Contact information

Room        S15 #06-12
Telephone   6874-6903
E-mail      hugh@comp.nus.edu.sg
GUI definition

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.

Original idea from research at SRI by Doug Engelbart.
GUI

Icons, buttons, dialogue/dialog boxes, windows on the screen.

User controls by moving a pointer on the screen and selecting objects.

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.
Official description of CS3283

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.
And...

✔ Graphical visualization
My expectation

✔ Read supporting papers, and

✔ maintain an active interest in GUI design and implementation.
## Assessment

<table>
<thead>
<tr>
<th>Assessment</th>
<th>Weighting</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assignments</td>
<td></td>
<td>35%</td>
</tr>
<tr>
<td>Ass1 Group</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Ass2 Individual</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>Ass3 Individual</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>Ass4 Group</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>Tutorials</td>
<td></td>
<td>5%</td>
</tr>
<tr>
<td>Mid-term</td>
<td>Closed book</td>
<td>10%</td>
</tr>
<tr>
<td>Final Exam</td>
<td>Open Book</td>
<td>50%</td>
</tr>
<tr>
<td>Total marks</td>
<td></td>
<td>100%</td>
</tr>
</tbody>
</table>
1. My textbook, and

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

Fundamental GUI concepts (1 lecture)
Design and programming techniques (3 lectures)
Cross platform GUI development (6 lectures)
Visualization techniques (2 lectures)

Enjoy the course!
Chapter 1

Module 1 - GUI concepts
GUI concepts

✔ Early user interfaces text based - fixed event ordering.

✔ GUI provides for complex interaction, and

✔ GUI 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.
How not to do GUI

[Image of a GUI showing a system error message: Application error: End date should be greater than or equal to start date. Start Date: 20/10/2001, End Date: 22/10/2001]
Two points

Try out your applications before delivering them. Ensure that error messages are precise, and indicate the next step.
How not to do GUI
General rules of GUI

Effective GUIs owe more to effective psychology than to effective programming.

✔ Not just icons - includes an abstract view

✔ No clash between views
EZ-link

Clash between

- User view: pay afterwards
- Bus co view: pay first
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.
Summary

Ensure correlation between What-u-c and What-u-think

KISS

Analogy, metaphor and icons

Always remember to include the U in GUI.
Do's and don’ts

Do follow standards
Do be predictable and responsive
Do be flexible
Don’t forget the user
Don’t forget the machine/environment
Don't assume things
Types of applications

Not all applications benefit from a GUI - consider embedded systems
However, there are areas that do benefit from a GUI:

- Immersive applications
- Office and business applications
- Interactive control systems
And visualization ...

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.
MAC platform
MAC platform
MAC platform
The X window system\(^1\) 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.

\(^1\)The system is called X, or the X window system. UNIX weenies insist that it is not called X-windows!
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:
Components of X

The X server
The X protocol
X clients
The Window manager(s)
The Display manager(s)
## Display manager

<table>
<thead>
<tr>
<th>Host</th>
<th>Users</th>
<th>Load</th>
</tr>
</thead>
<tbody>
<tr>
<td>manu.usp.ac.fj</td>
<td>14</td>
<td>0.3, 0.3, 0.2</td>
</tr>
<tr>
<td>opo.usp.ac.fj</td>
<td></td>
<td>Willing to manage</td>
</tr>
<tr>
<td>pc0169.usp.ac.fj</td>
<td></td>
<td>Willing to manage</td>
</tr>
<tr>
<td>teri.usp.ac.fj</td>
<td>0</td>
<td>0.0, 0.0, 0.0</td>
</tr>
<tr>
<td>walu.usp.ac.fj</td>
<td>1</td>
<td>0.0, 0.0, 0.0</td>
</tr>
</tbody>
</table>

(cancel, accept, ping)
Display manager

Welcome to opo
IRIX 6.3

Login: hugh
Password: |

or

Hugh Anderson
Window managers

or
Win32

✔ Win32 is the 32 bit successor of the Win16 API

✔ Win32 is a generic name for 4 (slightly) different APIs

✔ The Win32 API on Win95 is a subset of those on WinNT, so applications written for Win95 should be portable to WinNT.
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.
API for Win9X

Has three sections:

**KERNEL:** - the low level kernel services in kernel32.dll.

**GDI:** - Graphics Device Interface - drawing and printing in gdi32.dll.

**USER:** - User Interface controls, windows and messaging services in user32.dll.
Non-native platforms

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

Java/Swing
Web browser interfaces
Thin client systems
Sun Microsystems’s development of Java has always been done with portability issues in mind.

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.
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, loop, conditional),
Web browser interfaces

✔ The first web servers provided static pages of hypertext and images.

✔ Demand led to the specification of a standard for active page generation - CGI
CGI

✔ 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.
Warning

✗ You should be aware that poorly constructed CGI scripts can result in security problems for the server, and

✗ there is normally a process overhead for each script started.
Other web interfaces

Java applets, to allow processing at the browser, PHP (a server-side, cross-platform, HTML-embedded scripting language), or ASP (a scripting environment for Microsoft Internet Information Server in which you can combine HTML, scripts and reusable ActiveX server components).
Thin client systems
[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.
Motif application
Athena application
The ICS widget databook has a series of useful widgets to extend the basic Motif set, including ones for bar graphs and so on.
Summary of topics

In this module, we introduced the following topics:
Rules of GUI
Types of applications
Windowing/GUI environments
Widgets
Chapter 2

Module 2 - Design
Some fun
General issues of design

✔ Must design large things

✔ Design approaches an art form

✔ Principal concern with user
How not to design
Avoid doing things just because you know how to do them. Make your designs be driven by requirements.
The design process involves both specification of the behaviour of a product, and specification of the detailed techniques used to implement the product.

In each area, there exist a range of tools and techniques that can benefit any software product.
Role of designer

✔ Designer interacts with people

✔ Must relate designs back to requirements and constraints.

✔ Design must be readable, understandable, implementable.

✔ Designer uses abstraction.
Iconic abstraction

Text file using an icon:
Higher level abstractions

✔ Desktop

✔ Wastebasket
# Building blocks

<table>
<thead>
<tr>
<th>Button</th>
<th>Testbox</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Dismiss" /></td>
<td>This window is a text widget. It displays one or more lines of text and allows you to edit the text. Here is a summary of the things you might do.</td>
<td></td>
</tr>
<tr>
<td><img src="image" alt="Perl/Tk" /></td>
<td></td>
<td><img src="image" alt="Perl/Tk" /></td>
</tr>
</tbody>
</table>
# Building blocks

<table>
<thead>
<tr>
<th>Menu</th>
<th>Checkbox</th>
<th>Radiobutton</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open ...</td>
<td>![Checkbox](Wipers OK)</td>
<td>![Radiobutton](Point Size 10)</td>
</tr>
<tr>
<td>New</td>
<td></td>
<td>![Radiobutton](Point Size 12)</td>
</tr>
<tr>
<td>Save</td>
<td>![Checkbox](Brakes OK)</td>
<td>![Radiobutton](Point Size 18)</td>
</tr>
<tr>
<td>Save As ...</td>
<td></td>
<td>![Radiobutton](Point Size 24)</td>
</tr>
<tr>
<td>Setup ...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Print ...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quit</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Building blocks

<table>
<thead>
<tr>
<th>Scrollbar</th>
<th>Graph</th>
<th>Directory Tree</th>
</tr>
</thead>
</table>

**Scrollbar**

*This window is a text widget. It displays one or more lines of text and allows you to edit the text.*

**Graph**

![Graph Image]

**Directory Tree**

![Directory Tree Image]

*DirTree, display directory tree.*
Containers

![Diagram of container dimensions with options for Width and Height in cm, inches, % of Page, and % of Column. The values are 0 for both Width and Height.](image)
Sundries

✔ Cursors

✔ Fonts

✔ Colours

✔ Drag-n-drop

✔ Cut-n-paste.
Use cases

Designer imagines and proposes common scenarios\(^2\), and checks to see if the scenarios are *consistent*, and *complete*, tries out the scenarios on people to see if they work, tests the scenarios and attempts to quantify their behaviour.

\(^2\)Scenarios=Use\_cases. Use\_cases=scenarios.
F&R state diagram

Specify Pattern

Specify Change

Doit!

Enter

Escape

Enter

Escape

Enter

Cancel
GUI-style *find-and-replace*:
State diagram
The focus on detail related to the state of a dialog is not trivial.

✔ There is a well known example of a poorly constructed dialog, that contributed to the death of cancer patients in the US

http://sunnyday.mit.edu/therac-25.html
Modelling

Modelling: - Used to demonstrate the UI, without actually implementing the core software.

Dan Bricklen’s demo program (a demo copy is available at [http://www.brickin.com/](http://www.brickin.com/)) is worth looking at for modelling a user interface. There is an amusing demo called chiapaint.
Chiapaint
Modelling

It is also relatively easy to model a new UI using Tcl/Tk.
The principle features of OO technology are as follows:
Abstraction,
Information hiding,
Inheritance,
Polymorphism, and
Genericity
Some of these OO features provide a mechanism which supports the construction of better software...

- Create library
- Generalize it
- Design becomes the detailing of new classes derived from the generalized ones...
GUI design

GUI design has to meld four possibly conflicting elements:

**Software model** - structure of data and software

**User profile** - the types of users

**Product perception** - the mental image developed by user

**Product image** - the GUI - screenshots, descriptions or specifications

In general, a GUI is successful when the product perception matches the product image.
GUI specification/design

Our concern is to:

Develop a functional and behavioural response specification in terms of its cognitive aspects.

The functional and behavioural response specification is turned inside-out from a normal software specification. With a software behavioural model, we start with an analysis of states, events and actions, and specify the expected views as a result. With GUI specification, our orientation is to start with the views, and specify the states, events and actions associated with those views.
Basis for GUI design

One of the most characteristic elements of many GUI programs is the use of the event-driven software architecture.

When the designer adopts this paradigm, the GUI program is viewed as a series of response routines for particular events.
Design document

User requirement
Environment
Software constraints
Other constraints
Interface design
Overview
Interface description
Prototype screens
Functional specifications
Behavioural specifications
Testing methodology
Z can \textit{formally} specify complex GUI interactions.

Z tools test the specification

More details may be found in the handout, found at

\url{http://www.cs.virginia.edu/~jck/publications/zum.97.pdf}

It describes the interface to a nuclear reactor.
GUI designs

✔ With Therac-25, we saw how a poorly constructed interface led to deaths of patients...

✔ By contrast, a good interface may result in the reverse...
Examples of GUI designs

Here are some examples of different designs for similar things
Lyx Find-and-replace:
Nedit find-and-replace

String to Find: (use up arrow key to recall previous)

Replace With:

- Literal
- Case Sensitive Literal
- Regular Expression
- Search Forward
- Search Backward

Replace | Find | Replace All | R. In Selection | Cancel
Word find-and-replace

![Find and Replace dialog box](image-url)
Win98 file manager

Select an item to view its description.
KDE file manager

![KDE file manager screenshot](image-url)
Visualization design has a similar structure to GUI design - a difference being the focus on the use of analogy.
Basis for visualization design

Eick proposes the following guidelines:

1. Focus the visualization on task-specific user needs.

2. Use a whole-database overview display.

3. Encode the data using colour, shape, size, position.

4. Use drill-down, filters and multiple linked views

5. Use smooth animation for time
Visualization document

User requirement
Environment
Software constraints
Other constraints
Interface design
Overview
Interface description
Drill-down and other displays
Encoding
Testing methodologies
Examples of visualization

There are many examples of data visualizations, and I have just taken some from the world of network management - starting from simple graphical displays through to 3D images.
Graphs and diagramming
Compact visualization

![Compact Visualization Diagram]
3D graph
Abstract 3D view - SeeNet
Abstract 3D view - Flodar

[Image of 3D abstract view]
3D world-view
Summary of topics

In this module, we introduced the following topics:
The designer’s mindset
Specification and design, tools and methods
Examples of successful designs
Assignment

- Up to 3/group
- 3% of assignment mark/about 12% of final
- Development of a design/analysis document, with modelling
Your task...

✔ System with a GUI interface

✔ Help track disease

✔ View patient histories
  ✔ Select by region
  ✔ + other functions

✔ Design, not implement.
Submission

✔ Description of **system architecture**

✔ GUI **design/analysis document** concerned with the GUI interface.
Deliverables

A title page
Table of contents...
Introduction - non-technical
System architecture with justifications
GUI design/analysis document
Design/analysis document

Follow suggested structure?
User requirement, user profile, environment
Overview of the GUI interface
Description of the interface
Prototype screens
Functional spec
Behavioural spec
Justifications - relating back to user requirement.
A testing methodology
Note that this assignment does not require you to implement the application, just to design one, and to model the design with prototype screens.

You could use Java/Visual Basic/ a graphics editor... anything as long as you show screenshots.
Assessment

The assessment will be graded with the following weightings:

- Introduction 10%
- System architecture 25%
- GUI design 50%
- Extra 15%
Assessment

• The “Extra” component of the assessment is for submissions which show clear evidence of extra thought or care.

• In evaluating the “GUI design” component, I will also be looking for “justifications you can make for design decisions”.

Try to achieve clarity in your writing and take care in the structuring of the document.
GUI applications can be BIG

Hence concern with architecture
Architecture

Standalone
Shared file
Shared database
Web based
Simple
Scripting
Java
Web architecture

✔ Common to deliver applications via web browsers.

✔ MSIE/Navigator/iCab/Opera... different in implementation.
Standalone
Shared file

FileServer

High Speed Disk

CS3283 - Hugh Anderson's notes.
Shared database

DatabaseServer

High Speed Disk

CS3283 - Hugh Anderson's notes.
Web server
Active scripting
Java applet

An even more complex GUI application might be constructed using a series of interlinked web pages containing Java applets. The advantage of this, is two fold. The processing **load on the web server** may be **reduced**. The Java applet can directly\(^3\) communicate with a database server.

\(^3\)Note that there are some security concerns here.
Java applet

Web Server ➔

Database Server ➔

High Speed Disk ➔

Web Browser ➔
+ Java applet
Chapter 4

First steps
GUI programming

In elementary programming styles, there is a single thread-of-control

✔ GUI programs respond to events

✔ Restructuring programs as a group of callbacks.
GUI mainline

```c
#include <any GUI header files needed>

int main ()
{
    RegisterAllCallbacks ();
    LoopForever ();
}
```

CODE LISTING

GUICode.c
How not to ...

Don’t do it the hard way!
X API

quit

This is a label.
```c
#include <stdio.h>
#include <X11/Intrinsic.h>
#include <X11/StringDefs.h>
#include <X11/Xaw/Command.h>
#include <X11/Xaw/Paned.h>
#include <X11/Xaw/Label.h>

void quit_callback (widget, client_data, call_data)
    Widget widget;
    caddr_t client_data;
    caddr_t call_data;
    {
        exit (0);
    }

main (argc, argv)
    int argc;
    char *argv[];
    {
        /* main */
        Widget parent;
        Arg args[20];
        int n;
        Widget pane_widget, quit_widget;
        Widget label_widget;

        /* Set up top−level shell widget */
        parent = XtInitialize (argv[0], "Xaw1", NULL, 0, &argc, argv);

        /* Set up pane to control whole application */
        n = 0;
        pane_widget = XtCreateManagedWidget ("pane",
                                              panedWidgetClass, parent, args, n);

        /* Set up command widget to act as a push button */
        n = 0;
        quit_widget = XtCreateManagedWidget ("quit",
                                              commandWidgetClass, pane_widget, args, n);

        /* Set up a callback function */
        XtAddCallback (quit_widget, XtNcallback, quit_callback, (caddr_t) NULL);

        /* Set up label widget */
        n = 0;
        XtSetArg (args[n], XtNlabel, "This is a label.");
        n++;
        label_widget = XtCreateManagedWidget ("label",
                                              labelWidgetClass, pane_widget, args, n);

        /* Map widgets and handle events */
        XtRealizeWidget (parent);
        XtMainLoop ();
    }
```

CODE LISTING xaw1.c

CS3283 - Hugh Anderson's notes. Page number: 121
X compilation

```
gcc -o xawl xawl.c -lXt -lXaw
```
CODE LISTING

```
#include <windows.h>

int STDCALL
WinMain (HINSTANCE hInst, HINSTANCE hPrev, LPSTR lpCmd, int nShow)
{
    MessageBox ( NULL , "Hello, Windows! ", "Hello ", MB_OK);
    return 0;
}
```
Win32 compilation

gcc -oSimpleWin32 SimpleWin32.c -mwindows
Win32 application
Win32 application

#include <windows.h>
#include <string.h>

int STDCALL
WinMain (HINSTANCE hInst, HINSTANCE hPrev, LPSTR lpCmd, int nShow)
{
    HWND hwndMain;  /* Handle for the main window. */
    MSG msg;  /* A Win32 message structure. */
    WNDCLASSEX wndclass;  /* A window class structure. */
    char *szMainWndClass = "WinTestWin";
    memset (&wndclass, 0, sizeof (WNDCLASSEX));
    wndclass.lpszClassName = szMainWndClass;
    wndclass.cbSize = sizeof (WNDCLASSEX);
    wndclass.style = CS_HREDRAW | CS_VREDRAW;
    wndclass.lpfnWndProc = MainWndProc;
    wndclass.hInstance = hInst;
    wndclass.hIcon = LoadIcon (NULL, IDI_APPLICATION);
    wndclass.hIconSm = LoadIcon (NULL, IDI_APPLICATION);
    wndclass.hCursor = LoadCursor (NULL, IDC_ARROW);
    wndclass.hbrBackground = (HBRUSH) GetStockObject (WHITE_BRUSH);
    RegisterClassEx (&wndclass);
    hwndMain = CreateWindow (szMainWndClass, "Hello", WS_OVERLAPPEDWINDOW,
    CW_USEDEFAULT, CW_USEDEFAULT, CW_USEDEFAULT, NULL, NULL, hInst, NULL);
    ShowWindow (hwndMain, nShow);
    UpdateWindow (hwndMain);
    while (GetMessage (&msg, NULL, 0, 0)) {
        TranslateMessage (&msg);
        DispatchMessage (&msg);
    }
    return msg.wParam;
}
Win32 application

The full source code and a makefile is available at http://www.comp.nus.edu.sg/~cs3283/ftp/generic.tgz.
Win32 application
Win32 programming

OO GUI toolkits

No one object-oriented standard for GUI applications
Event handling

Frame

Button

Quit
GTK+

✔ GTK+ is a multi-platform toolkit

✔ By using CygWin GTK+ works on Win32.

✔ GTK+ is free software and part of the GNU Project.
GTK+ has an object-oriented architecture with component libraries:
GDK - A wrapper for low-level windowing functions.
GTK - An advanced widget set.
Glade application
MFC

✔ OO toolkit to access Win32

✔ DLL contains code for MFC

✔ Linked at runtime.

✔ Base class CObject
One characteristic of MFC programs is the use of Hungarian (prefix) notation for variable names. It is common to see MFC program variables prefixed with type identifiers. For example:

dLocalMax is a double variable
iLocalMin is an integer variable.
✔ Originally the graphical toolkit for Java was AWT, the **Abstract Windowing Toolkit**.

✔ It is fairly primitive, and the new Swing toolkit provides much more functionality.

✔ AWT is native code, with a Java API, but Swing is implemented on-top-of AWT.
Swing components inherit from `java.awt.component`, and the Swing classes that are similar to AWT classes are prefixed with the letter “J”.

- For example, the AWT `Button` class is renamed `JButton`.

- You can mix-and-match AWT and Swing components.
Java/Swing may be used in two distinct ways:
Producing a standalone application.
Producing an applet to run within a web browser.
One of the features of Swing is that it implements a pluggable look-and-feel.
The look-and-feel can even be changed dynamically.
Web interfaces

Categories:
Server-side dynamic pages
Server-side scripting
Client-side scripting
Client-side applets

We will look at some of these methods later in the course.
Scripting languages

✔ Scripting languages which can produce GUI interfaces are relatively easy to use.

✔ An effective strategy for building GUI applications is to write the GUI part in a scripting language, and to write the core ’difficult’ part in C.
Summary of topics

In this module, we introduced the following topics:
Programming styles to avoid
Event driven architectures
OO toolkits
Web-based systems
Scripting languages
Further study

Environment

✔ X - from UNIX/sunfire

✔ Cygwin - download `~cs3283/ftp/full.exe` (OLD)

✔ Native - download www.scriptics.com
Installing Tcl/Tk

✔ If you are using cygwin-b20, the wish interpreter is called cygwish80.exe.

✔ Find cygwish80.exe.

✔ Copy and call it wish8.0.exe for UNIX compatibility.
Installing Tcl/Tk

✔ In the first line of your tcl files, you should put
  #!/bin/wish8.0
Tcl/Tk demos

✔ If you download the file ~cs3283/ftp/demos.tar and extract it, you will have a series of Tcl/Tk widget examples in ~/Demos.

✔ Change into the directory ~/Demos, and type ./widget.
Tcl/Tk tutor

✔ There is a Tcl/Tk tutor, and many learn-to-program-Tcl/Tk documents available at many sites on the Internet - if you continue to have trouble, you may wish to try them.
✓ Tcl (Tool Command Language) is an interpreted scripting language

✓ Pronounced 'tickle'.

✓ Tk - X-window toolkit

✓ Wish - the windowing shell
Scripting language

- Difficult to define.
- Job control languages ...
- More powerful basic operations
- Regular-expression pattern matching
Scripting

✔ Normally interpreted

✔ One line equivalent to 100 lines of C.

✔ Overhead

Perl in web page developments, Tcl/Tk for GUI development
How not to!

Don’t use to the exclusion of other languages!

✔ Good and bad

✔ Array lookup ..

✔ Common to mix scripting and other languages.
Hello world

manu> wish
wish> button .quit -text "Hello World!" -command {exit}
.quit
wish> pack .quit
wish>
In a script

CODE LISTING

HelloWorld.tcl

#!/usr/local/bin/wish8.1 -f

button .quit -text "Hello World!" -command {exit}
pack .quit
Run it

Hello World!
Tcl first...then Tk
Tcl structure

✔ Tiny syntax

✔ Single *command* structure

✔ Set of rules

✔ All PROGRAM structures are implemented as *commands*.

✔ Level’ of the GUI interface is quite high.
If the first character of a command is #, it is a comment.
Tcl commands are just words separated by spaces. Commands return strings, and arguments are just further words.

```
command argument argument
command argument
```

Tcl commands are separated by a new line, or a semicolon, and arrays are indexed by text:

```
set a(a\ text\ index) 4
```
## Command examples

<table>
<thead>
<tr>
<th>Procedures</th>
<th>File Access</th>
<th>Miscellaneous</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>proc name {params} {body}</code></td>
<td><code>open &lt;name&gt;</code></td>
<td><code>source &lt;NameOfFile&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>read &lt;fileID&gt;</code></td>
<td><code>global &lt;varname&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>close &lt;fileID&gt;</code></td>
<td><code>catch &lt;command&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>cd &lt;directoryname&gt;</code></td>
<td><code>format &lt;fmtstrng&gt; &lt;val&gt;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>exec &lt;process&gt;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>return &lt;value&gt;</code></td>
</tr>
</tbody>
</table>
Spaces are important:

<table>
<thead>
<tr>
<th>expr 5*3</th>
<th>has a single argument</th>
</tr>
</thead>
<tbody>
<tr>
<td>expr 5 * 3</td>
<td>has three arguments</td>
</tr>
</tbody>
</table>
When the " or { character are first in the word.

".." disables a few of the special characters - for example space, tab, newline and semicolon, and

{..} disables everything except \{, \} and \nl.
Control structures

End up looking very like ’C’:

```
while {a==10} {
    set b [tst a]
}
```
Command

The []’s are replaced by the value returned by executing the Tcl command ’doit’.

```
set a [doit param1 param2]
```
Variable substitution

The dollar sign performs the variable value substitution. Tcl variables are strings.

<table>
<thead>
<tr>
<th>Command</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>set a 12b</code></td>
<td>a will be &quot;12b&quot;</td>
</tr>
<tr>
<td><code>set b 12$a</code></td>
<td>b will be &quot;1212b&quot;</td>
</tr>
</tbody>
</table>
Backslash

set a a\ string\ with\ spaces\ \\
and\ a\ new\ line
tclsh or wish?

✔ tclsh - a shell-like application that reads Tcl commands from its *standard input* or from a *file* and evaluates them.

✔ wish - the Tcl command language, the Tk toolkit, and a main program that reads commands from standard input or from a file.
Assignment

% set a 1
1
% set a
1
% set a 2
2
% set a
2
Assignment

% set a alphabetapruning
alphabetapruning
% set a
alphabetapruning
% set a alpha\ beta\ pruning
alpha beta pruning
% set a
alpha beta pruning
% set a alphapruning
alphapruning
% set a
alphapruning
% set a alpha\ beta\ pruning
alpha beta pruning
% set a
alpha beta pruning

(Backslash substitution)
Assignment

% set a {alpha
beta
pruning}
....
% set a
alpha
beta
pruning
Assignment

\% set a 22/3
22/3
\% set a
22/3
Assignment

% set a expr 22/3
wrong # args: should be "set varName ?newValue?"
% set a {expr 12/4.2}
expr 12/4.2
% set a [expr 12.2/33]
0.369696969697
% set a
0.369696969697

(Command substitution)
Assignment

```
% set a
0.369696969697
% set a [expr $a*33]
12.2
```

(Variable and Command substitution)
Assignment

% expr 1+2+3
6
% expr 1 + 2+3
6
Assignment - increment

```% set a 10
10
% incr a
11
% set a
11```
Assignment

\%
set g gallon
gallon
\%
set gallons $gs$
can’t read "gs": no such variable
\%
set gallons ${g}s$
gallons

(Variable substitution)
Assignment - append

% set a ${a}222
11222
% append a 333
11222333
Tcl/Tk list

Just a sequence of words:

```tcl
% set dow {Mon Tue Wed Thu Fri Party Sun}
Mon Tue Wed Thu Fri Party Sun
% lindex $dow 3
Thu
%
lindex $dow 1
Tue
```
Lists within lists

% set a {0 1 {2 x} {3 x y} 4}
0 1 {2 x} {3 x y} 4
% lindex $a 3
3 x y
Iteration over list

```bash
% foreach day $dow {
    puts Day\ of\ week\ is\ $day
}
Day of week is Mon
Day of week is Tue
Day of week is Wed
Day of week is Thu
Day of week is Fri
Day of week is Party
Day of week is Sun
```
Arrays

% set work(Mon) 8
8
% set work(Tue) 10
10
% foreach day $dow {
    puts $day\ I\ worked\ $work($day)hrs
}

Mon I worked 8hrs
Tue I worked 10hrs
can’t read "work(Wed)": no such element in array
<table>
<thead>
<tr>
<th>Mon</th>
<th>Tue</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

% array size work
% array names work
Summary

✔ Tcl

✔ commands
✔ quoting and substitutions
✔ assignment (set)
✔ Lists and arrays
Chapter 6

Tcl/Tk - 2
Summary

✔ Tcl
✔ commands
✔ quoting and substitutions
✔ assignment (set)
✔ Lists and arrays
Tcl/Tk list - recap

Just a sequence of words:

% set dow {Mon Tue Wed Thu Fri Party Sun}
Mon Tue Wed Thu Fri Party Sun
% lindex $dow 3
Thu
%
lindex $dow 1
Tue
Lists within lists - recap

```
% set a {0 1 {2 x} {3 x y} 4}
0 1 {2 x} {3 x y} 4
% lindex $a 3
3 x y
```
Iteration over list - recap

% foreach day $dow {
    puts Day\ of\ week\ is\ $day
}

Day of week is Mon
Day of week is Tue
Day of week is Wed
Day of week is Thu
Day of week is Fri
Day of week is Party
Day of week is Sun
List operators

- `sort <list> [options]`
- `split <string> [splitcharacters]`
- `concat <list> <list>`
- `lindex <list> <index>`
- ... + lots more
Sorting a list

% lsort $dow
Fri Mon Party Sun Thu Tue Wed
Split (string to list)

% split "/usr/local/bin/tclsh" /
{} usr local bin tclsh

% join {{} usr local bin} /
/usr/local/bin
Arrays - recap

```bash
% set work(Mon) 8
8
% set work(Tue) 10
10
% foreach day $dow {
    puts $day\ I\ worked\ $work($day)hrs
}
Mon I worked 8hrs
Tue I worked 10hrs
can't read "work(Wed)": no such element in array
```
% array size work
2
% array names work
Tue Mon
Multidimensional arrays

% set lect(Mon,8-10) cs2106
 cs2106
% set lect(Mon, 8-10) cs2106
 wrong # args: should be "set varName ?newValue?"
if then else

% if {3 == 3} {puts hi}
hi
% if {3 == 4} {puts hi}
% if {3 == 3} {puts hi}
hi
% if {"help" == "help"} {puts hi}
hi
while

% set c 3
3
% while {$c>=0} {
    puts "c is $c"
    set c [expr $c-1]
}
c is 3
c is 2
c is 1
c is 0
foreach

```bash
% foreach day $dow {
  puts $day\ I\ worked\ $work($day)hrs
}
Mon I worked 8hrs
Tue I worked 10hrs
...
```
Control - summary

if {test} {thenpart} {elsepart}
while {test} {body}
for {init} {test} {incr} {body}
continue
switch $x {a {a-part} b {b-part}... }
Procedures

% proc fac x {
    if {$x<=1} {return 1}
    expr $x * [fac [expr $x-1]]
}
% fac 4
24
% proc printerrorno {
    global errno
    puts "The error is $errno"
}

Parameter passing

✔ By value

✔ i.e. not arrays
Parameter passing

```bash
% set arr(1) 10
% set arr(2) 20
% proc prnt ar {
    puts $ar(1)
    puts $ar(2)
}
% prnt arr
  can't read "ar(1)": no such variable
```
Parameter passing

% proc prnt ar {
    upvar $ar arra
    puts $arra(1)
    puts $arra(2)
}

% prnt arr
10
100
Parameter passing

```bash
% proc prnt ar {
    upvar $ar arra
    set i 1
    while {$i<=[array size arra]} {
        puts $arra($i)
        incr i
    }
}

% prnt arr
10
20
```
End of Tcl, Start of Tk
Widget creation commands

First parameter is a 'dotted' name.

% label <name>  -
optional parameter pairs ...
% canvas <name>  -
optional parameter pairs ...
% button <name>  -
optional parameter pairs ...
% frame  <name>  -
optional parameter pairs ...
% ... and so on

The dot heirarchy indicates the relationships between the widgets.
New widget commands

When you create a widget ".b", a new command ".b" is created, which you can use to further communicate with it. The geometry managers in Tk assemble the widgets:

```python
% pack <name> .... where ....
```
Widget becomes command

canvas .c
pack .c
button .q -text Quit -command {exit}
pack .q
.q configure -foreground blue
.q configure -background red
Example 2

```bash
foreach r {raised sunken flat groove ridge} {
    frame .$r -width 15m -height 10m -relief $r -
    borderwidth 8
    pack .$r -side left -padx 2m -pady 2m
}
.flat configure -background blue
.flat configure -background gray
```
Example 2

![Diagram](image-url)
Example 3

```plaintext
radiobutton .cs2105 -text CS2105 -variable course -
value CS2105 -anchor w
radiobutton .cs2106 -text CS2106 -variable course -
value CS2106 -anchor w
pack .cs2105 .cs2106
```
Example 3
Example 3

set course CS2106
Tcl/Tk example software
Code is

```
#!/usr/local/bin/wish8.1 -f

text .log -width 60 -height 5 -bd 2 -relief raised
pack .log

button .buttonquit -text "Quit" -command exit
pack .buttonquit

button .buttondate -text "date" -command getdate
pack .buttondate

proc getdate {} {
    set result [exec date]
    .log insert end $result
    .log insert end 
}
```

CODE LISTING SimpleProg.tcl
Listbox

Tutorial...
Scrollbars

Tutorial...
Scales

Tutorial...
Tcl/Tk menus

✔ Make up a frame for the menu

✔ Add in the top level menu items

✔ For each top level item, add in the drop-menu items

✔ For each nested item, add in any cascaded menus.

✔ Remember to pack it...
### CODE LISTING

**Menus.tcl**

```tcl
#!/usr/bin/wish

frame .mbar −relief raised −bd 2
pack .mbar −side top −fill x

frame .dummy −width 10c −height 100
pack .dummy

menubutton .mbar.file −text File −underline 0 −menu .mbar.file.menu
menu .mbar.file.menu −tearoff 0
.mbar.file.menu add command −label "New..." −command "newcommand"
.mbar.file.menu add command −label "Open..." −command "opencommand"
m.bar.file.menu add separator
.mbar.file.menu add command −label Quit −command exit
pack .mbar.file −side left

menubutton .mbar.edit −text Edit −underline 0 −menu .mbar.edit.menu
menu .mbar.edit.menu −tearoff 1
.mbar.edit.menu add command −label "Undo..." −command "undocommand"
.mbar.edit.menu add separator
.mbar.edit.menu add cascade −label Preferences −menu .mbar.edit.menu.prefs
menu .mbar.edit.menu.prefs −tearoff 0
.mbar.edit.menu.prefs add command −label "Load default" −command "defaultprefs"
.mbar.edit.menu.prefs add command −label "Revert" −command "revertprefs"
pack .mbar.edit −side left

menubutton .mbar.help −text Help −underline 0 −menu .mbar.help.menu
menu .mbar.help.menu −tearoff 0
.mbar.help.menu add command −label "About ThisApp..." −command "aboutcommand"
pack .mbar.help −side right

proc aboutcommand {} {
    tk_dialog .win {About this program} "Hugh wrote it!" {} 0 OK
}
```

---

**CS3283 - Hugh Anderson's notes.**  
Page number: 224
Tcl/Tk menus

![Tcl/Tk menu example](image-url)
Tk canvas

✔ Draw items

✔ Items may be tagged

✔ Described in Robert Biddle’s “Using the Tk Canvas Facility”, a copy of which is found at ~cs3283/ftp/CS-TR-94-5.pdf.

✔ Dynamically created variable names (node$nodes).
Chapter 7

Tcl/Tk - 3, Java
Bind a Tcl/Tk script to an X event
Example 4

canvas .c -bd 3 -relief raised; pack .c
button .b -text BUTTON1; pack .b
bind .b <Enter> {.c config -cursor {clock}}
open /etc/passwd r
set f file4
while {[gets $f line] >=0} {
    puts $line
}
close $f
File I/O

✅ flush fileId

✅ seek fileId offset [start/current/end]

✅ tell fileId

✅ read fileId numbytes
file dirname name
file tail name
Demo tkpaint
#! /usr/local/bin/wish -f
set thistool rectangle
set thisop grow
set thiscolour black
button .exitbtn -bitmap @exit.xbm -command exit
button .squarebtn -bitmap @square.xbm -command setsquaretool
button .circlebtn -bitmap @circle.xbm -command setcircletool
button .shrnkbtn -bitmap @shrink.xbm -command "set thisop shrnk"
button .growbtn -bitmap @grow.xbm -command "set thisop grow"
button .printbtn -bitmap @print.xbm -command printit
button .colorbtn -bitmap @newcolour.xbm -command setanewcolour
canvas .net -width 400 -height 400 -background white -relief sunken
canvas .status -width 40 -height 40 -background white -relief sunken
pack .net -side bottom
pack .status -side right
pack .squarebtn .circlebtn -side left -ipadx 1m -ipady 1m -expand 1
pack .exitbtn .printbtn -side right -ipadx 1m -ipady 1m -expand 1
pack .colorbtn .shrnkbtn .growbtn -side right -ipadx 1m -ipady 1m -expand 1
bind .net <ButtonPress-1> {makenode %x %y}
.status create rectangle 10 10 37 37 -tag statusthingy -fill $thiscolour
set nodes 0; set oldx 0; set oldy 0;
Routines

```tcl
proc beginmove {x y} {
    global oldx oldy
    set oldx $x; set oldy $y
}

proc domove {item x y} {
    global oldx oldy
    .net move $item [expr "$x - $oldx"] [expr "$y - $oldy"]
    set oldx $x; set oldy $y
}

proc altersize {item x y z} {
    .net scale $item $x $y $z $z
}

proc printit {} {
    .net postscript -file "pic.ps"
}
```

CS3283 - Hugh Anderson's notes.
Routines

```tcl
proc makenode {x y} {
    global nodes oldx oldy thistool thiscolor
    set nodes [expr "$nodes+1"]
    set x1 [expr "$x−20"]
    set y1 [expr "$y−20"]
    set x2 [expr "$x+20"]
    set y2 [expr "$y+20"]
    if {[string compare $thistool "oval"] == 0} {
        .net create oval $x1 $y1 $x2 $y2 −tag node$nodes −fill $thiscolor
    } elseif {[string compare $thistool "rectangle"] == 0} {
        .net create rectangle $x1 $y1 $x2 $y2 −tag node$nodes −fill $thiscolor
    } .net bind node$nodes <Enter> "beginmove %x %y"
    .net bind node$nodes <Leave> "dothisop node$nodes %x %y"
    .net bind node$nodes <ButtonPress> "dothisop node$nodes %x %y"
    .net bind node$nodes <B3-Motion> "dothisop node$nodes %x %y"
}

proc dothisop {item x y} {
    global thisop
    if {[string compare $thisop "shrink"] == 0} {
        altersize $item $x $y 0.5
    } elseif {[string compare $thisop "grow"] == 0} {
        altersize $item $x $y 2.0
    }
```
Routines

```tcl
proc setcircletool {} {
    global thistool thiscolor
    set thistool oval
    .status delete statusthingy
    .status create oval 10 10 37 37 -tag statusthingy -fill $thiscolor
}

proc setsquaretool {} {
    global thistool thiscolor
    set thistool rectangle
    .status delete statusthingy
    .status create rectangle 10 10 37 37 -tag statusthingy -fill $thiscolor
}

proc setanewcolor {} {
    global thiscolor
    if {([string compare $thiscolor "black"] == 0)} {
        set thiscolor green
    } elseif {([string compare $thiscolor "green"] == 0)} {
        set thiscolor blue
    } elseif {([string compare $thiscolor "blue"] == 0)} {
        set thiscolor red
    } elseif {([string compare $thiscolor "red"] == 0)} {
        set thiscolor orange
    } elseif {([string compare $thiscolor "orange"] == 0)} {
        set thiscolor black
    }

    .status itemconfigure statusthingy -fill $thiscolor
}
```
End of Tk intro

Next assignment will be a Tk programming one...

and now...
Java and JFC

✔ Java - applications across a network

✔ Native-code interpreter for Java code.

✔ Core functions called JFC (Java Foundation Classes).

✔ JFC for GUIs, accessibility, 2D drawing...
AWT - the Abstract Windowing Toolkit.

AWT provides buttons, frames, dialogs...

Implemented in native code in the Java interpreter.
Swing

✔ Not implemented in native code

✔ Implemented in AWT.

✔ Swing and AWT coexist
Swing advantages

Consistent look-and-feel
Pluggable look-and-feel
High-level widgets
How not to...

✘ Watch interpreter overhead

✘ Internal thread scheduling
Getting started

http://java.sun.com/j2se/1.4.2/download.html

j2sdk1.4.2


Netbeans+j2sdk

http://www.comp.nus.edu.sg/~cs3283/ftp/Java/OpenAPIs/
Find the file called SwingSet2.jar. Then try:

```
java -jar SwingSet2.jar
```
Swing programming

✔ Use same strategy as used in Tcl/Tk.

✔ A good book that covers this material in detail is 

The JFC Swing Tutorial 
by Kathy Walrath and Mary Campione.
The toplevel components provided by Swing are:

**JApplet** - for applets within web pages

**JDialog** - for dialog boxes

**JFrame** - for building applications

All other Swing components derive from the **JComponent** class.
JComponent

**Tool tips** - little windows with explanations

**Pluggable look and feel** - as described

**Layour management** - items within the component

**Keyboard action management** - Hot keys and so on.

And other facilities

Swing implements an MVC architecture.
Pluggable look and feel
Pluggable look and feel
Pluggable look and feel
Pluggable look and feel

If you wished to use the WinXX look-and-feel, in the main of your application, you can make the following call:

```java
UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAndFeel");
```
Example application

```java
code t2.java

public class t2 extends javax.swing.JFrame {
    public t2() {
        initComponents();
    }
    private void initComponents() {
        jLabel2 = new javax.swing.JLabel();
        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent evt) {
                exitForm(evt);
            }
        });
        jLabel2.setText("Hello Singapore!");
        jLabel2.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        getContentPane().add(jLabel2, java.awt.BorderLayout.CENTER);
        pack();
    }
    private void exitForm(java.awt.event.WindowEvent evt) {
        System.exit(0);
    }
    public static void main(String args[]) {
        new t2().show();
    }
    private javax.swing.JLabel jLabel2;
}
```
Example application

✔ Instantiates a JLabel, and sets the text field.

✔ Call to getContentPane returns the contentPane object for the frame - this is a generic AWT container for components associated with each JFrame.

✔ The addWindowListener call is from java.awt.Window.
Example application

Hello Singapore!
Example applet

```java
public class HelloWorldApp extends javax.swing.JApplet {
    public HelloWorldApp() {
        initComponents();
    }
    private void initComponents() {
        JLabel1 = new javax.swing.JLabel();
        JLabel1.setText("Hello Singapore!");
        JLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        getContentPane().add(JLabel1, java.awt.BorderLayout.CENTER);
    }
    private javax.swing.JLabel jLabel1;
}
```
Example applet

✔ This code follows the same structure,

✔ Class extends a JApplet instead of a JFrame.

✔ Compile to get a HelloWorldApp.class file,

✔ Has to be referenced in a web page
Example applet

CODE LISTING

HelloWorldApp.txt

```html
The HelloWorld Applet

<EMBED type="application/x-java-applet;version=1.1.2"
    java_CODE="HelloWorldApp.class"
    java_ARCHIVE="applets.jar"
    WIDTH=400
    HEIGHT=50 />
</HTML>
```
Example applet
/* This method is called from within the constructor to
 * initialize the form.
 * WARNING: Do NOT modify this code. The content of this method
 * is always regenerated by the Form Editor.
 */

private void initComponents()
{
    JLabel2 = new javax.swing.JLabel();
    addWindowListener(new java.awt.event.WindowAdapter()
    {
        public void windowClosing(java.awt.event.WindowEvent e)
        {
            exitForm(e);
        }
    });
    JLabel2.setText("Hello Singapore!");
    JLabel2.setHorizontalAlignment(java.awt.SwingConstants.CENTER);
    getContentPane().add(JLabel2, java.awt.BorderLayout.CENTER);
    pack();
}
In this module, we introduced the following topics:
Tool sets for Java/Swing
The relationship between JFC, Java and Swing.
Simple first programs
Chapter 8

Java continued
MCQ test next week
This time
This place
Closed book
✔ Assignment #2 is for groups of 3

✔ Like extended version of tkpaint, but has

✔ menus
✔ persistence
✔ compound objects
Last week

- Tool sets for Java/Swing
- The relationship between JFC, Java and Swing.
- Simple first programs
This week

- Heirarchy
- Layout managers
- Simple first programs
Containment hierarchy

Top level provides panes for descendants to paint themselves

Control-Shift-F1 to view

t2[frame0,954,518,126x43,layout=java.awt... 
  javax.swing.JRootPane[,4,24,118x15,la... 
    javax.swing.JPanel[null.glassPane,... 
      javax.swing.JLayeredPane[null.laye... 
        javax.swing.JPanel[null.content... 
          javax.swing.JLabel[,0,0,118x...]
Containment hierarchy
Containment hierarchy

The glass pane: Intercepts input events for the root pane.

The layered pane: Serves to position its contents, which consist of the content pane and the optional menu bar.

The content pane: The container of the root pane’s visible components, excluding the menu bar.

The menu bar: The home for the root pane’s container’s menus.
### Containment Heirarchy

<table>
<thead>
<tr>
<th>Level</th>
<th>Container</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top-level</td>
<td>JFrame, JApplet, JDialog</td>
</tr>
<tr>
<td>Mid-level</td>
<td>JPanel, JScrollPane, JTabbedPane</td>
</tr>
<tr>
<td>Component-level</td>
<td>JButton, JLabel, ...</td>
</tr>
</tbody>
</table>
Containment hierarchy

Every GUI component must be part of a containment hierarchy\(^4\).

Each top-level container has

- a content pane, and an

- optional menu bar

\(^4\)To view the containment hierarchy for any frame or dialog, click its border to select it, and then press Control-Shift-F1. A list of the containment hierarchy will be written to the standard output stream.
Containment hierarchy

Java/Swing components are added to either the content pane or the menu bar.
Every component must be placed somewhere in this containment hierarchy, or it will not be visible.
Layout management

✔ Every container has a default layout manager
✔ It may be over-ridden
✔ A range of layout managers supplied
✔ These are AWT components, not Swing
BorderLayout

BorderLayout is the default layout manager for every content pane, and assists in placing components in the north, south, east, west, and center of the content pane.

```java
contentPane.add(new JButton("B1"), BorderLayout.NORTH);
```
BoxLayout

BoxLayout puts components in a single row or column. Here is code to create a centered column of components:

```java
pane.setLayout(new BoxLayout(pane, BoxLayout.Y_AXIS));
pane.add(label);
pane.add(Box.createRigidArea(new Dimension(0,5)));
pane.add(...);
```
CardLayout is for when a pane has different components at different times. You may think of it as a stack of same-sized cards.

cards = new JPanel();
cards.setLayout(new CardLayout());
cards.add(p1, BUTTONPANEL);
cards.add(p2, TEXTPANEL);
You can choose the top card to show:

```
CardLayout cl = (CardLayout)(cards.getLayout());
cl.show(cards, (String)evt.getItem());
```
Creating menus

The menu classes are descendants of `JComponent`, and may be used in any higher-level container class (`JApplet` and so on).
Creating menus

```java
public class menutest extends javax.swing.JFrame {
    public menutest() {
        initComponents();
    }
    private void initComponents() {
        jMenuBar1 = new javax.swing.JMenuBar();
        jMenu1 = new javax.swing.JMenu();
        jMenuItem1 = new javax.swing.JMenuItem();
        jMenuItem2 = new javax.swing.JMenuItem();
        jMenuItem3 = new javax.swing.JMenuItem();
        jMenu2 = new javax.swing.JMenu();
        jMenuItem4 = new javax.swing.JMenuItem();
        jMenu1.setText("File");
        jMenuItem1.setText("Open");
        jMenuItem1.addActionListener(
            new java.awt.event.ActionListener() {
                public void actionPerformed(java.awt.event.ActionEvent evt) {
                    jMenuItem3ActionPerformed(evt);
                }
            });
        jMenuItem2.setText("Close");
        jMenuItem3.addEventListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jMenuItem3ActionPerformed(evt);
            }
        });
        jMenuItem3.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jMenuItem3ActionPerformed(evt);
            }
        });
        jMenuItem1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jMenuItem3ActionPerformed(evt);
            }
        });
        jMenu1.add(jMenuItem1);
        jMenu1.add(jMenuItem2);
        jMenu1.add(jMenuItem3);
        jMenu1.setText("File");
        jMenuItem1.setText("Open");
        jMenuItem2.setText("Close");
        jMenuItem3.addActionListener(
            new java.awt.event.ActionListener() {
                public void actionPerformed(java.awt.event.ActionEvent evt) {
                    jMenuItem3ActionPerformed(evt);
                }
            });
        jMenuItem3.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jMenuItem3ActionPerformed(evt);
            }
        });
        jMenuItem1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jMenuItem3ActionPerformed(evt);
            }
        });
        jMenu2.setText("Edit");
        jMenuItem4.setText("Cut");
        jMenuItem4.addActionListener(
            new java.awt.event.ActionListener() {
                public void actionPerformed(java.awt.event.ActionEvent evt) {
                    jMenuItem3ActionPerformed(evt);
                }
            });
        jMenu2.add(jMenuItem4);
        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent evt) {
                exitForm(evt);
            }
        });
        setJMenuBar(jMenuBar1);
        pack();
    }
    private void jMenuItem3ActionPerformed(java.awt.event.ActionEvent evt) {
        System.exit(0);
    }
    private void exitForm(java.awt.event.WindowEvent evt) {
        System.exit(0);
    }
    public static void main(String args[]) {
        new menutest().show();
    }
}
```
Menus

The end result is:

![Menu Example](image.png)
Threads in Swing

✔ Java supports multi-threading

✔ We may have critical sections

✔ To create threads use **SwingWorker** or **Timer**.
Threads

Most Swing components are not thread safe - this means that if two threads call methods on the same Swing component, the results are not guaranteed.

The single-thread rule:

Swing components accessed by only one thread at a time.
A particular thread, the event-dispatching thread, is the one that normally accesses Swing components. To get access to this thread from another thread we can use `invokeLater()` or `invokeAndWait()`.
Threads

Many applications do not require threading, but if you do have threads, then you may have problems debugging your programs. However, you might consider using threads if:

Your application has to do some long task, or wait for an external event, without freezing the display.

Your application has to do something at fixed time intervals.
Implementing threads

The following two classes are used to implement threads:

1. **SwingWorker**\(^5\): To create a thread

2. **Timer**: Creates a timed thread

\(^5\)If you find that your distribution does not include SwingWorker.class, download and compile it.
To use SwingWorker, create a subclass of it, and in the subclass, implement your own `construct()` method. When you instantiate the SwingWorker subclass, the runtime environment creates a thread but does not start it. The thread starts when you invoke `start()` on the object.
Example

Here’s an example of using **SwingWorker** from the tutorial - an image is to be loaded over a network (given a URL). This may of course take quite a while, so we don’t block our main thread - (if we did this, the GUI may freeze).
SwingWorker example

```java
private void loadImage(final String imagePath,
                       final int index) {
    final SwingWorker worker = new SwingWorker() {
        final ImageIcon icon = null;
        public Object construct() {
            icon = new ImageIcon(getURL(imagePath));
            return icon;
        }
        public void finished() {
            Photo pic = (Photo)pictures.elementAt(index);
            pic.setIcon(icon);
            if (index == current)
                updatePhotograph(index, pic);
        }
    }
    worker.start();
}
```
The **Timer** class is used to repeatedly perform an operation. When you create a **Timer**, you specify its frequency, and you specify which object is the listener for its events. Once you start the timer, the action listener’s `actionPerformed()` method will be called for each event.
The event-dispatching thread is the main event-handling thread. It is normal for all GUI code to be called from this main thread, even if some of the code may take a long time to run. However - we have already mentioned that we should not delay the event-dispatching thread.

Swing provides a solution to this - the `InvokeLater()` method may be used to safely run code in the event-dispatching thread.
InvokeLater

The method requests that some code be executed in the event-dispatching thread, but returns immediately, without waiting for the code to execute.

```java
Runnable doWorkRunnable = new Runnable() {
    public void run() { doWork(); }
};
SwingUtilities.invokeLater(doWorkRunnable);
```
Handling events

Actions associated with Java/Swing components raise events - moving the mouse or clicking a JButton all cause events to be raised. The application program writes a listener method to process an event, and registers it as an event listener on the event source. There are different kinds of events, and we use different kinds of listener to act on them.
### Listener types

<table>
<thead>
<tr>
<th>Action</th>
<th>Listener type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Button click</td>
<td>ActionListener</td>
</tr>
<tr>
<td>A window closes</td>
<td>WindowListener</td>
</tr>
<tr>
<td>Mouse click</td>
<td>MouseListener</td>
</tr>
<tr>
<td>Mouse moves</td>
<td>MouseMotionListener</td>
</tr>
<tr>
<td>Component becomes visible</td>
<td>ComponentListener</td>
</tr>
<tr>
<td>Keyboard focus</td>
<td>FocusListener</td>
</tr>
<tr>
<td>List selection changes</td>
<td>ListSelectionListener</td>
</tr>
</tbody>
</table>
Listeners

The listener methods are passed an event object which gives information about the event and identifies the event source.
Event handlers

When you write an event handler, you must do the following:

- Specify a class
- Register an instance of the class as a listener
- Implement the methods
Specify class

Specify a class that either implements a listener interface or extends a class that implements a listener interface.

```java
public class MyClass implements ActionListener {
  ...
}
```
Register it

Register an instance of the class as a listener upon the components.

```java
Component.addActionListener(instanceOfMyClass);
```
Implement method

Implements the methods in the listener interface.

```java
public void actionPerformed(ActionEvent e) {
    ...//code that reacts to the action...
}
```
Event handling

Make sure that your event handler code executes quickly, or your program may seem to be slow.
In the sample code given so far, we have used window listeners to react if someone closes a window, but not to capture other sorts of events.
Handling events

```java
public class CheckBoxDemo extends JPanel {
    JCheckBox chinButton;
    JCheckBox glassesButton;
    StringBuffer choices;
    JLabel pic;
    public CheckBoxDemo () {
        chinButton = new JCheckBox("Chin");
        glassesButton = new JCheckBox("Glasses");
        chinButton.addItemListener(myListener);
        glassesButton.addItemListener(myListener);
        choices = new StringBuffer("−−h");
        pic = new JLabel(new ImageIcon("geek−" + choices.toString() + ".gif"));
        pic.setToolTipText(choices.toString());
        checkPanel = new JPanel();
        checkPanel.setLayout(new GridLayout(0, 1));
        checkPanel.add(chinButton);
        checkPanel.add(glassesButton);
        setLayout(new BorderLayout());
        add(checkPanel, BorderLayout.WEST);
        add(pic, BorderLayout.CENTER);
        frame = new JFrame("CheckBoxDemo");
        frame.addWindowListener(new WindowAdapter () {
            public void windowClosing (WindowEvent e) {
                System.exit (0);
            }
        });
    }
    class CheckBoxListener implements ItemListener {
        public void itemStateChanged (ItemEvent e) {
            int index = 0;
            char c = '−';
            Object source = e.getItemSelectable ();
            if (source == chinButton) {
                index = 0;
                c = 'c';
            } else if (source == glassesButton) {
                index = 1;
                c = 'g';
            } if (e.getStateChange () == ItemEvent.DESELECTED) {
                choices.setCharAt(index, c);
            } pic.setIcon (new ImageIcon("geek−" + choices.toString () + ".gif"));
            pic.setToolTipText (choices.toString ()�
        }
    }
    public static void main (String s[]) {
        JFrame frame = new JFrame("CheckBoxDemo");
        frame.addWindowListener (new WindowAdapter () {
            public void windowClosing (WindowEvent e) {
                System.exit (0);
            }
        });
    }
}
```
Example code

When you change either checkbox, an `itemListener` responds to the event and changes the graphic.
In this module, we introduced the following topics:
The containment hierarchy
Layout managers
Menus
Threading
Event handling
Chapter 9

Module 7
MCQ Test

✔ Closed book

✔ Closed computer

✔ 20 questions

✔ Leave if finished, but come back at 1:00 for lecture
Any questions?
✔ No more lecture material, but

✔ I will respond to questions with material as needed

✔ Following few weeks will have Java/Swing centered questions, and

✔ Assignment 3 (announced next week) will require Java/Swing, so get some practice in...
Common Gateway Interface

✔ CGI is a standard for helping web servers run external programs,

✔ and return dynamic web pages.

For example, a simple dynamic web page might return the current date and time, calculated by running the 'date' program, and formatting the results as a web page.
CGI script

**CODE LISTING**  
**mydate.cgi**

```bash
#!/bin/sh

cat <<EOM1
Content-type: text/html

<HTML><HEAD>
<TITLE>Output of data in HTML from CGI script</TITLE>
</HEAD><BODY>
<H1>Date:</H1>
EOM1
date

cat <<EOM2
</BODY></HTML>
EOM2
```
When this script is placed in the directory `public_cgi` in your home directory on one of the UNIX systems, then you may refer to

```
http://www-cgi.comp.nus.edu.sg:8000/~yourid/mydate.cgi
```
Date:

Thu Mar 21 09:28:16 GMT–8 2002
<HTML><HEAD></HEAD><TITLE>Output of data in HTML from CGI script</TITLE></HEAD><BODY><H1>Date:</H1>Thu Mar 21 09:28:16 GMT-8 2002</BODY></HTML>
CGI scripts

✔ No requirement for CGI program to be a shell script.

✔ *Perl* is very commonly used in this role.

✔ It should not take too long to process.
Environment variables

```perl
#!/usr/local/bin/perl

print "Content-type: text/html\n\n";
print <<EndOfHTML;
<html><head><title>Print Environment</title></head>
<body>
EndOfHTML

foreach $key (sort(keys %ENV)) {
   print "$key = $ENV{$key}<br>
";
}
print "</body></html>";
```
Environment variables

DOCUMENT_ROOT = /usr/local/apache/htdocs
GATEWAY_INTERFACE = CGI/1.1
HTTP_USER_AGENT = Mozilla/4.79 [en] (X11; U; Linux 2.2.16 i686)
...
QUERY_STRING =
...
TZ = Singapore
The form contents are found inside an environment variable called `QUERY_STRING`, as a series of `name/value` pairs. This mechanism is known as the `GET` mechanism, and a typical URL would look like this:

```
.../myform.cgi?name1=value1&name2=value2
```
An alternative mechanism is the **POST** mechanism, in which the **STDIN** of the CGI program is used to process the form data.
CGI form

CODE LISTING

form.html

```html
<html><head>Simple form</head>
<body>
<form action="env.cgi" method="GET">
    First Name:  <input type="text" name="First" size=30><p>
    Last Name:  <input type="text" name="Last" size=30><p>
    <select name="Home">
        <option>Singapore</option>
        <option>Malaysia</option>
        <option>Indonesia</option>
        <option>New Zealand</option>
        <option>The rest of the world!</option>
    </select>
    <input type="submit">
</form>
</body></html>
```
CGI form

Simple form

First Name: Hugh

Last Name: Anderson

New Zealand  Submit Query
When the form is submitted, the QUERY_STRING looks like this:

```
QUERY_STRING = First=Hugh&Last=Anderson&Home=New+Zealand
```

Within a CGI program, this series of name-value pairs may be used to return a dynamic web page based on this form data. Perl is a particularly useful language to use in this context - the QUERY_STRING can be split quickly into its component parts.
There are security issues with unrestricted CGI programs - since they run powerful programs (like perl and csh) with arbitrary parameters, they may be a source of (hacker) intrusion.

It is for this reason that CGI usage is restricted here at NUS.
PHP

✔ PHP is a server-side scripting language.

✔ It looks very like standard HTML scripts, but the server automatically interprets the PHP.

✔ There are no enhancements needed for browsers.
The two tags <<?php and ?> start and end a PHP script, and identify a PHP code segment.

The PHP code itself is a reasonably powerful programming language similar to Java, C and Perl, with functions, variables and so on.

PHP stands for PHP - Hypertext Preprocessor, a recursive acronym.
✔ Particularly useful to access databases.

✔ It is common to pair up PHP with MySQL, but PHP is not limited to one database type.

✔ For example if you wish to use PHP to access a Microsoft SQL server, you can install the ODBC support in the server machine, and access the server directly.
<?php

... 
mysql_pconnect("host","user","password")
    or die("Unable to connect to SQL server");
mysql_select_db("dbasename")
    or die("Unable to select database");
$numguests = mysql_query("SELECT COUNT(*) FROM guests")
    or die("Select Failed!");
...

?>
PHP security

✔ PHP suffers less from the security issue than perl or csh CGI scripts do.
/* @(#)Lissajous.java
 * Original version was written in 0.4 95/04/09
 * by Hugh Anderson for HotJava browser.
 * Updated by L. Gladney to Java 1.0 JDK on 4/13/97.
 * Patrick Chan (chan@scndprsn.Eng.Sun.COM) has suggested that it
 * would be nice if every point had a different display, so mouse
 * X motion now controls the ratio of frequencies, and mouse Y motion
 * controls the amplitude. */

import java.applet.Applet;
import java.awt.*;

public class Lissajous extends Applet implements Runnable {
    Thread animate = null;
    double pi = 3.14159265359;
    int fy = 100;
    int diffx = 0;
    int amp = 50, phase = 0;
    int fx = 50;
    int fy = 100;
    int diffx = 0;
    int delay = 50;
    int phase = 0;

    public void init() {
        resize(200, 200);
        // resize to fixed width, height
    }

    public void paint(Graphics g) {
        int X, Y, YY = 0, lastx = 0, lasty = 0, temp = 0, rev = 0;
        g.drawRect(0, 0, size().width - 1, size().height - 1);
        // outline
        if (fy < fx) {
            temp = fx;
            fx = fy;
            fy = temp;
            rev = 1;
        }
        for (int x = 0; x <= 360; x += 4) {
            Y = (int) (amp * Math.sin(x * 2.0 * pi / 360.0));
            X = (int) (amp * Math.sin(YY * 2.0 * pi / 360.0));
            if (rev == 1) { g.drawLine(lastx + 100, lasty + 100, X + 100, Y + 100); }
            else { g.drawLine(lasty + 100, lastx + 100, Y + 100, X + 100); }
            lastx = X;
            lasty = Y;
            if (rev == -1) { g.drawLine(lastx + 100, lasty + 100, X + 100, Y + 100); }
        }
        phase = YY;
        // Fix an error ... phase shouldn’t increase forever..... */
        if (phase < 0) { phase += 360; }
        if (phase >= 360) { phase -= 360; }
        g.drawString(fx + " : " + fy, 10, 20);
    }
}

CODE LISTING Lissajous1.java

Lissajous1.java

public class Lissajous extends Applet implements Runnable {
    Thread animate = null;
    double pi = 3.14159265359;
    int fy = 100;
    int diffx = 0;
    int amp = 50, phase = 0;
    int delay = 50;
    int phase = 0;

    public void init() {
        resize(200, 200);
        // resize to fixed width, height
    }

    public void paint(Graphics g) {
        int X, Y, YY = 0, lastx = 0, lasty = 0, temp = 0, rev = 0;
        g.drawRect(0, 0, size().width - 1, size().height - 1);
        // outline
        if (fy < fx) {
            temp = fx;
            fx = fy;
            fy = temp;
            rev = 1;
        }
        for (int x = 0; x <= 360; x += 4) {
            Y = (int) (amp * Math.sin(x * 2.0 * pi / 360.0));
            X = (int) (amp * Math.sin(YY * 2.0 * pi / 360.0));
            if (rev == 1) { g.drawLine(lastx + 100, lasty + 100, X + 100, Y + 100); }
            else { g.drawLine(lasty + 100, lastx + 100, Y + 100, X + 100); }
            lastx = X;
            lasty = Y;
            if (rev == -1) { g.drawLine(lastx + 100, lasty + 100, X + 100, Y + 100); }
        }
        phase = YY;
        // Fix an error ... phase shouldn’t increase forever..... */
        if (phase < 0) { phase += 360; }
        if (phase >= 360) { phase -= 360; }
        g.drawString(fx + " : " + fy, 10, 20);
    }
}
public void run() {
    while (true) {
        repaint();
        try {
            Thread.currentThread().sleep(delay); // delay
        }
        catch (Exception e) { };
    }
}

public void start() {
    if (animate == null) {
        animate = new Thread(this);
        animate.start();
    }
}

public void stop() {
    if (animate != null) {
        animate.stop();
        animate = null;
    }
}

public boolean mouseDown(Event e, int x, int y) {
    Graphics gc;
    gc = getGraphics();
    diffx = fx-x;
    System.out.println("Got a mouse event at " + x + "," + y);
    return true;
}

public boolean mouseDrag(Event e, int x, int y) {
    fx = x+diffx;
    if (fx <= 0) { fx = 1; }
    amp = y;
    return true;
}

public String getAppletInfo() {
    return "Lissajous by Hugh Anderson/Larry Gladney ";
}

public String[][] getParameterInfo() {
    String [][] info = {
        ["delay ","int ", "delay, default=50"]
    };
    return info;
}
Java

This code may be found at

http://olddept.physics.upenn.edu/courses/gladney/minicourse/lectures/lecture2.html

or locally at

http://www.comp.nus.edu.sg/~hugh/Lissajous/Lissajous.html
Java

You can find the source code [here](http://example.com).
Summary of topics

In this module, we introduced the following topics:
Web-based application architectures
CGI, PHP and Java
Chapter 10

Module 8
Visualization

✔ In visualization, we are concerned with *exploration*

✔ In computer-graphics, we are concerned with *rendering*
The use of 3D

✔ Analog with *real-world* physics.

✔ 10-fold improvement in item density with 3D.

✔ Familiarity with spatial location helps reduce visual clutter.

✔ Need sufficient visual cues.
Use of 3D
SGI in-house graphics system

Now a widely accepted graphics standard

Standard on UNIX and Windows

API supports rendering, buffering, anti-aliasing, shading, colouring, texture-mapping, a display list, Z-buffering...
OpenGL Application
#include <GL/glut.h>

void
Teapot (long grid)
{
    /* ... code to construct drawlist of teapot here. */
}

static void
Init (void)
{
    glEnable (GL_DEPTH_TEST);
    glLightModelfv (GL_LIGHT_MODEL_LOCAL_VIEWER, local_view);
    /* Lighting model, materials... */
}

static void
SpecialKey (int key, int x, int y)
{
    switch (key) {
    case GLUT_KEY_UP:
        rotX -= 20.0;
        glutPostRedisplay ();
        break;
    /* Move in other directions */
    }
}

static void
Draw (void)
{
    glClear (GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glPushMatrix ();
    /* ... translations ... */
    glCallList (teaList);
    glPopMatrix ();
    glutSwapBuffers ();
}

int
main (int argc, char **argv)
{
    glutInit (&argc, argv);
    type = GLUT_RGB | GLUT_DEPTH;
    type |= (doubleBuffer) ? GLUT_DOUBLE : GLUT_SINGLE;
    glutInitDisplayMode (type);
    glutInitWindowSize (300, 300);
    glutCreateWindow ("TeaPot");
    Init ();
    glutReshapeFunc (Reshape);
    glutKeyboardFunc (Key);
    glutSpecialFunc (SpecialKey);
    glutDisplayFunc (Draw);
    glutMainLoop ();
}
Java3D & VTK

✔ 3D OO toolkits

✔ VTK is open source
  ✔ C++ class library, and
  ✔ interface layers for Tcl/Tk, Java, and Python.
Network traffic application

To help answer questions such as the following:

Which segments carry the most traffic?
Which sections of the network are down?
At what times, and where do traffic bottlenecks occur?

...
Application elements

Following elements are represented:
Background: - to convince the viewer that the display is *three dimensional*...
Nodes: - a computer, a network device...
Traffic: - the amount of traffic flow...
Protocol: - the *type* of traffic...

...
Node representation
## Rendering speed

<table>
<thead>
<tr>
<th>Machine</th>
<th>Rendering speed</th>
<th>Computer (a)</th>
<th>Computer (b)</th>
<th>Computer (c)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workstation</td>
<td>485,000 Δ/sec</td>
<td>0.485 frames/sec</td>
<td>11.5 frames/sec</td>
<td>69 frames/sec</td>
</tr>
<tr>
<td>PC1</td>
<td>30,000 Δ/sec</td>
<td>0.03 frames/sec</td>
<td>0.71 frames/sec</td>
<td>4.3 frames/sec</td>
</tr>
<tr>
<td>PC2</td>
<td>11,000 Δ/sec</td>
<td>0.011 frames/sec</td>
<td>0.26 frames/sec</td>
<td>1.6 frames/sec</td>
</tr>
</tbody>
</table>
Some representation methods allow different levels of detail.

In VRML an object may be represented in different ways depending on how large it is.

If the object is near you, it could be represented in detail, but if it is a long way away, the representation could be as simple as a coloured square.
LOD {
  range [20]
  level [
    Shape{ #full detail 16 sided cone
      appearance Appearance { material Material { ... } }
      geometry Extrusion{ ... }.
    }
    Shape{ #low detail 4 sided cone
      appearance Appearance { material Material { ... } }
      geometry Extrusion{ ... }
    }
  ]
}

If the distance from the user to the object is smaller than the first range value specified, then the first version is drawn.

If the distance is greater than the last range specified, the last version is drawn.
Traffic and protocols

Draw a line between nodes.
A line indicates source and destination, but not the *amount* of traffic:

1. Colour coding (black through red to white for maximum traffic),

2. Line width, and

3. The length of partial lines, as discussed in Eick’s papers.
Partial lengths
Trend representation

✔ Graphing

✔ 4D visualization methods

✔ Encode previous on-top-of the current - visual echoes.
Systems

✔ CosmoPlayer VRML viewer,

✔ geomview.

The visualization is not dependant on the navigation or implementation method.
Aggregation Nodes

CS3283 - Hugh Anderson's notes.
Implementation #1

✔ A data collector

✔ A web page with...

✔ Java program loaded as an applet, and a

✔ VRML view of the network.
Web page

<html><head> <title>Sample 3DVNT Page</title> </head>
<center><H1>Sample 3DVNT Page</H1></center>
<center> <embed src="root.wrl" height="600" width="700"> </center>
<center> <applet code="View1.class" width="100" height="10" mayscript>
<PARAM name="segment" value="MACS">
<PARAM name="port" value="9876">
<PARAM name="host" value="opo.usp.ac.fj"> </applet> </center>
OK?
</html>
PROTO CLUSTER [] { ... }  # Cluster definition
PROTO KEYBOARD [] { ... }  # Keyboard definition
PROTO SCREEN [] { ... }  # Screen definition
PROTO GLOBE [] { ... }  # Traffic sphere definition
# Some setting up declarations
Background { skyColor .4 .66 1 }
NavigationInfo { type [ "EXAMINE", "ANY" ] speed 400 }
Viewpoint { position 0 400 0 orientation 0 1 0 4 description "Camera 1" }
# Lines, floors and roofs
DEF LINES Transform { ... }
DEF FLOORS Transform { ... }
DEF ROOFS Transform { ... }
# and then the nodes
DEF node1 Transform { ... }
DEF node2 Transform { ... }
# ... and so on ...
VRML nodes

DEF node1 Transform {
  translation 4350 150 4365
  rotation 0 1 0 4.71238
  children [
    KEYBOARD {}
    SCREEN {}
    DEF node1box Transform {
      children [
        Shape { ... }
      ]
    }
    DEF node1sphere Transform {
      scale 1 1 1
      children [
        Shape { ... }
      ]
    }
  ]
}

CS3283 - Hugh Anderson's notes.
// using the VRML External Interface.

```java
import java.applet.*;
import java.awt.*;
import java.util.*;
import vrml.external.field.*;
import vrml.external.exception.*;
import vrml.external.Node;
import vrml.external.Browser;
import java.io.*;

public class View1 extends Applet {

    public static final int DEFAULT_PORT = 9877;
    Browser browser;
    Socket s = null;
    DataInputStream in = null;
    String line;

    public void init() {
        System.out.println("Test init()...");
    }

    void SocketStart() throws java.io.IOException {
        String port = this.getParameter("port");
        int p = Integer.parseInt(port);
        try {
            String host = getCodeBase().getHost();
            System.out.println("Request came from: " + host);
            s = new Socket(host, p);
        }
        catch (UnknownHostException e) {
            System.out.println("No socket:" + e);
        }
    }

    public void start() {
        int count=0;
        Node node2sphere=null;
        Node appear=null;
        EventInSFVec3f[] scalein=new EventInSFVec3f[100];
        EventInSFColor[] appears=new EventInSFColor[100];
        float[] val = new float[3];
        int[] lastval = new int[100];
        int n;
        String id,vl;
        while (count != 100) {
            scalein[count] = null;
            appears[count] = null;
            count=count+1;
        }
        try {
            SocketStart();
        }
        catch (java.io.IOException e) {
            System.out.println("No socket:" + e);
        }
        System.out.println("Test start()...");
        browser = (Browser) vrml.external.Browser.getBrowser(this);
        System.out.println("Got the browser:" + browser);
        count = 0;
        try {
            in = new DataInputStream(s.getInputStream());
        }
    }
}
```
```java
while (true) {
    line = in.readLine();
    if (line == null) {
        System.out.println("Server closed connection.");
        break;
    }
    if (line.regionMatches(0, "n", 0, 1)) {
        if (n == line.indexOf("=", 1)) {
            id = line.substring(2, n);
            System.out.println(">>>" + id + "<<<");
            Integer a = Integer.valueOf(id);
            Integer b = Integer.valueOf(vl);
            if (scalein[a.intValue()] == null) {
                try {
                    node2sphere = browser.getNode("node" + id + "sphere");
                    System.out.println("Got the sphere node: " + node2sphere);
                } catch (InvalidNodeException e) {
                    System.out.println("PROBLEMS! node2sphere: " + e);
                }
                try {
                    tEventIn("scale");
                    tValue();
                } catch (InvalidNodeException e) {
                    System.out.println("PROBLEMS! (scalein): " + e);
                }
                try {
                    appear = browser.getNode("node" + id + "boxcolor");
                    System.out.println("Got the Boxcolor node: " + appear);
                } catch (InvalidNodeException e) {
                    System.out.println("PROBLEMS! appearance: " + e);
                }
                try {
                    appears[a.intValue()] = (EventInSFColor) appear.getEventIn("set_diffuseColor");
                    System.out.println("Got the Boxcolor color node: " + appears[a.intValue()]);
                } catch (InvalidNodeException e) {
                    System.out.println("PROBLEMS! appearance color: " + e);
                }
                if (b.intValue() == -1) {
                    val[0] = (float) 1.0;
                    val[1] = (float) 1.0;
                    val[2] = (float) 1.0;
                } else {
                    val[0] = (float) (b.intValue() * 20) + 1;
                    val[1] = (float) (b.intValue() * 20) + 1;
                    val[2] = (float) (b.intValue() * 20) + 1;
                }
                scalein[a.intValue()].setValue(val);
                if (b.intValue() == 0) {
                    if (lastval[a.intValue()] == 0) {
                        lastval[a.intValue()] = 1;
                        val[0] = (float) 0.0;
                        val[1] = (float) 0.0;
                        val[2] = (float) 0.0;
                        appears[a.intValue()].setValue(val);
                    }
                }
            }
        }
    } else {
        // System.out.println("***" + line + "***");
    }
}
```
```java
else {
    if (b.intValue() == 1) {
        val[0] = (float)0.1;
        val[1] = (float)0.1;
        val[2] = (float)0.1;
        appears[a.intValue()].setValue(val);
    }
    else {
        val[0] = (float)0.0;
        val[1] = (float)1.0;
        val[2] = (float)0.0;
        appears[a.intValue()].setValue(val);
    }
}
lastval[a.intValue()] = b.intValue();
// System.out.println(line);
}
```

```java
} catch (IOException e) { System.out.println("Reader: " + e); }
}

public Browser getBrowser() {
    return browser;
}
```
In this module, we introduced the following topics:
Visualization versus computer-graphics
OpenGL
(Briefly) Java3D, VTK
VRML/Java/EAI
Chapter 11

Code Samples
Assignment 2 - Code quality?

#!/usr/local/bin/wish
if {![info exists widgetDemo]} {
    error "This script should be run from the "widget" demo."
}
set w .plot
catch {destroy $w}
toplevel $w
wm title $w "Plot Demonstration"
wm iconname $w "Plot"
positionWindow $w
set c $w.c
label $w.msg −font $font −wraplength 4i −justify left −text "This window ..."
pack $w.msg −side top −fill x
frame $w.buttons
pack $w.buttons −side bottom −fill x −pady 2m
button $w.buttons.dismiss −text Dismiss −command "destroy $w"
button $w.buttons.code −text "See Code" −command "showCode $w"
pack $w.buttons.dismiss $w.buttons.code −side left −expand 1

canvas $c −relief raised −width 450 −height 300
for {set i 0} {$i <= 10} {incr i} {
    set x [expr {100 + ($i*30)}]
    $c create line $x 250 $x 245 −width 2
    $c create text $x 254 −text [expr 10*$i] −anchor n −font $plotFont
}
for {set i 0} {$i <= 5} {incr i} {
    set y [expr {250 − ($i*40)}]
    $c create line 100 $y 105 $y −width 2
    $c create text 96 $y −text [expr $i*50].0 −anchor e −font $plotFont
}
foreach point {{12 56} {20 94} {33 98} {32 120} {61 180} {75 160} {98 223}} {
    set x [expr {100 + (3*[lindex $point 0])}]
    set y [expr {250 − (4*[lindex $point 1])/5}]
    set item [ $c create oval [expr $x−6] [expr $y−6] [expr $x+6] [expr $y+6] −width 1 −outline black −fill SkyBlue2]
    $c addtag point withtag $item
}
$c bind point <Any−Enter> "$c itemconfig current −fill red"
$c bind point <Any−Leave> "$c itemconfig current −fill SkyBlue2"
$c bind point <B1> "plotDown $c %x %y"
$c bind point <B1−Motion> "plotMove $c %x %y"
set plot(lastX) 0
set plot(lastY) 0
proc plotDown {w x y} {
    global plot
    $w dtag selected
    $w addtag selected withtag current
    $w raise current
    set plot(lastX) $x
    set plot(lastY) $y
}

CODE LISTING Page 1/1
lect9.1.tcl
if {[![info exists widgetDemo]} { error "This script should be run from the \"widget\" demo." }

set w .plot
catch {destroy $w}
toplevel $w
wm title $w "Plot Demonstration"
wm iconname $w "Plot"
positionWindow $w

set c $w.c
canvas $c −relief raised −width 450 −height 300
pack $w.c −side top −fill x

set plotFont {Helvetica 18}
$c create line 100 250 400 250 −width 2
$c create text 225 20 −text "A Simple Plot" −font $plotFont −fill brown
for {set i 0} {$i <= 10} {incr i} {
    set x [expr {100 + ($i*30)}]
    $c create line $x 250 $x 245 −width 2
    $c create text $x 254 −text [expr 10*$i] −anchor n −font $plotFont
}
for {set i 0} {$i <= 5} {incr i} {
    set y [expr {250 − ($i*40)}]
    $c create line 100 $y 105 $y −width 2
    $c create text 96 $y −text [expr $i*50].0 −anchor e −font $plotFont
}
foreach point {{12 56} {20 94} {33 98} {32 120} {61 180}} {
    set x [expr {100 + (3*[lindex $point 0])}]
    set y [expr {250 − (4*[lindex $point 1])/5}]
    set item [ $c create oval [expr $x−6] [expr $y−6] [expr $x+6] [expr $y+6] −width 1 −outline black −fill SkyBlue2]
    $c addtag point withtag $item
}
$c bind point <ButtonRelease−1> "$c dtag selected"
$c bind <B1−Motion> "plotMove $c %x %y"

# plotDown −−
# This procedure is invoked when the mouse is pressed over one of the
# data points. It sets up state to allow the point to be dragged.
#
# Arguments:
# w − The canvas window.
# x, y − The coordinates of the mouse press.
proc plotDown {w x y} {
    global plot
    $w dtag selected
    $w addtag selected withtag current
    $w raise current
    set plot(lastX) $x
    set plot(lastY) $y
}
#!/usr/local/bin/wish
#
# This demonstration script creates a canvas widget showing a 2-D
# plot with data points that can be dragged with the mouse.
#

set w .px
catch {destroy $w}
label $w.m −font $font −wraplength 4i −justify left −text "This window ..."
pack $w.m −side top
frame $w.bs
pack $w.bs −side bottom −fill x −pady 2m
button $w.bs.dismiss −text Dismiss −command "destroy $w"
canvas $c −relief raised −width 450 −height 300
pack $w.c −side top −fill x
set pf {Helvetica 18}
$c create line 100 250 400 250 −width 2
$c create line 100 250 100 50 −width 2
for {set i 0} {$i <= 10} {incr i} {
    set x [expr {100 + ($i*30)}]
    $c create line $x 250 $x 245 −width 2
    $c create text $x 254 −text [expr 10*$i] −anchor n −font $pf
}
for {set i 0} {$i <= 5} {incr i} {
    set y [expr {250 − ($i*40)}]
    $c create line 100 $y 105 $y −width 2
    $c create text 96 $y −text [expr $i*50].0 −anchor e −font $pf
}
foreach pt {{12 56} {20 94} {33 98} {32 120} {61 180} {75 160} {98 223}} {
    set x [expr {100 + (3*[lindex $pt 0])}]
    set y [expr {250 − (4*[lindex $pt 1])/5}]
    set im [$c create oval [expr $x-6] [expr $y-6] 
             [expr $x+6] [expr $y+6] −width 1 −outline black 
             −fill SkyBlue2]
    $c addtag pt withtag $im
}
$c bind pt <Any-Enter> "$c itemconfig ct −fill red"
$c bind pt <Any-Leave> "$c itemconfig ct −fill SkyBlue2"
Write code clearly: Edit, document, comment...

##########################################################################
# GetCommandString( x,y,itemID ) : string
# Returns a string that is later executed as a
# command
# The parameters x and y are the current cursor
# position, and itemID is the closest visible
# item on the canvas .canv
# Requires: Uses global variable canvas .canv
# Ensures: Always returns a command of some sort
# Sets global variable ErrorID if there is
# any error...
# Last modified: 12/2/2004 - by Hugh
##########################################################################
Assignment 2 - debugging?

✔ Run wish, and then use source x.tcl

✔ ... then interact with running program...
Assignment 3

3 options:

1. Re-implement YOUR assignment 2

2. A simple (but actually useful) visualization

3. Image library assistant...
Assignment 3 (option a)

✔ The tricky thing is the graphics component

✔ Some help with it...
public void paintComponent(Graphics g) {
    super.paintComponent(g); //paint background
    //Paint a filled rectangle at user's chosen point.
    if (point != null) {
        g.drawRect(point.x, point.y , rectWidth-1, rectHeight-1);
        g.setColor(Color.yellow);
        g.fillRect(point.x+1,point.y+1,rectWidth-2,rectHeight-2);
    }
}
1. Basic/AWT - Abstract Graphics class

2. Java2D
Coordinate system

✔ Upper left of each component is (0,0)

✔ Behind the title bar of a window

✔ Container class has `getInsets` method

✔ Graphics objects contain methods for drawing
Graphics API

✔ Swing components have a method `paintComponent` which takes a graphics object as an argument

```java
public void paintComponent(Graphics g)
```

✔ Overide this to draw your objects.

✔ Also may call the `repaint()` method
Graphics class methods

clearRect(int x, int y, int width, int height);
draw3DRect(int x, int y, int width, int height, boolean raised);
drawImage(Image img, int x, int y, Color bgcolor, ImageObserver observer);
drawLine(int x1, int y1, int x2, int y2);
drawOval(int x, int y, int width, int height);
drawPolygon(int xPoints[], int yPoints[], int nPoints);
drawRect(int x, int y, int width, int height);
drawRoundRect(int x, int y, int width, int height, int arcWidth, int arcHeight);
drawString(String str, int x, int y);
Graphics class methods

```
fill3DRect(int x, int y, int width, int height, boolean raised);
fillArc(int x, int y, int width, int height, int startAngle, int arcAngle);
fillOval(int x, int y, int width, int height);
fillPolygon(int xPoints[], int yPoints[], int nPoints);
fillRect(int x, int y, int width, int height);
fillRoundRect(int x, int y, int width, int height, int arcWidth, int arcHeight);
```
Graphics class methods

Color getColor();
Font getFont();
FontMetrics getFontMetrics();
setColor(Color c);
setFont(Font font);
Graphics API

✔ Use JPanel instead of JComponent

✔ UI delegate (for look-and-feel painting) is called in JPanel

✔ UI delegate not called in JComponent
Text in Graphics API

✔ Note - you paint text using `drawString()`

✔ `getFontMetrics()` to get a `FontMetrics` object

```java
getHeight()
getAscent()
getDescent()
charWidth()
```

✔ and so on...
Assignment 3 (option b)

✔ Start with a large number (>1000000) points to be plotted, explored, displayed.

✔ If only a 1024*768 screen there are <1000000 points on screen.

✔ In some small region with (say) 10*10 points, there might be no difference between a display with 100 dots and one with 100000 dots.
Assignment 3 (option b)

✔ So...

✔ Tile the display

✔ Black and white? Colour?
Assignment 3 (option b)
Assignment 3 (option b)
Assignment 3 (option b)
Assignment 3 (option b)

✔ Must use a slider to change the tiling.

✔ May show different zoom levels, and locations of data

✔ Processing of other tilings in background using threads... (i.e. no pauses)
Assignment 3 (option c)

✔ Java *application* or a Java *applet*

✔ User interface to assist in the management of *large* numbers of images.

✔ Principally display TEXT information (spreadsheet),

✔ May also display small (thumbnail) versions of the images
Assignment 3 (option c)

✔ Database

✔ Special purpose editor for ...

✔ classifying,
✔ annotating and
✔ querying a large number of images.
Assignment 3 (option c)

✔ Image DSCN0100.JPG (Tim at a party): It is in

  ● “Friends”
  ● “Trip to NZ in Dec 2003”, which is itself in the section “Trips”
  ● “Hooligans”

✔ Main screen shows a list of images.
Assignment 3 (option c)

Editable and fixed annotation fields:

- The date and time the image was entered into the section (not editable).
- A unique identifier for the image
- A scrollable text box with (say) 5 visible lines of text description.
Assignment 3 (option c)

Minimum flow of operation:

1. create, locate and delete new sections,

2. import image(s), using selection or cut and paste.

3. edit image/section information annotations,

4. save and load new databases,

5. query the system with a text search.
Deliverables:

✔ Single (zipped) file with sourcecode, README, docs in PDF

✔ Documentation:

✔ A title page, Table of contents...
✔ A one page introduction to the application
✔ A one page technical section
✔ A one to three page section describing the interface
Assessment:

The assessment is as follows:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Documentation</td>
<td>15%</td>
</tr>
<tr>
<td>Code style/quality</td>
<td>35%</td>
</tr>
<tr>
<td>Operation of the interface</td>
<td>50%</td>
</tr>
</tbody>
</table>
Assignment 3 - code quality?
Debugging Java

✔ Netbeans debugger

✔ The java debugger `jdb`
Debugging Java

On suns...difficulty with versions of java and jdb and ddd

PATH=/usr/local/java/j2sdk1_3_1_02/bin:$PATH;export PATH
MFC

✔ Microsoft Foundation Classes - classes needed to produce GUI Windows programs.

✔ Development cycle - RAD, then editing.
MFC menus

A resource file for a simple File/Quit menu:

```c
#define MYAPP_EXIT 3210
MyApp MENU
    POPUP "File"
    {
        ITEM "Exit",MYAPP_EXIT
    }
}
In the `Create` call, you can do something like this:

```c
Create( NULL, "Example", ..., CRect(...), NULL, "MyApp" );
```

The `MYAPP_EXIT` message may be bound using the `DECLARE_MESSAGE_MAP()` macro, and with the following declaration:

```c
ON_COMMAND( MYAPP_EXIT, OnExit )
```
Message handler

```c++
afx_msg void CMenusWin::OnExit()
{
    SendMessage( WM_CLOSE );
}
```
#include <afxwin.h>

class CFirstWindow : public CFrameWnd {
public:
   CFirstWindow();
   ~CFirstWindow();
private:
   CStatic *m_pGreeting;
};

CFirstWindow::CFirstWindow()
{
   Create( NULL,
            "First Application",
            WS_OVERLAPPEDWINDOW,
            CRect( 100, 100, 400, 220 ) );
   m_pGreeting = new CStatic;
   m_pGreeting->Create( "Hello World!", // text
                        WS_CHILD | WS_VISIBLE | WS_BORDER |
                        SS_CENTER,
                        CRect( 80, 30, 200, 50 ),
                        this );
}

CFirstWindow::~CFirstWindow()
{
   delete m_pGreeting;
}

class CFirstApp : public CWinApp {
public:
   BOOL InitInstance()
   {
      m_pMainWnd = new CFirstWindow();
      m_pMainWnd->ShowWindow( m_nCmdShow );
      m_pMainWnd->UpdateWindow();
      return TRUE;
   }
};

FirstApp;
# Hungarian notation

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>c</code></td>
<td>Class declaration</td>
</tr>
<tr>
<td><code>m_</code></td>
<td>Class member variable</td>
</tr>
<tr>
<td><code>p</code></td>
<td>Pointer</td>
</tr>
<tr>
<td><code>n</code> or <code>i</code></td>
<td>Integer</td>
</tr>
<tr>
<td><code>On</code></td>
<td>Event or message handler</td>
</tr>
</tbody>
</table>
MFC class hierarchy
Chapter 12

Blending...
Blending languages

✔ SWIG - Simplified Wrapper and Interface Generator

✔ S/W tool that connects C and C++ with scripting languages

✔ Generates wrapper code that scripting languages use to access C/C++.

✔ Perl, Python, Tcl/Tk, Ruby, Java, OCAML, C# ...
Today though...

✓ Perl and Tk

✓ C and Tk

✓ VRML and Java

✓ Java and Tk
Perl and Tk

✔ Have to install the Tk module for perl

✔ Download Tk-804.026.tar.gz, and then

✔ perl Makefile.pl XFT=1
✔ make
✔ make install
GUI
Perl has a Tk module:

```perl
use Tk;
my $currentslice = 0;
my $currentpp    = 0;
my $disptype    = 2;
my $main = new MainWindow;

<<SetupMenu>>
<<SetupFileMenu>>
<<SetupEditMenu>>
<<SetupViewMenu>>

$main->configure(-menu =>$menubar);

<<SetupScrolledMainArea>>

MainLoop;

<<FileOpenDialogBox>>
```
Menu bar
CODE LISTING

SetupMenu.pl

$menubar = $main−>Menu;
$filemenu = $menubar−>cascade(−label=>"File" );
$editmenu = $menubar−>cascade(−label=>"Operate" );
$viewmenu = $menubar−>cascade(−label=>"View" );
$helpmenu = $menubar−>cascade(−label=>"Help");
$helpmenu−>command(−command => \&about_choice,
−label => "About TkMenu...",
−underline => 0);

CS3283 - Hugh Anderson’s notes.

Page number: 411


Menu items

```
$filename->command(-command => sub { fileDialog( $main, 'open' );
    printf "Opening $thisfile\n";
    readfile($thisfile);
    writeFile($thisfile . "ppx");},
    -label => "Open...",
    -underline => 0);
$filename->separator;
$filename->command(-label => "Exit",
    -command => \&exit_choice,
    -underline => 1);
```

CODE LISTING
```
SetUpFileMenu.pl
```

CS3283 - Hugh Anderson's notes.
Edit menu

CODE LISTING

SetUpEditMenu.pl

```perl
$editmenu->command(-command => sub {Tp($currentslice,1,1);},
                    -label => "Crank with widening...",
                    -underline => 0);
$editmenu->command(-command => sub {Tp($currentslice,1,10);},
                    -label => "Crank with widening (10X)...",
                    -underline => 0);
$editmenu->command(-command => sub {Tp($currentslice,0,1);},
                    -label => "Crank...",
                    -underline => 0);
$editmenu->command(-command => sub {Tp($currentslice,0,10);},
                    -label => "Crank (10X)...",
                    -underline => 0);
$editmenu->command(-command => sub {Cousot($currentslice);},
                    -label => "Cousot...",
                    -underline => 0);
$editmenu->separator;
$editmenu->command(-command => sub {widening($currentslice);},
                    -label => "Widen...",
                    -underline => 0);
```
View menu

<table>
<thead>
<tr>
<th>CODE LISTING</th>
<th>SetUpViewMenu.pl</th>
</tr>
</thead>
<tbody>
<tr>
<td>$viewmenu-&gt;command(-command =&gt; sub { $disptype=0; display($currentslice,0); },</td>
<td></td>
</tr>
<tr>
<td>-label =&gt; &quot;Show LRS...&quot;,</td>
<td></td>
</tr>
<tr>
<td>-underline =&gt; 0);</td>
<td></td>
</tr>
<tr>
<td>$viewmenu-&gt;command(-command =&gt; sub { $disptype=1; display($currentslice,1); },</td>
<td></td>
</tr>
<tr>
<td>-label =&gt; &quot;Show Actual Transform...&quot;,</td>
<td></td>
</tr>
<tr>
<td>-underline =&gt; 0);</td>
<td></td>
</tr>
<tr>
<td>$viewmenu-&gt;command(-command =&gt; sub { $disptype=2; display($currentslice,2); },</td>
<td></td>
</tr>
<tr>
<td>-label =&gt; &quot;Show Equations...&quot;,</td>
<td></td>
</tr>
<tr>
<td>-underline =&gt; 0);</td>
<td></td>
</tr>
<tr>
<td>$viewmenu-&gt;command(-command =&gt; sub { $disptype=3; display($currentslice,3); },</td>
<td></td>
</tr>
<tr>
<td>-label =&gt; &quot;Show Default Transform...&quot;,</td>
<td></td>
</tr>
<tr>
<td>-underline =&gt; 0);</td>
<td></td>
</tr>
<tr>
<td>$viewmenu-&gt;command(-command =&gt; sub { $disptype=4; display($currentslice,4); },</td>
<td></td>
</tr>
<tr>
<td>-label =&gt; &quot;Show Full formulae...&quot;,</td>
<td></td>
</tr>
<tr>
<td>-underline =&gt; 0);</td>
<td></td>
</tr>
<tr>
<td>$viewmenu-&gt;command(-command =&gt; sub { $disptype=5; display($currentslice,5); },</td>
<td></td>
</tr>
<tr>
<td>-label =&gt; &quot;Show only the program...&quot;,</td>
<td></td>
</tr>
<tr>
<td>-underline =&gt; 0);</td>
<td></td>
</tr>
<tr>
<td>$viewmenu-&gt;separator;</td>
<td></td>
</tr>
<tr>
<td>$viewmenu-&gt;command(-command =&gt; sub { if ($currentslice&gt;0) {</td>
<td></td>
</tr>
<tr>
<td>$currentslice=$currentslice-1;</td>
<td></td>
</tr>
<tr>
<td>if ($currentpp==0) {</td>
<td></td>
</tr>
<tr>
<td>$currentpp=$codesize;</td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
</tr>
<tr>
<td>$currentpp=$currentpp-1;</td>
<td></td>
</tr>
<tr>
<td>display($currentslice,$disptype);</td>
<td></td>
</tr>
<tr>
<td>}},</td>
<td></td>
</tr>
<tr>
<td>-label =&gt; &quot;Go back...&quot;,</td>
<td></td>
</tr>
<tr>
<td>-underline =&gt; 0);</td>
<td></td>
</tr>
<tr>
<td>$viewmenu-&gt;command(-command =&gt; sub { if ($currentslice+1&lt;$maxslice) {</td>
<td></td>
</tr>
<tr>
<td>$currentslice=$currentslice+1;</td>
<td></td>
</tr>
<tr>
<td>$currentpp=$currentpp+1;</td>
<td></td>
</tr>
<tr>
<td>if ($currentpp==$codesize) {</td>
<td></td>
</tr>
<tr>
<td>$currentpp=0;</td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
</tr>
<tr>
<td>display($currentslice,$disptype);</td>
<td></td>
</tr>
<tr>
<td>}},</td>
<td></td>
</tr>
<tr>
<td>-label =&gt; &quot;Go forward...&quot;,</td>
<td></td>
</tr>
<tr>
<td>-underline =&gt; 0);</td>
<td></td>
</tr>
</tbody>
</table>
sub exit_choice {
    exit;
}

sub fileDialog {
    my $w = shift;
    my $operation = shift;
    my $types; my $file;
    @types = ("Code files", '.pp'),
             ("Work files", '.ppx'),
             ("All files", '*');
    $file = $w−>getOpenFile(−filetypes => \@types);
    if (defined $file and $file ne '') {
        $thisfile = $file;
    }
}

CODE LISTING
FileOpenDialogBox.pl
http://www.comp.nus.edu.sg/~cs3283/ftp/original.pl

It may be run by typing “perl original.pl”.
C and Tk

✔ There are various ways...

✔ Call C within Tk
✔ Call Tk within C
✔ Communication
C and Tk

✔ This one is a C program

✔ It loads and runs a Tk program along with its interpreter.

✔ Tk in turn can call-back C procedures if needed
Viewer
#include <stdio.h>
#include <tcl.h>
#include <tk.h>

char  tclprog[] = "\n  proc fileDialog {w} {\n    set types {\n      { "Image files" {.gif} }\n      { "All files" {*} }\n    }\n    set file [tk_getOpenFile −filetypes $types −parent $w]\n    set glb_tx [image width picture]\n    set glb_ty [image height picture]\n    .c configure −width $glb_tx −height $glb_ty\n    .c create image 1 1 −anchor nw −image picture −tags "myimage"\n  }\n}
frame .mbar −relief raised −bd 2\nframe .dummy −width 10c −height 0\npack .mbar .dummy −side top −fill x\nmenubutton .mbar.file −text File −underline 0 −menu .mbar.file.menu\nmenu .mbar.file.menu −tearoff 1\n.mbar.file.menu add command −label "Open..." −command "fileDialog ."\n.mbar.file.menu add separator\n.mbar.file.menu add command −label "Quit" −command "destroy ."
pack .mbar.file −side left\ncanvas .c −bd 2 −relief raised\npack .c −side top −expand yes −fill x\nbind . <Control−c> {destroy .}\nbind . <Control−q> {destroy .}\nfocus .mbar;";

int
main (argc, argv)
{
  Tk_Window mainWindow;
  Tcl_Interp *tcl_interp;
  char **argv;
  Tk_Window mainWindow;
  Tcl_Interp *tcl_interp = Tk_CreateInterp ();
  tcl_interp = Tcl_CreateInterp ();
  if (Tcl_Init (tcl_interp) != TCL_OK || Tk_Init (tcl_interp) != TCL_OK) {
    if (Tcl_Init (tcl_interp) != TCL_OK) {
      if (tcl_interp->result) {
        printf (stderr, "%s", tcl_interp->result);
      } else {
        printf (stderr, "TCL_Init failed: %s", argv[0], tcl_interp->result);
      }
      exit (1);
    } else {
      printf (stderr, "Tk_Init failed: %s", argv[0], tcl_interp->result);
      exit (1);
    }
  }
  mainWindow = Tk_MainWindow (tcl_interp);
  if (mainWindow == NULL) {
    printf (stderr, "%s", tcl_interp->result);
    exit (1);
  }
  Tcl_Eval (tcl_interp, tclprog);
  Tk_MainLoop ();
  exit (1);
}
Compilation

On a Win32 system, we compile this as:

```
gcc -o myexe CplusTclTk.c -mwindows -ltcl80 -ltk80
```

On a UNIX system we use:

```
gcc -o myexe CplusTclTk.c -ltk -ltcl -lX11 -lm -ldl
```
VRML and Java

VRML is a scene description language

“The EAI allows you to control the contents of a VRML browser window embedded in a web page from a Java (tm) applet on the same page.”

http://www.frontiernet.net/~imaging/eaifaq.html

(People poser and Tiny3D are OK but there are problems...)

VRML and Java3DVNT
VRML software

<table>
<thead>
<tr>
<th>CODE LISTING</th>
<th>defaulthtml.txt</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;html&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;head&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;title&gt;</td>
<td>Sample 3DVNT Page &lt;/title&gt;</td>
</tr>
<tr>
<td>&lt;/head&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;center&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;H1&gt;</td>
<td>Sample 3DVNT Page &lt;/H1&gt;</td>
</tr>
<tr>
<td>&lt;/center&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;center&gt;</td>
<td>embed src=&quot;root.wrl&quot; height=&quot;600&quot; width=&quot;700&quot;&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;center&gt;</td>
<td>applet code=&quot;View1.class&quot; width=&quot;100&quot; height=&quot;10&quot; mayscript&gt;</td>
</tr>
<tr>
<td>&lt;PARAM name=&quot;segment&quot; value=&quot;MACS&quot;&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;PARAM name=&quot;port&quot; value=&quot;9876&quot;&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;PARAM name=&quot;host&quot; value=&quot;opo.usp.ac.fj&quot;&gt;</td>
<td></td>
</tr>
<tr>
<td>OK?</td>
<td></td>
</tr>
<tr>
<td>&lt;/center&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;/html&gt;</td>
<td></td>
</tr>
</tbody>
</table>
The *root.wrl* file

<table>
<thead>
<tr>
<th>CODE LISTING</th>
<th>root.wrl</th>
</tr>
</thead>
</table>
| PROTO CLUSTER [] { ... }  # Cluster definition  
| PROTO KEYBOARD [] { ... }  # Keyboard definition  
| PROTO SCREEN [] { ... }  # Screen definition  
| PROTO GLOBE [] { ... }  # Traffic sphere definition  
| # Some setting up declarations  
| Background { skyColor .4 .66 1 }  
| NavigationInfo { type [ "EXAMINE", "ANY" ] speed 400 }  
| Viewpoint { position 0 400 0 orientation 0 1 0 4 description "Camera 1" }  
| # Lines, floors and roofs  
| DEF LINES Transform { ... }  
| DEF FLOORS Transform { ... }  
| DEF ROOFS Transform { ... }  
| # and then the nodes  
| DEF node1 Transform { ... }  
| DEF node2 Transform { ... }  
| # ... and so on ... |
VRML nodes

```vrml
DEF node1 Transform {
   translation 4350 150 4365
   rotation 0 1 0 4.71238
   children [
      KEYBOARD {}
      SCREEN {}
      DEF node1box Transform {
         children [
            Shape {
               appearance Appearance {
                  material DEF node1boxcolor Material {
                     diffuseColor 0.8 0.8 0.8
                  }
               }
            }
         }  
         geometry Box { size 50 50 50 }  
      }  
      DEF node1sphere Transform {
         scale 1 1 1
         children [
            Shape {
               appearance GLOBE {}
               geometry Sphere { radius 1 }
            }
         ]  
      }  
   ]  
}
```
The java software maintains a link to a remote data collector, uses the EAI to modify the images in the VRML view.
// using the VRML External Interface.
import java.applet.*;
import java.awt.*;
import java.util.*;
import vrml.external.*;
import vrml.external.browser;
import java.io.*;

public class View1 extends Applet {
    // public static final int DEFAULT_PORT = 9877;
    Browser browser;
    Socket s = null;
    DataInputStream in = null;
    String line;

    public void init() {
        System.out.println("Test.init()...");
    }

    void SocketStart() throws java.io.IOException {
        String port = this.getParameter("port");
        int p = Integer.parseInt(port);
        try {
            String host = getCodeBase().getHost();
            System.out.println("Request came from: " + host);
            s = new Socket(host, p);
        }
        catch (UnknownHostException e) {
            System.out.println("No socket: " + e);
        }
        catch (java.io.IOException e) {
            System.out.println("No socket: " + e);
        } finally {
            System.out.println("Test.start()...");
            browser = (Browser) vrml.external.Browser.getBrowser(this);
            System.out.println("Got the browser: " + browser);
        }
    }

    public void start() {
        int count=0;
        Node node2sphere=null;
        Node appear=null;
        EventInSFVec3f[] scalein=new EventInSFVec3f[100] ;
        EventInSFColor[] appears=new EventInSFColor[100] ;
        float[] val = new float[3] ;
        int[] lastval = new int[100] ;
        int n;
        String id,vl;
        while (count != 100) {
            scalein[count] = null;
            appears[count] = null;
            count=count+1;
        }
        catch (java.io.IOException e) {
            System.out.println("No socket: " + e);
        }
    }
}

CS3283 - Hugh Anderson's notes.
Page number: 428
```java
while (true) {
    line = in.readLine();
    if (line == null) {
        System.out.println("Server closed connection.");
        break;
    }
    if (line.regionMatches(0, "n", 0, 1)) {
        n = line.indexOf(32, 2);
        // System.out.println(">>"+id+"<<");
        vl = line.substring(n+1);
        Integer a = Integer.valueOf(id);
        Integer b = Integer.valueOf(vl);
        if (scalein[a.intValue()] == null) {
            try {
                node2sphere = browser.getNode("node"+id+"sphere");
                System.out.println("Got the sphere node: " + node2sphere);
            } catch (InvalidNodeException e) {
                System.out.println("PROBLEMS! node2sphere: " + e);
            }
            try {
                scalein[a.intValue()] = (EventInSFVec3f) node2sphere.getEventIn("scale");
                System.out.println("Got the sphere scale node: " + appears[a.intValue()]);
            } catch (InvalidNodeException e) {
                System.out.println("PROBLEMS! (scalein): " + e);
            }
            try {
                appear = browser.getNode("node"+id+"boxcolor");
                System.out.println("Got the Boxcolor node: " + appear);
            } catch (InvalidNodeException e) {
                System.out.println("PROBLEMS! appearance: " + e);
            }
            try {
                appears[a.intValue()] = (EventInSFColor) appear.getEventIn("set_diffuseColor");
                System.out.println("Got the Boxcolor color node: " + appears[a.intValue()]);
            } catch (InvalidNodeException e) {
                System.out.println("PROBLEMS! appearance color: " + e);
            }
        } else {
            val[0] = (float)(b.intValue()*20)+1;
            val[1] = (float)(b.intValue()*20)+1;
            val[2] = (float)(b.intValue()*20)+1;
            scalein[a.intValue()].setValue(val);
            if ((b.intValue()==0) != (lastval[a.intValue()]==0)) {
                val[0] = (float)0.8;
                val[1] = (float)0.8;
                val[2] = (float)0.8;
                appears[a.intValue()].setValue(val);
            }
        }
    } else if ((b.intValue()==-1)) {
        val[0] = (float)1.0;
        val[1] = (float)1.0;
        val[2] = (float)1.0;
    } else {
        val[0] = (float)0.0;
        val[1] = (float)0.5;
        val[2] = (float)0.5;
        scalein[a.intValue()].setValue(val);
    }
}
```
else {
    if (b.intValue()==-1) {
        val[0] = (float)0.1;
        val[1] = (float)0.1;
        val[2] = (float)0.1;
        appears[a.intValue()].setValue(val);
    } else {
        val[0] = (float)0.0;
        val[1] = (float)1.0;
        val[2] = (float)0.0;
        appears[a.intValue()].setValue(val);
    }
    lastval[ a.intValue()]=b.intValue();
    //
    System.out.println(line);
    }
} catch (IOException e) { System.out.println("Reader: " + e); }

public Browser getBrowser() {
    return browser;
}
Java/Tk code

✔ As for C.... we have
  ✔ Java calls to Tk
  ✔ Tk calls to Java

✔ Partial implementation of Tk inside Java/Swing
This window displays a canvas widget containing a simple 2-dimensional plot. You can doctor the data by dragging any of the points with mouse button 1.

A Simple Plot
Java/Tk code

![Diagram of a simple plot]

A Simple Plot
```
canvas $c −relief raised −width 450 −height 300
  set plotFont {Helvetica 18}
...
  $c create line 100 250 100 50 −width 2
  $c create text 225 20 −text "A Simple Plot" −font $plotFont −fill brown
...
  for {set i 0} {$i <= 10} {incr i} {
    set x [expr {100 + ($i*30)}]
    $c create line $x 250 $x 245 −width 2
    $c create text $x 254 −text [expr 10*$i] −anchor n −font $plotFont
  }
...
  $c bind point <Any−Enter> "$c itemconfig current −fill red"
...
  proc plotMove {w x y} {
    global plot
    $w move selected [expr $x−plot(lastX)] [expr $y−plot(lastY)]
    set plot(lastX) $x
    set plot(lastY) $y
  }
```
The JavaKit code

```
C = new canvas(jf, width, height, bgColor);
plotFont = "−font [fontName Helvetica fontSize 18 fontStyle plain]";
...
C.create("−item line −coords [100 250 100 50] −width 2.0");
C.create("−item text −coords [225 20] −text "A Simple Plot\" " + plotFont");
...
for(int i = 0; i <= 10; i++) {
  int x = 100 + (i*30);
  C.create("−item line −coords [" + x + " 250 " + x + " 245] −width 2.0");
  C.create("−item text −coords [" + (x+9) + " 264] −text " + (10*i) + " " + plotFont");
}
...
C.bind("point", new command("enter") {
  public void action(eventInfo ei) { C.itemConfig("current", "−fill red"); }});
...
public void plotMove(canvas w, double x, double y) {
  w.move("selected", x − lastX, y − lastY);
  lastX = x; lastY = y;
}
```
Summary of topics

✔ Tk blended with C
✔ Tk blended with perl
✔ VRML, Java and the EAI
✔ Tk blended with Java
Outline

1 Module 1 - GUI concepts
2 Module 2 - Design
3 Architecture
4 First steps
5 Tcl/Tk - 1
6  Tcl/Tk - 2

7  Tcl/Tk - 3, Java

8  Java continued

9  Module 7

10 Module 8

11 Code Samples

12 Blending...