

*M*icrosoft
*F*oundation
of
*C*lasses

周敬斐 老師
Curtis Chou

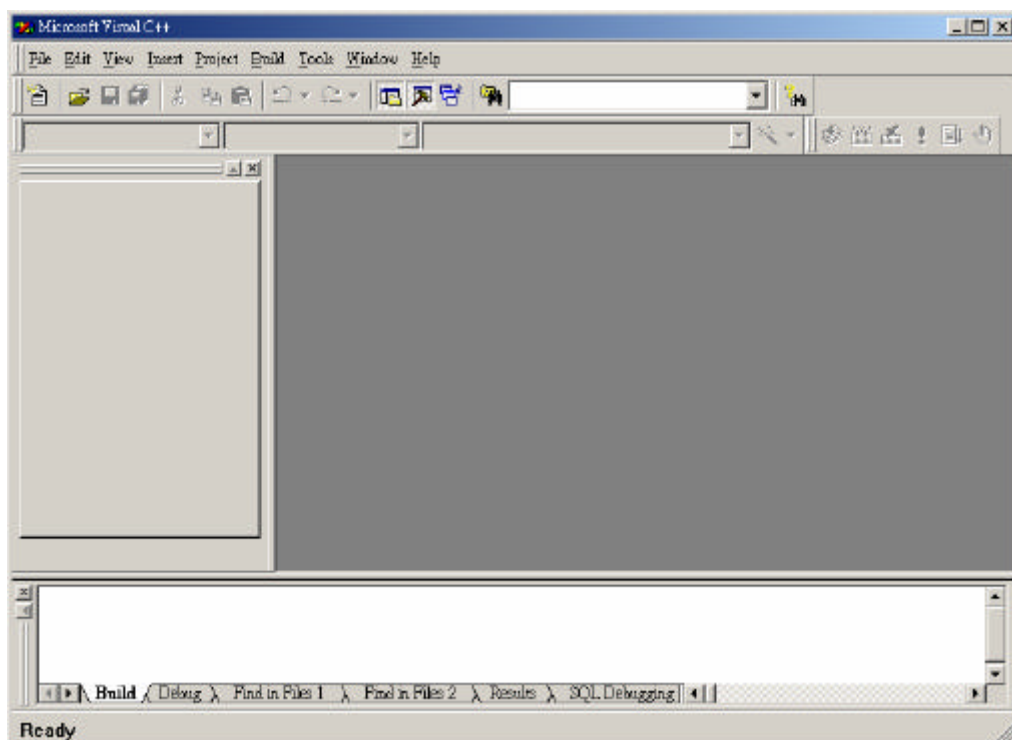
4.1 使用多行編輯方塊.....	2
4.2 使用核選方塊.....	2
4.3 使用圓形按鈕(RADIO BUTTONS)	2
4.4 使用列示方塊(LIST BOX).....	2

一、C++ 的開發環境

1.1 關於這門課程

這門課程主要探討視窗程式使用 Microsoft Visual C++ 6.0 搭配 MSDN。同學在家中必須要有 VC++ 與 MSDN，這樣子在學習的過程中會比較順利。

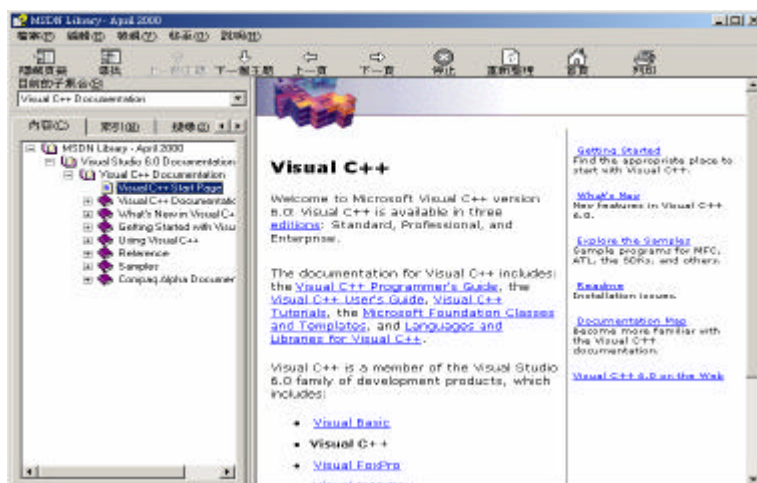
1.2 整合開發介面

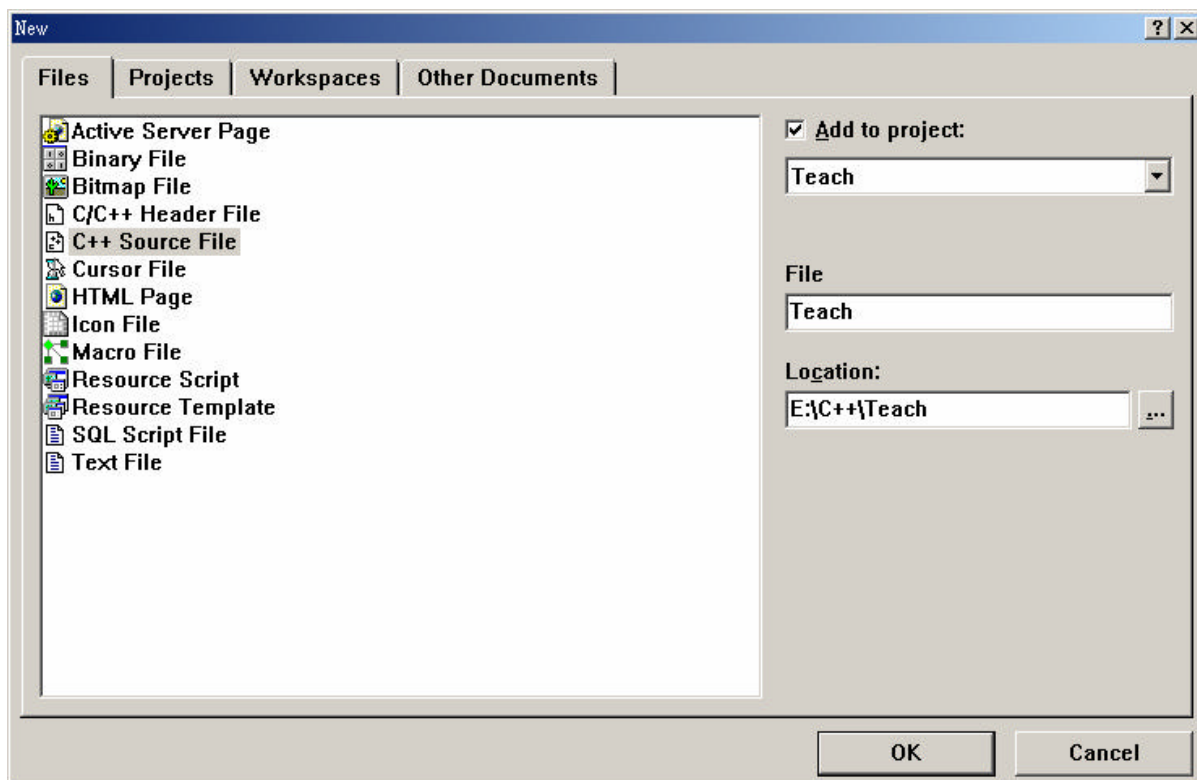


微軟提供了一組強大的開發軟體，就如講義上所看到的是微軟的 VC++ 開發環境的畫面，這種整合開發介面（IDE）最早是由 BORLAND 公司所提出的，後來幾乎所有的程式開發都走向 IDE 的介面。這樣子可以大幅提昇程式的寫作以及除錯並且執行程式。

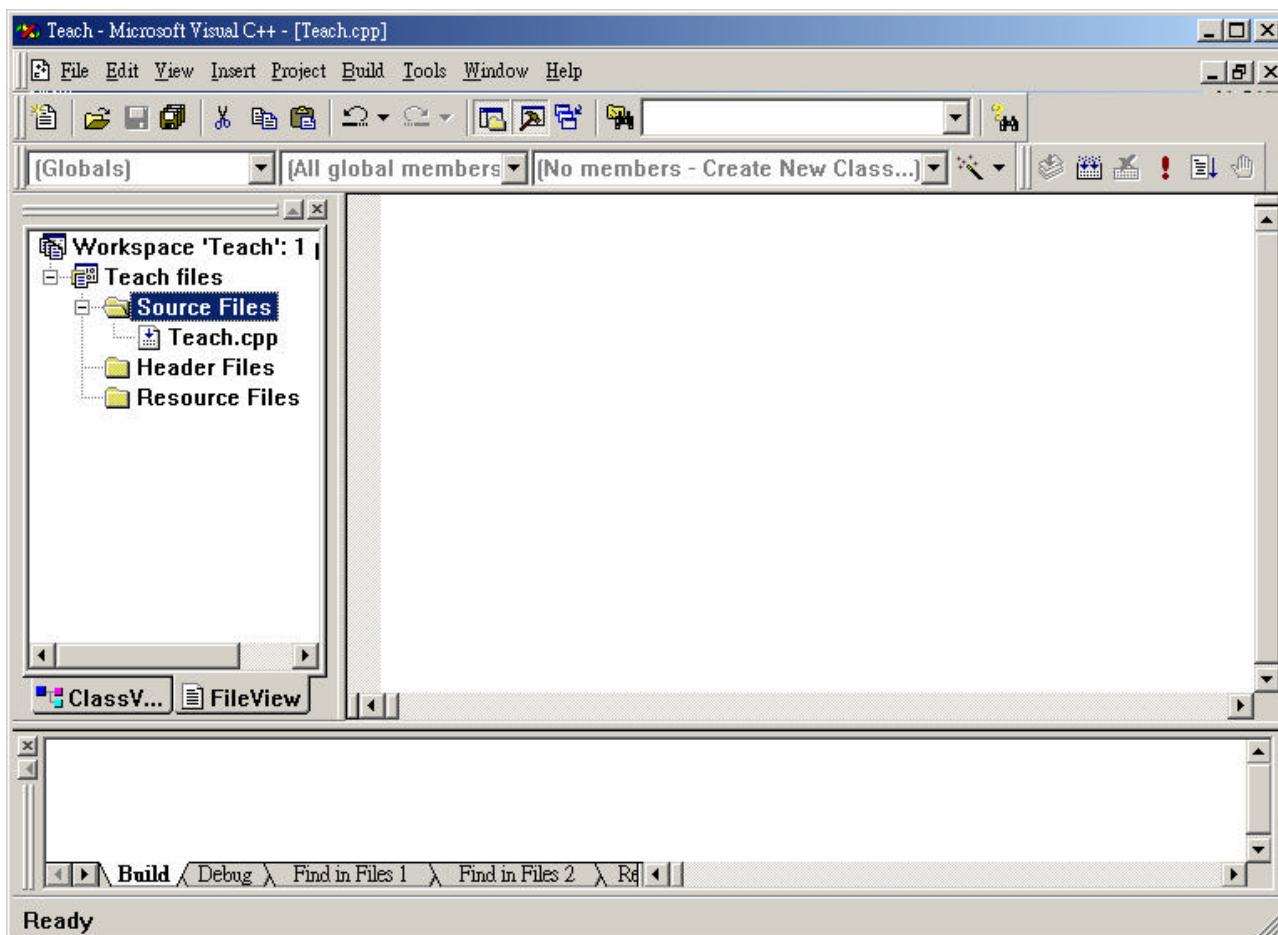
1.3 線上的 Visual C++ 說明文件

透過使用 MSDN 查詢相關函數的參數與使用範例，MSDN 多達 13GB 以上的說明文件可供參考。其中 C++ 多達五百多 MB。MSDN 將來絕對成為你不可或缺的好幫手，不信你等著吧。





完成之後應該有下面的視窗才對。



二、MFC 程式設計概念

2.1 關於 MFC 程式設計

2.1.1 早期的程式設計(DOS)

2.1.2 視窗的程式設計(GUI)

2.1.3 Microsoft Foundation of Classes(MFC)

2.2 MFC Classes 類別

CObject*	所有物件的基底類別
CGdiObject*	Graphic device interface class.
CPen	Class used for drawing pattern / color
CBrush	Class that represents fill patterns
CFont	Font class.
CBitmap	Bitmap class
CDC*	裝置容器基底類別
CClientDC	Client area device context class.
CPaintDC	Painting area device context class/
CMenu	Menu class
CCmdTarget*	事件訊息目標基底類別
CWnd*	視窗基底類別
CDialog	Dialog box window class
CStatic	Static control class.
CButton	Button control class.
CListBox	List box control class
CComboBox	Combo box control class
CEdit	Edit box control class
CFrameWnd	Frame Window class.
CWinThread*	工作排程基底類別
CWinApp	視窗應用類別

2.3 訊息

2.3.1 What is the Message.

當 GUI 事件發生時候，Windows OS 自動送出訊息給應用程式，程式也進行相關的回應動作，這種程式設計我們稱為事件導向的程式設計。

2.3.2 How the Message Operate?

1. Window 中每個訊息都被定義了一個獨一無二的符號，我稱稱為 [message identifier]。
2. 而訊息將由程式交由訊息處理器處理 [message handler]。
3. 這中間是透過一種叫做 [message map] 的構造完成的。

2.3.3 Declaring a Message Map

1. DECLARE_MESSAGE_MAP 巨集
2. 宣告方式
BEGIN_MESSAGE_MAP(owner-class-name, base -class-name)
..
END_MESSAGE_MAP()

2.3.4 Naming Convention

1. 我們經常在訊息前面加上 WM_
2. ex. WM_PAINT
3. WM is Window Message

2.3.5 Receive Window Message

1. In message map we will declare a handler .
2. ex. `ON_WM_PAINT()`
3. 處理訊息必須要在前面加上 ON。
4. `ON_WM_PAINT()` is MFC-defined marco .

2.3.6 Handling message by ON COMMAND

1. When you defined a identifier you have to suit it into right message handler.
2. To archive such task you should using
`ON COMMAND(identifier , function)`

2.4 匈牙利命名法

ar	Array	m_	class member variable
b	BOOL (int) or bool	n or i	Integer
c	char	p	Pointer
C	class	s	String
d	double	sz	String with NULL termination
l	long	s_	static class member variable
lp	long pointer		

2.5 建立一個簡單的 C + + 程式使用 MFC

```
//一個最簡單的 MFC 應用程式: welcome.h
//Author: Curtis Chou

class CWelcomeWnd : public CFrameWnd {
public:
    CWelcomeWnd();
    ~CWelcomeWnd();
private:
    CStatic *m_pHello;
};

//一個最簡單的 MFC 應用程式: welcome.cpp
//Author: Curtis Chou

#include <afxwin.h>
#include "welcome.h"

CWelcomeWnd::CWelcomeWnd(){
    Create( NULL,
        "歡迎",
        WS_OVERLAPPEDWINDOW,
        CRect( 200, 200, 460, 400) );
    m_pHello = new CStatic;
    m_pHello->Create(
        "歡迎來到 MS Visual C++的世界!",
        WS_CHILD | WS_VISIBLE | WS_BORDER | SS_CENTER,
        CRect( 40, 50, 200, 120),
        this );
}

CWelcomeWnd::~CWelcomeWnd(){
    delete m_pHello;
}

//宣告應用程式類別
class CWelcomeApp : public CWinApp {
public:
    BOOL InitInstance(){
        m_pMainWnd = new CWelcomeWnd ;
        m_pMainWnd -> ShowWindow( m_nCmdShow );
        m_pMainWnd -> UpdateWindow();
        return TRUE;
    }
};

CWelcomeApp hi;
```

2.5.1 MFC Frame Window Programming

1. MFC 應用程式架構 (主架構視窗)
3. Detail of class CFrameWnd , CWinApp

2.5.2 CFrameWnd::Create()

```

BOOL Create( LPCTSTR lpszClassName, LPCTSTR lpszWindowName,
            DWORD dwStyle = WS_OVERLAPPEDWINDOW, const RECT& rect =
            rectDefault, CWnd* pParentWnd = NULL, LPCTSTR lpszMenuName =
            NULL, DWORD dwExStyle = 0, CCreateContext* pContext = NULL );
Return Value

```

2.5.3 CStatic::Create()

```

BOOL Create( LPCTSTR lpszText, DWORD dwStyle, const RECT& rect, CWnd*
pParentWnd, UINT nID = 0xffff );
Return Value

```

Nonzero if successful; otherwise 0.

Parameters

IpszText Specifies the text to place in the control. If NULL, no text will be visible.

dwStyle

Specifies the static control 's window style. Apply any combination of [static control styles](#) to the control.

rect

Specifies the position and size of the static control. It can be either a RECT structure or a CRect object.

pParentWnd

Specifies the CStatic parent window, usually a CDialog object. It must not be NULL.

nID

Specifies the static control 's control ID.

2.5.4 Window Style

- `WS_BORDER` Creates a window that has a border.
- `WS_CAPTION` Creates a window that has a title bar (implies the `WS_BORDER` style). Cannot be used with the `WS_DLGFRAME` style.
- `WS_CHILD` Creates a child window. Cannot be used with the `WS_POPUP` style.
- `WS_CLIPCHILDREN` Excludes the area occupied by child windows when you draw within the parent window. Used when you create the parent window.
- `WS_CLIPSIBLINGS` Clips child windows relative to each other; that is, when a particular child window receives a paint message, the `WS_CLIPSIBLINGS` style clips all other overlapped child windows out of the region of the child window to be updated. (If `WS_CLIPSIBLINGS` is not given and child windows overlap, when you draw within the client area of a child window, it is possible to draw within the client area of a neighboring child window.) For use with the `WS_CHILD` style only.
- `WS_DISABLED` Creates a window that is initially disabled.

- **WS_DLGBORDER** Creates a window with a double border but no title.
- **WS_GROUP** Specifies the first control of a group of controls in which the user can move from one control to the next with the arrow keys. All controls defined with the **WS_GROUP** style **FALSE** after the first control belong to the same group. The next control with the **WS_GROUP** style starts the next group (that is, one group ends where the next begins).
- **WS_HSCROLL** Creates a window that has a horizontal scroll bar.
- **WS_MAXIMIZE** Creates a window of maximum size.
- **WS_MAXIMIZEBOX** Creates a window that has a Maximize button.
- **WS_MINIMIZE** Creates a window that is initially minimized. For use with the **WS_OVERLAPPED** style only.
- **WS_MINIMIZEBOX** Creates a window that has a Minimize button.
- **WS_OVERLAPPED** Creates an overlapped window. An overlapped window usually has a caption and a border.
- **WS_OVERLAPPEDWINDOW** Creates an overlapped window with the **WS_OVERLAPPED**, **WS_CAPTION**, **WS_SYSMENU**, **WS_THICKFRAME**, **WS_MINIMIZEBOX**, and **WS_MAXIMIZEBOX** styles.
- **WS_POPUP** Creates a pop-up window. Cannot be used with the **WS_CHILD** style.
- **WS_POPUPWINDOW** Creates a pop-up window with the **WS_BORDER**, **WS_POPUP**, and **WS_SYSMENU** styles. The **WS_CAPTION** style must be combined with the **WS_POPUPWINDOW** style to make the Control menu visible.
- **WS_SYSMENU** Creates a window that has a Control-menu box in its title bar. Used only for windows with title bars.
- **WS_TABSTOP** Specifies one of any number of controls through which the user can move by using the TAB key. The TAB key moves the user to the next control specified by the **WS_TABSTOP** style.
- **WS_THICKFRAME** Creates a window with a thick frame that can be used to size the window.
- **WS_VISIBLE** Creates a window that is initially visible.
- **WS_VSCROLL** Creates a window that has a vertical scroll bar.

2.5.5 Static Style

- `SS_BLACKFRAME` Specifies a box with a frame drawn with the same color as window frames. The default is black.
- `SS_BLACKRECT` Specifies a rectangle filled with the color used to draw window frames. The default is black.
- `SS_CENTER` Designates a simple rectangle and displays the given text centered

in the rectangle. The text is formatted before it is displayed. Words that would extend past the end of a line are automatically wrapped to the beginning of the next centered line.

- **SS_GRAYFRAME** Specifies a box with a frame drawn with the same color as the screen background (desktop). The default is gray.
- **SS_GRAYRECT** Specifies a rectangle filled with the color used to fill the screen background. The default is gray.
- **SS_ICON** Designates an icon displayed in the dialog box. The given text is the name of an icon (not a filename) defined elsewhere in the resource file. The `nWidth` and `nHeight` parameters are ignored; the icon automatically sizes itself.
- **SS_LEFT** Designates a simple rectangle and displays the given text flush-left in the rectangle. The text is formatted before it is displayed. Words that would extend past the end of a line are automatically wrapped to the beginning of the next flush-left line.
- **SS_LEFTNOWORDWRAP** Designates a simple rectangle and displays the given text flush-left in the rectangle. Tabs are expanded, but words are not wrapped. Text that extends past the end of a line is clipped.
- **SS_NOPREFIX** Unless this style is specified, Windows will interpret any ampersand (&) characters in the control's text to be accelerator prefix characters. In this case, the ampersand (&) is removed and the next character in the string is underlined. If a static control is to contain text where this feature is not wanted, **SS_NOPREFIX** may be added. This static-control style may be included with any of the defined static controls. You can combine **SS_NOPREFIX** with other styles by using the bitwise OR operator. This is most often used when filenames or other strings that may contain an ampersand (&) need to be displayed in a static control in a dialog box.
- **SS_RIGHT** Designates a simple rectangle and displays the given text flush-right in the rectangle. The text is formatted before it is displayed. Words that would extend past the end of a line are automatically wrapped to the beginning of the next flush-right line.
- **SS_SIMPLE** Designates a simple rectangle and displays a single line of text flush-left in the rectangle. The line of text cannot be shortened or altered in any way. (The control's parent window or dialog box must not process the `WM_CTLCOLOR` message.)
- **SS_USERITEM** Specifies a user-defined item.
- **SS_WHITEFRAME** Specifies a box with a frame drawn with the same color as the window background. The default is white.
- **SS_WHITERECT** Specifies a rectangle filled with the color used to fill the window background. The default is white.


```

// Fig. 2.11: menus.cpp
// create menus with MFC
#include <afxwin.h>           // MFC application framework
#include <strstream.h>        // string stream
#include <iomanip.h>          // I/O manipulators
#include "menus_ids.h"       // application message ID symbols
#include "CMenuWin.h"

CMenuWin::CMenuWin()          // construct window
    : m_str( m_szText, TEXT_SIZE ) // initialize ostrstream
{
    Create( NULL, "Menus Example", WS_OVERLAPPEDWINDOW,
        CRect( 0, 0, 200, 200 ), NULL, "Food" );

    m_nChicken = m_nFish = 0;
    m_nGingerale = m_nRootbeer = 0;
    m_dTotal = 0.0;
}

// count each type of item ordered, compute total bill
void CMenuWin::tally( int &nCount, double dAmount )
{
    nCount++;
    m_dTotal += dAmount;
}

// afx_msg precedes each message handler function
afx_msg void CMenuWin::OnExit()
{
    SendMessage( WM_CLOSE );
}

afx_msg void CMenuWin::OnDoFood(UINT nFood)
{
    switch (nFood)
    {
    case IDM_CHICKEN:
        tally( m_nChicken, 2.25 );
        break;
    case IDM_FISH:
        tally( m_nFish, 1.80 );
        break;
    case IDM_GINGERALE:
        tally( m_nGingerale, .80 );
        break;
    case IDM_ROOTBEER:
        tally( m_nRootbeer, .80 );
        break;
    }
}

afx_msg void CMenuWin::OnShowTotal()

```

功能表 (2 之 6)

```
{
    m_str.seekp( 0 );           // reset output string
    m_str << setprecision( 2 )
        << setiosflags( ios::fixed | ios::showpoint )
        << "          $" << m_dTotal << ends; // stopper

    // display new dialog box with output string
    MessageBox( m_szText, "Your total is:" );
    m_dTotal = 0.0;
}

afx_msg void CMenuWin::OnClearTotal()
{
    m_dTotal = 0.0;
    MessageBox( "          $0.00", "Cleared Order" );
}

BEGIN_MESSAGE_MAP( CMenuWin, CFrameWnd )

    ON_COMMAND( IDM_EXIT, OnExit )

    ON_COMMAND_RANGE( IDM_CHICKEN, IDM_ROOTBEER, OnDoFood )

    ON_COMMAND( IDM_SHOW_TOTAL, OnShowTotal )
    ON_COMMAND( IDM_CLEAR_TOTAL, OnClearTotal )

END_MESSAGE_MAP()

class CMenuApp : public CWinApp {
public:
    BOOL InitInstance()           // called by CWinApp::CWinApp
    {
        m_pMainWnd = new CMenuWin;           // create window
        m_pMainWnd->ShowWindow( m_nCmdShow ); // make it visible
        m_pMainWnd->UpdateWindow();           // force refresh
        return TRUE;                          // report success
    }
}

} menuApp;                               // calls CWinApp::CWinApp
```

功能表 (3 之 6)

```
// Fig. 2.11: menus_ids.h
// define messages used by menus.cpp and menus.rc
#define IDM_EXIT      2000

#define IDM_CHICKEN    2021
#define IDM_FISH       2022

#define IDM_GINGERALE  2041
#define IDM_ROOTBEER   2042

#define IDM_SHOW_TOTAL 2051
#define IDM_CLEAR_TOTAL 2052
```

功能表 (4 之 6)

```
// Fig. 2.11 menus.rc
// resource script for menus example
#include <afxres.h>
#include "menus_ids.h"

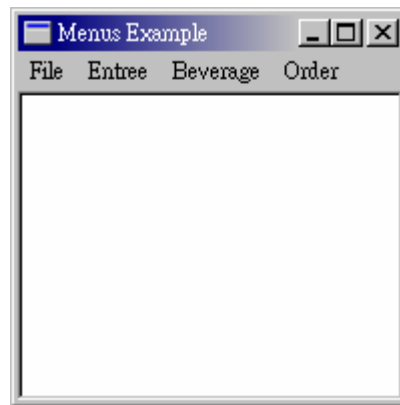
Food MENU
{
    POPUP "File"
    {
        MENUITEM "Exit", IDM_EXIT
    }

    POPUP "Entree"
    {
        MENUITEM "Chicken", IDM_CHICKEN
        MENUITEM "Fish", IDM_FISH
    }

    POPUP "Beverage"
    {
        MENUITEM "Ginger Ale", IDM_GINGERALE
        MENUITEM "Root Beer", IDM_ROOTBEER
    }

    POPUP "Order"
    {
        MENUITEM "Show Total", IDM_SHOW_TOTAL
        MENUITEM "Clear Total", IDM_CLEAR_TOTAL
    }
}
```

功能表 (5 之 6)



功能表(6之6) 執行結果

2.9 對話方塊 DIALOG

```
// Fig. 2.12: CAdditionDialog.h
// Addition program with MFC dialog box

class CAdditionDialog : public CDialog {
public:
    CAdditionDialog()
        : CDialog( "Addition" ), m_nTotal( 0 ) {}

    afx_msg void OnAdd();           // clicked the "Add" button
    afx_msg void OnClear();         // clicked the "Clear" button

private:
    int m_nTotal;                   // sum of numbers

    DECLARE_MESSAGE_MAP()
};
```

對話方塊 (1 of 6)

```
// Fig. 2.12: addition.cpp
// Addition program with MFC dialog box
#include <afxwin.h>
#include "addition_ids.h"
#include "CAdditionDialog.h"

// clicked the "Add" button
afx_msg void CAdditionDialog::OnAdd()
{
    const TEXT_SIZE = 16;
    char szText[ TEXT_SIZE + 1 ]; // buffer for conversions

    // get addresses of Edit Box Controls

    CEdit *pTotal = ( CEdit * ) ( GetDlgItem( IDC_TOTAL ) );
    CEdit *pNum    = ( CEdit * ) ( GetDlgItem( IDC_NUMBER ) );

    pNum->GetWindowText( szText, TEXT_SIZE ); // get Number
```

對話方塊 (2 of 6)

```

m_nTotal += atoi( szText );           // add binary value

itoa( m_nTotal, szText, 10 );         // convert total to text

pTotal->SetWindowText( szText ); // display total
pNum->SetWindowText( "" );           // clear input
pNum->SetFocus();                     // next input to Number
}

// clicked the "Clear" button
afx_msg void CAdditionDialog::OnClear()
{
    CEdit *pTotal = ( CEdit * ) ( GetDlgItem( IDC_TOTAL ) );
    CEdit *pNum    = ( CEdit * ) ( GetDlgItem( IDC_NUMBER ) );

    m_nTotal = 0;                      // clear the total
    pTotal->SetWindowText( "" ); // clear the edit box
    pNum->SetFocus();                // next input to Number
}

BEGIN_MESSAGE_MAP( CAdditionDialog, CDialog )
    ON_COMMAND( IDC_ADD, OnAdd )
    ON_COMMAND( IDC_CLEAR, OnClear )
END_MESSAGE_MAP()

// dialog-based application
class CAdditionApp : public CWinApp {
public:
    BOOL InitInstance()
    {
        CAdditionDialog additionDialog;
        additionDialog.DoModal(); // run dialog
        return FALSE;            // finished
    }
} addition;
```

對話方塊 (3 of 6)

```

// Fig 2.12: addition_ids.h
// Define Message Numbers

#define IDC_NUMBER 2000
#define IDC_ADD    2001
#define IDC_TOTAL  2002
#define IDC_CLEAR  2003

```

對話方塊 (4 of 6)

```
// Fig. 2.12: Addition.rc
// resource script for addition example
#include "afxres.h"
#include "addition_ids.h"

Addition DIALOG 50, 50, 130, 130
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU

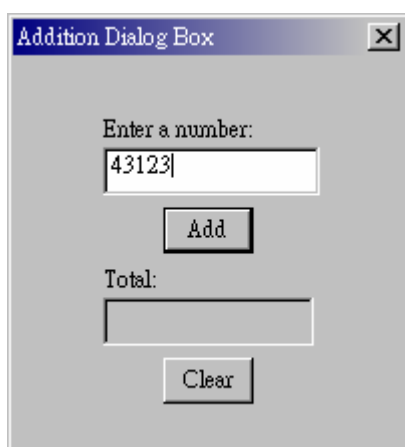
CAPTION "Addition Dialog Box"
{
    LTEXT        "Enter a number:", IDC_STATIC, 30, 20, 50, 8
    EDITTEXT     IDC_NUMBER, 30, 30, 72, 16, ES_NUMBER

    DEFPUSHBUTTON "Add", IDC_ADD, 50, 50, 30, 15

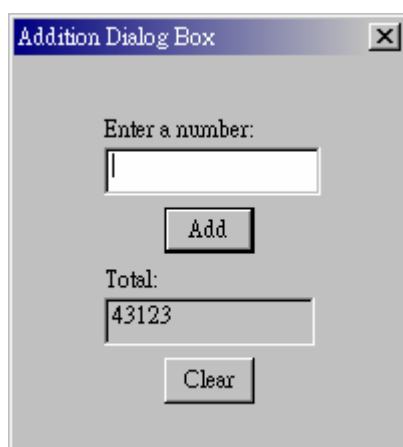
    LTEXT        "Total:", IDC_STATIC, 30, 70, 20, 8
    EDITTEXT     IDC_TOTAL, 30, 80, 70, 16,
                ES_READONLY | NOT WS_TABSTOP

    PUSHBUTTON   "Clear", IDC_CLEAR, 50, 100, 30, 15,
                NOT WS_TABSTOP
}
```

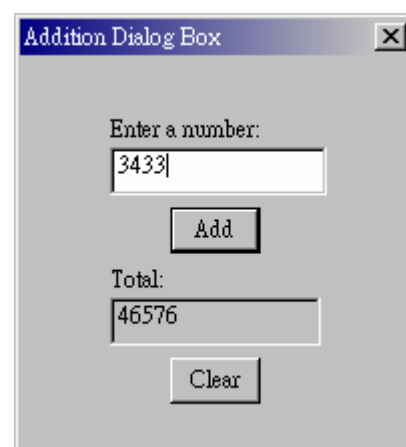
對話方塊 (5 of 6)



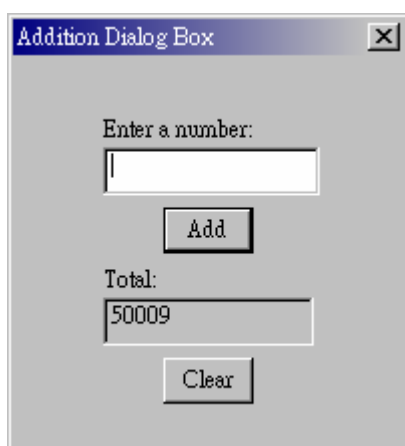
1.



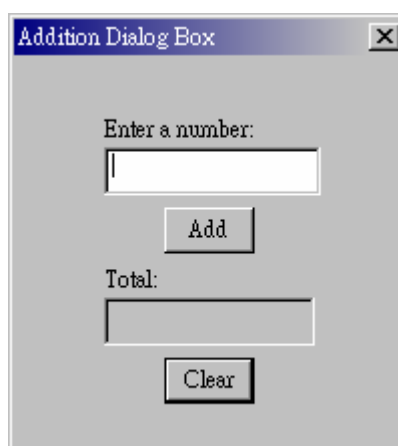
2.



3.



4.



5.


```
// Fig. 3.1: login.cpp
// login dialog
#include <afxwin.h>
#include "login_ids.h"
#include "CLoginDialog.h"

CLoginDialog::CLoginDialog( char *lpszName )
: CDialog( lpszName )
{
    m_szUserId[ 0 ] = m_szPassword[ 0 ] = '\0';
}

// clicked the "Log in" button
afx_msg void CLoginDialog::OnLogin()
{
    CEdit *pUserId;
    CEdit *pPassword;

    // get user ID value
    pUserId = ( CEdit * ) GetDlgItem( IDC_USERID );
    pUserId->GetWindowText( m_szUserId, 16 );

    // get password value
    pPassword = ( CEdit * ) GetDlgItem( IDC_PASSWORD );
    pPassword->GetWindowText( m_szPassword, 12 );

    // validate password
    int j = strcmp( m_szPassword, "PassWord" );

    if ( *m_szUserId != '\0' && j == 0 ) {
        MessageBox( m_szUserId, "Access Granted",
                    MB_ICONINFORMATION );
        EndDialog( IDOK );          // Report login success
    }
    else {
        MessageBox( "Invalid userid or password",
                    "Login error",
                    MB_ICONEXCLAMATION );

        pPassword->SetWindowText( "" ); // clear password
        pUserId->SetFocus();           // cursor in userid
    }
}

BEGIN_MESSAGE_MAP( CLoginDialog, CDialog )
    ON_COMMAND( IDC_LOGIN, OnLogin )    // Log in handler
END_MESSAGE_MAP()
```

對話方塊視窗 (2 之 5)

```
// after validating user, main application continues
CMainWin::CMainWin()
{
    Create( NULL, "Application Window",
            WS_OVERLAPPEDWINDOW,
            rectDefault );
}

// launch login dialog
void CMainWin::login()
{
    CLoginDialog loginDialog( "Login" );

    if ( loginDialog.DoModal() != IDOK )
        SendMessage( WM_CLOSE );    // login failed
}

// application creates main window, requests login dialog
class CLoginApp : public CWinApp {
public:
    BOOL InitInstance()
    {
        CMainWin *pMainWnd = new CMainWin;    // create window
        m_pMainWnd = pMainWnd;                // set main window
        pMainWnd->ShowWindow( m_nCmdShow );    // make visible
        pMainWnd->UpdateWindow();              // force refresh

        pMainWnd->login();                     // login dialog
        return TRUE;                          // report success
    }
};

loginApp;                                     // Fig03.01: login.rc
```

對話方塊視窗 (3 之 5)

```
// Fig 3.1: login_ids.h
// define message numbers

#define IDC_USERID    2000
#define IDC_PASSWORD  2001
#define IDC_LOGIN     2002
```

對話方塊視窗 (4 之 5)

```
// Fig03.01: login.rc
// login resources
#include <afxres.h>
#include "login_ids.h"

Login DIALOG 50, 50, 130, 130
CAPTION "User Authorization"
{
    LTEXT      "Enter userid:",
               IDC_STATIC, 30, 20, 50, 8
    EDITTEXT   IDC_USERID, 30, 30, 70, 16

    LTEXT      "Password:",
               IDC_STATIC, 30, 50, 40, 8
    EDITTEXT   IDC_PASSWORD, 30, 60, 70, 16,
               ES_PASSWORD

    DEFPUSHBUTTON "Log in", IDC_LOGIN, 50, 100, 30, 15
}
```

對話方塊視窗 (5 之 5)

3.3 滑鼠訊息處理

```
// Fig 3.3: CMouseWin.h
// Mouse coordinates display

class CMouseWin : public CFrameWnd {
public:
    CMouseWin();

    // display mouse coordinates and mouse button status
    void showPoint( UINT uFlags, CPoint point );

    // mouse button handlers
    afx_msg void OnLButtonDown( UINT uFlags, CPoint point );
    afx_msg void OnRButtonDown( UINT uFlags, CPoint point );
private:
    DECLARE_MESSAGE_MAP()
};
```

滑鼠訊息處理 (1 之 3)

```
// Fig 3.3: mouse.cpp
// Mouse coordinates display
#include <afxwin.h>
#include <strstrea.h>
#include "CMouseWin.h"

CMouseWin::CMouseWin()
{
    Create( NULL, "Mouse Example", WS_OVERLAPPEDWINDOW );
}
```

滑鼠訊息處理 (2 之 3)

[illegible]

滑鼠訊息處理 (3 之 3)

3.4 鍵盤訊息處理

```
// Fig. 3.4: CKeyboardWin.h
// keyboard input example

const int LINES = 24;           // maximum number of lines
const int LINE_LENGTH = 64;     // maximum characters per line
const int LINE_HEIGHT = 16;     // pixels between lines

// application window
class CKeyboardWin : public CFrameWnd {
public:
    CKeyboardWin();

    // refresh window when requested to by the system
    afx_msg void OnPaint();

    // process each character typed on keyboard
    afx_msg void OnChar( UINT uChar, UINT uRepCnt, UINT uFlg );
private:
    char m_asText[ LINES ][ LINE_LENGTH ]; // text to paint
    int m_anLen[ LINES ];                  // keep track of line lengths
    int m_nLine;                           // line receiving keystrokes
    DECLARE_MESSAGE_MAP()
};
```

```
// Fig. 3.4: keyboard.cpp
// keyboard input example
#include <afxwin.h>
#include "CKeyboardWin.h"

// initialize main window
CKeyboardWin::CKeyboardWin()
{
    m_nLine = 0;                // empty text array
    m_anLen[ m_nLine ] = 0;     // empty line of text

    Create( NULL, "Keyboard Example", WS_OVERLAPPEDWINDOW,
           CRect( 0, 0, 200, 200 ) );
}

// refresh window when requested to by the system
afx_msg void CKeyboardWin::OnPaint()
{
    CPaintDC dc( this );      // get device context

    int nPosition = m_anLen[ m_nLine ]++;

    m_asText[ m_nLine ][ nPosition ] = '_'; // make a cursor
}
```

```

for ( int ln = 0; ln <= m_nLine; ln++ ) // paint lines
    dc.TextOut( 1, LINE_HEIGHT * ln,
        m_asText[ ln ], m_anLen[ ln ] );

m_anLen[ m_nLine ]--; // remove cursor
}

// process each character typed on keyboard
afx_msg void CKeyboardWin::OnChar( UINT uChar, UINT uRepCnt, UINT uFlg ){
    switch ( uChar ) {
        case '\r': // start new line
            m_nLine++;

            if ( m_nLine >= LINES )
                m_nLine = 0; // wrap around

            m_anLen[ m_nLine ] = 0;
            break;
        case '\b': // backspace erases previous char
            if ( m_anLen[ m_nLine ] > 0 )
                m_anLen[ m_nLine ]--;
            break;
        default:
            int nPosition = m_anLen[ m_nLine ]++;

            m_asText[ m_nLine ][ nPosition ] = uChar;

            if ( m_anLen[ m_nLine ] >= LINE_LENGTH ) {

                if ( ++m_nLine >= LINES )
                    m_nLine = 0; // wrap around

                m_anLen[ m_nLine ] = 0;
            }
    }
    InvalidateRect( NULL ); // send WM_PAINT message
}

BEGIN_MESSAGE_MAP( CKeyboardWin, CFrameWnd )
    ON_WM_CHAR() // listen for any key press message
    ON_WM_PAINT() // listen for paint message
END_MESSAGE_MAP()

// application class creates window
class CKeyboardApp : public CWinApp {
public:
    BOOL InitInstance(){
        m_pMainWnd = new CKeyboardWin; // create window
        m_pMainWnd->ShowWindow( m_nCmdShow ); // make it visible
        m_pMainWnd->UpdateWindow(); // force refresh
        return TRUE; // report success
    }
} keyboardApp;

```

3.5 字串輸出

```
// Fig. 3.5: CTextWin.h
// Text display in the client area of a window

class CTextWin : public CFrameWnd {
public:
    CTextWin();

    // Refresh window when requested to by the system
    afx_msg void OnPaint();
private:
    DECLARE_MESSAGE_MAP()
};
```

```
// Fig. 3.5: text.cpp
// Text display in the client area of a window
#include <afxwin.h>
#include "CTextWin.h"

char *aszText[] =
{
    "Welcome to C++ and MFC!",
    "This is text in the client area.",
    " ",
    "<---Neatly Centered--->",
    NULL
};

// Create window for displaying text
CTextWin::CTextWin()
{
    Create( NULL, "Text Example", WS_OVERLAPPEDWINDOW,
           CRect( 100, 100, 400, 300 ) );
}

// Refresh window when requested to by the system
afx_msg void CTextWin::OnPaint()
{
    CPaintDC dc( this );    // get paint display context

    CRect rect;
    GetClientRect( &rect ); // get size of client area

    int nX = rect.right / 2;    // centered horizontally
    int nY = rect.bottom / 4;   // 1/4 for top margin

    // display lines of text centered in region of screen
    for ( int nLine = 0; aszText[ nLine ] != NULL; nLine++ )
    {
        int nLength = strlen( aszText[ nLine ] );
```

```

CSize nCSizeText = dc.GetTextExtent( aszText[ nLine ],
                                       nLength );

dc.SetTextColor( RGB( 255, 0, 0 ) ); // red text
dc.TextOut( nX - nCSizeText.cx / 2, // center text
            nY,                      // vertical offset
            aszText[ nLine ],        // text to display
            nLength );               // char count

nY += nCSizeText.cy; // advance to next line
}
}

BEGIN_MESSAGE_MAP( CTextWin, CFrameWnd )
    ON_WM_PAINT() // OnPaint handles WM_PAINT message
END_MESSAGE_MAP()

// application creates main window
class CTextApp : public CWinApp {
public:
    BOOL InitInstance()
    {
        m_pMainWnd = new CTextWin; // create window
        m_pMainWnd->ShowWindow( m_nCmdShow ); // make it visible
        m_pMainWnd->UpdateWindow(); // force refresh
        return TRUE; // report success
    }
} textApp;

```

四、MFC 圖形化使用者介面 (GUI)

4.1 使用多行編輯方塊

```
// Fig. 4.1: CEditTextDialog.h
// multiline edit text example

const int MAX_TEXT = 128;

class CEditTextDialog : public CDialog {
public:
    CEditTextDialog( char *lpszName );
    afx_msg void OnCount();    // clicked the "Count" button

private:
    char m_szText[ MAX_TEXT + 1 ];

    DECLARE_MESSAGE_MAP()
};
```

```
// Fig. 4.1: edittext.cpp
// multiline edit text example
#include <afxwin.h>
#include <strstream.h>
#include "edittext_ids.h"
#include "CEditTextDialog.h"

// Dialog constructor
CEditTextDialog::CEditTextDialog( char *lpszName )
    : CDialog( lpszName )    // base class constructor
{
    m_szText[ 0 ] = '\0';
}

// count the characters in the edit text control
afx_msg void CEditTextDialog::OnCount()
{
    // get address of edit control
    CEdit *pText = ( CEdit * ) GetDlgItem( IDC_TEXT );
    pText->GetWindowText( m_szText, MAX_TEXT );

    // display length of text read from edit text control
    static char szBuf[ 20 ];
    static ostrstream str( szBuf, 20);

    str.seekp( 0 );
    str << "Text length = " << strlen( m_szText ) << ends;
    pText->SetWindowText( szBuf );
}

BEGIN_MESSAGE_MAP( CEditTextDialog, CDialog )
```

```

        ON_COMMAND( IDC_COUNT, OnCount )
    END_MESSAGE_MAP()

    // start dialog-based application
    class CEditApp : public CWinApp {
    public:
        BOOL InitInstance()
        {
            CEditTextDialog editTextDialog( "EditText" );
            editTextDialog.DoModal();        // run dialog
            return FALSE;                    // finished
        }
    } editApp;

```

```
// Fig 4.1: edittext_ids.h
// define edit text message identifiers

#define IDC_TEXT      2001
#define IDC_COUNT     2002
```

[illegible]

4.2 使用核選方塊

```
// Fig. 4.2: CCheckBoxDialog.h
// check box example

class CCheckBoxDialog : public CDialog {
public:
    CCheckBoxDialog( char *lpszName );

    afx_msg void OnOK();      // clicked the "OK" button

private:
    // helper function combines two MFC calls in one
    int GetButtonStatus( int nId );

    DECLARE_MESSAGE_MAP()
};
```

```
// Fig. 4.2: checkbox.cpp
// check box example
#include <afxwin.h>
#include <strstream.h>
#include "checkbox_ids.h"
#include "CCheckBoxDialog.h"

CCheckBoxDialog::CCheckBoxDialog( char *lpszName )
    : CDialog( lpszName ) {}

// helper function combines two MFC calls in one
int CCheckBoxDialog::GetButtonStatus( int nId )
{
    CButton *pCButton = ( CButton * ) GetDlgItem( nId );
    return pCButton->GetCheck();
}

// overriding OnOK function
afx_msg void CCheckBoxDialog::OnOK() // clicked "OK" button
{
    static char sMessage[ 64 ];
    static ostrstream str( sMessage, 64 );
    str.seekp( 0 );

    int nChocolate = GetButtonStatus( IDC_CHOCOLATE );
    int nVanilla   = GetButtonStatus( IDC_VANILLA );
    int nStrawberry = GetButtonStatus( IDC_STRAWBERRY );

    str << "Flavor(s) Selected:\n\n";

    if ( nChocolate )
```

```

        str << "    Chocolate\n";

    if ( nVanilla )
        str << "    Vanilla\n";

    if ( nStrawberry )
        str << "    Strawberry\n";

    if ( nChocolate + nVanilla + nStrawberry == 0 )
        str << "    None\n";

    str << ends;

    MessageBox( sMessage, "Ice Cream" );
}

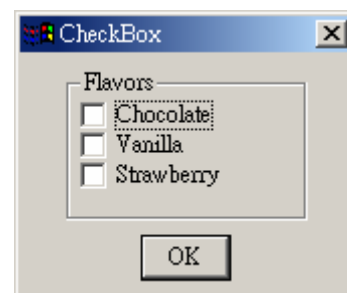
BEGIN_MESSAGE_MAP( CCheckBoxDialog, CDialog )
    ON_COMMAND( IDOK, OnOK )
END_MESSAGE_MAP()

// start application, create dialog window
class CCheckBoxApp : public CWinApp {
public:
    BOOL InitInstance()
    {
        CCheckBoxDialog checkBoxDialog( "CCheckBox" );
        checkBoxDialog.DoModal();           // run dialog window
        return FALSE;                       // exit
    }
}

} checkBoxApp;

```

// Fig. 4.2:



```
// Fig. 4.2: checkbox_ids.h
// check box message identifiers
```

```
// Fig. 4.2: checkbox.rc
// check box resource file
#include <afxres.h>
#include "checkboxbox_ids.h"
```

```
#define IDC_CHOCOLATE 2200
#define IDC_VANILLA 2201
#define IDC_STRAWBERRY 2202
```

```

CCheckBox DIALOG 50, 50, 110, 80
CAPTION "CheckBox"
{
    GROUPBOX "Flavors", IDC_STATIC, 15, 5, 80, 50

    AUTOCHECKBOX "Chocolate", IDC_CHOCOLATE, 20, 15, 60, 10
    AUTOCHECKBOX "Vanilla", IDC_VANILLA, 20, 25, 60, 10
    AUTOCHECKBOX "Strawberry", IDC_STRAWBERRY, 20, 35, 60, 10

    DEFPUSHBUTTON "OK", IDOK, 40, 60, 30, 15
}

```

4.3 使用圓形按鈕(Radio Buttons)

```
// Fig. 4.3: CRadioButtonDialog.h
// radio button example

class CRadioButtonDialog : public CDialog
{
public:
    CRadioButtonDialog( char *lpszName )
        : CDialog( lpszName ), m_lpszFlavor( "" ),
          m_lpszContainer( "" ) {}

    afx_msg void OnOK();      // clicked the "OK" button

private:
    char *m_lpszFlavor;
    char *m_lpszContainer;
    char m_szOrder[ 64 ];

    DECLARE_MESSAGE_MAP()
};
```

```
// Fig. 4.3: radiobutton.cpp
// radio button example
#include <afxwin.h>
#include "radiobutton_ids.h"
#include "CRadioButtonDialog.h"

// clicked the "OK" button
afx_msg void CRadioButtonDialog::OnOK()
{
    // get flavor ID of selected radiobutton from group
    int nFlavor = GetCheckedRadioButton( IDC_CHOCOLATE,
                                          IDC_STRAWBERRY );

    switch ( nFlavor ) {
        case IDC_CHOCOLATE:
            m_lpszFlavor = "Chocolate";
            break;
        case IDC_STRAWBERRY:
            m_lpszFlavor = "Strawberry";
            break;
        default:
            m_lpszFlavor = "Vanilla";
    }

    // get container ID of selected radiobutton from group
    int nContainer = GetCheckedRadioButton( IDC_CONE,
                                             IDC_CUP );

    switch ( nContainer ) {
        case IDC_CONE:
            m_lpszContainer = "Cone";
```

```

        break;
    default:
        m_lpszContainer = "Cup";
    }

    // concatenate the flavor and container in a string
    strcpy( m_szOrder, m_lpszFlavor );
    strcat( m_szOrder, " " );
    strcat( m_szOrder, m_lpszContainer );

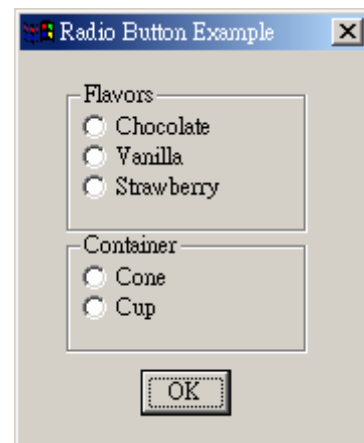
    MessageBox( m_szOrder, "Ice Cream Order:" );
}

BEGIN_MESSAGE_MAP( CRadioButtonDialog, CDialog )
    ON_COMMAND( IDOK, OnOK )
END_MESSAGE_MAP()

// start application, create dialog window
class CRadioButtonApp : public CWinApp
{
public:
    BOOL InitInstance()
    {
        CRadioButtonDialog radioButtonDialog( "CRadioButton" );
        radioButtonDialog.DoModal(); // run dialog window
        return FALSE;                // exit
    }
} radioButtonApp;

```

// Fig. 4.3: radioButtonApp



```
// Fig. 4.3: radiobutton.rc
// radio button resource file
#include <afxres.h>
#include "radiobutton_ids.h"
```

```
CRadioButton DIALOG 50, 50, 115, 130
CAPTION "Radio Button Example"
```

```
{
GROUPBOX          "Flavors",      IDC_STATIC,      15, 10, 80, 50
AUTORADIOBUTTON  "Chocolate",   IDC_CHOCOLATE,  20, 20, 60, 10,
WS_GROUP
AUTORADIOBUTTON  "Vanilla",     IDC_VANILLA,    20, 30, 60, 10
AUTORADIOBUTTON  "Strawberry",  IDC_STRAWBERRY,20, 40, 60, 10

GROUPBOX          "Container",   IDC_STATIC,      15, 60, 80, 40
AUTORADIOBUTTON  "Cone",         IDC_CONE,        20, 70, 60, 10,
WS_GROUP
AUTORADIOBUTTON  "Cup",          IDC_CUP,         20, 80, 60, 10

DEFPUSHBUTTON    "OK",           IDOK,            40,105, 30, 15
}
```

```
// Fig. 4.3: radiobutton_ids.h
// radio button message identifiers
#define IDC_CHOCOLATE 2200
#define IDC_VANILLA 2201
#define IDC_STRAWBERRY 2202

#define IDC_CONE 2210
#define IDC_CUP 2211
```

4.4 使用列示方塊(List Box)

```
// Fig. 4.4: CListBoxDialog.h
// list box control example

class CListBoxDialog : public CDialog {
public:
    CListBoxDialog( char *lpszName ) : CDialog( lpszName ) {}

    BOOL OnInitDialog();

    afx_msg void OnAdd();
    afx_msg void OnClear();

private:
    static char *s_alpszChoices[];

    DECLARE_MESSAGE_MAP()
};
```

```
// Fig. 4.4: listbox.cpp
// list box control example
#include <afxwin.h>
#include "listbox_ids.h"
#include "CListBoxDialog.h"

BOOL CListBoxDialog::OnInitDialog()
{
    // call base class initialization first
    CDialog::OnInitDialog();

    // Get address of List Box
    CListBox *pChoices;
    pChoices = ( CListBox * ) GetDlgItem( IDC_CHOICES );

    // add items to list box
    for ( int i = 0; s_alpszChoices[ i ] != NULL; i++ )
        pChoices->AddString( s_alpszChoices[ i ] );

    return TRUE;
}

// clicked the "Add" button
afx_msg void CListBoxDialog::OnAdd()
{
    CListBox *pChoices;
    pChoices = ( CListBox * ) GetDlgItem( IDC_CHOICES );

    int iCurSel = pChoices->GetCurSel();
```

```

if ( iCurSel == LB_ERR ) {
    MessageBox( "Select an item.", "Choices",
                MB_ICONWARNING );
    return;
}

CListBox *pSelected;
pSelected = ( CListBox * ) GetDlgItem( IDC_SELECTED );
char szText[ 32 ];
pChoices->GetText( iCurSel, szText );
pSelected->AddString( szText );
}

```

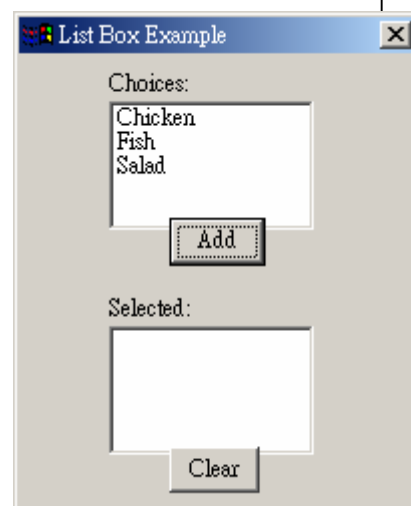
```
afx_msg void CListBoxDialog::OnClear()
```

```
{
    CListBox *pSelected;
    pSelected = ( CListBox * ) GetDlgItem( IDC_SELECTED );
    pSelected->ResetContent();    // clear list box
}
```

```
char *CListBoxDialog::s_alpszChoices[]
    = { "Chicken", "Fish", "Salad", NULL };
```

```
BEGIN_MESSAGE_MAP( CListBoxDialog, CDialog )
    ON_COMMAND( IDC_ADD,    OnAdd )
    ON_COMMAND( IDC_CLEAR, OnClear )
END_MESSAGE_MAP()
```

```
class CListBoxApp : public CWinApp {
public:
    BOOL InitInstance()
    {
        CListBoxDialog listBoxDialog(
            listBoxDialog.DoModal();
        return FALSE;
    }
} listBoxApp;
```



```
// Fig. 4.4: listbox_ids.h
// listbox message ID definitions
```

```
#define IDC_CHOICES 2101
#define IDC_SELECTED 2102

#define IDC_ADD 2103
#define IDC_CLEAR 2104
```

```
// Fig 4.4: listBox.rc
// ListBox resource file
```

```
#include "afxres.h"
#include "listbox_ids.h"
```

```

ListBoxDialog DIALOG    20, 20, 132, 150
CAPTION "List Box Example"
{
    LTEXT        "Choices:", IDC_STATIC,    30,    5, 50,    8
    LISTBOX      IDC_CHOICES,                30,    15 68,    48

    DEFPUSHBUTTON "Add", IDC_ADD,            50,    54 32,    16

    LTEXT        "Selected:", IDC_STATIC,    30,    80 50,    8
    LISTBOX      IDC_SELECTED,                30,    90 68,    48,
                WS_VSCROLL

    PUSHBUTTON   "Clear", IDC_CLEAR,         50,    130, 30,    15
}

```

4.5 使用 Combo Box

```
// Fig. 4.5: CComboBoxDialog.h
// combo box control example

class CComboBoxDialog : public CDialog {
public:
    CComboBoxDialog( char *lpszName ) : CDialog( lpszName ) {}

    BOOL OnInitDialog();
    afx_msg void OnAdd();      // clicked the "Add" button
    afx_msg void OnClear();    // clicked the "Clear" button

private:
    static char *s_alpszChoices[];

    DECLARE_MESSAGE_MAP()
};
```

```
// Fig. 4.5: combobox.cpp
// combo box control example
#include <afxwin.h>
#include "combobox_ids.h"
#include "CComboBoxDialog.h"

BOOL CComboBoxDialog::OnInitDialog()
{
    CDialog::OnInitDialog();
    CComboBox *pChoices;
    pChoices = ( CComboBox * ) GetDlgItem( IDC_CHOICES );

    // add strings to combo box
    for ( int i = 0; s_alpszChoices[ i ] != NULL; i++ )
        pChoices->AddString( s_alpszChoices[ i ] );

    return TRUE;
}

// clicked the "Add" button
afx_msg void CComboBoxDialog::OnAdd()
{
    CComboBox *pChoices;
    pChoices = ( CComboBox * ) GetDlgItem( IDC_CHOICES );

    int iCurSel = pChoices->GetCurSel();

    if ( iCurSel == CB_ERR ) {
        MessageBox( "Select an item.", "Choices",
                    MB_ICONWARNING );
    }
}
```

Combo Box Example

Choices:

Add

Selected:

Clear

```
// Fig. 4.5: combobox_ids.h
// combo box message identifiers
```

```
#define IDC_CHOICES 2101
#define IDC_ADD 2102
#define IDC_SELECTED 2103
#define IDC_CLEAR 2104
```

```
// Fig 4.5: combobox.rc
// combo box resource file
#include <afxres.h>
#include "combobox ids.h"
```

```

ComboBoxDialog DIALOG 20, 20, 132, 150
CAPTION "Combo Box Example"
{
    LTEXT          "Choices:", IDC_STATIC, 30, 5, 50, 8
    COMBOBOX       IDC_CHOICES, 30, 15, 68, 48,
                  CBS_SORT | CBS_DROPDOWNLIST

    DEFPUSHBUTTON  "Add", IDC_ADD, 50, 54, 30, 15

    LTEXT          "Selected:", IDC_STATIC, 30, 80, 50, 8
    LISTBOX        IDC_SELECTED, 30, 90, 68, 48,
                  WS_VSCROLL

    PUSHBUTTON     "Clear", IDC_CLEAR, 50, 130, 30, 15
}

```

五、MFC 圖形程式設計

5.1 繪製矩形及圓形

```
// Fig. 5.2: shapes.h
// Draw shapes

class CShapesWin : public CFrameWnd {
public:
    CShapesWin();

    // refresh window when requested to by the system
    afx_msg void OnPaint();

private:
    DECLARE_MESSAGE_MAP()
};
```

```
// Fig. 5.2: shapes.cpp
// Draw shapes and colors example
#include <afxwin.h>
#include "shapes.h"

CShapesWin::CShapesWin()
{
    Create( NULL, "Draw Shapes" );
} // end CShapesWin()

// refresh window when requested to by the system
afx_msg void CShapesWin::OnPaint()
{
    CPaintDC dc( this ); // get device context for paint

    // get current window size
    CRect rect;
    GetClientRect( &rect ); // get current client area size

    int x = rect.right / 4; // quarter x
    int y = rect.bottom / 4; // quarter y

    // draw a rectangle giving left, top, right, bottom edges
    dc.Rectangle( x, y, 2 * x, 2 * y );

    // create solid red brush and make it the current brush
    CBrush redBrush;
    redBrush.CreateSolidBrush( RGB( 255, 0, 0 ) );
    CBrush *pBrushSv = dc.SelectObject( &redBrush );

    // draw an ellipse, giving bounding rectangle
    dc.Ellipse( 2 * x, 2 * y, // left, top coord
```

```

        3 * x, 3 * y ); // right, bottom

    dc.SelectObject( pBrushSv ); // revert to original brush
} // end OnPaint()

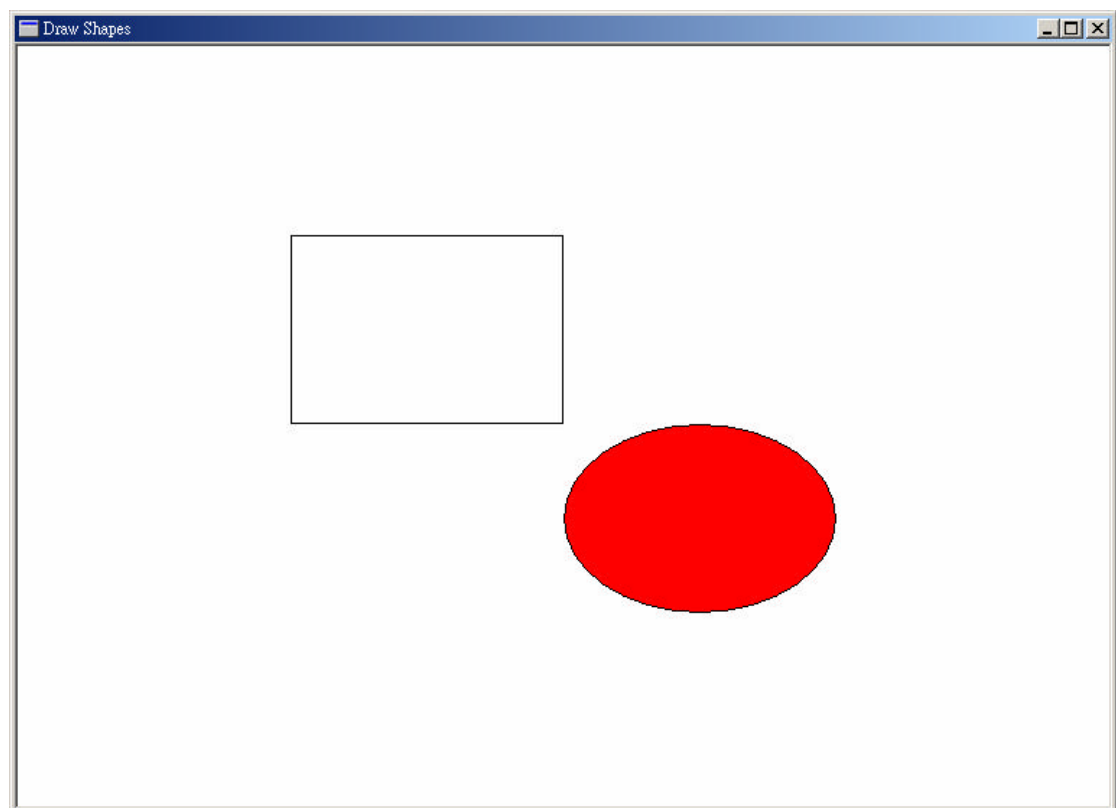
BEGIN_MESSAGE_MAP( CShapesWin, CFrameWnd )
    ON_WM_PAINT() // OnPaint() when screen changes
END_MESSAGE_MAP()

class CShapesApp : public CWinApp {
public:

    BOOL InitInstance()
    {
        m_pMainWnd = new CShapesWin; // create window
        m_pMainWnd->ShowWindow( m_nCmdShow ); // make visible
        m_pMainWnd->UpdateWindow(); // force refresh

        return TRUE; // run until user closes program
    }
} shapesApp;

```



5.2 畫線

```
// Fig. 5.3: lines.h
// Draw lines

class CLinesWin : public CFrameWnd {
public:
    CLinesWin();          // window constructor

    // refresh window when requested to by the system
    afx_msg void OnPaint();

private:
    CBrush m_greenBrush;    // green brush
    CPen m_redPen;          // red pen

    DECLARE_MESSAGE_MAP()
};
```

```
// Fig 5.3: lines.cpp
// line drawing example

#include <afxwin.h>        // MFC application framework
#include "lines.h"        // application class

CLinesWin::CLinesWin() // main window
{
    Create( NULL, "Draw Lines" ); // Frame window with title

    // create a pen for solid, 2 pixel wide, red lines
    m_redPen.CreatePen( PS_SOLID, 2, RGB( 255, 0, 0 ) ); // red
    m_greenBrush.CreateSolidBrush( RGB( 0, 255, 0 ) ); // green
}

// refresh window when requested to by the system
afx_msg void CLinesWin::OnPaint()
{
    CPaintDC dc( this ); // get device context for paint

    // get current window size
    CRect rect;
    GetClientRect( rect ); // get current client area size

    int x = rect.right / 4; // quarter x
    int y = rect.bottom / 4; // quarter y

    // draw rectangle, a line at a time
    dc.MoveTo( x, y ); // set starting point
    dc.LineTo( 3 * x, y ); // draw top line
    dc.LineTo( 3 * x, 2 * y ); // draw right line
    dc.LineTo( x, 2 * y ); // draw bottom line
    dc.LineTo( x, y ); // draw left line
```

```

x *= 2;           // point to the middle of the window
y *= 2;

// create a list of points for a shape

CPoint alpPoints[] = { // arrow shape
    CPoint( x + 00, y + 10 ),
    CPoint( x + 30, y + 50 ),
    CPoint( x + 10, y + 50 ),
    CPoint( x + 10, y + 70 ),
    CPoint( x - 10, y + 70 ),
    CPoint( x - 10, y + 50 ),
    CPoint( x - 30, y + 50 ),
    CPoint( x + 00, y + 10 )
};

const int POLY_POINTS           // array size
    = sizeof( alpPoints ) / sizeof( alpPoints[ 0 ] );

// select solid green brush, saving default brush
CBrush *pBrushSv = dc.SelectObject( &m_greenBrush );

// select red pen, saving default pen
CPen *pPenSv = dc.SelectObject( &m_redPen );

// display a polygon from a list of points
dc.Polygon( alpPoints, POLY_POINTS );

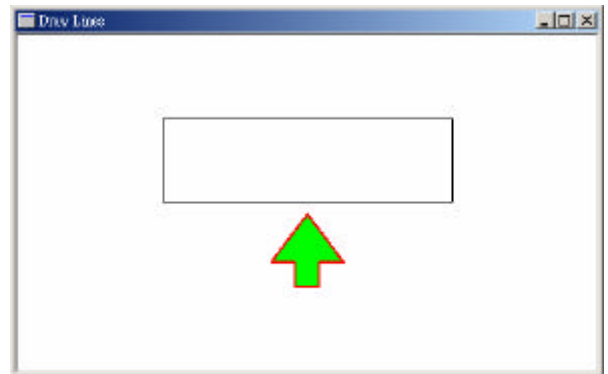
// restore default pen and brush
dc.SelectObject( pBrushSv );
dc.SelectObject( pPenSv );
}

BEGIN_MESSAGE_MAP( CLinesWin, CFrameWnd )
    ON_WM_PAINT()           // OnPaint() when screen changes
END_MESSAGE_MAP()

class CDrawApp : public CWinApp {
public:

    BOOL InitInstance()
    {
        m_pMainWnd = new CLinesWin;           // create
        m_pMainWnd->ShowWindow( m_nCmdShow ); // make vi
        m_pMainWnd->UpdateWindow();           // force r
        return TRUE;                          // run until user closes pro
    }
} linesApp;

```



5.3 計時器（做一個動態改變顏色的視窗）

```
// Fig. 5.4: colors.h
// Draw with many colors

class CColorsWin : public CFrameWnd {
public:
    CColorsWin();

    // refresh window when requested to by the system
    afx_msg void OnPaint();

    // cyclic timer tells when to update colors
    afx_msg void OnTimer();

    // release resources at end of program
    afx_msg void OnDestroy();

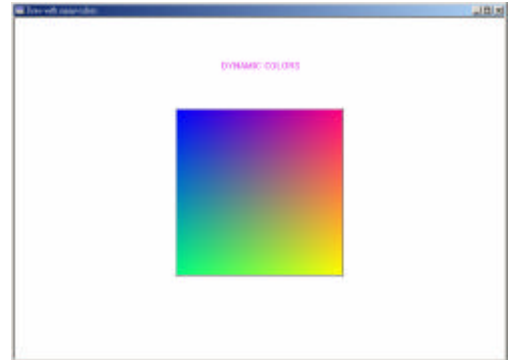
private:
    // draw a colorful square a pixel at a time
    void colorSquare();

    char *m_lpszText;           // text to draw
    int m_nLength;              // text length

    int m_cRed;                 // red component of a color
    int m_cGreen;               // green component of a color
    int m_cBlue;                // blue component of a color

    CDC m_memDC;                // memory device context
    CBitmap m_bitmap;           // memory bitmap

    DECLARE_MESSAGE_MAP()
};
```



```
// Fig 5.4: colors.cpp
// Draw with many colors example
#include <afxwin.h>
#include "colors.h"

CColorsWin::CColorsWin()
{
    Create( NULL, "Draw with many colors" );

    m_lpszText = "DYNAMIC COLORS";
    m_nLength = strlen( m_lpszText );
    m_cRed = m_cGreen = m_cBlue = 0;

    // get physical screen limits
    int xMax = GetSystemMetrics( SM_CXSCREEN );
    int yMax = GetSystemMetrics( SM_CYSCREEN );
```

```
CClientDC dc( this );    // get client area device context

// create a memory device context and bitmap to save
// the color square so it can be copied quickly
m_memDC.CreateCompatibleDC( &dc );
m_bitmap.CreateCompatibleBitmap( &dc, xMax, yMax );
m_memDC.SelectObject( &m_bitmap );
colorSquare();    // compute once rather than on each tick
} // end CColorsWin()

// draw a colorful square a pixel at a time
void CColorsWin::colorSquare()
{
    // step through red intensity values
    for ( int cRed = 0, x = 0;    // red increases with x
          cRed <= 255;
          cRed++, x++ )
        // step through green intensity values
        for ( int cGreen = 0, y = 0; // green increases with y
              cGreen <= 255;
              cGreen++, y++ )
        {
            // make blue balance the average of red and green
            int cBlue = 255 - ( cRed + cGreen ) / 2;

            // set pixel (x,y) to RGB value
            m_memDC.SetPixel( x, y, RGB( cRed, cGreen, cBlue ) );
        }
}

// refresh window when requested to by the system
afx_msg void CColorsWin::OnPaint()
{
    CPaintDC dc( this );    // get device context for paint

    // get current window size
    CRect rect;
    GetClientRect( rect ); // get current client area size

    int x = rect.right / 2; // middle x
    int y = rect.bottom / 4; // quarter y

    // display text with dynamically changing colors
    dc.SetTextColor( RGB( m_cRed, m_cGreen, m_cBlue ) );
    dc.SetBkColor( RGB( 255, 255, 255 ) );
    dc.TextOut( x - 60, y / 2, m_lpszText, m_nLength );

    x -= 128;    // center 256 pixel square in window
    y += 10;    // color square position
}
```

```
// copy color square from memory bitmap to screen bitmap
dc.BitBlt( x, y,                // destination
           256, 256,            // width, height
           &m_memDC, 0, 0,      // source, x, y
           SRCCOPY );           // straight copy

// draw a shaded rectangle around the color square
dc.Draw3dRect( x - 2, y - 2,    // left, top
               260, 260,        // width, height
               RGB( 150, 150, 150 ), // gray left, top
               RGB( 10, 10, 10 ) ); // dark gray shadow
} // end OnPaint()

// cyclic timer tells when to update colors
afx_msg void CColorsWin::OnTimer()
{
    // cycle through colors in prime number steps
    m_cRed = ( m_cRed + 3 ) % 256;
    m_cGreen = ( m_cGreen + 7 ) % 256;
    m_cBlue = ( m_cBlue + 11 ) % 256;

    InvalidateRect( NULL, FALSE ); // redraw all, no erase
}

// release resources at end of program
afx_msg void CColorsWin::OnDestroy()
{
    KillTimer( 1 ); // release timer 1
}

BEGIN_MESSAGE_MAP( CColorsWin, CFrameWnd )
    ON_WM_TIMER() // OnTimer() when timer ticks
    ON_WM_PAINT() // OnPaint() when screen changes
    ON_WM_DESTROY() // OnDestroy() when program ends
END_MESSAGE_MAP()

class CColorsApp : public CWinApp {
public:
    BOOL InitInstance()
    {
        m_pMainWnd = new CColorsWin; // create window
        m_pMainWnd->ShowWindow( m_nCmdShow ); // make visible
        m_pMainWnd->UpdateWindow(); // force refresh

        // start timer id 1, 30 milliseconds, use WM_TIMER
        if ( !m_pMainWnd->SetTimer( 1, 30, NULL ) )
            return FALSE; // quit on failure
        return TRUE; // run until user closes program
    }
} colorsApp;
```

5.4 圖片顯示

```
// Fig. 5.5: CImageWin.h
// display a bitmap image

class CImageWin : public CFrameWnd {
public:
    CImageWin();

    // refresh window when requested to by the system
    afx_msg void OnPaint();
private:
    CBitmap m_bmpCool; // bitmap stores image
    CDC m_memDC;       // memory device context

    DECLARE_MESSAGE_MAP()
};
```

```
// Fig. 5.5: image.cpp
// display a bitmap image
#include <afxwin.h>
#include "CImageWin.h"

CImageWin::CImageWin()
{
    // create frame window
    Create( NULL, "Display an Image", WS_OVERLAPPEDWINDOW );

    // load bitmap image from resources by name
    m_bmpCool.LoadBitmap( "COOL_BMP" );

    // device context of client area of window
    CClientDC dc( this );

    // create memory device context to hold image
    m_memDC.CreateCompatibleDC( &dc );
    m_memDC.SelectObject( &m_bmpCool );
}

// refresh window when requested to by the system
afx_msg void CImageWin::OnPaint()
{
    CPaintDC dc( this ); // get paint device context

    // bit block transfer image from memory DC to paint DC
    dc.BitBlt( 0, 0, 300, 300, &m_memDC, 0, 0, SRCCOPY );
}

BEGIN_MESSAGE_MAP( CImageWin, CFrameWnd )
    ON_WM_PAINT()
END_MESSAGE_MAP()
```

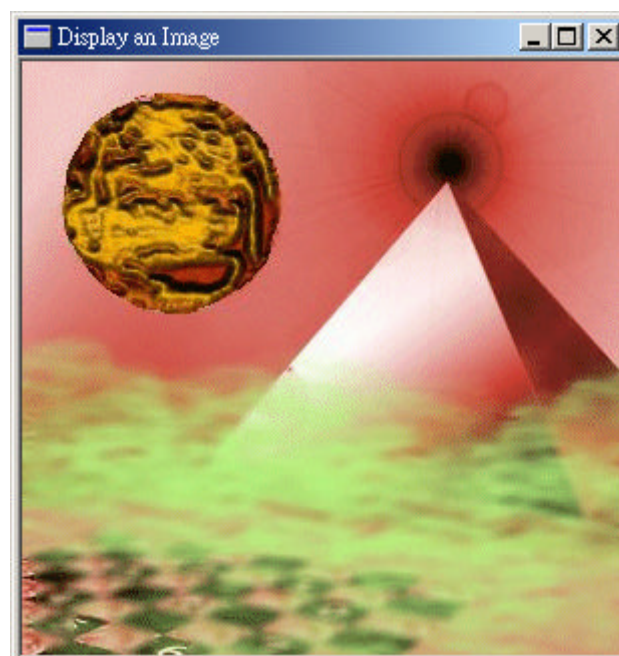
```
class CImageApp : public CWinApp {
public:

    BOOL InitInstance()
    {
        m_pMainWnd = new CImageWin;           // create window
        m_pMainWnd->ShowWindow( m_nCmdShow ); // make visible
        m_pMainWnd->UpdateWindow();            // force refresh

        return TRUE;
    }
} imageApp;
```

```
// Fig. 5.5: image.rc
// bitmap image resource file

// bitmap name BITMAP file name
COOL_BMP BITMAP cool.bmp
```



5.5 字型

```
// Fig. 5.6: CFontWin.h
// manipulate fonts
```

```
class CFontWin : public CFrameWnd {
public:
    CFontWin();

    // refresh window when requested to by the system
    afx_msg void OnPaint();

private:
    CFont m_sysFont;           // system font
    CFont m_italicFont;        // italic font

    DECLARE_MESSAGE_MAP()
};
```

```
// Fig. 5.6: fonts.cpp
// manipulate fonts
#include <afxwin.h>
#include "CFontWin.h"

CFontWin::CFontWin()
{
    // finish Windows frame in MFC CFrameWnd
    Create( NULL, "Font Example" );

    // create client area device context for this window
    CClientDC dc( this );

    // make default font
    m_sysFont.CreateStockObject( SYSTEM_FONT );

    // make 32 bit high bold italic font
    m_italicFont.CreateFont( 32, 0, 0, 0, FW_BOLD, 1, 0, 0, 0,
                           0, 0, 0, 0, "Times New Roman" );
}

// refresh window when requested to by the system
afx_msg void CFontWin::OnPaint()
{
    static char *aText[] = // sample text to display
    {
        "Welcome to C++ and MFC!",
        " ",
        "Height is 32 pixels.",
        "Font weight is Bold.",
        "Italic is set.",
        "Face name is Times New Roman.",
        NULL
    };
};
```



```

CPaintDC dc( this );    // get paint device context

// select italic font for text display
CFont *pFontSv = dc.SelectObject( &m_italicFont );

int x = 10;    // top margin
int y = 10;    // left margin

// display lines of text centered in region of screen
for ( int nLine = 0; aText[ nLine ] != 0; nLine++ )
{
    int nLength = strlen( aText[ nLine ] );

    // get size of text in current font
    CSize nCSizeText
        = dc.GetTextExtent( aText[ nLine ], nLength );

    // set text color, display text at computed position
    dc.SetTextColor( RGB( 255, 64, 64 ) ); // light red text
    dc.TextOut( x, y, aText[ nLine ], nLength );

    y += nCSizeText.cy; // advance to next line
}

char *lpszText = "This line uses SYSTEM_FONT.";

// select system font for text display
dc.SelectObject( &m_sysFont );
dc.SetTextColor( RGB( 0, 0, 0 ) ); // black text
y += 20;    // space down the screen
dc.TextOut( 10, y, lpszText, strlen( lpszText ) );
dc.SelectObject( pFontSv );    // original font
}

BEGIN_MESSAGE_MAP( CFontWin, CFrameWnd )
    ON_WM_PAINT()
END_MESSAGE_MAP()

class CFontApp : public CWinApp {
public:

    BOOL InitInstance()
    {
        m_pMainWnd = new CFontWin;    // create window
        m_pMainWnd->ShowWindow( m_nCmdShow ); // make visible
        m_pMainWnd->UpdateWindow();    // force refresh
        return TRUE;
    }
} fontApp;

```