Testing / Running DirectX9 code in Microsoft Visual C++ 2010 Express

I selected one program from Chapter 2 (Draw2D) to test compiling and running using DirectX9.

- Create an empty Window Project named TestDirectX9
- Created the file WinMain.cpp and copied the contents from book version
- Copy the file Texture.bmp to the same directory where the WinMain.cpp is saved

You know you have successfully compiled, linked and executed when you see the application open a window that appears as follows:

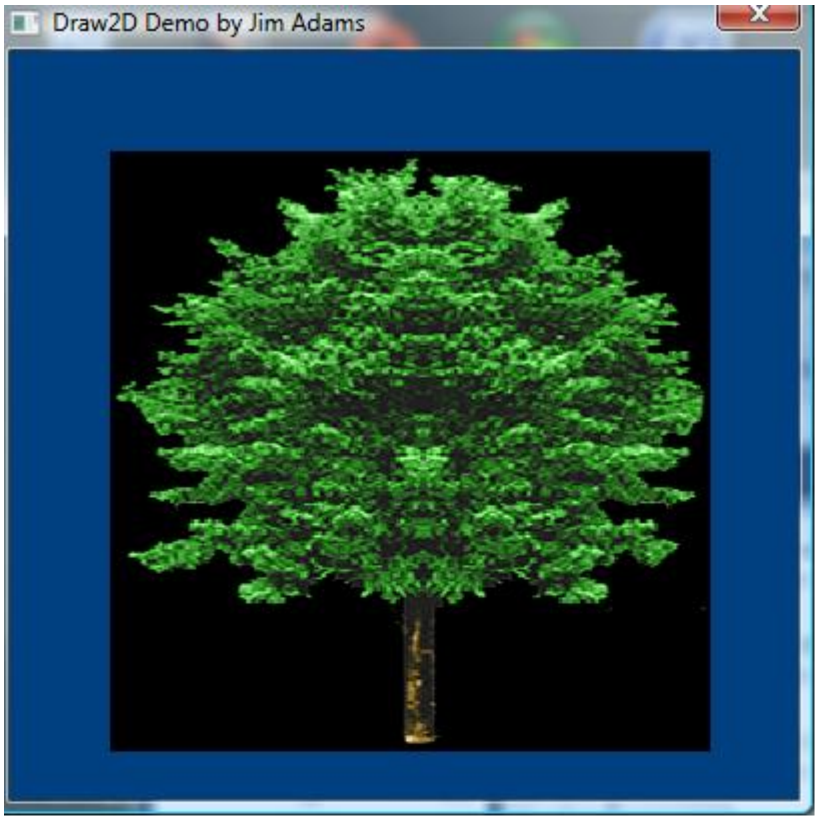


Figure 1 - Results of running Draw2D

The code is as follows:

Table 1- Code for Draw2D

```
/**************************************************
WinMain.cpp
Chapter 2 2-D Drawing Demo (Draw2D)

Programming Role-Playing Games with DirectX, 2nd Edition
```

```
by Jim Adams (Jan 2004)

Required libraries:
  D3D9.LIB and D3DX9.LIB
**************************************************/

// Include files
#include <windows.h>
#include <stdio.h>
#include "d3d9.h"
#include "d3dx9.h"

#ifdef _DEBUG

#   pragma comment(lib, "d3dx9d.lib")

#else

#   pragma comment(lib, "d3dx9.lib")

#endif

#pragma comment(lib, "d3d9.lib")

// Window handles, class and caption text
HWND          g_hWnd;
HINSTANCE     g_hInst;
static char   g_szClass[]   = "Draw2DClass";
static char   g_szCaption[] = "Draw2D Demo by Jim Adams";

// The Direct3D and Device object
IDirect3D9       *g_pD3D       = NULL;
IDirect3DDevice9 *g_pD3DDevice = NULL;

// The 2-D vertex format and descriptor
typedef struct {
  FLOAT x, y, z;     // 2-D coordinates
  FLOAT rhw;         // rhw
  FLOAT u, v;        // Texture coordinates
} sVertex;
#define VERTEXFVF (D3DFVF_XYZRHW | D3DFVF_TEX1)

// Vertex buffer
IDirect3DVertexBuffer9 *g_pVB = NULL;

// Texture
IDirect3DTexture9 *g_pTexture = NULL;

// Function prototypes
int PASCAL WinMain(HINSTANCE hInst, HINSTANCE hPrev,          \
                   LPSTR szCmdLine, int nCmdShow);
long FAR PASCAL WindowProc(HWND hWnd, UINT uMsg,              \
                           WPARAM wParam, LPARAM lParam);

BOOL DoInit();
BOOL DoShutdown();
BOOL DoFrame();
BOOL SetupMeshes();
```

```
int PASCAL WinMain(HINSTANCE hInst, HINSTANCE hPrev,          \
                   LPSTR szCmdLine, int nCmdShow)
{
  WNDCLASSEX wcex;
  MSG        Msg;

  g_hInst = hInst;

  // Create the window class here and register it
  wcex.cbSize        = sizeof(wcex);
  wcex.style         = CS_CLASSDC;
  wcex.lpfnWndProc   = WindowProc;
  wcex.cbClsExtra    = 0;
  wcex.cbWndExtra    = 0;
  wcex.hInstance     = hInst;
  wcex.hIcon         = LoadIcon(NULL, IDI_APPLICATION);
  wcex.hCursor       = LoadCursor(NULL, IDC_ARROW);
  wcex.hbrBackground = NULL;
  wcex.lpszMenuName  = NULL;
  wcex.lpszClassName = g_szClass;
  wcex.hIconSm       = LoadIcon(NULL, IDI_APPLICATION);
  if(!RegisterClassEx(&wcex))
    return FALSE;

  // Create the Main Window
  g_hWnd = CreateWindow(g_szClass, g_szCaption,
        WS_CAPTION | WS_SYSMENU,
        0, 0, 400, 400,
        NULL, NULL,
        hInst, NULL );
  if(!g_hWnd)
    return FALSE;
  ShowWindow(g_hWnd, SW_NORMAL);
  UpdateWindow(g_hWnd);

  // Run init function and return on error
  if(DoInit() == FALSE)
    return FALSE;

  // Start message pump, waiting for signal to quit
  ZeroMemory(&Msg, sizeof(MSG));
  while(Msg.message != WM_QUIT) {
    if(PeekMessage(&Msg, NULL, 0, 0, PM_REMOVE)) {
      TranslateMessage(&Msg);
      DispatchMessage(&Msg);
    }
    if(DoFrame() == FALSE)
      break;
  }

  // Run shutdown function
  DoShutdown();

  UnregisterClass(g_szClass, hInst);

  return Msg.wParam;
}
```

```
long FAR PASCAL WindowProc(HWND hWnd, UINT uMsg,                 \
                           WPARAM wParam, LPARAM lParam)
{
  switch(uMsg) {
    case WM_DESTROY:
      PostQuitMessage(0);
      return 0;

  }

  return DefWindowProc(hWnd, uMsg, wParam, lParam);
}

BOOL DoInit()
{
  D3DPRESENT_PARAMETERS d3dpp;
  D3DDISPLAYMODE        d3ddm;
  BYTE *Ptr;
  sVertex Verts[4] = {
      {  50.0f,  50.0f, 1.0f, 1.0f, 0.0f, 0.0f },
      { 350.0f,  50.0f, 1.0f, 1.0f, 1.0f, 0.0f },
      {  50.0f, 350.0f, 1.0f, 1.0f, 0.0f, 1.0f },
      { 350.0f, 350.0f, 1.0f, 1.0f, 1.0f, 1.0f }
    };

  // Do a windowed mode initialization of Direct3D
  if((g_pD3D = Direct3DCreate9(D3D_SDK_VERSION)) == NULL)
    return FALSE;
  if(FAILED(g_pD3D->GetAdapterDisplayMode(D3DADAPTER_DEFAULT, &d3ddm)))
    return FALSE;
  ZeroMemory(&d3dpp, sizeof(d3dpp));
  d3dpp.Windowed = TRUE;
  d3dpp.SwapEffect = D3DSWAPEFFECT_DISCARD;
  d3dpp.BackBufferFormat = d3ddm.Format;
  d3dpp.EnableAutoDepthStencil = FALSE;
  if(FAILED(g_pD3D->CreateDevice(D3DADAPTER_DEFAULT, D3DDEVTYPE_HAL, g_hWnd,
                                  D3DCREATE_SOFTWARE_VERTEXPROCESSING,
                                  &d3dpp, &g_pD3DDevice)))
    return FALSE;

  // Create the vertex buffer and set data
  g_pD3DDevice->CreateVertexBuffer(sizeof(sVertex)*4, 0,         \
                          VERTEXFVF, D3DPOOL_DEFAULT,            \
                          &g_pVB, NULL);
  g_pVB->Lock(0,0, (void**)&Ptr, 0);
  memcpy(Ptr, Verts, sizeof(Verts));
  g_pVB->Unlock();

  // Load the texture map
  D3DXCreateTextureFromFile(g_pD3DDevice, "Texture.bmp", &g_pTexture);

  return TRUE;
}

BOOL DoShutdown()
{
  // Release vertex buffer
  if(g_pVB != NULL)
    g_pVB->Release();

  // Release texture
  if(g_pTexture != NULL)
    g_pTexture->Release();

  // Release device and 3D objects
  if(g_pD3DDevice != NULL)
    g_pD3DDevice->Release();

  if(g_pD3D != NULL)
    g_pD3D->Release();

  return TRUE;
}

BOOL DoFrame()
{
  // Clear device backbuffer
  g_pD3DDevice->Clear(0, NULL, D3DCLEAR_TARGET,                \
                      D3DCOLOR_RGBA(0,64,128,255), 1.0f, 0);

  // Begin scene
  if(SUCCEEDED(g_pD3DDevice->BeginScene())) {

    // Set the vertex stream, shader, and texture
    g_pD3DDevice->SetStreamSource(0, g_pVB, 0, sizeof(sVertex));
    g_pD3DDevice->SetFVF(VERTEXFVF);
    g_pD3DDevice->SetTexture(0, g_pTexture);

    // Draw the vertex buffer
    g_pD3DDevice->DrawPrimitive(D3DPT_TRIANGLESTRIP, 0, 2);

    // Release texture
    g_pD3DDevice->SetTexture(0, NULL);

    // End the scene
    g_pD3DDevice->EndScene();
  }

  // Display the scene
  g_pD3DDevice->Present(NULL, NULL, NULL, NULL);

  return TRUE;
}
```

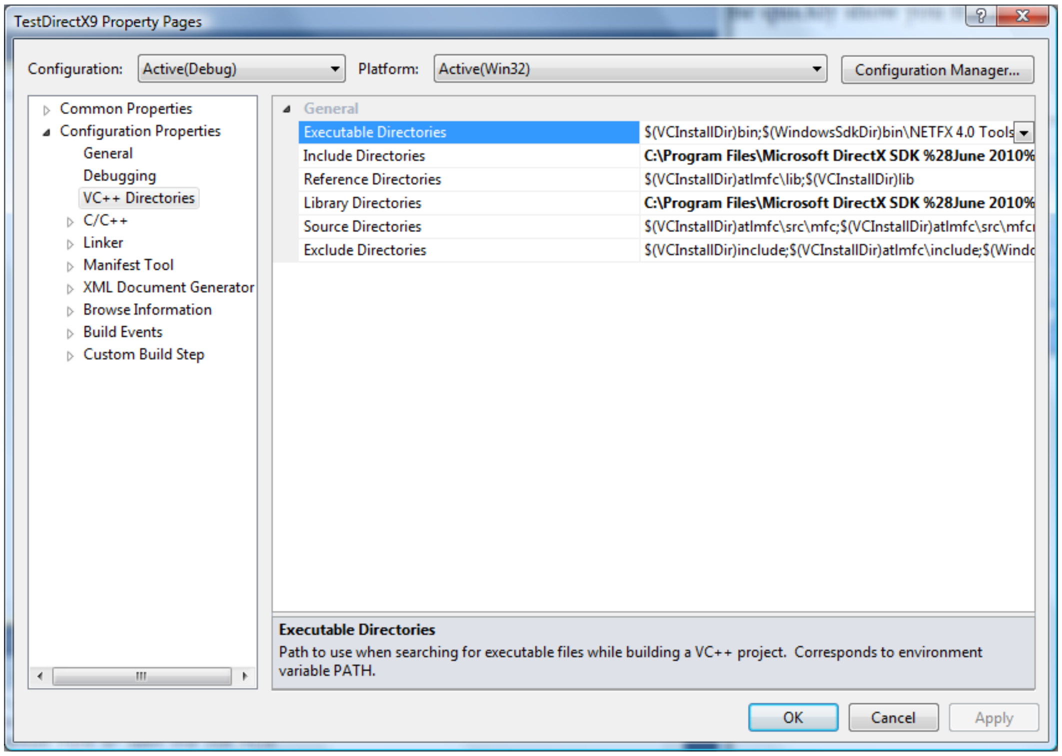
- Added the following pragma's to the code:

```
#ifdef _DEBUG

#   pragma comment(lib, "d3dx9d.lib")

#else
```

```
#   pragma comment(lib, "d3dx9.lib")
#endif
#pragma comment(lib, "d3d9.lib")
```

The above was added after all the #include statements

- In Project | Properties menu



TestDirectX9 Property Pages

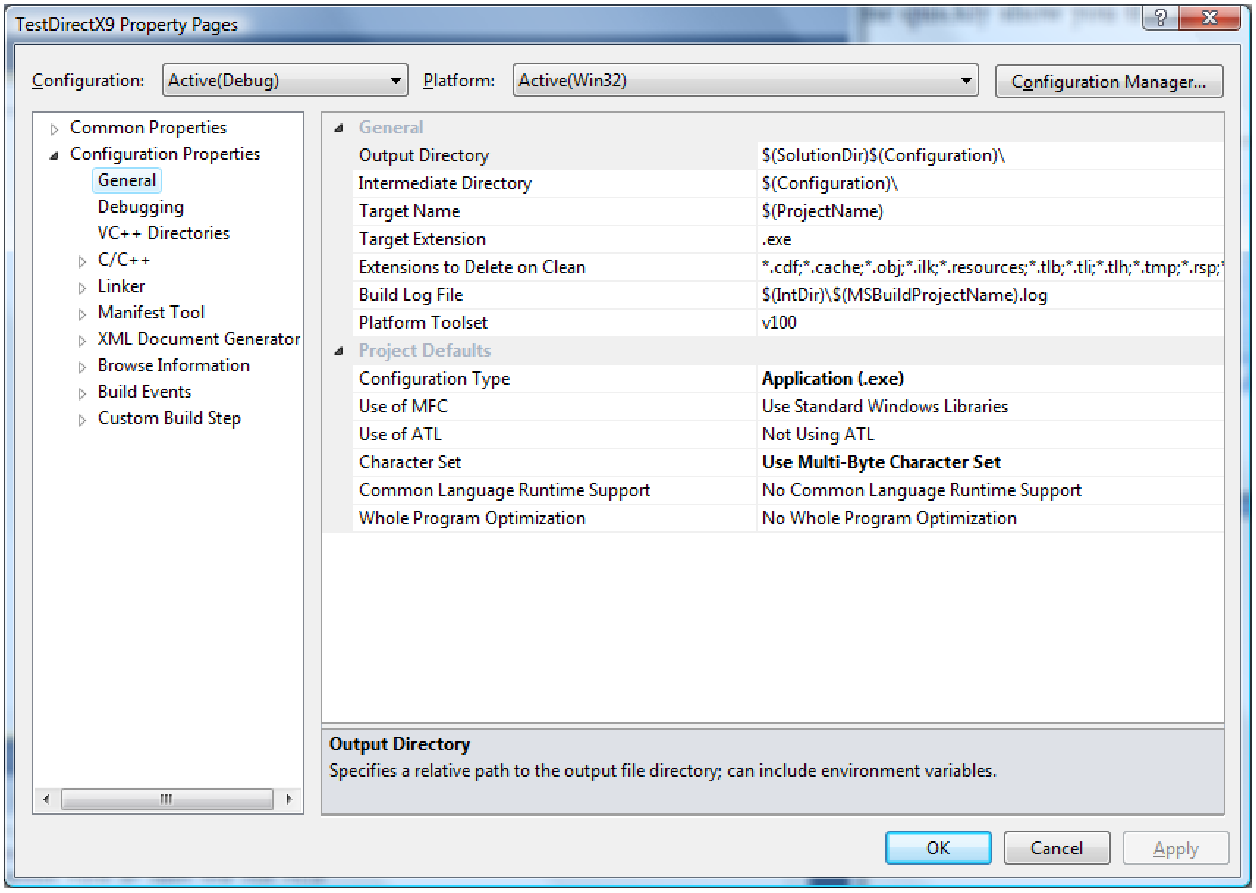
Configuration: Active(Debug) Platform: Active(Win32) Configuration Manager...

General

Executable Directories	$(VCInstallDir)bin;$(WindowsSdkDir)bin\NETFX 4.0 Tools
Include Directories	**C:\Program Files\Microsoft DirectX SDK %28June 2010%**
Reference Directories	$(VCInstallDir)atlmfc\lib;$(VCInstallDir)lib
Library Directories	**C:\Program Files\Microsoft DirectX SDK %28June 2010%**
Source Directories	$(VCInstallDir)atlmfc\src\mfc;$(VCInstallDir)atlmfc\src\mfc
Exclude Directories	$(VCInstallDir)include;$(VCInstallDir)atlmfc\include;$(Windo

Executable Directories
Path to use when searching for executable files while building a VC++ project. Corresponds to environment variable PATH.

I added “C:\Program Files\Microsoft DirectX SDK (June 2010)\Include” directory to the “Include Directories” and “C:\Program Files\Microsoft DirectX SDK(Jun 2010)\Lib\x86” to the “Library Directories”



General	
Output Directory	$(SolutionDir)$(Configuration)\
Intermediate Directory	$(Configuration)\
Target Name	$(ProjectName)
Target Extension	.exe
Extensions to Delete on Clean	*.cdf;*.cache;*.obj;*.ilk;*.resources;*.tlb;*.tli;*.tlh;*.tmp;*.rsp;
Build Log File	$(IntDir)\$(MSBuildProjectName).log
Platform Toolset	v100
Project Defaults	
Configuration Type	**Application (.exe)**
Use of MFC	Use Standard Windows Libraries
Use of ATL	Not Using ATL
Character Set	**Use Multi-Byte Character Set**
Common Language Runtime Support	No Common Language Runtime Support
Whole Program Optimization	No Whole Program Optimization

Output Directory
Specifies a relative path to the output file directory; can include environment variables.

- Changed Character Set to “Use Multi-Byte Character Set”

Note: The above assumes that you installed the latest Microsoft DirectX SDK (June 2010). This was the highest version I could install given my operating system (Windows Vista). The SDK supports DirectX9, DirectX10, and DirectX11.