DirectX 11

Introduction

Origins

Prior to DirectX, game makers were struggling with hardware incompatibilities, making it almost impossible for everyone to enjoy the same games.

As the industry faced the need for standardization, Microsoft introduced the Windows Game SDK for Windows 95, which became DirectX 1.

Hardware manufacturers have to adhere to this API, and provide the driver layer for their devices accordingly.

What Is DirectX?

- Microsoft’s collection of application programming interfaces (APIs)

- Designed to give developers a low-level interface to the PC hardware running Windows-based operating systems

- Ensuring compatibility across different sets of PC hardware

DirectX 9

Support for the use of much longer shader programs than before with pixel and vertex shader version 2.0.

Fixed-function pipeline, which is essentially a set of rendering states and algorithms built into the API that allowed for the rendering of objects using common effects.

Render geometry, enable lighting by setting a few properties and rendering states, transform the geometry, and so forth by calling a few Direct3D function calls.

DirectX 10

The most apparent change in Direct3D 10 was removing the fixed function pipeline was removed in favor of programmable shaders within graphics hardware.

Shaders are code written specifically to customize how the graphics hardware processes geometry.

The fixed function pipeline was limited to whatever was built into the API, whereas shaders allow us to create any effect.

DirectX 11

- General-purpose computing on the GPU using the new API DirectCompute
- True multi threaded rendering support
- New hardware tessellation
- Shader Model 5.0 and object-oriented programming concepts for shaders
- BC6 (sometimes called BC6H) and BC7 for texture compression of HDR and LDR images, respectively
- Increased texture resolution sizes

COMPONENTS

Components of DirectX

Direct2D
DirectWrite
DXGI
Direct3D

XAudio2
XAct3
XInput
XNA Math
Windows Game Explorer

DirectCompute
DirectSetup
DirectInput

Direct2D

Direct2D is used for 2D graphics within Win32 applications. It is capable of rendering high-performance vector graphics.

DirectWrite

DirectWrite is used for fonts and text rendering within a Direct2D application.

DXGI

The DirectX Graphics Infrastructure, also known as DXGI, is used for the creation of Direct3D swap chains and the enumeration of device adapters.

Direct3D

Direct3D is used for all 3D graphics in DirectX. It is also the API that receives the most attention and updates.

XAudio2

XAudio2 is a lower-level audio processing API that is part of the XDK (Xbox Development Kit) and, now, the DirectX SDK. XAudio2 is the replacement for DirectSound.

XAC3

Is a higher-level audio processing API built on top of XAudio2. Allows developers to use the Cross-Platform Audio Creation Tool to author sounds in their application.

XInput

Input API for the XDK and the DirectX SDK and is used for processing of input from all Xbox 360 controllers. Essentially any controller you can use with your Xbox 360 can be used with the PC, and XInput is the API you use for working with these devices. These devices include not only Xbox gamepad controllers but also Rock Band and Guitar Hero instrument controllers, Big Button controllers, arcade sticks, and so much more. XInput is the replacement for DirectInput.

XNA Math

The new XNA Math is not an API but rather a math library that implements optimized math operations that are common to video games. XNA Math uses SIMD (Single Instruction Multiple Data) to perform multiple operations with a single instruction call. The XNA Math library is available to the Xbox 360 and to Windows PCs.

DirectCompute

Is a new API added to DirectX 11 that allows for general purpose multi threading computing using the GPU. The GPU has the ability to process many tasks in parallel, such as physics, video compression and decompression, audio processing, and much more.

DirectSetup

DirectSetup gives the functionality to install the latest version of DirectX on the user’s computer. It also has the ability to check the latest installed version of DirectX.

Windows Game Explorer

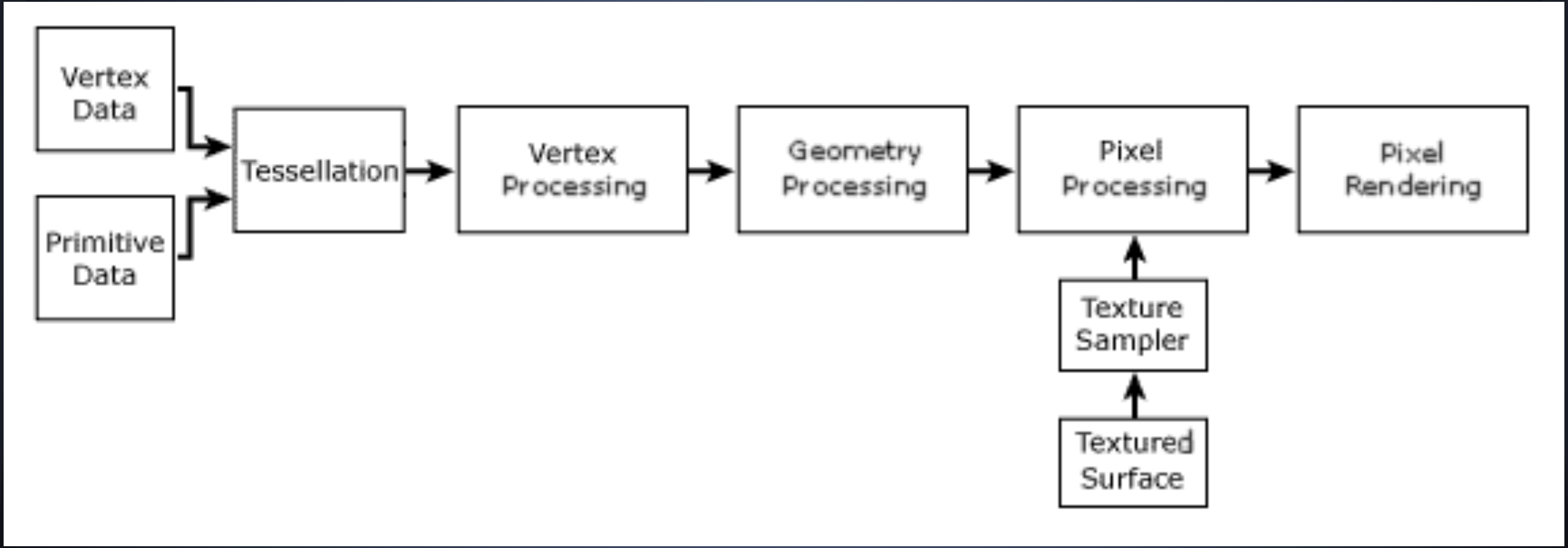
The Games Explorer is a feature of Windows Vista and Windows 7 that allows developers to present their games on those OS's. The Games Explorer handles things such as the game's display, title, rating, description, region-specific box art, content ratings (e.g., M for Mature, T for Teens, etc.), game statistics and notifications, parental controls, and more. The DirectX SDK provides plenty of information on how to use the Games Explorer for your own games and could be very useful when it comes time to ship a game.

DirectInput

DirectInput is an API for detecting input with keyboards, mice, and joysticks. Today XInput is used for all game controllers. According to the DirectX SDK, DirectInput will remain in its current form until new technologies replace it.

PIPELINE

DirectX 9 Fixed pipeline

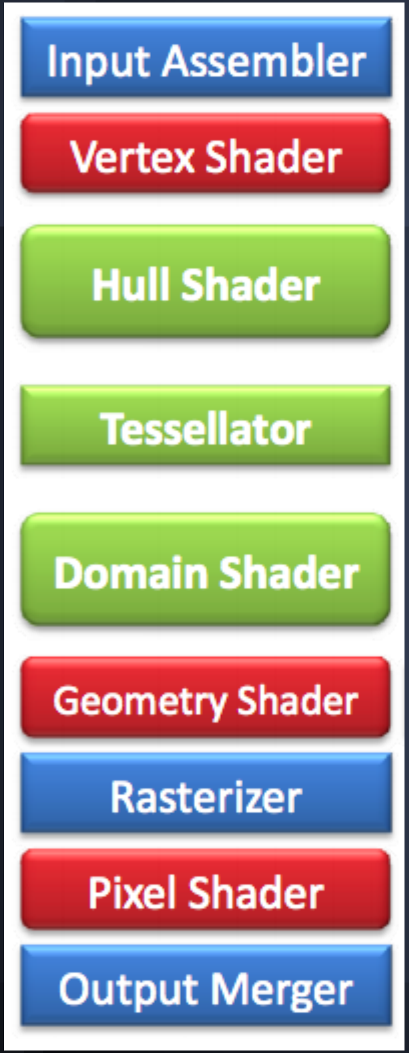


DirectX 9 Fixed pipeline components

Vertex Data	Untransformed model vertices are stored in vertex memory buffers.
Primitive Data	Geometric primitives, including points, lines, triangles, and polygons, are referenced in the vertex data with index buffers.
Tessellation	The tesselator unit converts higher-order primitives, displacement maps, and mesh patches to vertex locations and stores those locations in vertex buffers.
Vertex Processing	Direct3D transformations are applied to vertices stored in the vertex buffer.

DirectX 9 Fixed pipeline components

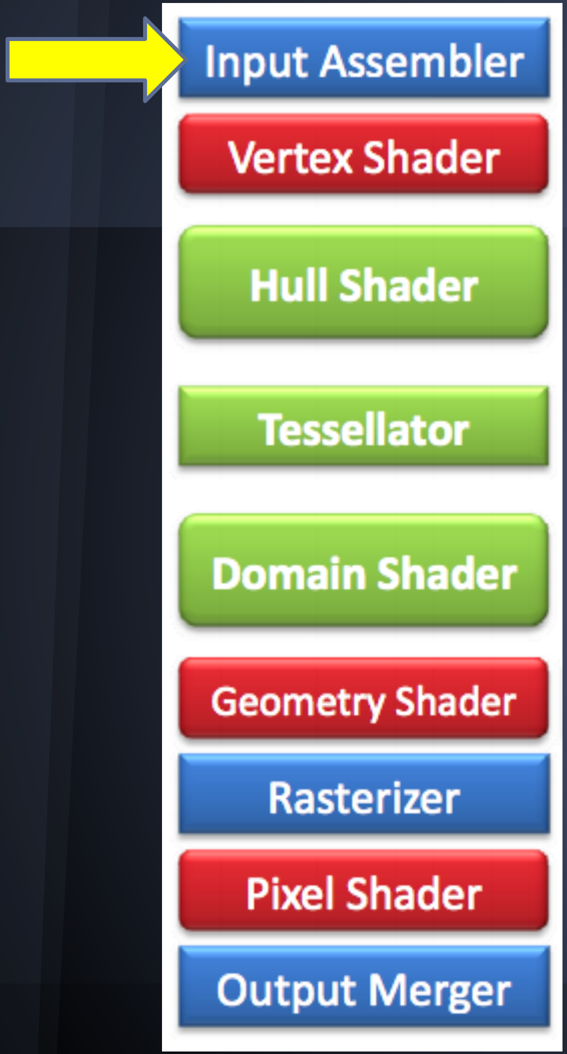
Geometry Processing	Clipping, back face culling, attribute evaluation, and rasterization are applied to the transformed vertices.
Textured Surface	Texture coordinates for Direct3D surfaces are supplied to Direct3D through the **IDirect3DTexture9** interface.
Texture Sampler	Texture level-of-detail filtering is applied to input texture values.
Pixel Processing	Pixel shader operations use geometry data to modify input vertex and texture data, yielding output pixel color values.
Pixel Rendering	Final rendering processes modify pixel color values with alpha, depth, or stencil testing, or by applying alpha blending or fog. All resulting pixel values are presented to the output display.



DirectX 11 Pipeline stages

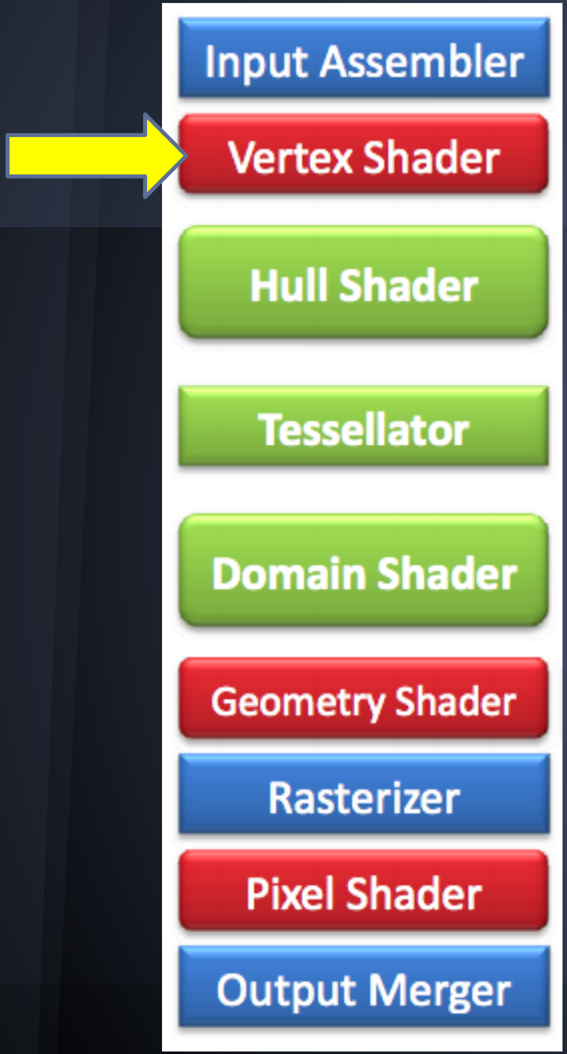
- On red there are the programmable shaders introduced on DirectX 10

- On green are the new pipeline stages introduced on DirectX 11



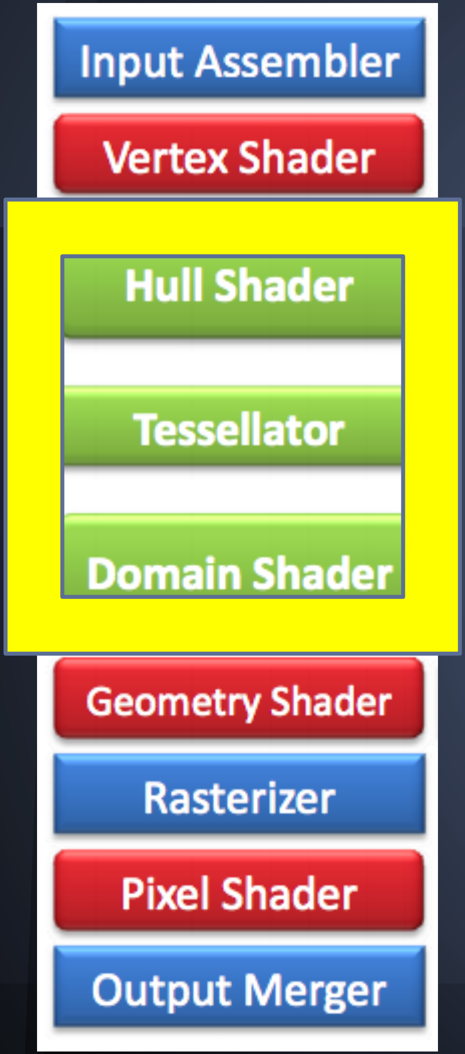
Input Assembler

Can be thought of as the building-block stage. In this stage we set the geometry we are going to render along with the necessary information Direct3D needs to perform that task.



Vertex Shader

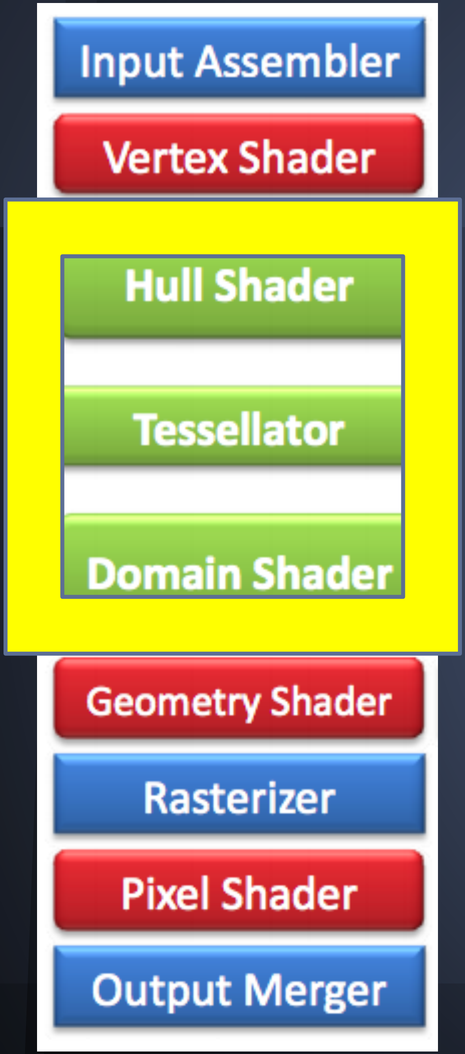
A vertex is a single point that makes up a shape, such as a triangle. In a vertex shader we can run code that operates on each vertex, much of which depends on the effect we are setting up for. A vertex shader always takes a single vertex as input and outputs a single vertex. This vertex data was supplied by the data set using the input assembler.

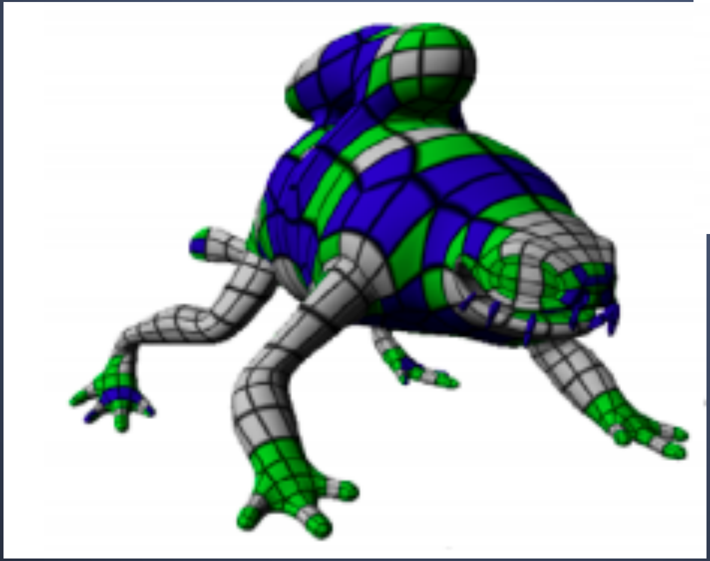


Tesselation

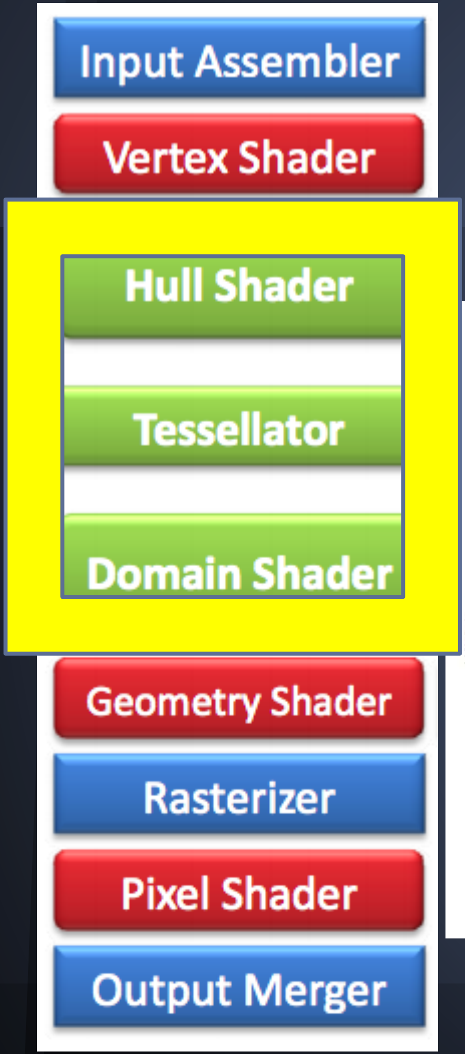
Tessellation is an advanced topic that uses two new shaders to Direct3D called the hull and domain shaders. Hardware tessellation, in a nutshell, is the process of taking input geometry and increasing or decreasing its level of detail. This allows for very high polygonal models to be rendered in real time with polygon counts in the hundreds of thousands or even the millions. By having the hardware create the detail of the geometry, the application only has to submit a small amount of data that defines the low-level model. This model can be in the form of patches, which is a 3D modeling. The hardware takes this low-level model and makes it higher level. Subdividing polygons means to take a polygon and divide it into smaller pieces.

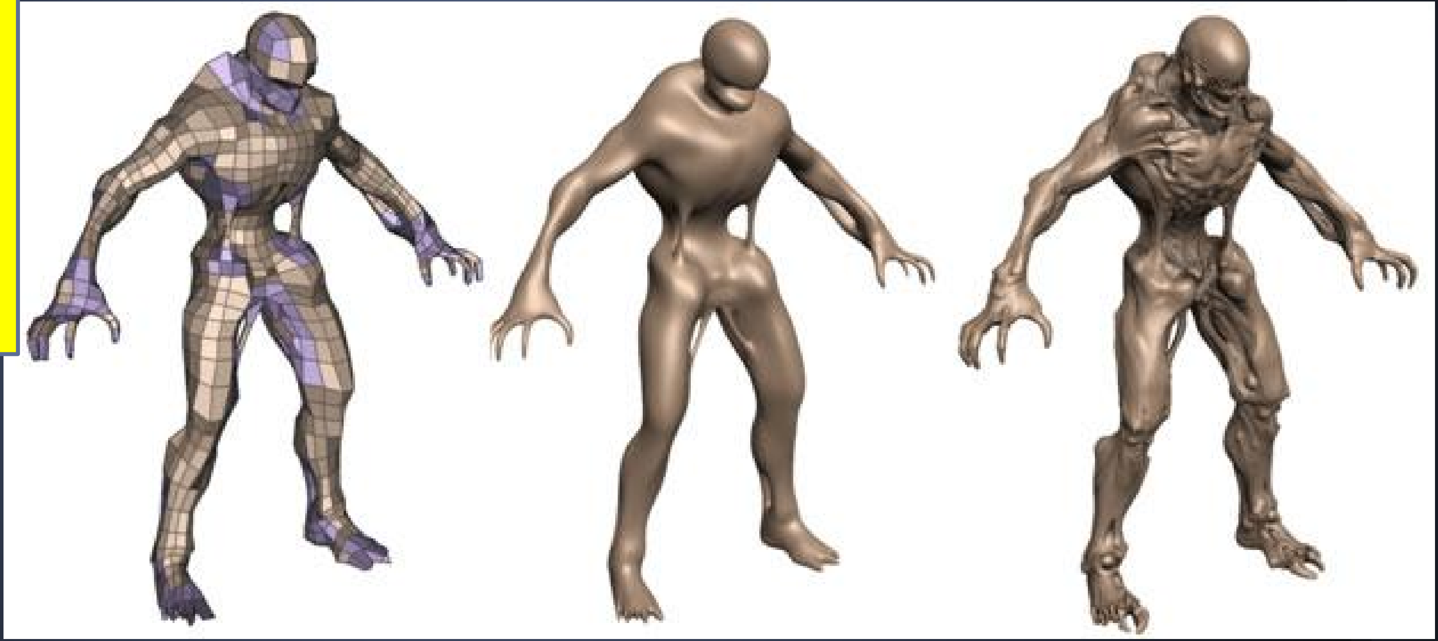
Tesselation



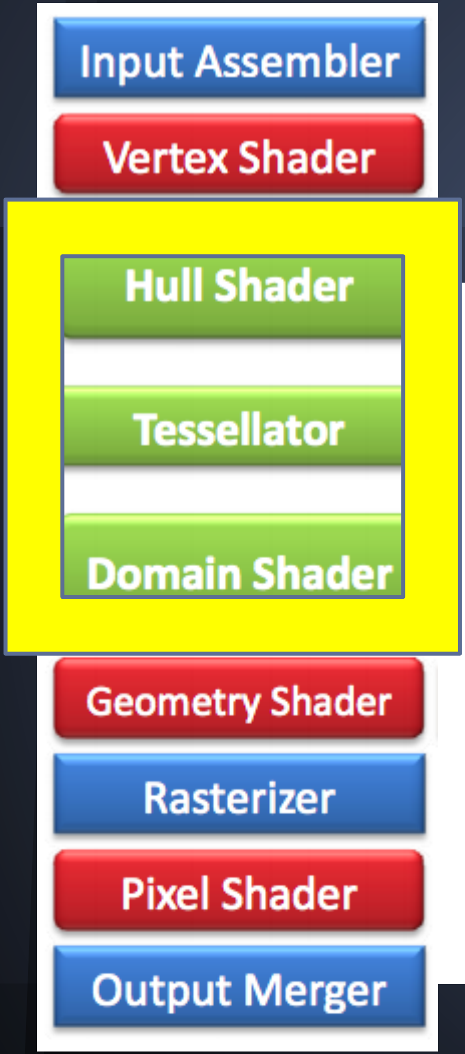


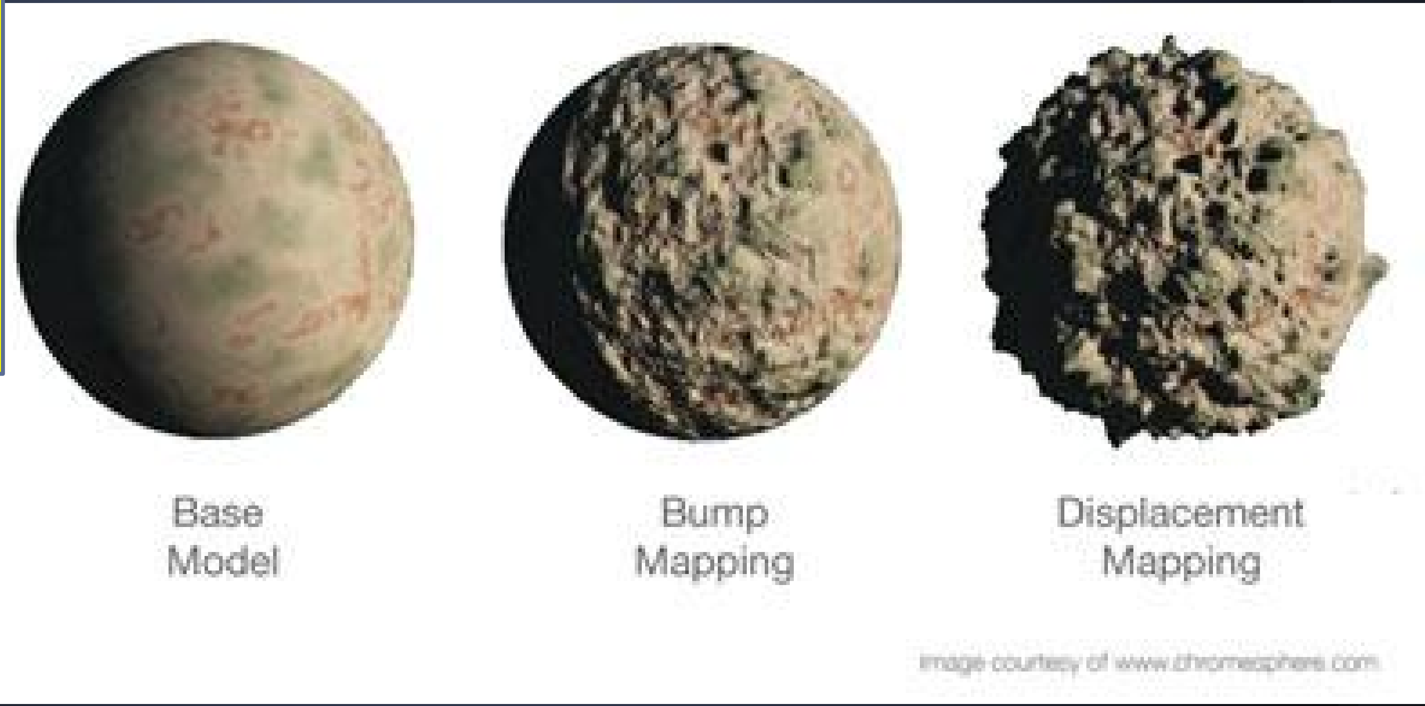
Tesselation

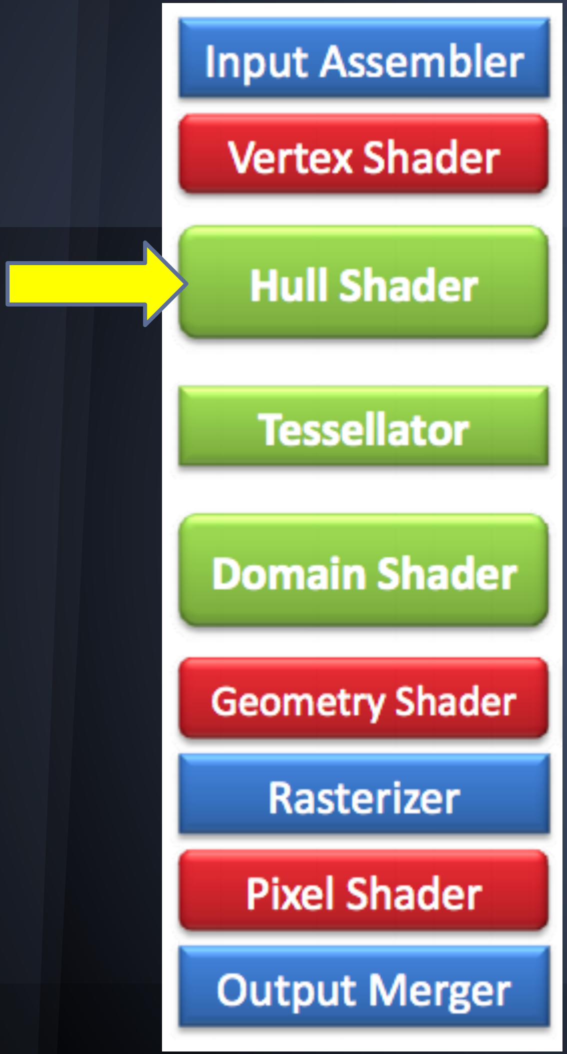




Tesselation

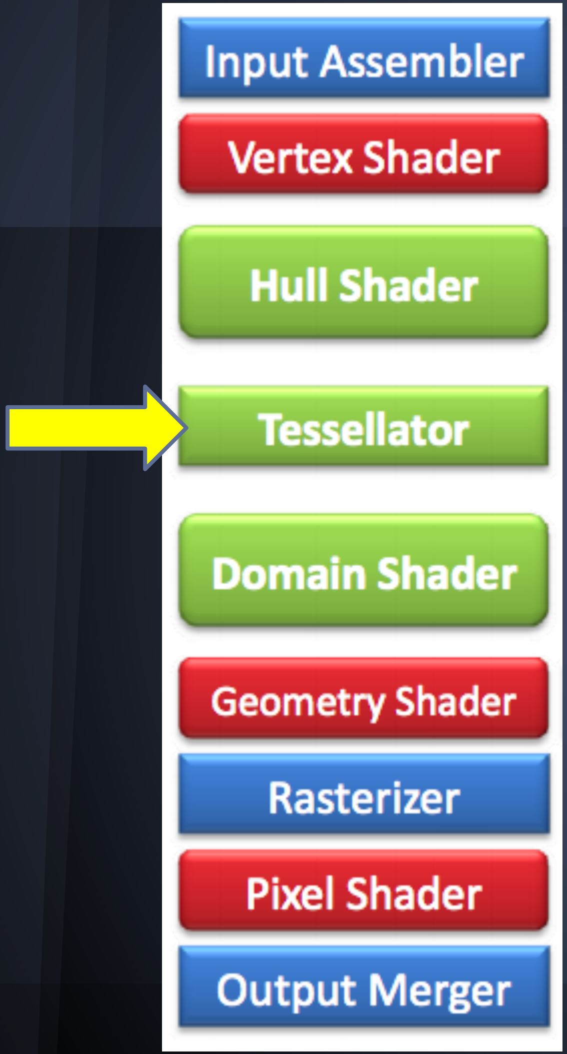






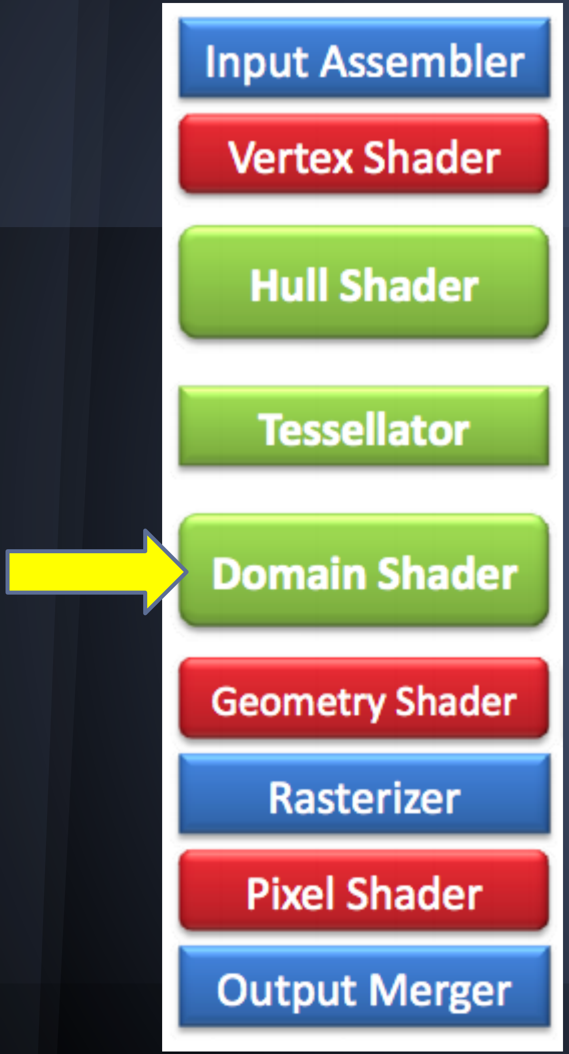
Hull Shader

Operates once per patch from the input assembler. Can transform input control points that make up a patch into output control points. The hull shader can perform other setup for the fixed-function tessellator stage. For example, can output tess factors, which are numbers that indicate how much to tessellate.



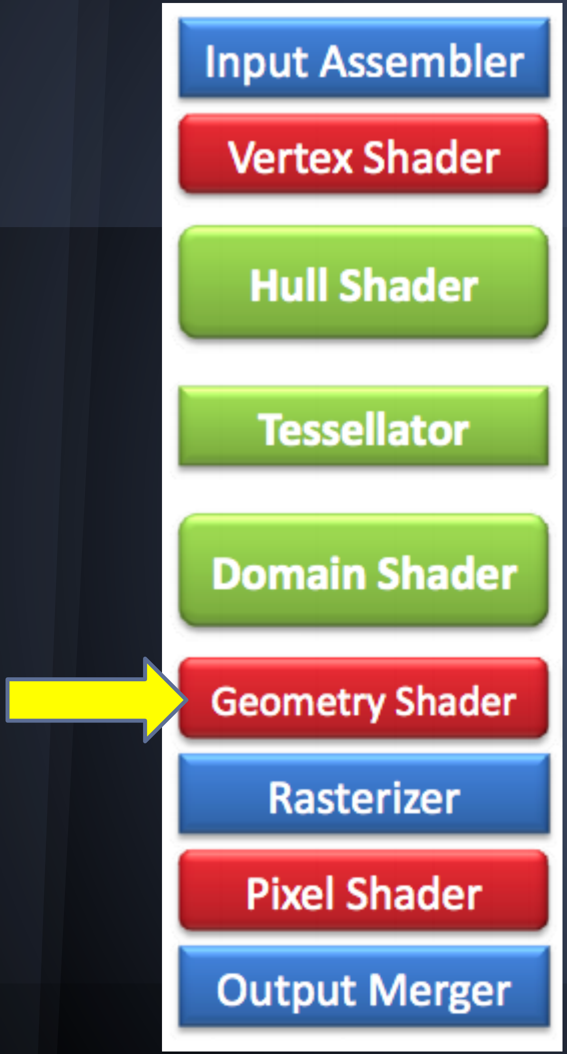
Tesselator

Is a fixed-function unit whose operation is defined by declarations in the hull shader. Operates once per patch that is output by the hull shader. The hull shader generates tess factors, which are numbers that notify the tessellator how much to tessellate (generate geometry and connectivity) over the domain of the patch.



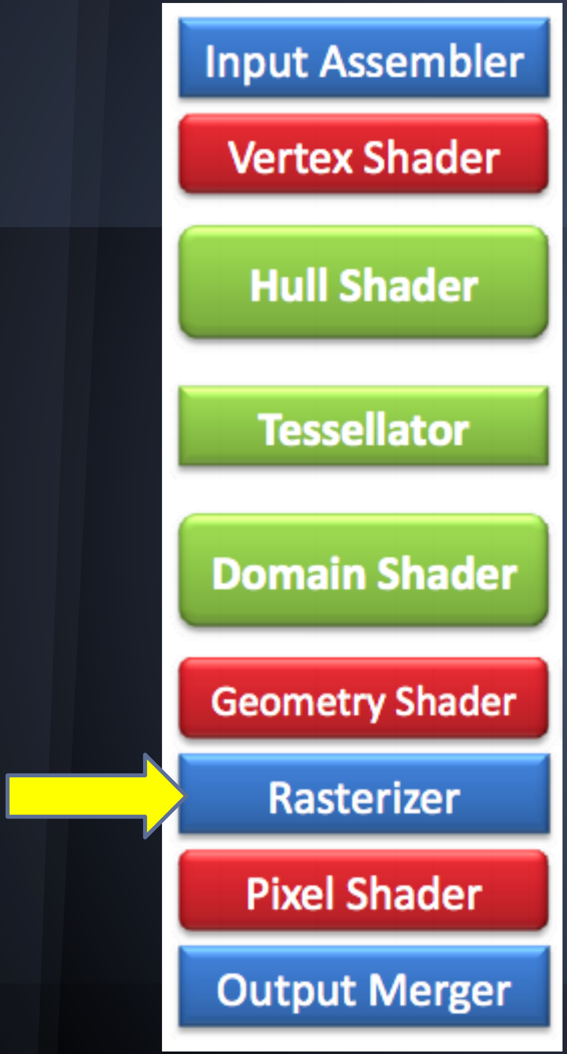
Domain shader

Is invoked once per vertex, which is generated by the tessellator. Each invocation is identified by its coordinate on a generic domain. The role is to turn that coordinate into something tangible (such as, a point in 3D space) for use down stream of the domain shader. Each domain shader invocation for a patch also accesses shared input of all the hull shader output (such as, output control points).



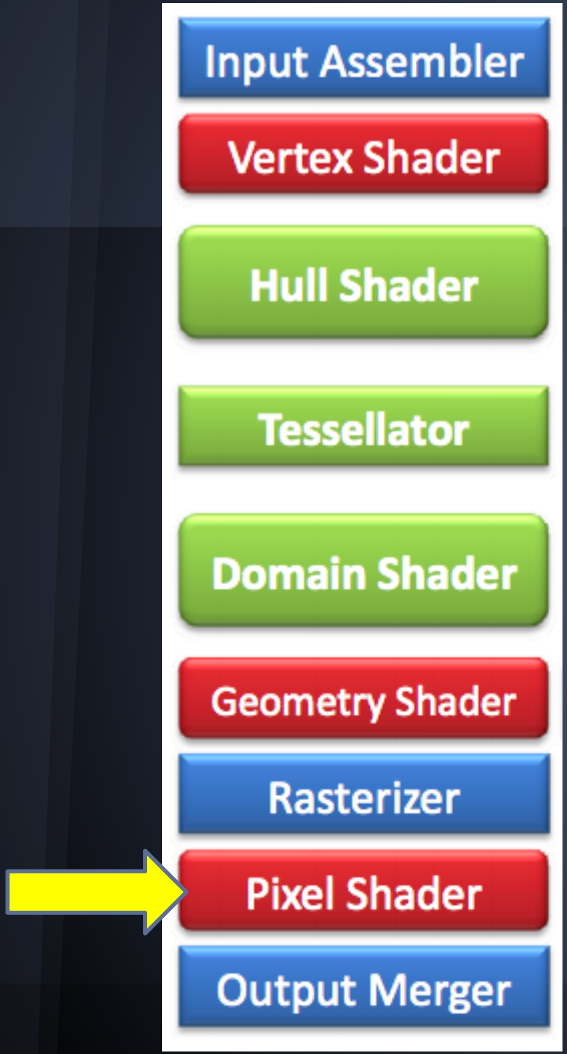
Geometry shader

Tesselation and Geometry shaders are optional. Geometry shader operate on entire shapes such as triangles, whereas the vertex shader operates on a single point of a shape. The geometry shader has the ability to essentially create or destroy geometry as needed, which depends largely on the effect you are trying to create.



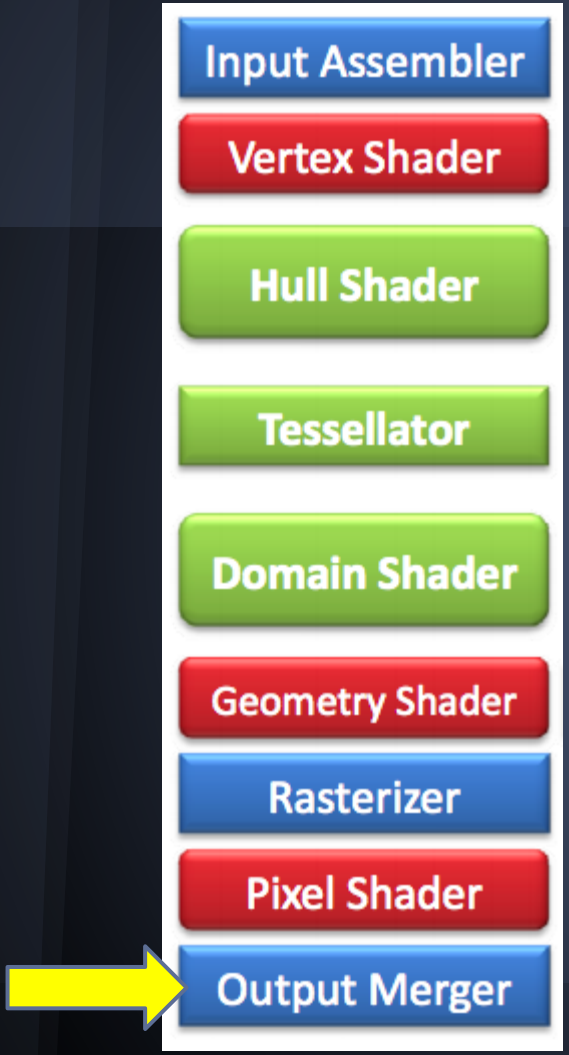
Rasterizer

Has the job of determining what pixels are visible through clipping and culling geometry, setting up the pixel shaders, and determining how the pixel shaders will be invoked.



Pixel Shader

This shader receives the geometric data from all previous stages and is used to shade the pixels (sometimes referred to as fragments) that comprise that shape. The output of the pixel shader is a single color value that will be used by the final stage to build the final image displayed to the screen. The input is technically interpolated data—that is, data that is generated between the points (vertices) of a shape.



Output Merger

Takes all of the output pieces from the other stages of the pipeline and builds up the final image to send to the screen.

TOOLS

DirectX Tools

Sample browser and documentation

Cross-Platform Audio Creation Tool

Game Definition File Editor

PIX

Caps viewer

Diagnostic Tools

Texture Tool

Error lookup

Control panel

Texture Converter

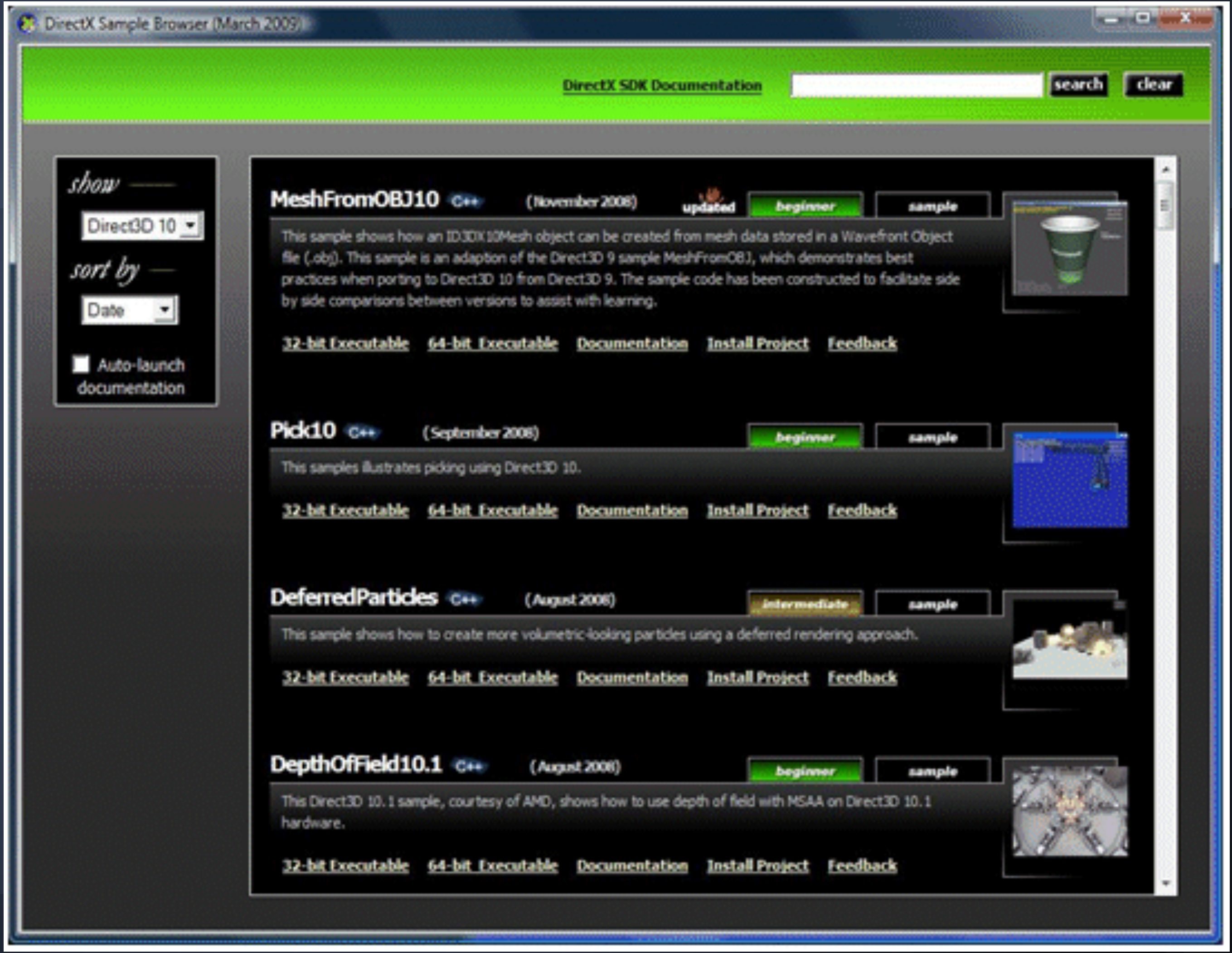
Sample Browser and documentation

The DirectX SDK Sample Browser is a tool that displays all of the example demos, technical papers, tutorials, articles, and tools that come with the DirectX SDK.

This tool is not going to be available anymore as a standalone app, because starting with Windows 8, DirectX SDK is included as a part of the Windows SDK.

However this tool is the best option for start learning DirectX, because has attached screenshots, executables and documentation besides the source code.

Sample Browser



DirectX Sample Browser (March 2009)

DirectX SDK Documentation

show: Direct3D 10

sort by: Date

[ ] Auto-launch documentation

MeshFromOBJ10 C++ (November 2008) updated beginner sample

This sample shows how an ID3DX10Mesh object can be created from mesh data stored in a Wavefront Object file (.obj). This sample is an adaption of the Direct3D 9 sample MeshFromOBJ, which demonstrates best practices when porting to Direct3D 10 from Direct3D 9. The sample code has been constructed to facilitate side by side comparisons between versions to assist with learning.

32-bit Executable 64-bit Executable Documentation Install Project Feedback

Pick10 C++ (September 2008) beginner sample

This samples illustrates picking using Direct3D 10.

32-bit Executable 64-bit Executable Documentation Install Project Feedback

DeferredParticles C++ (August 2008) intermediate sample

This sample shows how to create more volumetric-looking particles using a deferred rendering approach.

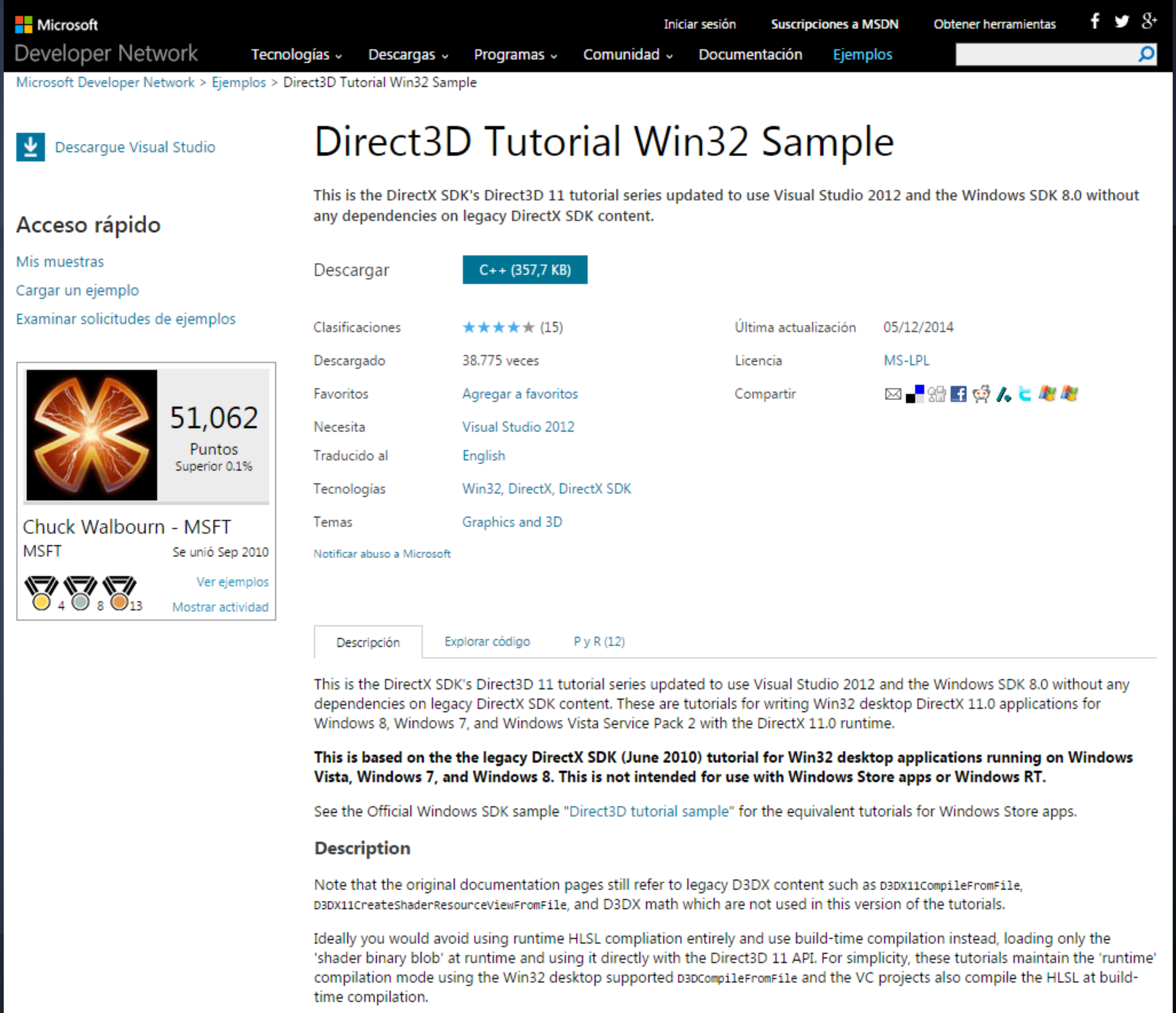
32-bit Executable 64-bit Executable Documentation Install Project Feedback

DepthOfField10.1 C++ (August 2008) beginner sample

This Direct3D 10.1 sample, courtesy of AMD, shows how to use depth of field with MSAA on Direct3D 10.1 hardware.

32-bit Executable 64-bit Executable Documentation Install Project Feedback

New Sample



Direct3D Tutorial Win32 Sample

This is the DirectX SDK's Direct3D 11 tutorial series updated to use Visual Studio 2012 and the Windows SDK 8.0 without any dependencies on legacy DirectX SDK content.

Descargar	C++ (357,7 KB)		
Clasificaciones	★★★★★ (15)	Última actualización	05/12/2014
Descargado	38.775 veces	Licencia	MS-LPL
Favoritos	Agregar a favoritos	Compartir	
Necesita	Visual Studio 2012		
Traducido al	English		
Tecnologías	Win32, DirectX, DirectX SDK		
Temas	Graphics and 3D		

Notificar abuso a Microsoft

Descripción | Explorar código | P y R (12)

This is the DirectX SDK's Direct3D 11 tutorial series updated to use Visual Studio 2012 and the Windows SDK 8.0 without any dependencies on legacy DirectX SDK content. These are tutorials for writing Win32 desktop DirectX 11.0 applications for Windows 8, Windows 7, and Windows Vista Service Pack 2 with the DirectX 11.0 runtime.

This is based on the the legacy DirectX SDK (June 2010) tutorial for Win32 desktop applications running on Windows Vista, Windows 7, and Windows 8. This is not intended for use with Windows Store apps or Windows RT.

See the Official Windows SDK sample "Direct3D tutorial sample" for the equivalent tutorials for Windows Store apps.

Description

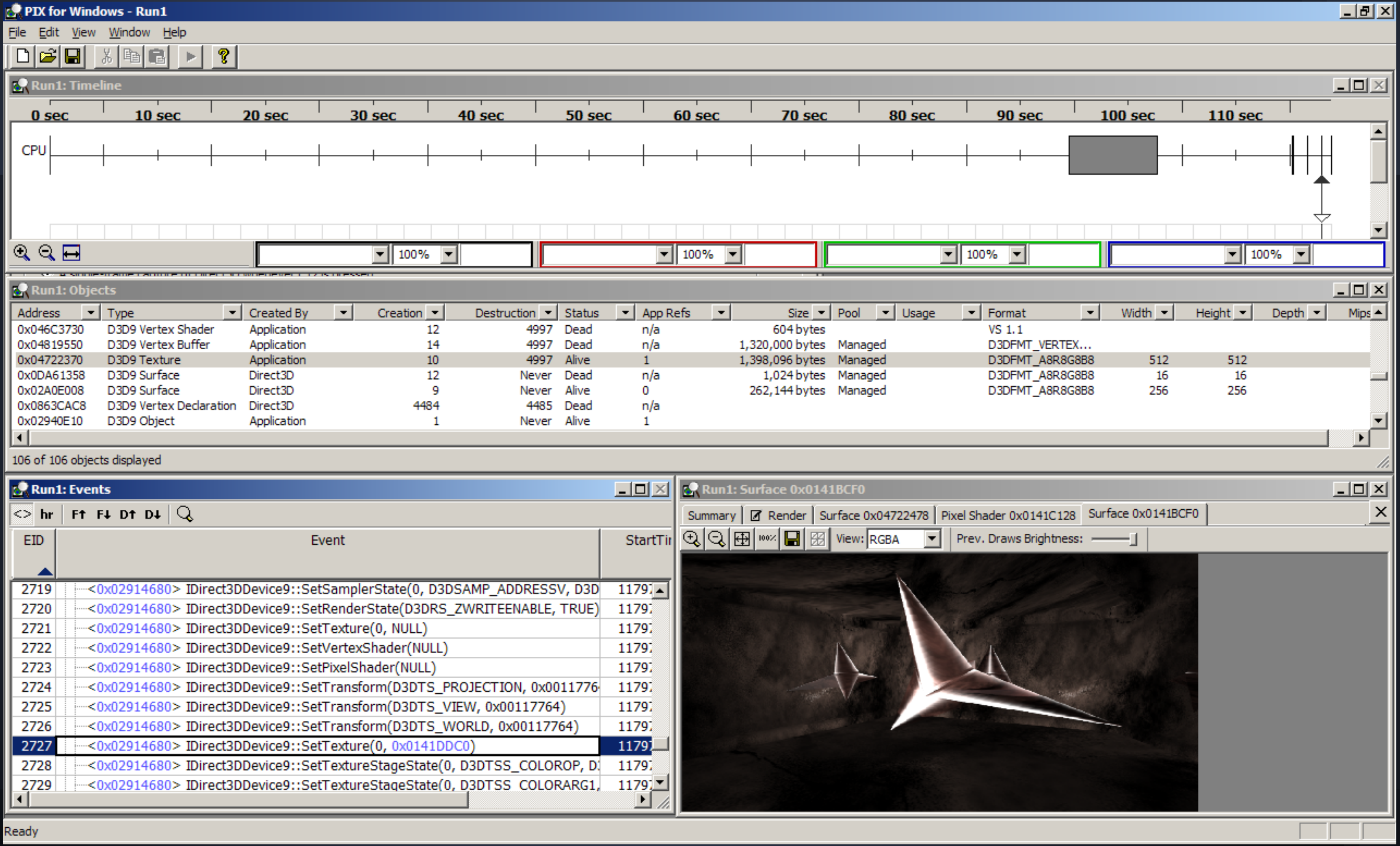
Note that the original documentation pages still refer to legacy D3DX content such as `D3DX11CompileFromFile`, `D3DX11CreateShaderResourceViewFromFile`, and D3DX math which are not used in this version of the tutorials.

Ideally you would avoid using runtime HLSL compliation entirely and use build-time compilation instead, loading only the 'shader binary blob' at runtime and using it directly with the Direct3D 11 API. For simplicity, these tutorials maintain the 'runtime' compilation mode using the Win32 desktop supported `D3DCompileFromFile` and the VC projects also compile the HLSL at build-time compilation.

PIX

PIX is a tool used for the debugging and analysis of Direct3D applications as they are executing. PIX can give valuable information such as API calls, timing statistics, and mesh information before and after transformation, to name a few. PIX can also be used for the debugging of shader code on the GPU, along with breakpoints and the ability to step through code.

PIX


PIX for Windows - Run1
File Edit View Window Help
Run1: Timeline
0 sec 10 sec 20 sec 30 sec 40 sec 50 sec 60 sec 70 sec 80 sec 90 sec 100 sec 110 sec
CPU
100% 100% 100% 100%
Run1: Objects

Address	Type	Created By	Creation	Destruction	Status	App Refs	Size	Pool	Usage	Format	Width	Height	Depth	Mips
0x046C3730	D3D9 Vertex Shader	Application	12	4997	Dead	n/a	604 bytes			VS 1.1				
0x04819550	D3D9 Vertex Buffer	Application	14	4997	Dead	n/a	1,320,000 bytes	Managed		D3DFMT_VERTEX...				
0x04722370	D3D9 Texture	Application	10	4997	Alive	1	1,398,096 bytes	Managed		D3DFMT_A8R8G8B8	512	512		
0x0DA61358	D3D9 Surface	Direct3D	12	Never	Dead	n/a	1,024 bytes	Managed		D3DFMT_A8R8G8B8	16	16		
0x02A0E008	D3D9 Surface	Direct3D	9	Never	Alive	0	262,144 bytes	Managed		D3DFMT_A8R8G8B8	256	256		
0x0863CAC8	D3D9 Vertex Declaration	Direct3D	4484	4485	Dead	n/a								
0x02940E10	D3D9 Object	Application	1	Never	Alive	1								

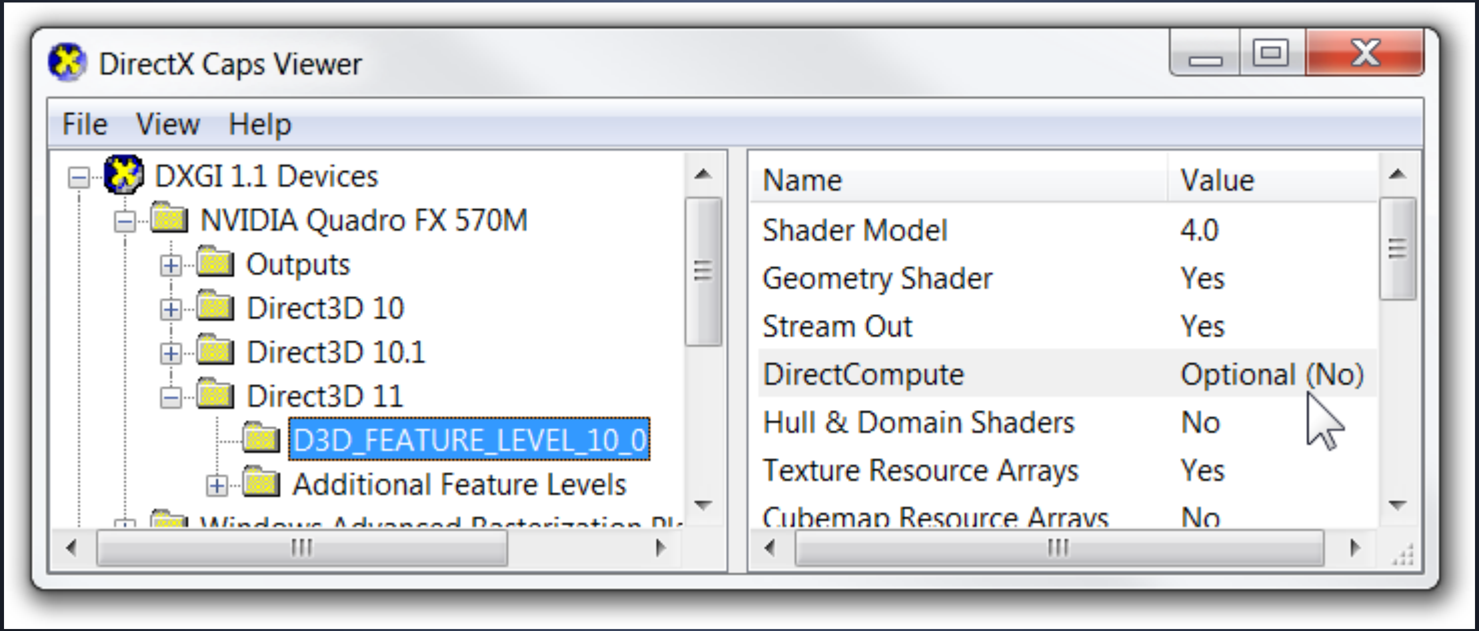
106 of 106 objects displayed

EID	Event	StartTi
2719	<0x02914680> IDirect3DDevice9::SetSamplerState(0, D3DSAMP_ADDRESSV, D3D	1179
2720	<0x02914680> IDirect3DDevice9::SetRenderState(D3DRS_ZWRITEENABLE, TRUE)	1179
2721	<0x02914680> IDirect3DDevice9::SetTexture(0, NULL)	1179
2722	<0x02914680> IDirect3DDevice9::SetVertexShader(NULL)	1179
2723	<0x02914680> IDirect3DDevice9::SetPixelShader(NULL)	1179
2724	<0x02914680> IDirect3DDevice9::SetTransform(D3DTS_PROJECTION, 0x0011776	1179
2725	<0x02914680> IDirect3DDevice9::SetTransform(D3DTS_VIEW, 0x00117764)	1179
2726	<0x02914680> IDirect3DDevice9::SetTransform(D3DTS_WORLD, 0x00117764)	1179
2727	<0x02914680> IDirect3DDevice9::SetTexture(0, 0x0141DDC0)	1179
2728	<0x02914680> IDirect3DDevice9::SetTextureStageState(0, D3DTSS_COLOROP, D	1179
2729	<0x02914680> IDirect3DDevice9::SetTextureStageState(0, D3DTSS_COLORARG1,	1179

Caps Viewer

Shows information about the hardware’s capabilities by way of detailing its support for Direct3D, DirectDraw, DirectSound, and DirectInput. Every piece of information about what the hardware supports and its version are displayed by this tool.

Caps Viewer

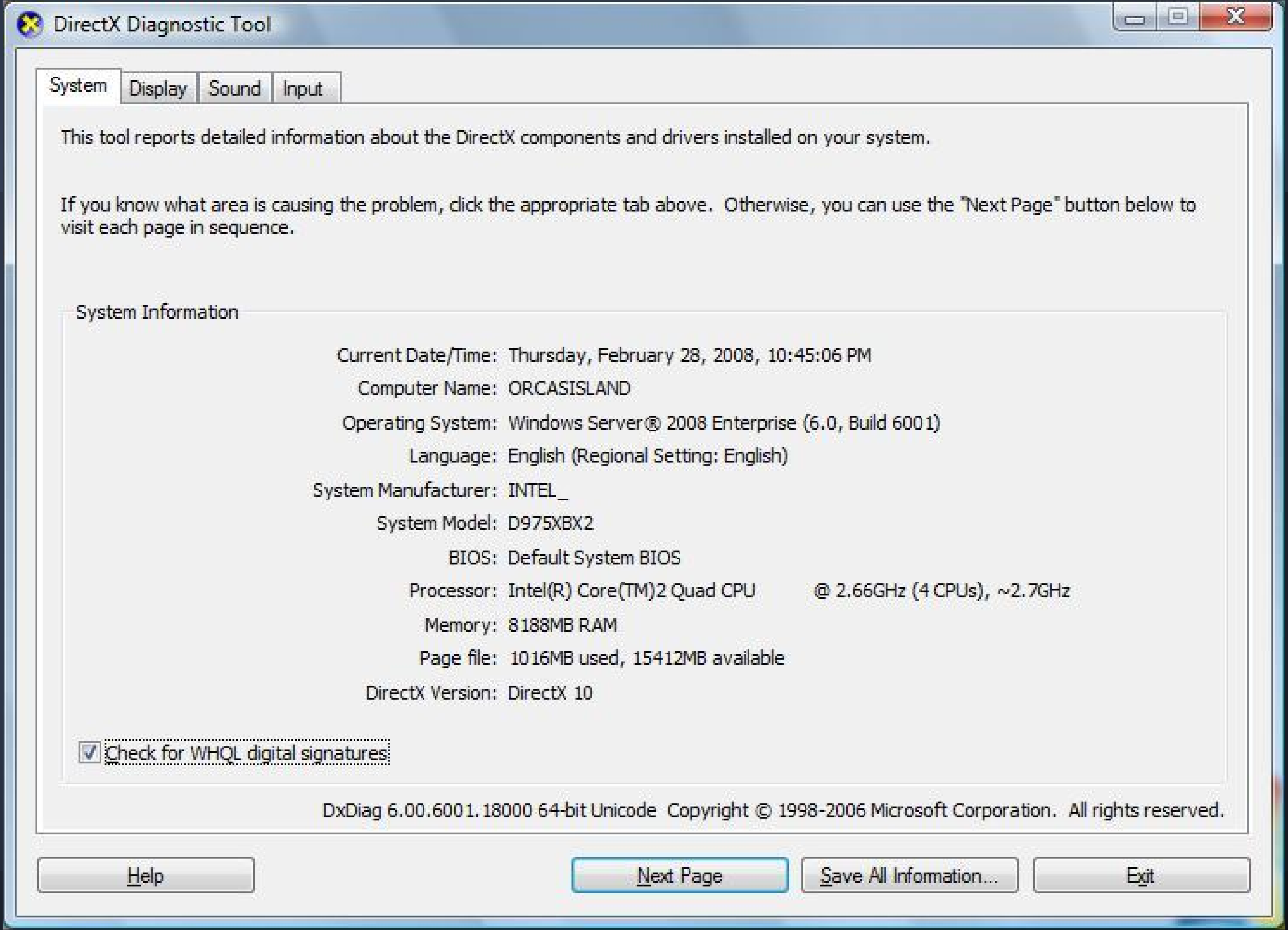


Name	Value
Shader Model	4.0
Geometry Shader	Yes
Stream Out	Yes
DirectCompute	Optional (No)
Hull & Domain Shaders	No
Texture Resource Arrays	Yes
Cubemap Resource Arrays	No

Diagnostic tool

Is used to test various components of DirectX to see if they are working properly. Reports that can be saved to a file and/or sent to Microsoft.

Diagnostic Tool



This tool reports detailed information about the DirectX components and drivers installed on your system.

If you know what area is causing the problem, click the appropriate tab above. Otherwise, you can use the "Next Page" button below to visit each page in sequence.

System Information

Current Date/Time:	Thursday, February 28, 2008, 10:45:06 PM
Computer Name:	ORCASISLAND
Operating System:	Windows Server® 2008 Enterprise (6.0, Build 6001)
Language:	English (Regional Setting: English)
System Manufacturer:	INTEL_
System Model:	D975XBX2
BIOS:	Default System BIOS
Processor:	Intel(R) Core(TM)2 Quad CPU @ 2.66GHz (4 CPUs), ~2.7GHz
Memory:	8188MB RAM
Page file:	1016MB used, 15412MB available
DirectX Version:	DirectX 10

[✓] Check for WHQL digital signatures

DxDiag 6.00.6001.18000 64-bit Unicode Copyright © 1998-2006 Microsoft Corporation. All rights reserved.

Texture tool

The DirectX Texture Tool is used to convert images to Direct3D texture formats that use DXTn compression. This tool is deprecated since it only supports texture formats supported by DirectX 9, not DirectX 10 or 11.

Texture Converter

Used for convert an image from one format to another. The file formats that can be seen include: BMP, JPEG, DDS, TGA, PNG, DIB, HDR, PMF.

The Texture Converter works by right-clicking on an image (or multiple images) in the Windows Explorer and selecting Convert to File Format from the drop- down list.

It’s possible to use the texture conversion command-line tool called TexConv.exe or TexConvEx.exe for Direct3D 10 and 11 textures.

Error lookup tool

Displays a description of any error code received while running a DirectX application. Error code can be searched in this application, for a detailed description. Not all errors are clear, and this tool can be useful. Can be found in the Utilities folder of your DirectX SDK installation.

Control Panel

Located in the Utilities folder of the DirectX SDK, is used to examine and modify the properties of various DirectX components. With the Control Panel tool you can:

- Enable the debug layer of Direct3D 10/11
- Change the debug output level and settings for use during development n View driver information
- View hardware support information
- View version information for the various components

Cross-Platform Audio Creation Tool

GUI tool (also available with the DirectX SDK is a command-line alternative) for creating audio files used by XACT3, which is DirectX’s high-level audio API/component.

Game Definition File Editor

Used to create localized game definition files for Windows. This information is displayed on the Games Explorer, and include the game’s release date, its Games Explorer icon and box art, its rating (e.g., Teen, Mature, etc.), its name and description, and a host of other properties.

Test 1: DirectX 11 introduction

goo.gl/forms/CalEq14BVM