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

**Design/analysis document**

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.

Architecture

✔ 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.
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.

1. The processing load on the web server may be reduced.
2. The Java applet can directly communicate with a database server.

Note that there are some security concerns here.
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.

CODE LISTING

GUICode.c

#include <any GUI header files needed>
int main () {
    RegisterAllCallbacks ();
    LoopForever ();
}

How not to ...

Don’t do it the hard way!

X API

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)
{  
    /* main */    
    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 ();    }
```

**Win32 API**

```c
#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 -o SimpleWin32 SimpleWin32.c -mwindows
```
```c
#include <windows.h>
#include <string.h>

int WINAPI 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, CW_USEDEFAULT,
        NULL, NULL, hInst, NULL);
    ShowWindow (hwndMain, nShow);
    UpdateWindow (hwndMain);

    while (GetMessage (&msg, NULL, 0, 0)) {
        TranslateMessage (&msg);
        DispatchMessage (&msg);
    }
    return msg.wParam;
}
```

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

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.

Java/Swing

- 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

- 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:

1. Producing a standalone application.
2. Producing an applet to run within a web browser.

One of the features of Swing is that it implements a plug-gable 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