uncluttered.

Humans are particularly good at navigating systems which have some analogy to things they know - for example the use of the desktop metaphor is well established and works well in most cultures. Icons are also useful, but shouldn't be abused.

To summarize:

1. Ensure correlation between What-u-c and What-u-think
2. KISS
3. Analogy, metaphor and icons

Always remember to include the U in GUI.

1.2.1 Do’s and don’ts

The following tips are gleaned from various sources, and are provided to help you know what you *should* do with your GUIs:

- **Do follow standards** - for example **quit** is the bottom item on the leftmost menu and so on. In general people prefer new applications to follow established standards.
- **Do be predictable and responsive** - events should have an immediate response, and should operate in well understood and predictable fashion.
- **Do be flexible** - the interface should admit a wide variation of allowable sequences, rather than forcing a particular ordering. An undo facility is often helpful here.
- **Don’t forget the user** - it is important to consider the pyschological, physical, and social attributes of people when designing user interfaces for them.
- **Don’t forget the machine/environment** - but watch for nasty implementation details showing up in user interfaces.
- **Dont assume things** - like “*the user will know how to ...*” - provide informative help and clear actions.

1.3 Types of applications

Not all applications benefit from a GUI. Consider the area of embedded control systems (such as lift controllers, washing machine controllers) - these systems certainly have a user interface which requires careful design, but not necessarily a GUI. In this course, we will not consider this sort of interface.

However, there are many application areas that do benefit from a GUI:

- **Immersive applications:** - games, medical imaging, avatar based CSCW and so on.
- **Office and business applications:** - for use by anyone.
- **Interactive control systems:** - flight systems, remote control and so on.

In addition, a newer application area involves the use of visualization to examine large data sets:

- **Data mining:** - delving into some set of data.

Finally, there is the use of GUI in WAP enabled devices and on PDAs. This is a specialist topic, which will not be covered in this course.

1.4 Native environments/platforms

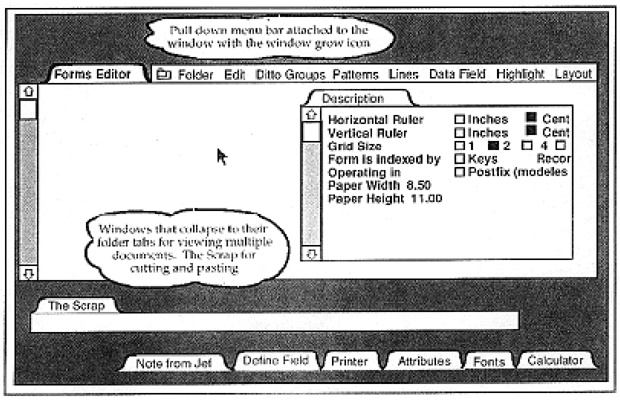
Early user interfaces were primitive things which barely disguised the underlying machines - nearly always text-based, and fixed in operation.

Contrary to popular public opinion, “Windows” is not the only GUI window system. The Macintosh system, and the UNIX X window system both predate Microsoft’s GUI system, and each have interesting features that have yet to be added to “Windows”. For example the UNIX X window system allows a clear separation between the display and the processing, whereas “Windows” display and processing must be on the same machine.

By far the most common GUI windowed environment on the desktop is the one found in Win95/98, and our programming API for it is known as Win32.

1.4.1 MacOS

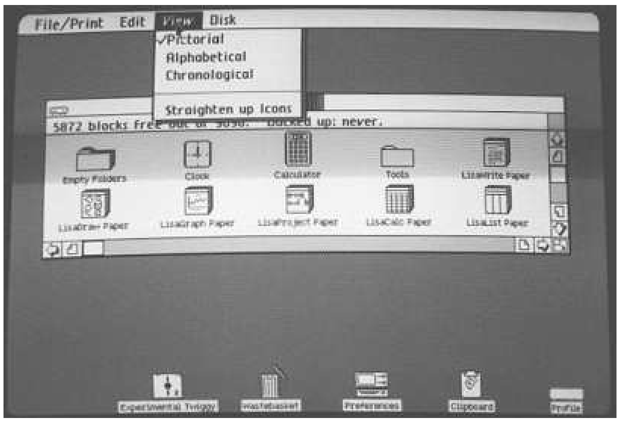
It is interesting to see the development of the Macintosh window system from an early prototype in 1979 through to its current incarnation - MacOSX.


Pull down menu bar attached to the window with the window grow icon

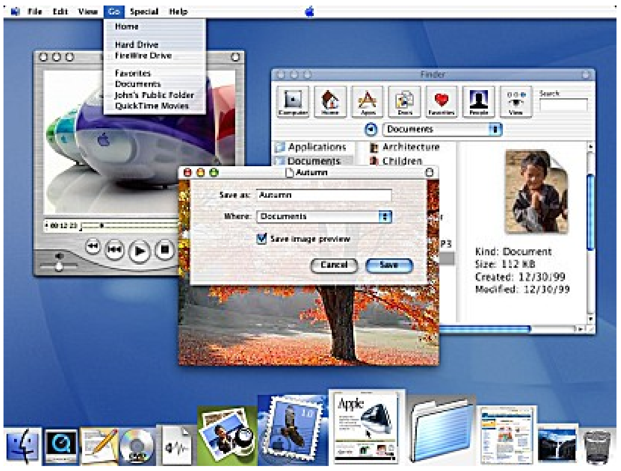
Windows that collapse to their folder tabs for viewing multiple documents. The Scrap for cutting and pasting

Horizontal Ruler ☐ Inches ■ Cent
Vertical Ruler ☐ Inches ■ Cent
Grid Size ☐ 1 ■ 2 ☐ 4 ☐
Form Is indexed by ☐ Keys Recor
Operating in ☐ Postfix (modeles
Paper Width 8.50
Paper Height 11.00

Note that this first prototype used folder tabs in a way that is no longer done, and did not appear to use icons. Within a year, the first commercial Macintosh system had the following interface:



Macintosh operating systems have a single *user*, single *display* orientation, although the latest version of the OS is built on top of a multi-user (UNIX) kernel. The latest version has some interesting features - most notable is the application docking bar *magnification* feature:



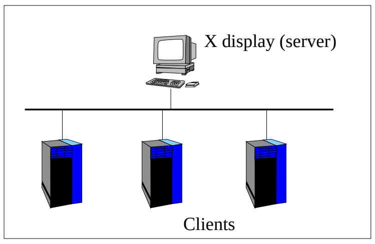
Save as: Autumn
Where: Documents
Save image preview

Kind: Document
Size: 112 KB
Created: 12/30/99
Modified: 12/30/99

1.4.2 X

The X window system[2] is a sophisticated and well developed system which allows for multimedia software and hardware components to be distributed around a network. At its simplest level, it allows a program and its display to be on different computers.

The architectural view of X is a little peculiar. The designers view the display as central to the system, and the software running on the display is called the X-server:



From the diagram, we can easily identify three essential components of X:

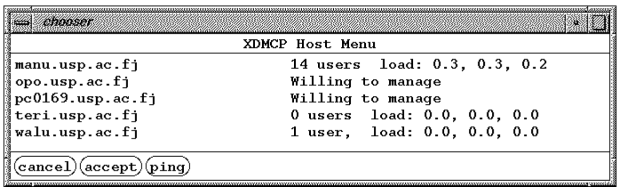
1. **The X server** - providing the high resolution graphical display(s[3]), keyboard and mouse.
2. **The X protocol** - providing standard communication methods between the distributed software components.
3. **X clients** - programs with graphical display output, running on (perhaps) many machines.

[2]The system is called X, or the X window system. UNIX weenies insist that it is **not** called X-windows!
[3]Mechanism, not policy!

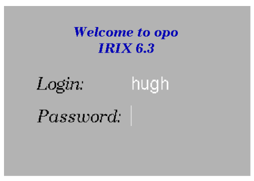
There are two other components:

- **The Window manager(s)** - providing decorations around each window.
- **The Display manager(s)** - providing access to the system.

The display manager controls access to displays. The diagram shows a simple display manager allowing selection of one of a number of hosts. When you select a host, you are presented with a login window for that particular host.

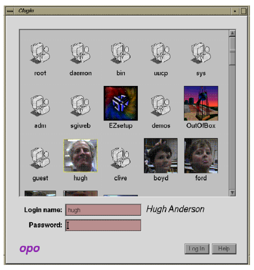


```
                    XDMCP Host Menu
manu.usp.ac.fj              14 users  load: 0.3, 0.3, 0.2
opo.usp.ac.fj               Willing to manage
pc0169.usp.ac.fj            Willing to manage
teri.usp.ac.fj              0 users  load: 0.0, 0.0, 0.0
walu.usp.ac.fj              1 user,  load: 0.0, 0.0, 0.0
```



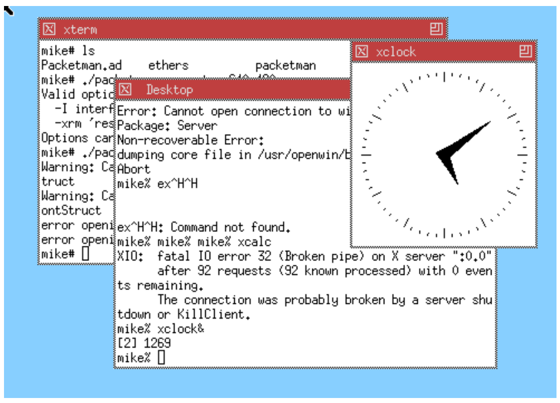
Login: hugh
Password:

or



Login name: hugh Hugh Anderson
Password:

There are of course other display managers, and login windows. There are also many different window managers:

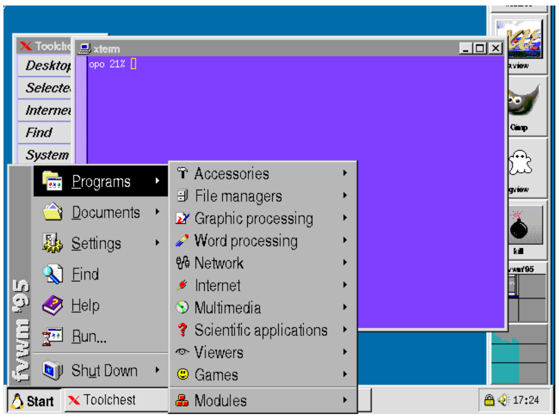


```
mike# ls
Packetman.ad    ethers          packetman
mike# ./pac
Valid optic
  -I interf
  -xrm 'res
Options car
mike# ./pac
Warning: Ca
truct
Warning: Ca
ontStruct
error openi
error openi
mike# 
```

```
Error: Cannot open connection to wi
Package: Server
Non-recoverable Error:
dumping core file in /usr/openwin/b
Abort
mike% ex^H^H


ex^H^H: Command not found.
mike% mike% mike% xcalc
XIO:  fatal IO error 32 (Broken pipe) on X server ":0.0"
      after 92 requests (92 known processed) with 0 even
ts remaining.
      The connection was probably broken by a server shu
tdown or KillClient.
mike% xclock&
[2] 1269
mike% 
```

or



1.4.3 Win32

Win32 is the 32 bit successor of the Win16 API - the original Windows Application Programming Interface. Win32 is a generic name for 4 (slightly) different APIs - providing standard function calls for accessing the GUI, file system, processes and so on. The Win32 API on Win95 is a subset of those on WinNT, so applications written for Win95 should be portable to WinNT. The reverse is not always true, but most WinNT applications can run on Win95/98.

The normal way for you to access Win32 functions is by using a precompiled library from a C program. C programmers include a set of header files, and applications link at run time to the Win32 DLLs.

The API for Win9X has three sections:

- **KERNEL:** - the low level kernel services in user32.dll.
- **GDI:** - Graphics Device Interface - drawing and printing in gdi32.dll.
- **USER:** - User Interface controls, windows and messaging services in kernel32.dll.

In Windows NT these services are kernel calls.

1.5 Non-native environments/platforms

The following systems can be used to provide a consistent environment that is independant of the host operating systems:

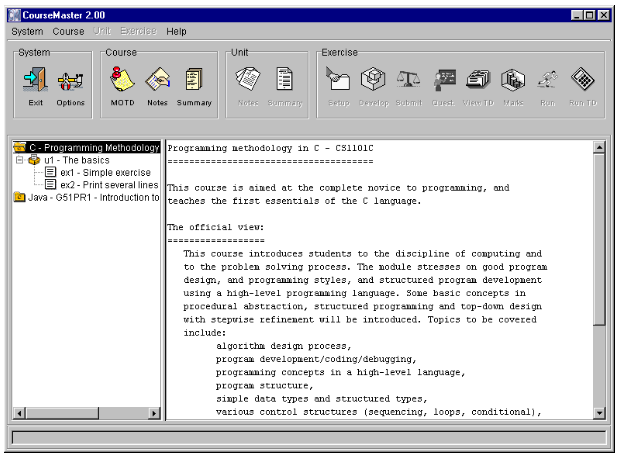
- Java/Swing
- Web browser interfaces
- Thin client systems

Each is discussed in turn in the following sections.

1.5.1 Java

Sun Microsystem's development of Java has always been done with portability issues in mind. The JVM (Java Virtual Machine) is available on all platforms, and runs reasonably well on them. There are some unresolved issues - particularly in the areas of security, fonts and efficiency - for example the Blackdown distribution of Java for Linux is reputed to be about ten times slower than Sun's in some areas of its use.

However - despite this - it is relatively easy to write a portable application - for delivery either as an applet in a web page, or as a standalone application. The **swing** windowing toolkit is the Java API for GUI development.



```
Programming methodology in C - CS1101C
======================================

This course is aimed at the complete novice to programming, and
teaches the first essentials of the C language.

The official view:
==================
   This course introduces students to the discipline of computing and
   to the problem solving process. The module stresses on good program
   design, and programming styles, and structured program development
   using a high-level programming language. Some basic concepts in
   procedural abstraction, structured programming and top-down design
   with stepwise refinement will be introduced. Topics to be covered
   include:
         algorithm design process,
         program development/coding/debugging,
         programming concepts in a high-level language,
         program structure,
         simple data types and structured types,
         various control structures (sequencing, loops, conditional),
```

The all-java standalone application shown above is used for automated laboratory assessment of programs.

1.5.2 Web browser interfaces

The first web servers provided static pages of hypertext and images, but fairly quickly, demand led to the specification of a standard for active page generation - known as CGI - the Common Gateway Interface.

CGI specifies how to pass arguments to a program on a server as part of the HTTP request. The program might then look up a database before generating some HTML to pass back to the browser. A CGI program can be any program which can accept command line arguments - Perl is a common choice for writing these programs.

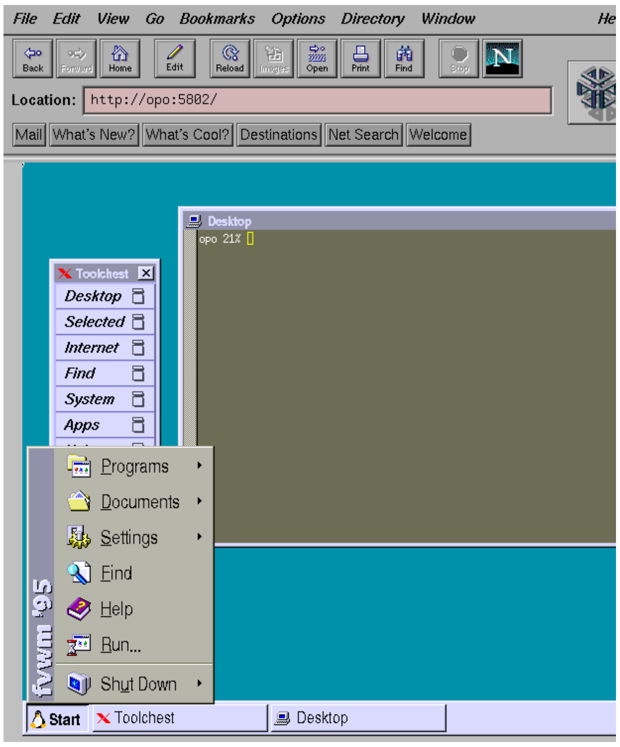
You should be aware that poorly constructed CGI scripts can result in security problems for the server, and also that there is normally a process overhead for each script started. More recently there have been various other web/interface techniques - for example the use of:

1. Java applets, to allow processing at the browser,
2. PHP (a server-side, cross-platform, HTML-embedded scripting language), or
3. ASP (a scripting environment for Microsoft Internet Information Server in which you can combine HTML, scripts and reusable ActiveX server components).

In this course we will look at the use of Java in this role.

1.5.3 Thin client systems

A relatively recent development involves moving the X-server (or equivalent) from the machine with the display to a larger machine, and then using a smaller computer to actually display the data. Here is a thin client written in Java, running as an applet in a web page:



Location: http://opo:5802/

```
opo 21%
```

File Edit View Go Bookmarks Options Directory Window He
Back Forward Home Edit Reload Images Open Print Find Stop
Mail What's New? What's Cool? Destinations Net Search Welcome
Desktop
Toolchest
Desktop
Selected
Internet
Find
System
Apps
Programs
Documents
Settings
Find
Help
Run...
Shut Down
fvwm '95
Start
Toolchest
Desktop

1.6 Widget sets

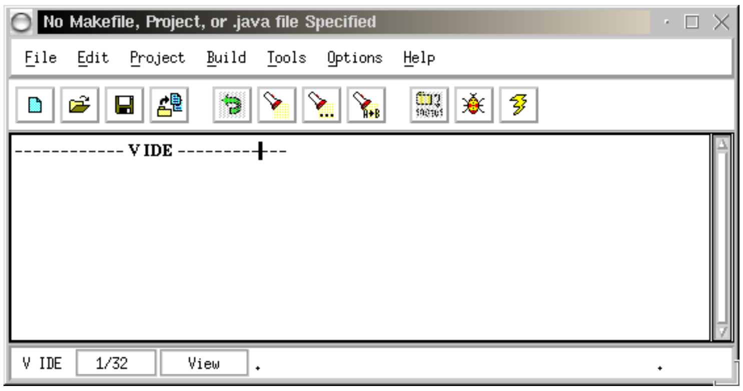
One definition of a *widget* is:

> *[possibly evoking "window gadget"] In graphical user interfaces, a combination of a graphic symbol and some program code to perform a specific function. E.g. a scroll-bar or button. Windowing systems usually provide widget libraries (sets) containing commonly used widgets drawn in a certain style and with consistent behaviour.*

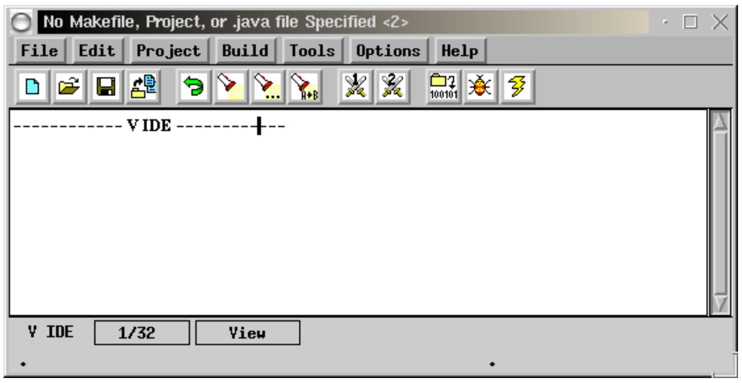
When we use different widget sets, our applications have a slightly different look-and-feel.

The following two screenshots are of the same application - the first linked with the Motif widget set, and the second with the Athena Widget set - notice the differences in the look of the two applications.

Motif:


No Makefile, Project, or .java file Specified
File
Edit
Project
Build
Tools
Options
Help
------------ V IDE ---------+--
V IDE
1/32
View

Athena:


No Makefile, Project, or .java file Specified <2>
File
Edit
Project
Build
Tools
Options
Help
------------ V IDE ---------+--
V IDE
1/32
View

The ICS widget databook has a series of useful widgets to extend the basic Motif set, including ones for bar graphs and so on.

1.7 Summary of topics

In this module, we introduced the following topics:

- Rules of GUI
- Types of applications
- Windowing/GUI environments
- Widgets

Questions for Module 1

1. List three rules of things-to-avoid when developing GUI applications. For each rule, give an example which demonstrates the problem.
2. State one way in which an application written for the X-window environment may be different from an application written for a Win32 environment.
3. Find one example of a GUI application with a clear use of metaphor. Describe the application and the metaphor.
4. Research: What is a DLL?
5. Research: Why is java considered more secure for use in a distributed environment than (say) C?
6. Research: What is the principal use of PHP?

Further study

- The hall of shame:
  http://www.iarchitect.com/mshame.htm.
- A brief history of HCI:
  http://www.comp.nus.edu.sg/˜cs3283/ftp/BriefHistoryOfHCI.ps.gz.

1.8 Sample Assignment - design

This is just a sample assignment. Assignment 1 for this semester's CS3283 will be distributed in class.

Task:

- Identify a GUI application which you think you can improve.
- Design an improvement.
- State how you would *test* the improvement.

Deliverables:

- A title page containing your name(s) and matriculation number(s).
- A two to five page document containing
  - A description of the application - perhaps with screenshots.
  - A description of that part of the application that needs improving, clearly stating *why* you think it needs to be changed.
  - A description of the improvement that you would make, clearly stating *why* you think this change would be better.
  - A testing methodology for the change that you want to make.

Note that this assignment does not require you to implement any change, just to describe one that you would make.