**Windows 10과 Visual Studio 2015에서의 Direct3D 11 데모**

Frank Luna

April 4, 2016

[www.d3dcoder.net](http://www.d3dcoder.net/)

Windows 8에서 DirectX를 사용할 때 더 이상 분리된 SDK를 사용하지 않습니다. 대신에 DirectX가 OS에 이미 내장되어 있으며, Windows 업데이트를 통해 최신 버전을 얻을 수 있습니다. 더 나아가, D3DX 라이브러리는 제거되었으며, Windows에서 제공하지 않습니다. XNA 수학 라이브러리도 이제 DirectX Math라는 이름으로 바뀌었습니다. 또한 dxerr.lib는 Visual Studio 2015와 더 이상 호환이 되지 않습니다. 그래서 책의 Direct3D 11 데모들을 빌드하고 실행시키는 데에 필요한 전략은 아래의 것들로 요약될 수 있습니다.

1.  데모가 사용하는 D3DX 기능들의 대체를 찾으세요. 들리는 것보단 쉽습니다. DirectX팀은 많은 D3DX 기능들을 오픈 소스로 제공하기 때문입니다.

2. DirectX Math 로 바꾸세요. 라이브러리가 꽤 비슷해서 간단합니다. 그냥 DirectX Math 라이브러리를 추가하고 DirectX 네임스페이스를 추가하세요.

3. dxerr.lib의 대체방법을 찾던가 그 기능을 사용하지 마세요. 책의 데모에서는 아래의 에러 찾는 코드밖에 사용하지 않습니다.

         #define HR(x)                                               \

        {                                                           \

            HRESULT hr = (x);                                       \

            if(FAILED(hr))                                          \

            {                                                       \

                DXTrace(\_\_FILEW\_\_, (DWORD)\_\_LINE\_\_, hr, L#x, true); \

            }                                                       \

        }

2010년 6월 버전의 DirectX SDK를 설치해 D3DX와 XNA Math라이브러리를 얻는 방법도 있습니다만, 이 방법은 해결해야 할 몇 가지 충돌사항을 만들어냅니다. 게다가 이 SDK는 지금으로부터 6년이나 된지라 오래된 SDK를 쓰려고 우리한테 필요 없는 것까지 접근해야 합니다.

**Downloads**

DirectX 11책은 Effects 프레임워크를 사용합니다. Effects 프레임워크의 오픈 소스 버전은 Windows 8이나 그 상위 버전의 윈도우에서도 사용할 수 있으며, 아래의 주소에서 찾을 수 있습니다.

<https://github.com/Microsoft/FX11>

소스를 다운로드하고, Effects11\_2015 솔루션 파일을 열어 라이브러리를 다시 빌드하세요(Visual Studio 2015를 사용해 디버그 모드, 릴리즈 모드 둘 다). 디버그모드에서 생성된 파일은 *Effects11d.lib*, 릴리즈 모드에서 생성도딘 파일은 *Effects11.lib*로 이름을 바꿔주세요. 그리고 *d3dx11effect.h*, *Effects11d.lib,* *Effects11.lib* 을 책의 Common 디렉토리에 복사해주세요(오래된 파일과 교체).

다음으로, 우리는 texture들을 불러오는 다른 방법이 필요합니다. texture들을 불러오기 위해 사용되던 D3DX 코드는 오픈소스로 열려있습니다.

<https://github.com/Microsoft/DirectXTex>

*DDSTextureLoader을 가져다가 DDSTextureLoader.h/.cpp* 파일을 책의 Common디렉토리에 넣고, 당신의 프로젝트에 추가하세요. 우리는 디스크에서 texture들을 불러오는데 이 코드를 사용할 겁니다.

마지막으로, 우리는 dxerr.lib의 대체 방법이 필요합니다. 이 블로그의 게시글은 우리가 왜 이것을 해야 하는지 설명해주고 있어요: <https://blogs.msdn.microsoft.com/chuckw/2012/04/24/wheres-dxerr-lib/>. 이 링크는 dxerr\_nov2015.zip으로의 링크입니다. dxerr.h/.cpp의 압축파일이며, dxerr.lib의 우리가 필요로 하는 기능을 얻기 위해 이 파일들을 책의 Common 디렉토리에 넣으면 됩니다. 이 두 파일을 당신의 프로젝트에 포함하세요.

D3DX에는 쓸모 있는 메쉬 기반의 코드가 있으니 사용하려고 하실 수도 있습니다. 다행히 DirectX팀이 이 코드도 배포해줬어요. 여기 블로그 게시글과 git저장소 주소입니다:

<https://blogs.msdn.microsoft.com/chuckw/2014/06/27/directxmesh/>

<https://github.com/Microsoft/DirectXMesh>

관심 있으신 분들은 Chuck Walbourn의 블로그에서 DirectX의 변경사항을 시간순서대로 읽을 수 있어요: <https://blogs.msdn.microsoft.com/chuckw/>

**Project Changes**

'C/C++ > 일반 > 추가 포함 디렉터리'에 책의 Common폴더를 추가하세요

 '링커 > 일반 > 추가 라이브러리 디렉터리'에 책의 Common 폴더를 추가하세요

**Code Changes**

이 항목에서, 'BlendDemo'를 빌드하고 실행시키는데 필요한 변경사항을 설명하겠습니다.

 모든 #include<d3dx11.h>를 지우고, '링커 > 입력 > 추가 종속성'에서 d3dx11.lib와 d3dx11d.lib를 지우세요. 이제는 D3DX11을 위해서 DirectX를 추가하지 않아도 되요.

 '링커 > 입력 > 추가 종속성'에서 dxerr.lib를 지우세요. 대신 프로젝트에 dxerr.h/.cpp를 추가하고 #include <dxerr.h>를 #include "dxerr.h"로 바꾸세요.

 d3dUtil.h파일에 #include "DDSTextureLoader.h"를 추가하세요.

 모든 #include <xnamath.h>를 #include <DirectXMath.h>와 #include <DirectXPackedVector.h>로 바꾸세요.

 모든 DirectX math 타입들(XMVECTOR, XMFLOAT4X4, XMMatrixTranspose, 등등.)은 앞에 "DirectX::"가 붙어야 해요. 얘들은 이제 다 DirectX 네임스페이스 안에 들어있거든요.

 XMCOLOR에 관련된 것들은 "DirectX::PackedVector::"가 앞에 붙어야 해요. 얘넨 다들 DirectX::PackedVector안에 들어있거든요.

 D3DX11INLINE들을 전부 지우세요. 그냥 지워도 되고 inline으로 교체해도 됩니다.

 지금은 d3dUtil.h에서 d3dHelper::CreateTexture2DArraySRV 함수를 지우세요. texture배열을 만들기 위해서 texassemble툴을 사용할 수 있어요(<https://directxtex.codeplex.com/wikipage?title=Texassemble&referringTitle=Texconv>).

 d3dUtil.h파일에서 \_\_FILE\_\_(DXTrace함수의 인자로 들어간다)을 \_\_FILEW\_\_로 바꿔주세요. 파일을 wide string으로 얻어올 수 있어요.

 d3dUtil.h에서 ExtractFrustumPlanes 함수를 아래의 코드로 바꿔주세요(변경사항은 인자로 받아온 행렬을 XMFLOAT4X4로 바꾸는 부분입니다. 이로써 우리는 오버로딩된 parenthesis 오퍼레이터를 사용할 수 있습니다.).

void ExtractFrustumPlanes(XMFLOAT4 planes[6], CXMMATRIX T)

{

XMFLOAT4X4 M;

XMStoreFloat4x4(&M, T);

//

// Left

//

planes[0].x = M(0,3) + M(0,0);

planes[0].y = M(1,3) + M(1,0);

planes[0].z = M(2,3) + M(2,0);

planes[0].w = M(3,3) + M(3,0);

//

// Right

//

planes[1].x = M(0,3) - M(0,0);

planes[1].y = M(1,3) - M(1,0);

planes[1].z = M(2,3) - M(2,0);

planes[1].w = M(3,3) - M(3,0);

//

// Bottom

//

planes[2].x = M(0,3) + M(0,1);

planes[2].y = M(1,3) + M(1,1);

planes[2].z = M(2,3) + M(2,1);

planes[2].w = M(3,3) + M(3,1);

//

// Top

//

planes[3].x = M(0,3) - M(0,1);

planes[3].y = M(1,3) - M(1,1);

planes[3].z = M(2,3) - M(2,1);

planes[3].w = M(3,3) - M(3,1);

//

// Near

//

planes[4].x = M(0,2);

planes[4].y = M(1,2);

planes[4].z = M(2,2);

planes[4].w = M(3,2);

//

// Far

//

planes[5].x = M(0,3) - M(0,2);

planes[5].y = M(1,3) - M(1,2);

planes[5].z = M(2,3) - M(2,2);

planes[5].w = M(3,3) - M(3,2);

// Normalize the plane equations.

for(int i = 0; i < 6; ++i)

{

    XMVECTOR v = XMPlaneNormalize(XMLoadFloat4(&planes[i]));

    XMStoreFloat4(&planes[i], v);

}

}

 texture를 불러오는 함수인 D3DX11CreateShaderResourceViewFromFile 를 DirectX::CreateDDSTextureFromFile로 바꿔주세요. 예를 들어 "BlendDemo"에서 우리는 아래의 코드를

HR(D3DX11CreateShaderResourceViewFromFile(md3dDevice,

    L"Textures/grass.dds", 0, 0, &mGrassMapSRV, 0 ));

HR(D3DX11CreateShaderResourceViewFromFile(md3dDevice,

    L"Textures/water2.dds", 0, 0, &mWavesMapSRV, 0 ));

HR(D3DX11CreateShaderResourceViewFromFile(md3dDevice,

    L"Textures/WireFence.dds", 0, 0, &mBoxMapSRV, 0 ));

이렇게 바꿀 거에요.

ID3D11Resource\* texResource = nullptr;

HR(DirectX::CreateDDSTextureFromFile(md3dDevice,

    L"Textures/grass.dds", &texResource, &mGrassMapSRV));

ReleaseCOM(texResource); // view saves reference

HR(DirectX::CreateDDSTextureFromFile(md3dDevice,

    L"Textures/water2.dds", &texResource, &mWavesMapSRV));

    ReleaseCOM(texResource); // view saves reference

HR(DirectX::CreateDDSTextureFromFile(md3dDevice,

    L"Textures/WireFence.dds", &texResource, &mBoxMapSRV));

ReleaseCOM(texResource); // view saves reference

이렇게 하면 "BlendDemo"는 잘 작동할 겁니다.

==========================================================

# Direct3D 11 Book Demos with Windows 10 and Visual Studio 2015

Frank Luna

April 4, 2016

[www.d3dcoder.net](http://www.d3dcoder.net/)

Recently, I’ve received emails about issues with getting the Direct3D 11 book demos to compile using Visual Studio 2015.  This article will explain the necessary steps.  But first a little background on why things broke.

Starting with Windows 8, DirectX is no longer a separate SDK that you download and install; instead, DirectX ships with the OS and gets updated through Windows Update.  Moreover, the D3DX library has been deprecated and does not ship with Windows, and the XNA Math library now goes by the name DirectX Math.  Also, starting with Visual Studio 2015, the dxerr.lib is no longer compatible with Visual Studio 2015.  So our strategy to get the Direct3D 11 demos to build and run can be summarized as follows:

1.      Find replacements for the D3DX functionality the demos use.  This is easier than it sounds; as the DirectX team has open sourced a lot of the D3DX functionality.

2.      Switch to using DirectX Math—this is pretty easy as the libraries are pretty much the same, we just need to account for DirectX Math putting everything in the DirectX namespace.

3.      Find a replacement for dxerr.lib or stop using its functionality.  We used this library for our error tracing code:

        #define HR(x)                                               \

        {                                                           \

            HRESULT hr = (x);                                       \

            if(FAILED(hr))                                          \

            {                                                       \

                DXTrace(\_\_FILEW\_\_, (DWORD)\_\_LINE\_\_, hr, L#x, true); \

            }                                                       \

        }

Note that an alternative strategy would be to install the old June 2010 DirectX SDK to get D3DX and XNA Math, but it creates some additional conflicts that need to be solved, and that SDK is 6 years old at the time of this writing, so we should take the approach that does not require us to download an old SDK.

## Downloads

The DirectX 11 book uses the Effects framework.  An open source version of the Effects framework that is compatible with the DirectX versions that ship with Windows 8 and above can be found here:

<https://github.com/Microsoft/FX11>

Download the source, open the Effects11\_2015 solution file and rebuild the libraries in debug and release mode using Visual Studio 2015.  Name the debug one *Effects11d.lib* and the release one *Effects11.lib*.  Copy the files: *d3dx11effect.h*, *Effects11d.lib* and *Effects11.lib* into the book’s*Common* directory (overwriting the old ones).

            Next, we need a replacement for loading textures.  Again, the D3DX code used for loading textures has been open sourced:

<https://github.com/Microsoft/DirectXTex>

Pull *DDSTextureLoader*, drop *DDSTextureLoader.h/.cpp* into the book’s *Common* directory, and add these files to your project.  We will use this code to load textures from disk.

            Finally, we need a replacement for dxerr.lib.  This blog post summarizes the details of why we need this: <https://blogs.msdn.microsoft.com/chuckw/2012/04/24/wheres-dxerr-lib/>.  In short, the preceding link has a link to *dxerr\_nov2015.zip*, which contains two files *dxerr.h* and *dxerr.cpp*that we can drop in the book’s Common directory to get the dxerr.lib functionality we need.  Add these two files to your project.

Readers may recall that D3DX also had some useful mesh based code.  Fortunately, the DirectX team has released this code as well.  Here is the blog post and a link the repository:

<https://blogs.msdn.microsoft.com/chuckw/2014/06/27/directxmesh/>

<https://github.com/Microsoft/DirectXMesh>

For those interested, you can read a lot about the historical changes of DirectX at Chuck Walbourn’s blog: <https://blogs.msdn.microsoft.com/chuckw/>

## Project Changes

         In the C/C++ Property Pages > General, add the book’s *Common* directory to the “Additional Include Directories” list.

         In the Linker Property Pages > General, add the book’s *Common* directory to the “Additional Library Directories” list.

## Code Changes

In this section, we describe the changes needed to get the “BlendDemo” to build and run.

         Remove all #include <d3dx11.h> and library dependencies to d3dx11.lib/d3dx11d.lib, as D3DX11 is no longer included with DirectX.

         Remove the library dependency to dxerr.lib.  Add dxerr.h/.cpp to the project and change #include <dxerr.h> to #include “dxerr.h.”

         Add a #include “DDSTextureLoader.h” in d3dUtil.h.

         Replace all references to #include <xnamath.h> with #include <DirectXMath.h> and #include <DirectXPackedVector.h>.

         All DirectX math types (XMVECTOR, XMFLOAT4X4, XMMatrixTranspose, etc.) will need to be prefixed with “DirectX::” because they now live in the DirectX namespace.

         All references to XMCOLOR need to be prefixed with DirectX::PackedVector because XMCOLOR lives in the DirectX::PackedVector namespace.

         Remove all references to D3DX11INLINE.  You can just delete it or replace with inline.

         Remove the helper function d3dHelper::CreateTexture2DArraySRV from *d3dUtil.h* for now.  You can use the texassemble tool (<https://directxtex.codeplex.com/wikipage?title=Texassemble&referringTitle=Texconv>) to create a texture array offline, which would be the preferred process anyway.

         In *d3dUtil.h*, replace \_\_FILE\_\_ in the DXTrace call with \_\_FILEW\_\_ so we get the file as a wide string.

         In *d3dUtil.h*, rewrite ExtractFrustumPlanes like so (the change is to convert the input matrix to XMFLOAT4X4 so we can use the overloaded parenthesis operator to get the elements):

void ExtractFrustumPlanes(XMFLOAT4 planes[6], CXMMATRIX T)

{

XMFLOAT4X4 M;

XMStoreFloat4x4(&M, T);

//

// Left

//

planes[0].x = M(0,3) + M(0,0);

planes[0].y = M(1,3) + M(1,0);

planes[0].z = M(2,3) + M(2,0);

planes[0].w = M(3,3) + M(3,0);

//

// Right

//

planes[1].x = M(0,3) - M(0,0);

planes[1].y = M(1,3) - M(1,0);

planes[1].z = M(2,3) - M(2,0);

planes[1].w = M(3,3) - M(3,0);

//

// Bottom

//

planes[2].x = M(0,3) + M(0,1);

planes[2].y = M(1,3) + M(1,1);

planes[2].z = M(2,3) + M(2,1);

planes[2].w = M(3,3) + M(3,1);

//

// Top

//

planes[3].x = M(0,3) - M(0,1);

planes[3].y = M(1,3) - M(1,1);

planes[3].z = M(2,3) - M(2,1);

planes[3].w = M(3,3) - M(3,1);

//

// Near

//

planes[4].x = M(0,2);

planes[4].y = M(1,2);

planes[4].z = M(2,2);

planes[4].w = M(3,2);

//

// Far

//

planes[5].x = M(0,3) - M(0,2);

planes[5].y = M(1,3) - M(1,2);

planes[5].z = M(2,3) - M(2,2);

planes[5].w = M(3,3) - M(3,2);

// Normalize the plane equations.

for(int i = 0; i < 6; ++i)

{

    XMVECTOR v = XMPlaneNormalize(XMLoadFloat4(&planes[i]));

    XMStoreFloat4(&planes[i], v);

}

}

         Replace texture loading calls D3DX11CreateShaderResourceViewFromFile with the call DirectX::CreateDDSTextureFromFile.  For example, in the “Blend Demo” we replaced

HR(D3DX11CreateShaderResourceViewFromFile(md3dDevice,

    L"Textures/grass.dds", 0, 0, &mGrassMapSRV, 0 ));

HR(D3DX11CreateShaderResourceViewFromFile(md3dDevice,

    L"Textures/water2.dds", 0, 0, &mWavesMapSRV, 0 ));

HR(D3DX11CreateShaderResourceViewFromFile(md3dDevice,

    L"Textures/WireFence.dds", 0, 0, &mBoxMapSRV, 0 ));

with

ID3D11Resource\* texResource = nullptr;

HR(DirectX::CreateDDSTextureFromFile(md3dDevice,

    L"Textures/grass.dds", &texResource, &mGrassMapSRV));

ReleaseCOM(texResource); // view saves reference

HR(DirectX::CreateDDSTextureFromFile(md3dDevice,

    L"Textures/water2.dds", &texResource, &mWavesMapSRV));

    ReleaseCOM(texResource); // view saves reference

HR(DirectX::CreateDDSTextureFromFile(md3dDevice,

    L"Textures/WireFence.dds", &texResource, &mBoxMapSRV));

ReleaseCOM(texResource); // view saves reference

 At this point, the “Blend Demo” should build and run.