Alan Shreve
EECS 487
Special Assignment Paper
3/26/2007

Direct 3D: Origins and Evolution

Direct3D, a graphics API part of the larger DirectX package offered by Microsoft, remains the only other major alternative to OpenGL. Used widely among computer game developers, Direct3D has made inroads into the 3D graphics world despite woefully buggy early versions and outspoken criticism from multiple sources in the graphics community. The DirectX software continues to grow more usable, efficient, and feature-rich with each version as Microsoft strives to preserve one of the important consumer incentives of its Windows platform. The newest version of DirectX was released recently as part of Microsoft's latest operating system, Windows Vista, where it powers the oft-criticized new Aero GUI. But the earliest versions of DirectX were much smaller and less integral parts of the Windows platform, aimed at game developers who had previously written games in DOS.

Despite the release of Windows 95, the majority of computer games were still being released for DOS, due to the slow performance of the Windows API for rendering graphics. In an effort to retain developers' interests in writing games for Microsoft platforms, an API for communicating directly to the hardware device drivers, bypassing all of Windows' layers of indirection was devised. Microsoft purchased the majority of the original code for this project from a small company called RenderMorphics. This code formed the base for the initial release of DirectX.

The game developers community widely rejected the first, and subsequent versions of DirectX. The early versions were mired in slow, buggy implementations which used an awkward and complex

interface. Microsoft went through many iterations, releasing versions two, three, and skipping four entirely before arriving at an API that was acceptable to use in DirectX 5.

Before the release of DirectX 5, rendering through DirectX was accomplished through a system of “execute buffers.” An execute buffer is a packet of information containing a list of all vertices which would be rendered, followed by a series of instructions which described how the vertices were to be rendered. DirectX 5 introduced an interface called DrawPrimitive, which was much less complex and easier to use, yet still had comparative performance. For the first time, DirectX became a viable platform for developing Windows games.

A more usable version of the DirectX API and the increasing sales of the computer game market set the stage for a battle of 3D APIs that would embroil the computer graphics community. Choice of which graphics API (Direct3D and OpenGL) to use when developing a new game quickly became an important choice. Tensions mounted over allegations of relative performance of the APIs, and continued in flurries of posts to mailing lists and newsgroups. Some of the most important industry figures came out attacking Microsoft and the Direct3D API.

John Carmack, lead software designer at id Software, in December of 1996, posted a searing criticism of the Direct3D API, saying, “Direct-3D IM is a horribly broken API. It inflicts great pain and suffering on the programmers using it, without returning any significant advantages. I don't think there is ANY market segment that D3D is apropriate for, OpenGL seems to work just fine for everything from quake to softimage. There is no good technical reason for the existance of D3D.” He urged Microsoft to reconsider their direction and to instead focus on implementing a good, working version of the OpenGL API. Following Carmack, multiple letters were sent by many of the important

programmers in the computer graphics community to Microsoft encouraging them to support only a single API: OpenGL. Microsoft responded with its own form of evangelism, including multiple demonstrations in which they claimed superior performance to OpenGL. These claims were often found to be based in misleading methods or faulty bases of comparison. A good deal of argument over speed was centered around the performance of software renderers, which have since been made relatively obsolete with the advent of cheap, and powerful GPUs.

In spite of the game development community's outcry, and Direct3D's rocky beginnings, DirectX has become the more popular API for building computer games. DirectX won over game developers for multiple reasons which usually boiled down to simple pragmatism. The ability of a complete suite of tools to use for game development outside of Direct3D, for controlling audio, input devices, and other aspects made the API more appealing. The lack of a large market for games on Apple computers made forfeiting cross-platform possibilities a minor sacrifice. Furthermore, DirectX supported the ability to query the hardware for its abilities, whereas OpenGL would simply revert to software rendering, which for any graphics intensive computer game is not useful.

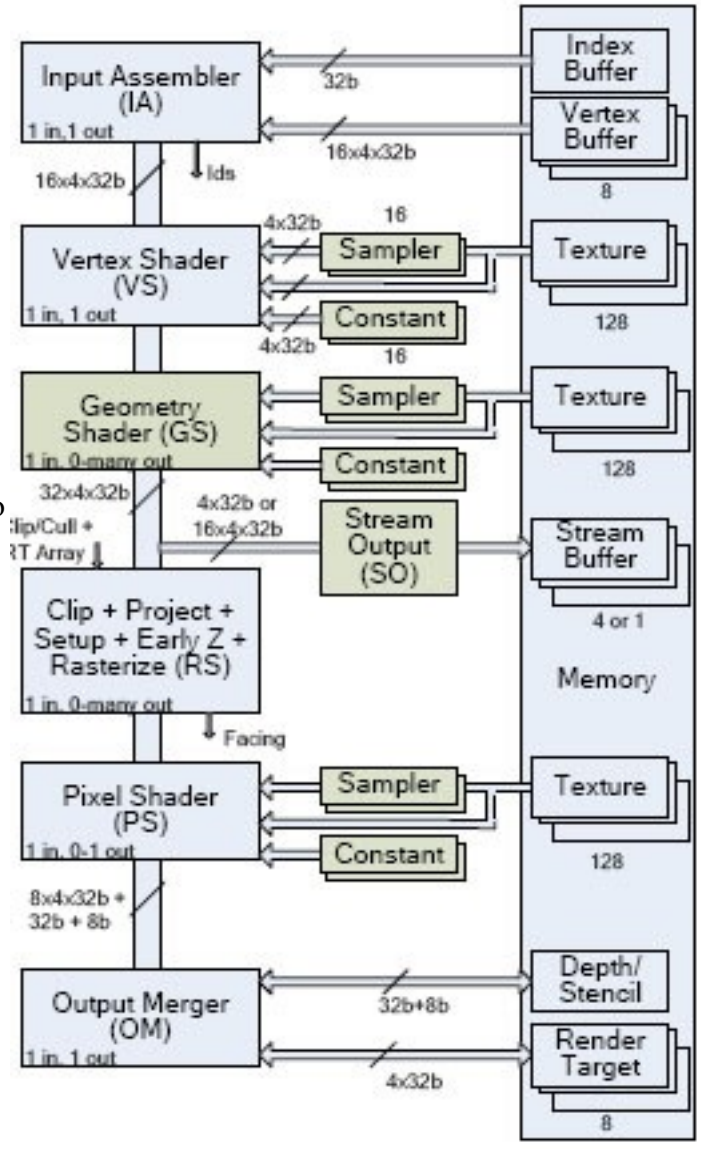
Direct3D has continued evolving through its versions. DirectX is now built around the COM (Component Object Model) framework, and thus can be used in any language which is COM aware. (C++, C#, Visual Basic) It is even possible to use DirectX in C, though often painful. Further, DirectX has made strides alongside OpenGL to provide an interface for custom shaders to be used in parts of the graphics pipeline. And although Microsoft has attempted to avoid major changes in the framework, Direct3D's powerful 2D API known as DirectDraw, was deprecated with the release of DirectX 8. Moving forward, DirectX 10 has just been released, which includes more improvements to the DirectX

suite.

Direct3D 10 is the result of three years of Microsoft development alongside input from application writers and hardware vendors. Direct3D 10 makes a number of important changes from its predecessors including a modified graphics pipeline in the hopes that it will afford applications programmers greater performance and flexibility.

The Direct3D pipeline includes an entire new programmable stage of the pipeline termed the geometry shader (GS). A GS program takes as input one primitive and allows the output of between 0 and 1024 primitives of any type. It is executed after the vertex shader, and before rasterization. Although the input assembler (IA) and output merger (OM) stages were considered for programmability, Direct3D designers ultimately reject the increased versatility citing performance concerns, thus leaving those stages as fixed functionality.

Programmable pipeline stages have also been improved to all use the same virtual machine for calculations, which designers call a "common core." The design of this common core includes input and output registers, temporary registers and resource binding points. The core is controlled by an assembly-



The Direct3D 10 Pipeline

like language of 32 bit instructions. However, it is not intended for application programmers to write shaders in this assembly language. Rather, Microsoft has introduced a new version of its shading language HLSL (High Level Shading Language) which it hopes will be used by application programmers to write custom shaders without the knowledge of the underlying guts of the implementation.

The latest advances in Direct3D have made it a 3D API which is now of similar import as Open GL. Much ill will over Microsoft's previous practices and closed-source, vendor-lock in strategies remain, so looking to the future it will be interest to see in which direction Microsoft moves next and how the graphics community responds.

References:

http://download.microsoft.com/download/f/2/d/f2d5ee2c-b7ba-4cd0-9686-b6508b5479a1/direct3d10_web.pdf

http://www.gamedev.net/reference/articles/article1775.asp

http://www.azillionmonkeys.com/windoze/OpenGLvsDirect3D.html