# OpenGL with Glut 的設定與執行

- 系統需求
  1. Win98 或 Window NT 4.0 版以上
  2. Visual Studio 6.0 版或 Visual Studio .NET 以上
- 複製 glut32.lib 檔至以下位置(根據開發環境)
  1. (Visual Studio 6.0) Microsoft Visual Studio/VC98/Lib/
  2. (Visual Studio 2003) Microsoft Visual Studio .NET 2003/Vc7/PlatformSDK/Lib
  3. (Visual Studio 2005) Microsoft Visual Studio 8/VC/PlatformSDK/Lib/
  4. (Visual Studio 2008) Microsoft Visual Studio 9.0/VC/Lib/
- 確認系統有所需 DLL 檔：OPENGL32.DLL, GLU.DLL, GLU32.DLL, GLUT32.DLL
  1. (Win98) C:/WINDOWS/SYSTEM/
  2. (Windows NT) C:/WINNT/SYSTEM32/
  3. (Windows XP) C:/WINDOWS/system32/
- 確認開發環境中有所需標頭檔：GL.H, GLU.H, GLAUX.H, GLUT.H
  1. (Visual Studio 6.0) Microsoft Visual Studio/VC98/Include/GL/
  2. (Visual Studio 2003) Microsoft Visual Studio .NET 2003/Vc7/PlatformSDK/Include/GL/
  3. (Visual Studio 2005) Microsoft Visual Studio 8/VC/PlatformSDK/Include/GL/
  4. (Visual Studio 2008) Microsoft Visual Studio 9.0/VC/Include/GL/
- 為應用程式引入標頭檔

```
#include <windows.h>
#include <gl/gl.h>
#include <gl/glu.h>
#include <gl/glaux.h>
#include <gl/glut.h>
```

- 連結函式庫 LIB 檔：opengl32.lib, glaux.lib, glu32.lib, glut32.lib [1,2,3,4]
  1. (Visual Studio 6.0)

     見圖 1，進入 Menu 中的 Project -> Settings。

     見圖 2，選擇 Link 標籤，在 “Object/library modules:” 的最後面加上四個 lib 檔。

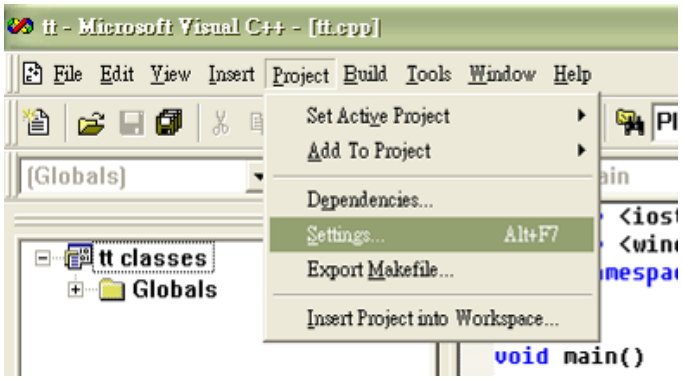


圖 1

[1] All functions that define OpenGL are defined. All functions begin with the prefix *gl.*

[2] AUX library. A toolkit that provides a platform-independent framework for calling OpenGL functions. All functions begin with the prefix *aux*.

[3] Utility library contains utility functions for making tasks easier, such as drawing spheres, disks, and cylinders. All functions begin with the prefix *glu.*

[4] GL Utility Toolkit (GLUT) that provides the minimum functionality that should be expected in any modern windowing system.

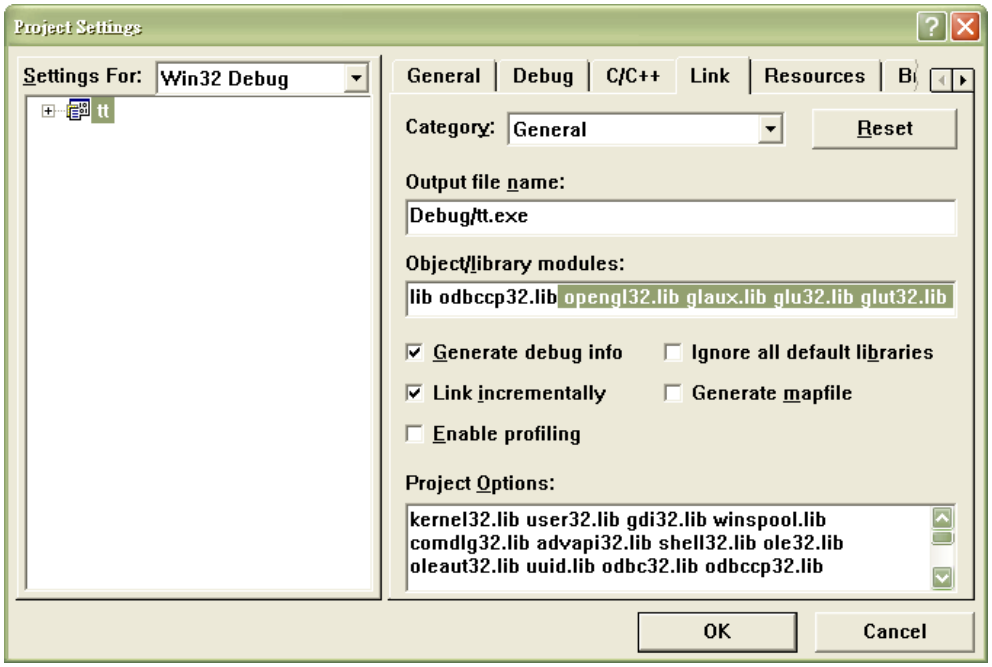


圖 2

2. (Visual Studio .net 2003, 2005, 2008)

在類別檢視(Class view)視窗中，在目前的專案(Project)的標籤上按滑鼠右鍵，在 Context menu 上選擇屬性(Property)。

見圖 3，在對話盒的左邊，選擇「連結器(Linker)」，再選擇子項目「輸入」，然後在「其他相依性」中輸入上列出的四個函式庫。(以空白間隔)

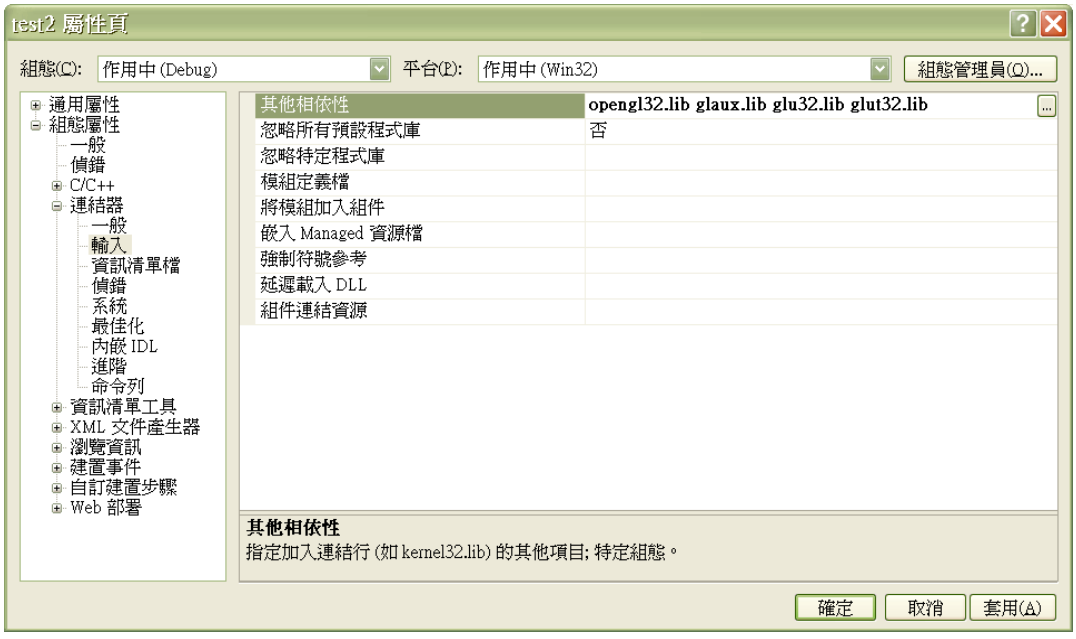


圖 3

3. (適用於任一版本)

在原始碼檔案的開始部分(在引入標頭檔的部分之後)，寫入以下巨集指令：

```
#pragma comment(lib, "opengl32.lib")
#pragma comment(lib, "glu32.lib")
#pragma comment(lib, "glaux.lib")
#pragma comment(lib, "glut32.lib")
```

# OpenGL SuperBible 4th Ed 範例程式之環境設定

至課程教學網站中【檔案下載】區下載【OpenGL SuperBible 內附程式庫】

- 複製 freeglut_static.lib 檔至以下位置(根據開發環境)
  1. (Visual Studio 2003)　Microsoft Visual Studio .NET 2003/Vc7/PlatformSDK/Lib/
  2. (Visual Studio 2005)　Microsoft Visual Studio 8/VC/PlatformSDK/Lib/
  3. (Visual Studio 2008)　Microsoft Visual Studio 9.0/VC/Lib/
- 複製所需標頭檔(.h)與 C++檔(.cpp)至以下位置(根據開發環境)
  1. (Visual Studio 2003)　Microsoft Visual Studio .NET 2003/Vc7/PlatformSDK/Include/GL/shared/
  2. (Visual Studio 2005)　Microsoft Visual Studio 8/VC/PlatformSDK/Include/GL/shared/
  3. (Visual Studio 2008)　Microsoft Visual Studio 9.0/VC/Include/GL/shared/

- 為應用程式引入所需之標頭檔

如：

```
#include <gl/shared/gltools.h>
#include <gl/shared/math3d.h>
#include <gl/shared/math3d.cpp>
...
```

## 再加入標頭檔後，還需要做三件事才能順利執行範例程式

## (若直接開啟專案檔則不需做以下動作)

在類別檢視(Class view)視窗中，在目前的專案(Project)的標籤上按滑鼠右鍵，在Context menu上選擇屬性(Property)，會產生屬性頁之對話盒，如圖4所示。而將左上角【組態】的部分會依其模式調整成Debug或Release，要記得兩個模式都有要加入的部分。

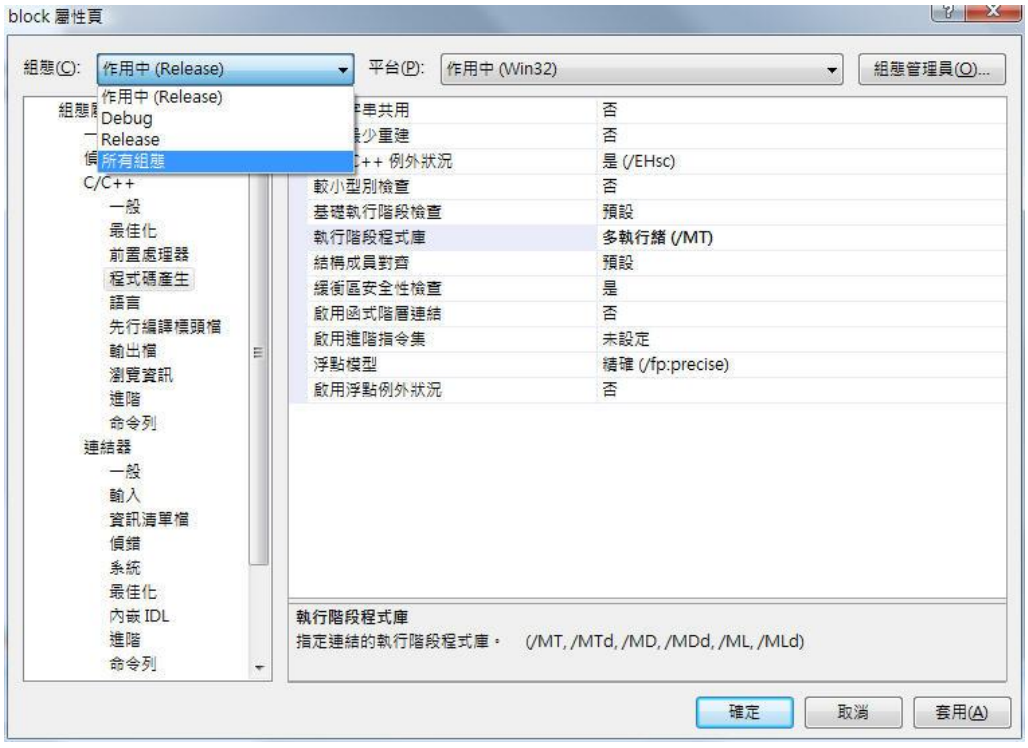


圖4

1. 見圖5，在【組態屬性】→【C/C++】→【前置處理器】內的【前置處理器定義】中加入屬性值
   【Debug】模式下：WIN32;_DEBUG;_CONSOLE
   【Release】模式下：WIN32;NDEBUG;_CONSOLE

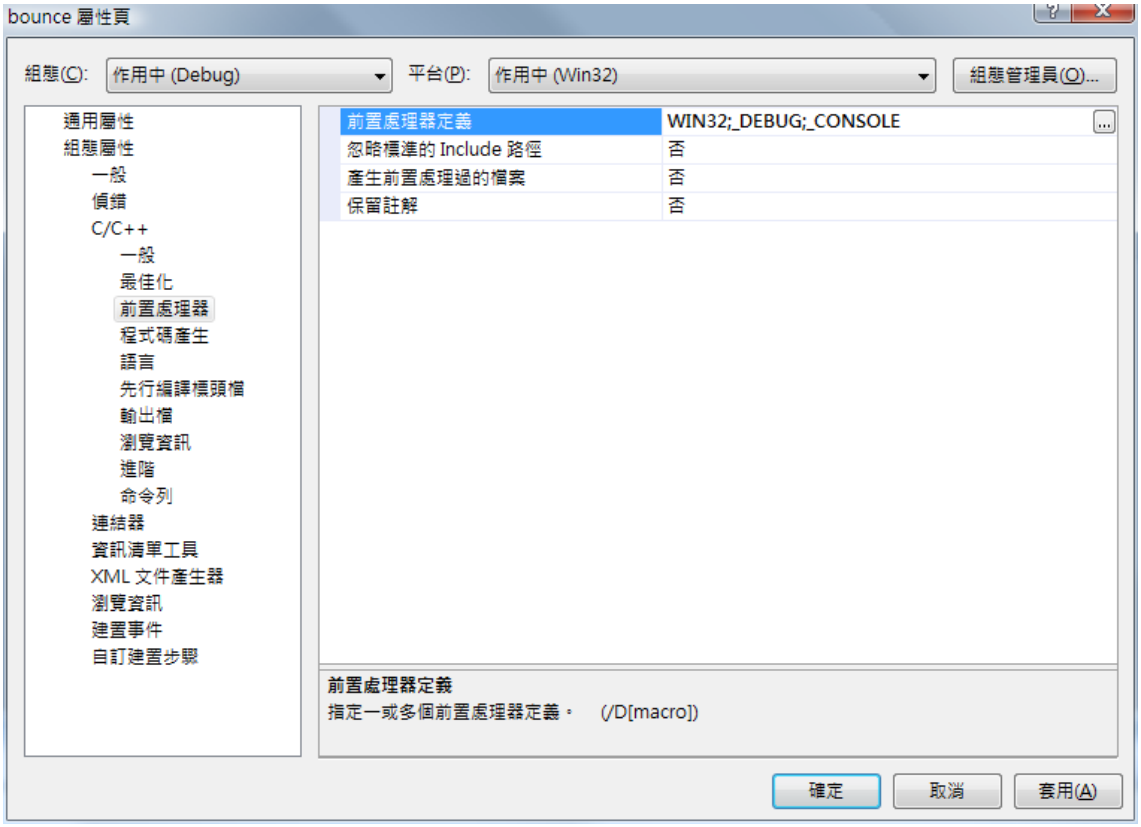

圖5

2. 見圖6，在【組態屬性】→【C/C++】→【程式碼產生】內的【執行階段程式庫】中的屬性改成
   【Debug】模式下：【多執行緒偵錯 (/MTd)】
   【Release】模式下：【多執行緒 (/MT)】

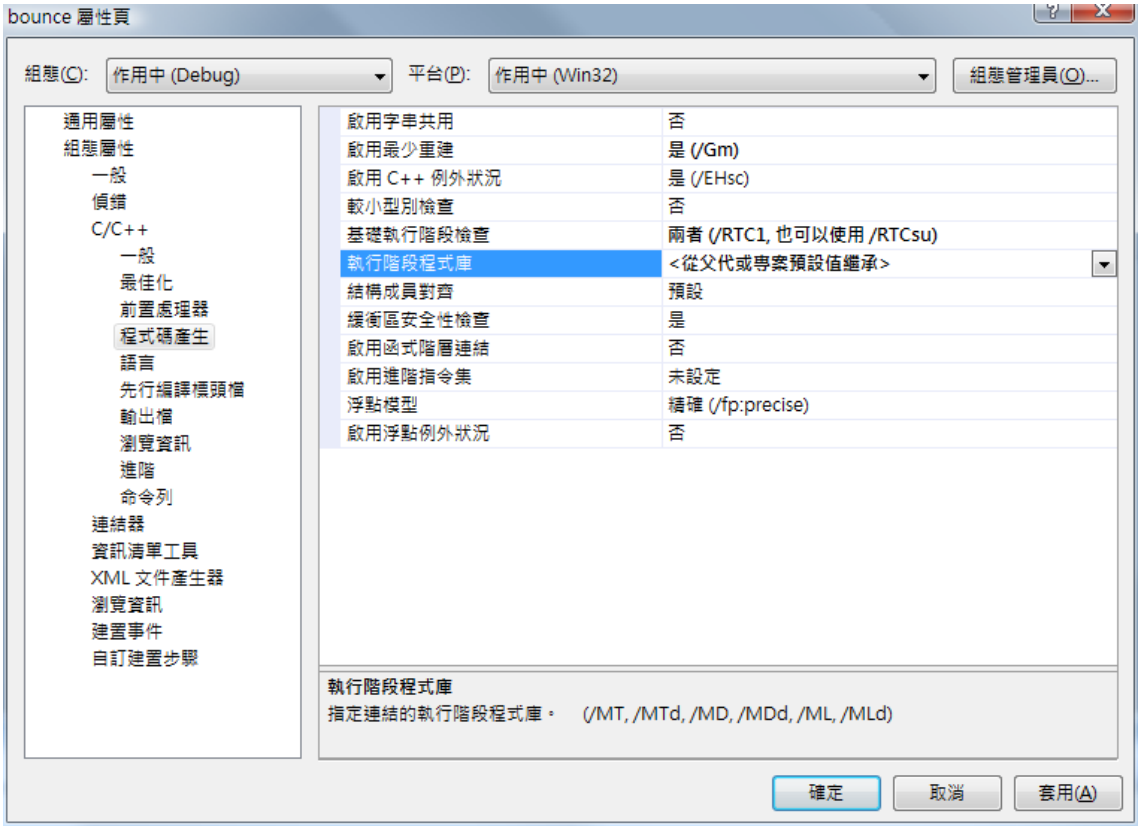

圖 6

3. 見圖7，在【組態屬性】→【連結器】→【輸入】內的【忽略特定程式庫】中加入需忽略的lib

【Debug】模式下：LIBC.LIB

【Release】模式下：LIBC.LIB

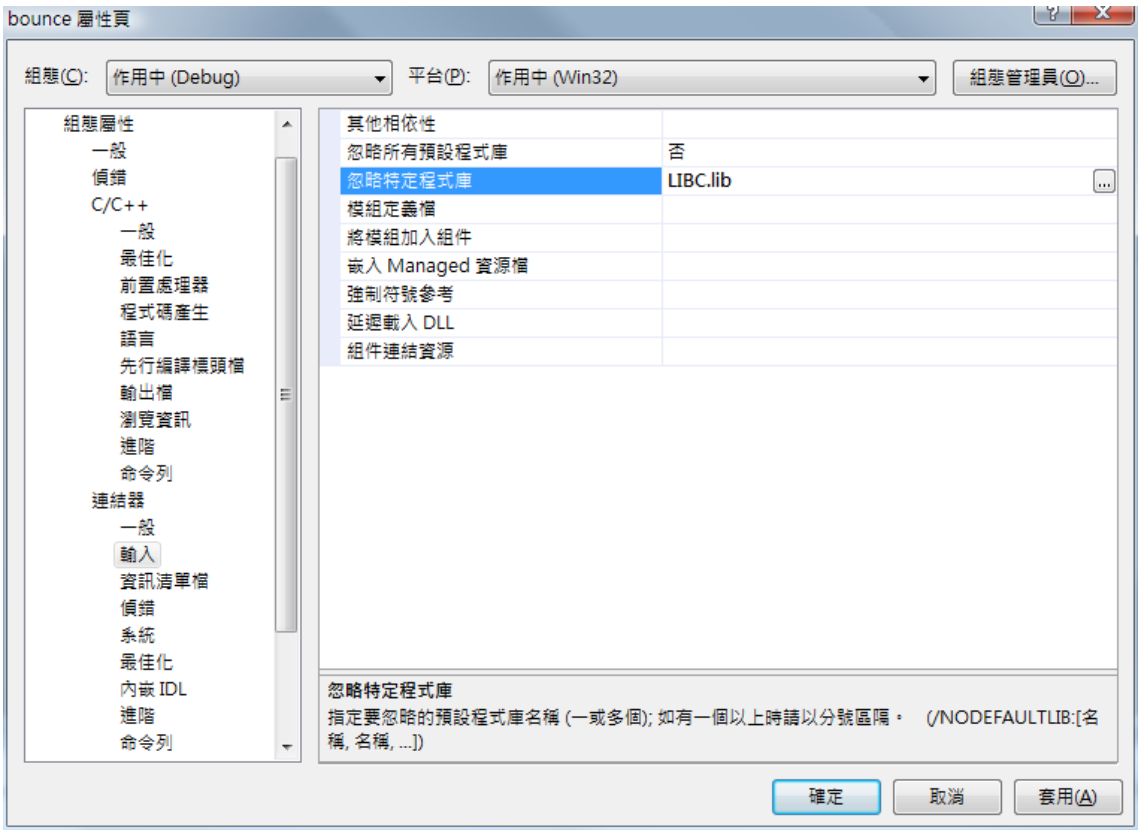


圖 7

# 將 OpenGL 加入 CView (MFC)

➢ 引入標頭檔

```
#include <gl/gl.h>
#include <gl/glu.h>
#include <gl/glaux.h>
#include <gl/glut.h>
```

➢ 專案的連結器中加入

opengl32.lib, glaux.lib, glu32.lib, glut32.lib

➢ view 中加入 2 個成員、2 個函數、4 個訊息對應函數

```
HDC m_hdc;
HGLRC m_rc;
bool SetDCPixelFormat(HDC hDC);
void RenderScene(void);
afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
afx_msg void OnSize(UINT nType, int cx, int cy);
afx_msg void OnDestroy();
afx_msg BOOL OnEraseBkgnd(CDC* pDC);
```

---

```
bool CglView::SetDCPixelFormat(HDC hDC)
{
    PIXELFORMATDESCRIPTOR pixelDesc;

    pixelDesc.nSize         = sizeof(PIXELFORMATDESCRIPTOR);
    pixelDesc.nVersion = 1;

    pixelDesc.dwFlags  =    PFD_DRAW_TO_WINDOW | PFD_SUPPORT_OPENGL |
                            PFD_DOUBLEBUFFER | PFD_STEREO_DONTCARE;

    pixelDesc.iPixelType        = PFD_TYPE_RGBA;
    pixelDesc.cColorBits        = 32;
    pixelDesc.cRedBits          = 8;
    pixelDesc.cRedShift         = 16;
    pixelDesc.cGreenBits        = 8;
    pixelDesc.cGreenShift       = 8;
    pixelDesc.cBlueBits         = 8;
    pixelDesc.cBlueShift        = 0;
    pixelDesc.cAlphaBits        = 0;
    pixelDesc.cAlphaShift       = 0;
```

```
	pixelDesc.cAccumBits		= 64;
	pixelDesc.cAccumRedBits		= 16;
	pixelDesc.cAccumGreenBits	= 16;
	pixelDesc.cAccumBlueBits	= 16;
	pixelDesc.cAccumAlphaBits	= 0;
	pixelDesc.cDepthBits		= 32;
	pixelDesc.cStencilBits		= 8;
	pixelDesc.cAuxBuffers		= 0;
	pixelDesc.iLayerType		= PFD_MAIN_PLANE;
	pixelDesc.bReserved		= 0;
	pixelDesc.dwLayerMask		= 0;
	pixelDesc.dwVisibleMask		= 0;
	pixelDesc.dwDamageMask		= 0;

	int m_GLPixelIndex = ChoosePixelFormat( hDC, &pixelDesc);
	if (m_GLPixelIndex==0) // Let's choose a default index.
	{
		m_GLPixelIndex = 1;
		if (DescribePixelFormat(hDC, m_GLPixelIndex,
					sizeof(PIXELFORMATDESCRIPTOR), &pixelDesc)==0)
		{
			return FALSE;
		}
	}

	if (SetPixelFormat( hDC, m_GLPixelIndex, &pixelDesc)==FALSE)
	{
		return FALSE;
	}
	return TRUE;
}
```

---

```
void CglView::RenderScene(void)
{
	// Combine m_hdc and m_rc
	wglMakeCurrent(m_hdc,m_rc);
	// Clear buffers
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );

	// Draw
	glColor3f(1,1,1);
```

```
	glBegin(GL_TRIANGLES);
	glVertex3f(0,0,-10);
	glVertex3f(100,0,-10);
	glVertex3f(100,100,-10);
	glEnd();

	if (SwapBuffers(m_hdc)==FALSE)
	{
		DWORD errorCode = GetLastError();
		CString errorStr;
		errorStr.Format(_T("SwapBuffers returned error code %d."), errorCode);
		AfxMessageBox(errorStr);
		return;
	}
}
```

---

```
int CglView::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
	if (CView::OnCreate(lpCreateStruct) == -1)
		return -1;

	// TODO:  在此加入特別建立的程式碼
	// Get HDC
	m_hdc = GetDC()->GetSafeHdc();

	// Choose pixel format
	if( !SetDCPixelFormat(m_hdc) )
		return -1;

	// Create RC
	m_rc = wglCreateContext(m_hdc);

	// Basic setup here ...
	wglMakeCurrent(m_hdc,m_rc);

	// Clear color
	glClearColor(0.2f,0.2f,0.2f,1.0f);
	// Shade mode
	glShadeModel(GL_FLAT);
	// Winding
	glFrontFace(GL_CCW);
```

```
	// Culling
	glEnable(GL_CULL_FACE);
	// Depth test
	glEnable(GL_DEPTH_TEST);
	// Polygon mode
	glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);

	return 0;
}
```

---

```
void CglView::OnDestroy()
{
	CView::OnDestroy();

	// TODO: 在此加入您的訊息處理常式程式碼
	if(m_hdc != 0)
	{
		::wglMakeCurrent(m_hdc,NULL);
		m_hdc=0;
	}
	if(m_rc != 0)
	{
		::wglDeleteContext(m_rc);
		m_rc=0;
	}
}
```

---

```
void CglView::OnSize(UINT nType, int cx, int cy)
{
	CView::OnSize(nType, cx, cy);

	// TODO: 在此加入您的訊息處理常式程式碼
	wglMakeCurrent(m_hdc,m_rc);

	glViewport(0, 0, cx, cy);

	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	glOrtho(-200, 200, -200, 200, 200, -200);
```

```
	glMatrixMode(GL_MODELVIEW);
	glLoadIdentity();
}
```

---

```
BOOL CglView::OnEraseBkgnd(CDC* pDC)
{
	// TODO:  在此加入您的訊息處理常式程式碼和(或)  呼叫預設值

	return 1;//CView::OnEraseBkgnd(pDC);
}
```

---

```
void CglView::OnDraw(CDC* /*pDC*/)
{
	CglDoc* pDoc = GetDocument();
	ASSERT_VALID(pDoc);
	if (!pDoc)
		return;

	// TODO:  在此加入原生資料的描繪程式碼
	RenderScene();
}
```

# 將 OpenGL 加入 CDialog (MFC)

- 建立一個類別(ex. COpenGLCtrl)繼承於 CWnd
- 進行 “將 OpenGL 加入 CView (MFC)” 的步驟，不過，最後的 OnDraw()改為 OnPaint()。

```
void COpenGLCtrl::OnPaint()
{
	CPaintDC dc(this); // device context for painting
	// TODO: 在此加入您的訊息處理常式程式碼
	// 不要呼叫圖片訊息的CWnd::OnPaint()
	RenderScene();
}
```

- 再增加一個函數

```
void COpenGLCtrl::Create(CRect rect,CWnd* parent)
{
	DestroyWindow();

	CString className = AfxRegisterWndClass(
		CS_HREDRAW | CS_VREDRAW | CS_OWNDC,
		NULL,
		(HBRUSH)GetStockObject(BLACK_BRUSH),
		NULL);

	CreateEx(
		0,
		className,
		_T("OpenGL"),
		WS_CHILD | WS_VISIBLE | WS_CLIPSIBLINGS | WS_CLIPCHILDREN,
		rect,
		parent,
		0);
}
```

- 在需要 OpenGL 的 Dialog 中加入 Static Text 或 Picture Control 的控制項。
  修改 ID，不能用預設的 IDC_STATIC。 (ex. IDC_OPENGL)
  屬性設定如下：
  - Static Text:
    Visible → “False”
  - Picture Control:
    Transparent → “True”

- 在 dialog 的 .h 檔中引入前面新增的類別

```
#include "OpenGLCtrl.h"
```

- 宣告該類別物件於 dialog 類別中

```
COpenGLCtrl glCtrl;
```

- 在 dialog 的 OnInitDialog()中建立繪圖區，完成。

---

```
BOOL CglDialogDlg::OnInitDialog()
{
	CDialog::OnInitDialog();

	// 將[關於...] 功能表加入系統功能表。

	// IDM_ABOUTBOX 必須在系統命令範圍之中。
	ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
	ASSERT(IDM_ABOUTBOX < 0xF000);

	CMenu* pSysMenu = GetSystemMenu(FALSE);
	if (pSysMenu != NULL)
	{
		CString strAboutMenu;
		strAboutMenu.LoadString(IDS_ABOUTBOX);
		if (!strAboutMenu.IsEmpty())
		{
			pSysMenu->AppendMenu(MF_SEPARATOR);
			pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
		}
	}

	// 設定此對話方塊的圖示。當應用程式的主視窗不是對話方塊時，
	// 框架會自動從事此作業
	SetIcon(m_hIcon, TRUE);			// 設定大圖示
	SetIcon(m_hIcon, FALSE);		// 設定小圖示
```

```
    // TODO:  在此加入額外的初始設定
    CRect rect;
    GetDlgItem(IDC_OPENGL)->GetWindowRect(rect);
    ScreenToClient(rect);
    glCtrl.Create(rect, this);

    return TRUE;  // 傳回TRUE，除非您對控制項設定焦點
}
```