Module 9

MFC

The Microsoft Foundation Classes provide all the classes needed to produce GUI (Microsoft) Windows programs. A typical development cycle with MFC involves using a rapid application development tool such as the wizard found in Visual C++, and then modifying the resultant source code. The RAD development cycle is relatively easy, but unfortunately, the second phase is not.

9.1 MFC menus

There are many ways to create menus in MFC, but it is common to use a special menu resource file. A resource file for a simple File/Quit menu might look like this:

```
#define MYAPP_EXIT 3210
MyApp MENU
    POPUP "File"
    {
       MENUITEM "Exit",MYAPP_EXIT
    }
}
```

In the **`Create`** call, you can do something like this:

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

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

```
ON_COMMAND( MYAPP_EXIT,OnExit )
```

Finally, we need a message handler:

```
afx_msg void CMenusWin::OnExit()
{
    SendMessage( WM_CLOSE );
}
```

9.2 MFC Programming

Here is a simple initial example of an MFC application built using Microsoft Visual C++, modified from the Deitel&Deitel MFC book:

The code to do this is here:

CODE LISTING **FirstApp.cpp**

```
#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;
```

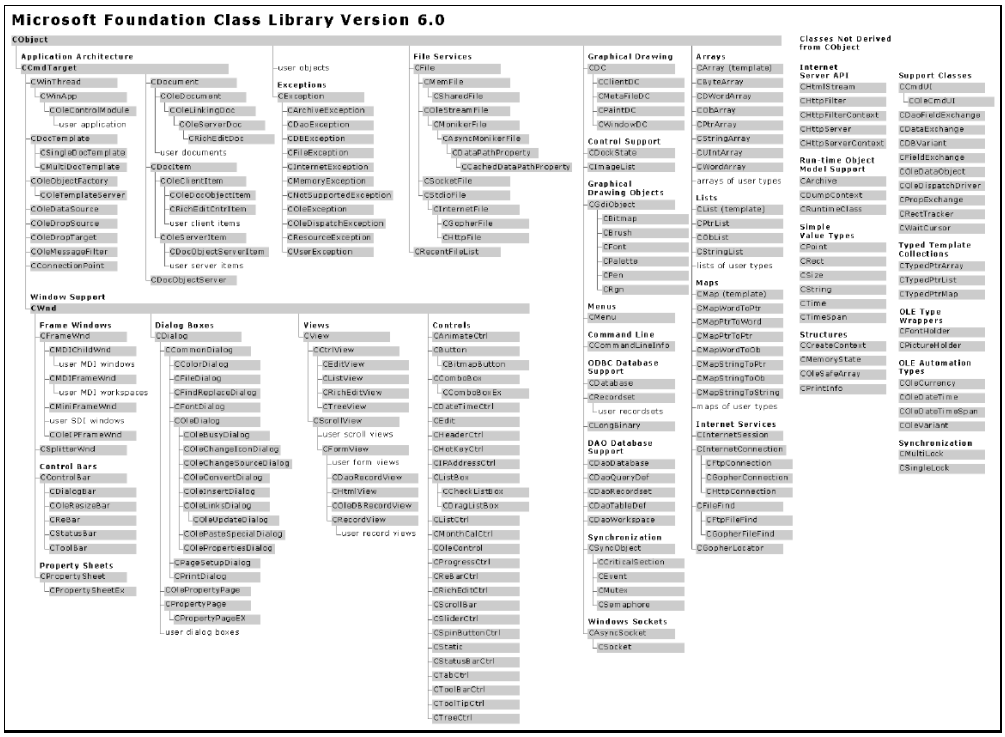
Note the use of Hungarian notation:

Prefix	Meaning
`C`	Class declaration
`m_`	Class member variable
`p`	Pointer
`n` or `i`	Integer
`On`	Event or message handler

This appears relatively easy, just instantiating a **`Cstatic`** object (a simple window). We can use much the same techniques to create dialogs (**`CDialog`**) or drawing windows (**`CFrameWnd`**)

9.3 MFC class hierarchy

The following heirarchy diagram shows the MFC components.



9.4 Summary of topics

In this module, we introduced the following topics:

- MFC
- MFC class heirarchy
- Simple programming

Tutorial 8 - questions for week 14 (April 10, 2002)

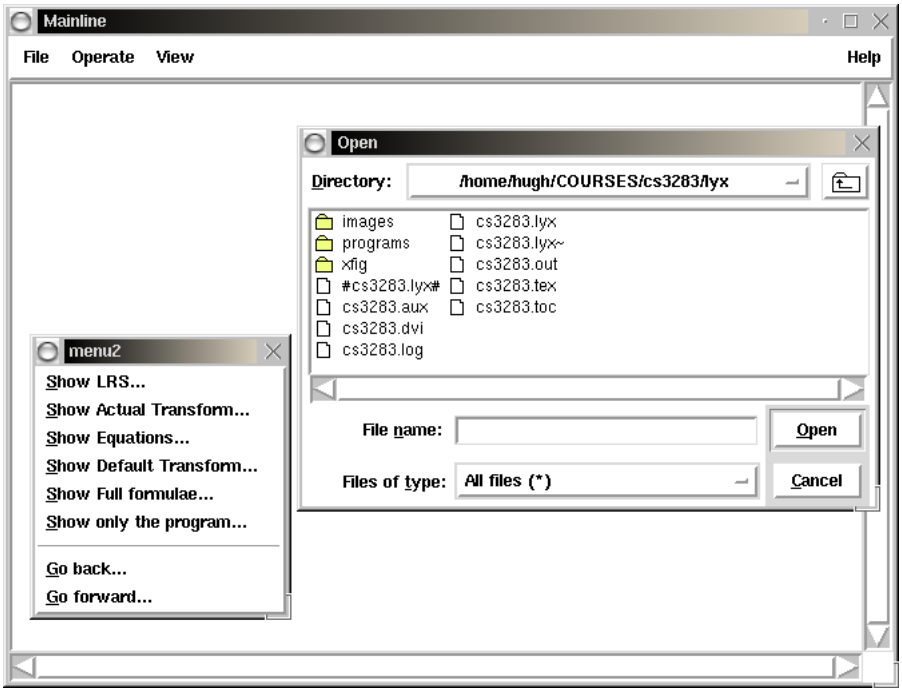
1. Give a minimal menu driven application for MFC.
2. Compare the MFC message model with the Java Event model.
3. Outline a strategy for porting an MFC program to UNIX.
4. Outline a strategy for porting a Tcl/Tk program to MFC.

Appendix B

Case study: GUI implementation

Here is an example of a *quick-n-dirty* GUI interface for a prototype application used for analysis of software. The prototype application uses libraries of C and fortran (!) routines to do numerical analysis, and a script written in perl to glue these all together, and transform complex data structures from binary to readable forms.

The GUI interface is easy to use and represents about 5% of the effort in developing the application.



B.1 Perl/Tk code

Perl has a Tk module - the perl interface to Tk - and it provides most of the user interface. The mainline is quite simple:

CODE LISTING **mainline.pl**

```
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>>
```

We then set up the menu bar. In this code we use the cascade model.

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);
```

And then we set up the menu items in each menu. First the file menu:

CODE LISTING **SetUpFileMenu.pl**

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

Then the edit menu:

CODE LISTING **SetUpEditMenu.pl**

```
$editmenu->command(-command => sub {Tp($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);
```

Finally the view menu:

CODE LISTING **SetUpViewMenu.pl**

```
$viewmenu->command(-command => sub {$disptype=0;display($currentslice,0);},
                   -label => "Show LRS...",
                   -underline => 0);
$viewmenu->command(-command => sub {$disptype=1;display($currentslice,1);},
                   -label => "Show Actual Transform...",
                   -underline => 0);
$viewmenu->command(-command => sub {$disptype=2;display($currentslice,2);},
                   -label => "Show Equations...",
                   -underline => 0);
$viewmenu->command(-command => sub {$disptype=3;display($currentslice,3);},
                   -label => "Show Default Transform...",
                   -underline => 0);
$viewmenu->command(-command => sub {$disptype=4;display($currentslice,4);},
                   -label => "Show Full formulae...",
                   -underline => 0);
$viewmenu->command(-command => sub {$disptype=5;display($currentslice,5);},
                   -label => "Show only the program...",
                   -underline => 0);
$viewmenu->separator;
$viewmenu->command(-command => sub { if ($currentslice>0) {
                                        $currentslice=$currentslice-1;
                                        if ($currentpp==0) {
                                           $currentpp=$codesize;
                                        }
                                        $currentpp=$currentpp-1;
                                        display($currentslice,$disptype);
                                     }},
                   -label => "Go back...",
                   -underline => 0);
$viewmenu->command(-command => sub { if ($currentslice+1<$maxslice){
                                        $currentslice=$currentslice+1;
                                        $currentpp=$currentpp+1;
                                        if ($currentpp==$codesize) {
                                           $currentpp=0;
                                        }
                                        display($currentslice,$disptype);
                                     }},
                   -label => "Go forward...",
                   -underline => 0);
```

Here is the code for an OpenFile dialog box:

CODE LISTING **FileOpenDialogBox.pl**

```
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;
   }
}
```

You can find a copy of this code at

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

It may be run by typing "**perl original.pl**".