Tutorial: Migrating Your Apps to DirectX* 12 – Part 1

Introduction

With the release of Microsoft's* new operating system, Windows 10*, the core graphics technology is upgraded to the latest DirectX 12*. To help programmers who have experience in DirectX get familiar with DirectX 12 and start as soon as possible to develop games, Microsoft, Intel, and Suzhou Snail company absorbed the developing experience of PC DirectX 12 in "King of Wushu" by Suzhou Snail and made a cooperative effort to write this white paper. We hope to bring cutting-edge first-hand materials to all developers who are enthusiastic about DirectX technology. The technical adviser of this white paper is Jianye Lu, a senior graphics expert and project manager of DirectX 12 from Microsoft. Technical experts from all three companies collaborated on every chapter.

Chapter 1: Yingguang Mei (Microsoft)

Chapter 2: Yingguang Mei (Microsoft)

Chapter 3: Wenwei Lv (Snail)

Chapter 4: Sheng Guo (Intel) and Wenwei Lv (Snail)

Chapter 5: Kai Wang (Intel) and Wenwei Lv (Snail)

Currently, this is the first version of the white paper. We are continuelly adding and updating this tutorial, and would particularly like to get your suggestions and opinions. Here is our contact information:

Yingguang Mei：yimei@microsoft.com

Sheng Guo：sheng.guo@intel.com

Wenwei Lv：lvww@snailgame.net

Contents

Chapter 1 Overview of DirectX 12

DirectX is a programming interface set that Microsoft uses to control the underlying hardware and handle multi-media tasks (especially games and video) on Windows operating system platform. DirectX API is divided into multiple components. APIs within different components are independent and updated separately, which ensure that the games only need to use necessary functions. Each component has the capabilities to access different hardware which include graphics card, sound card, GPU and other input devices that have standard interfaces like joystick and mouse.

1.1 Overview of DirectX 12

DirectX 12 mainly includes the following components:

Table 1.1: DirectX 12components

Direct2D	It is used for 2D graphics rendering. It is a high-performance vector rending library.
DirectWrite	It utilizes the Direct2D application to render fonts and texts
Direct3D	It is used to build all of the 3D graphics in DirectX. It is the API that receives the most attention and the most frequently updated.
XAudio2	It is a low-level audio processing API.
XACT3	It is an advanced audio processing API built on XAudio2.
XInput	It is used to process all inputs from Xbox game console, etc.
DirectCompute	It is a new API set newly added to DirectX 11. GPU is allowed to execute the general-purpose multi-threaded computing.

1.1.1 Development environment and hardware configuration of DirectX 12

Table 1.2: Development environment and hardware configuration of DirectX 12

Development environment

- ▼ Software installation
  - Windows 10
  - Visual Studio 2013 + Update4 / Visual Studio 2015 (has better DX12 debug tool)
  - DirectX 12 SDK

- ▼ Installation of drivers
  - AMD Driver
  - Intel Driver
  - NVIDIA Desktop / Mobile Driver

Hardware configuration

- CPU: Intel Core i5-4670K
- Memory: 16GB+ RAM
- Graphics card (support Feature Level 11.1 and above):
  - NVIDIA: GTX 970/980 (4GB+ RAM) / GTX TITAN (4GB+ RAM)
  - AMD: R9 290 (4GB+ RAM)
  - INTEL: graphics card is built into CPU.HasWell and above.
- Hard drive: 1TB

1.1.2 Direct3D 12

Direct3D is the most important part of DirectX components. With the release of DirectX 12, Direct3D was also updated to the latest generation --Direct3D 12. Compared to the previous version, Direct3D 12 is much faster, more efficient, and be able to support richer scenes, more objects, more cool features to take full advantage of the mainstream GPUs.

More efficient performance

Direct3D 12 provides a lower than usual hardware abstraction layer so that games and applications are able to significantly improve CPU utilization and multi-threaded load balancing. Direct3D 12 allows games and applications to manage memory to a great extent by themselves. In addition, using the new features of Direct3D 12 greatly reduced GPU overhead and improved the presentation effects of games and applications. These new features include: command queues and lists, resource descriptor table, pipeline state object and so on. Moreover, Direct3D 12 also supports many new pipeline rendering technologies, such as conservative rasterization algorithms, volume-tiled resources, and raster-order views.

More powerful hosts

Regarding the rendering function, Direct3D 12 is a super set of Direct3D 11. Direct3D 12 retains the rendering function of Direct3D 11, meanwhile achieves better performance on the multi-core CPU and GPU of modern graphics cards, which greatly improves the efficiency of the host platform API. Based on Direct3D 12, modern hosts ensure the full use of all CPU and GPU core resources so as to exploit their most powerful rendering potential.

Better tools

Direct3D 12 provides a better PIX tool for debugging and analysis of D3D application runtime. In addition to tracking a lot of useful information (such as API calls and time statistics), debugging shader code on GPU, setting up break points and debugging in the code, testing the performance of Xbox applications was also greatly improved compared to what it was before. The latest PIX tool provides complete graphical debugging environment for DirextX applications.

1.2 Core concept of Direct3D 12

1.2.1 Work Submission

The execution of the command list after its submission is asynchronous to the CPU. That is, after the program submits command list to command queue, the CPU will directly go back to the program instead of waiting for the command to be executed, which ensures that the CPU can submit a rendering command list at any time. Direct3D 12 provides synchronous primitives that communicate between the CPU and GPU to know the status of rendering tasks executed by the GPU.

This type of command list is very efficient. API commands are converted directly into the original GPU commands, which can minimize the burden of the display driver. This is a high-performance approach of presenting rendering. To achieve such performance, when building the rendering command list in the final release version of an application, the core program of display driver should perform only a minimal amount of error checking work. During development, you can use the debugging tools to ensure rendering is correct. In order to facilitate input validation, analysis and debug, you can also use a debug layer that use Direct3D 12 API to conduct a comprehensive status examination to know the errors and warnings (such as "invalid rendering operation " error and "undefined rendering operation” warning, etc.).

1.2.2 Memory Management

The data processed by Direct3D 12 API is stored in a memory object called resource heap.

A Resource heap may be present in the local graphic memory or system memory. It has some characteristics like having a certain amount of cache and being accessible by CPU or GPU. An application can control the properties and the allocation of resource heap memory through the use of API parameters created by a variety of heaps.

Within a resource stack, a program can freely allocate resources, such as texture and buffer. This is a relatively lightweight operation. Creating resources in an existing resource heap will not cause actual memory allocation operation. In order to let GPU access these resources correctly, applications need to create resource descriptor views, such as: shader resource views or unordered access views.

To read or write resource heap memory of CPU, an application must map a heap-based resource for CPU to access.

In order to ensure continuous data allocation of GPU memory and correct operation of the pipeline operation in Diect3D 12, ring buffer or similar technologies are generally adopted. API provides synchronization primitives to ensure this type of pipeline operation.

1.3 Important changes from Direct3D 11 to Direct3D12

Please check the following site on MSDN for getting the important difference between Direct3D 12 and Direct3D 11
https://msdn.microsoft.com/en-us/library/windows/desktop/dn899194(v=vs.85).aspx

1.4 Create the first DirectX 12 program

Please check the following site on MSDN for creating a basic Direct3D 12 component
https://msdn.microsoft.com/en-us/library/windows/desktop/dn859356(v=vs.85).aspx