**Direct3D의 그리기 연산**

**6-1. 정점과 입력 배치**

- 원하는 자료(‘특성’)를 가진 커스텀 **정점 형식(vertex format)**을 만들려면 우선 그러한 자료를 담을 구조체를 정의

**※ 입력 배치 서술(input layout description)** - 정점의 각 성분으로 무엇을 해야 하는지를 Direct3D에 알려주는 수단이며, **D3D12\_INPUT\_LAYOUT\_DESC**라는 구조체로 대표.

**※ D3D12\_INPUT\_ELEMENT\_DESC 배열** – 각 원소는 정점 구조체의 각 성분을 서술하고 이 배열의 원소들과 정점 구조체의 성분들은 일대일로 대응한다.

**6-2. 점정 버퍼**

- GPU가 점들의 배열에 접근하려면, 그 정점들은 **버퍼(Buffer)**라 불리는 **GPU 자원(ID3D12Resource)**에 넣어 두어야 함.

- 응용 프로그램에서 정점 같은 자료 원소들의 배열을 GPU에 제공해야 할 때에는 항상 버퍼를 사용.

- **정점 기하구조(=프레임마다 변하지 않는 기하구조)**를 그릴 때에는 최적의 성능을 위해 정점 버퍼들을 **기본 힘(D3D12\_HEAP\_TYPE\_DEFAULT)**에 넣는다.

- 정점 버퍼를 초기화한 후에는 GPU만 버퍼의 정점들을 읽으므로(기하구조를 그리려고), 기본 힙에 넣는 것이 합당하고, CPU는 기본 힙에 있는 정점 버퍼를 수정하지 못한다.

- 실제 정점 버퍼 자원을 생성하는 것과 더불어, 응용 프로그램은 **D3D12\_HEAP\_TYPE\_UPLOAD** 형식의 힙에 임시 업로드용 버퍼 자원을 생성해야 함.

- 정점 버퍼를 파이프라인에 묶으려면 정점 버퍼 자원을 서술하는 **정점 버퍼 뷰**를 만들어야 한다.

- 정점 버퍼 뷰에는 서술자 힙이 필요하지 않다.

- 정점 버퍼와 이에 대한 뷰를 생성했으면, 정점 버퍼를 파이프라인의 한 입력 슬롯에 묶을 수 있다. 그러면 정점들이 파이프라인의 입력 조립기 단계로 공급됨.

- 입력 슬롯에 묶은 정점 버퍼는 다시 변경하지 않는 한 계속 그 입력 슬롯에 묶임.

**※ 정점 버퍼(vertex buffer)** – 정점들을 저장하는 버퍼.

**※ 범용 GPU 자원으로서의 버퍼에서 너비(width)** – 가로 길이가 아닌 버퍼의 바이트 개수를 뜻.

**※ D3D12\_VERTEX\_BUFFER\_VIEW\_DESC** – 정점 버퍼 뷰를 대표하는 구조체.

**※ ID3D12GraphicsCommandList::DrawInstanced 메서드** – 정점들을 실제로 그리는 함수.

**6-3 색인과 색인 버퍼**

- GPU가 색인들의 배열에 접근할 수 있으려면 색인들을 버퍼 GPU 자원(ID3D12Resource)에 넣어둬야 함.

- 색인 버퍼를 파이프라인에 묶으려면 색인 버퍼 자원을 서술하는 **색인 버퍼 뷰**를 만들어야 함.

- 정점 버퍼 뷰와 마찬가지로 **서술자 힙이 필요 없다.**

- 색인 버퍼는 **Id3D12CommandList::SetIndexBuffer 메서드**를 통해서 입력 조립기 단계에 묶음.

**※ 색인 버퍼** – 색인(Index)들을 담는 버퍼

**※ D3D12\_INDEX\_BUFFER\_VIEW** – 색인 버퍼 뷰를 대표하는 형식

**※ ID3D12GraphicsCommandList::DrawIndexedInstanced 메서드** – 색인들을 통해 기본도형을 그리는 함수.

**※ 기준 정점 위치(base vertex location)** – 전역 정점 버퍼에서 한 물체의 첫 번째 정점의 위치.

**6-4 예제 정점 셰이더**

- 본질적으로 셰이더는 하나의 함수.

- HLSL에서 함수는 항상 인라인화.

- 첫 입력 매개 변수는 정점 셰이더의 **입력 서명(input signature)**을 형성.

- **입력 매개변수의 의미소**( :POSITION, :COLOR 등)는 정점 구조체의 멤버들을 정점 셰이더 입력 매개변수들에 대응시키는 역할.

- **출력 매개변수의 의미소**(:SV\_POSITION, :COLOR 등)는 정점 셰이더의 출력을 파이프라인의 다음 단계( 기하 셰이더 또는 픽셀 셰이더)의 해당 입력에 대응시키는 역할.

**※ HLSL(High level shading language)** – 셰이더 작성 언어

**※ SV** – 해당 값이 system value 의미소임을 뜻하고 해당 정점 셰이더 출력 성분이 정점의 위치(동차 절단 공간에서의)를 담고 있음.

**6-5 픽셀 셰이더**

- 픽셀 단편마다 실행

- 주어진 입력으로부터 픽셀 단편의 색상을 계산.

- 후면 버퍼의 한 픽셀에는 최종적으로 그 픽셀이 될 수 있는 ‘후보’로서의 픽셀 단편들이 여러 개 존재할 수 있다.

**※ SV\_TARGET 의미소** – 해당 함수의 반환 값의 형식이 랜더 대상(render target)의 형식과 일치해야 함을 의미.

**6.6 상수 버퍼**

**6.6.1 상수 버퍼의 생성**

- 상수 버퍼는 CPU가 프레임당 한 번 갱신하는 것이 일반적.

- 상수 버퍼는 기본 힙이 아닌, 업로드 힙에 만들어야 함.

- 크기가 **반드시 최소 하드웨어 할당 크기( 256 byte ) 의 배수**여야 함.

- 어떤 물체를 그릴 때가 되면, 이 버퍼에서 해당 물체를 위한 상수들이 있는 부분 영역을 서술하는 상수 버퍼 뷰를 파이프라인에 묶음.

**※ 상수 버퍼(constant buffer)** – 셰이더 프로그램에서 참조하는 자료를 담는 GPU 자원(ID3D12Resource)의 예.

**6.6.2 상수 버퍼의 갱신**

- 자료를 올리려면 먼저 자원 자료를 가리키는 포인터를 **Map 메서드**를 호출하여 얻는다.

- 시스템 메모리에 있는 자료를 상수 버퍼에 복사하려면 memcpy 이용.

- 해당 메모리 해제하기 전에 **unmap** 을 호출.

**6.6.3 업로드 버퍼 보조 클래스**

**6.6.4 상수 버퍼 서술자.**

- 상수 버퍼 서술자는 **D3D12\_DESCIPTOR\_HEAP\_TYPE\_CBV\_SRV\_UAV** 형식의 서술자 힙에 담김.

- 상수 버퍼, 셰이더 자원 뷰(SRV) , 순서 없는 접근 뷰(UAV) 서술자들을 섞어 담을 수 있음.

- 상수 버퍼 뷰를 생성하려면 **D3D12\_CONSTANT\_BUFFER \_VIEW\_DESC 인스턴스**를 채운 후에 **ID3D12Device::CreateConstantBufferView**를 호출.

- **D3D12\_CONSTANT\_BUFFER \_VIEW\_DESC 구조체**는 상수 버퍼 자원 중 HLSL 상수 버퍼 구조체에 묶일 부분을 서술.

**※ D3D12\_DESCRIPTOR\_HEAP\_FLAG\_SHADER\_VISIBLE 플래그** – 셰이더 프로그램에서 이 서술자들에 접근할 것임을 뜻함.

**6.6.5 루트 서명과 서술자 테이블**

- 셰이더 프로그램들은 특정 종류의 자원들이 렌더링 파이프라인에 묶인 상태에서 그리기 호출이 실행되었다고 기대함.

- 자원들은 특정 레지스터 슬롯에 묶이며, 셰이더 프로그램들은 슬롯들을 통해 자원들에 접근.

- 루트 서명은 응용 프로그램이 렌더링 파이프라인에 묶을 자원들을 정의하기만 하고, 실제로 자원들을 묶지 않는다.

- **ID3D12GraphicsCommandList::SetGraphicsRootDescriptorTable**을 호출해서 서술자 테이블 파이프라인에 묶는다.

- 루터 서명을 바꾸면 기존 모든 바인딩이 사라짐.

**※ 루터 서명(Root signature)** – 그리기 호출 전에 응용 프로그램이 반드시 렌더링 파이프라인에 묶어야 하는 자원들이 무엇이고 그 자원들이 셰이더 입력 레지스터들에 어떻게 대응되는지를 정의하며 루트 서명은 반드시 그리기 호출에 쓰이는 셰이더와 호환되어야 함.

**※ ID3D12RootSignature** – 루트 서명을 대표하는 인터페이스.

**※ 서술자 테이블(descriptor table)** – 서술자 힙 안에 있는 연속된 서술자들의 구간을 지정.

**6.7 셰이더의 컴파일**

- 이식성 있는 바이트 코드로 컴파일 되어야 함.

- 그래픽 드라이버는 그 바이트 코드를 다시 시스템의 GPU에 맞게 최적의 네이티브 명령들로 컴파일 ([AT11])

**※ ID3DBlob** – 범용 메모리 버퍼를 나타내는 형식..

**6.7.1 오프라인 컴파일**

**6.7.2 어셈블리 코드 생성**

**6.7.3 Visual Studio를 이용한 오프라인 셰이더 컴파일**

**6.8 래스터화기 상태**

- 래스터화 단계는 **래스터화기 상태(rasterizer state)**를 통해서 구성하고 이 상태를 대표하는 것은 **D3D12\_RASTERIZER\_DESC 구조체.**

**6.9 파이프라인 상태 객체**

- 렌더링 파이프라인의 상태를 제어하는 대부분의 객체는 **파이프라인 상태 객체(pipeline state object, PSO)**라고 부르는 집합체(aggregate)를 통해 지정.

- 이 모든 객체를 하나의 집합체로서 렌더링 파이프라인에 지정하는 이유는 성능 때문.

- 하나의 집합체로 지정하는 덕에 Direct3D는 모든 상태가 호환되는지 미리 검증할 수 있다.

- 드라이버는 하드웨어 상태의 프로그래밍을 위한 모든 코드를 미리 생성 가능.

- 응용 프로그램이 파이프라인 상태의 대부분을 하나의 집합체로서 지정하므로, 드라이버는 파이프라인의 프로그래밍에 필요한 모든 코드를 초기화 시점에서 지정할 수 있다.

- **Direct3D는 기본적으로 하나의 상태 기계(state machine).**

- 어떤 POS를 명령 목록에 묶었다면, 다른 PSO가 묶이기까지는 그 POS가 계속 적용 됨.

**※ ID3D12PipelineState** – PSO 를 대표하는 인터페이스.

**6.10 기하구조 보조 구조체**

- 하나의 기하구조 그룹을 정의하는 정점 버퍼와 색인 버퍼를 한데 엮는 편리한 보조 구조체.

- 실제 정점 자료와 색인 자료를 시스템 메모리에 유지해서 CPU가 그 자료를 언제라도 읽을 수 있게 하는 역할.