**DirectX를 이용한**

**3D Game 프로그래밍 입문**

**< 개요 >**

Direct3D는 3D가속 하드웨어를 이용해 3D 세계를 표현할 수 있게 돕는

저수준 그래픽 API를 말함.

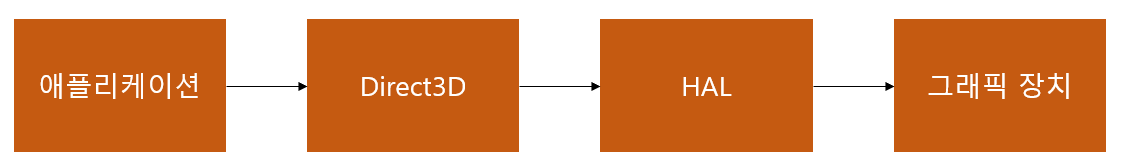


그림 1 애플리케이션, Direct3D, 하드웨어 간의 관계

**< HAL(Hardware Abstraction Layer) >**

HAL은 장치로 하여금 특별한 작업을 수행하도록 하는 장치 고유의 코드로, 이를 통해 Direct3D가 각 장치의 세부적인 부분을 제어할 필요가 없도록 하고, 하드웨어 장치에 독립적인 규약을 확립하는 것이 가능함. 장치 제조사에서는 자신들의 장치가 지원하는 모든 기능을 사용하여 HAL을 구현한다.

다만, Direct3D가 정의하지만 장치에서 지원하지 않는 기능들은 포함되지 않으며, HAL에서 구현되지 않은 기능을 사용하는 Direct3D함수를 호출하면 오류가 발생한다.

**< REF 장치 >**

개발자의 장치에서는 지원하지 않지만 Direct3D에서 제공하는 기능을 이용하는 프로그램을 작성하고자 할 때, 모든 Direct3D API를 소프트웨어로 애뮬레이트하는 레퍼런스 래스터라이저(REF)를 제공하고 있음. 이를 통해 지원하지 않는 Direct3D기능을 이용하는 코드를 작성하고 테스트할 수 있음.

REF 장치는 순수한 개발 목적으로 제공된다는 것에 주의.

REF장치는 SDK에만 포함되며, 최종 사용자에게는 배포할 수 없음.

또한 테스트 이외의 목적으로 이용하기에는 상당히 느림.

**< D3DDEVTYPE >**

HAL 장치는 D3DDEVTYPE열거형의 멤버인 D3DDEVTYPE\_HAL로 지정된다.

비슷하게 REF장치 역시 D3DDEVTYPE열거형의 멤버인 D3DDEVTYPE\_REF로 지정된다.

**< COM(컴포넌트 객체 모델, Component Object Model) >**

COM은 DirectX를 프로그래밍 언어로부터 독립적으로 만들어주며, 하위 호환성을 갖출 수 있게 하는 기술. 보통 인터페이스로 부르며, C++ 클래스와 비슷하게 이용됨.

다른 COM 인터페이스의 메서드나 특수한 함수를 통해 COM 인터페이스의 포인터를 얻는 것이며(빌려쓴다.), 직접 COM인터페이스를 만드는 것이 아님.

모든 COM인터페이스는 IUnknown COM 인터페이스에서 기능을 상속받음.

또한, 사용이 모두 끝나면 직접 삭제하지 않고 인터페이스에서 제공하는 Release 메서드를 호출해야 함.

COM 객체는 자신의 메모리 관리는 스스로 수행.

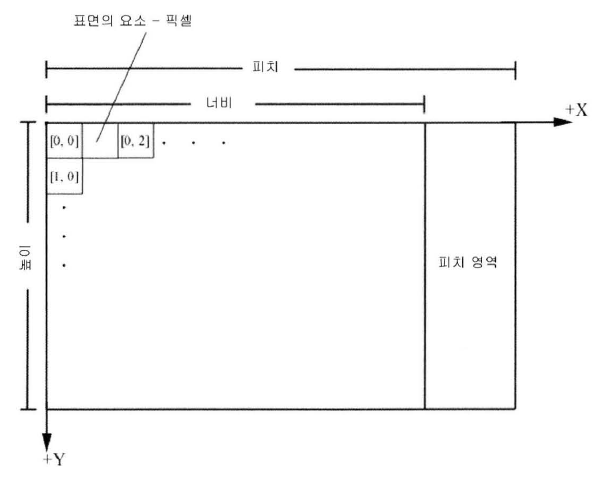
|  |
| --- |
| COM 인터페이스는 접두어로 대문자 I를 가짐. |

**< 기본적인 그래픽 개념과 Direct3D 형에 대한 이해 >**

**표면(Surface)**

표면은 Direct3D가 주로 2D 이미지 데이터를 보관하는 데 이용하는 픽셀의 행렬.

그림에서는 표면 데이터를 행렬로 그렸지만, 실제로 픽셀데이터는 선형 배열에 보관됨.

****

표면의 너비와 높이는 픽셀 단위로 계산됨.

피치는 바이트로 계산되며, 하드웨어 구현에 따라서는 너비보다 길 수도 있음.

따라서 피치=너비 \* sizeof(pixelFormat)라고 가정할 수는 없음.

코드에서 표면을 이용하는 데는 IDirect3DSurface9 인터페이스가 사용됨.

**IDirect3DSurface9**

중요한 메서드들

* LockRect : 표면 메모리로의 포인터를 제공. 여기서 약간의 포인터 연산을

거치면 표면내의 각 픽셀을 읽고 쓸 수 있게 됨.

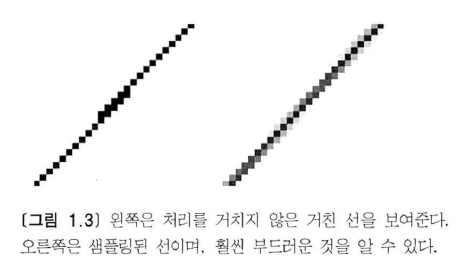
* UnlockRect: LockRect를 호출하고 표면 메모리에 대한 작업이 끝난 뒤에는

이 메서드를 호출하여 표면의 잠금을 해제해야 함.

* GetDesc: 표면에 대한 정보를 D3DSURFACE\_DESC구조체를 통해 얻는다.

**멀티 샘플링**

멀티 샘플링은 픽셀 매트릭스로 이미지를 표현할 때 나타나는 거친 이미지를 부드럽게 만드는 데 이용되는 기술. 가장 일반적으로 안티 앨리어싱을 위한 표면 멀티 샘플링이 있음.



멀티 샘플링은 응용프로그램의 속도를 지나치게 떨어뜨림.

멀티 샘플링을 이용하고자 한다면 IDirect3D9::CheckDeviceMultiSampleType 메서드를 통해 그래픽 카드에서 지원하는 멀티 샘플링 타입과 품질 레벨에 대해 확인해야 함.

**D3DMULTISAMPLE\_TYPE**

D3DMULTISAMPLE\_TYPE 열거형은 표면에 적용할 멀티 샘플링 레벨을 지정할 수 있도록 하는 값들로 구성되어 있음.

* D3DMULTISAMPLE\_NONE : 멀티 샘플링을 지정하지 않음.
* D3DMULTISAMPLE\_1\_SAMPLE, … , D3DMULTISAMPLE\_16\_SAMPLE :

1에서 16까지의 멀티 샘플링 레벨을 지정함.

**픽셀포맷**

표면이나 텍스처를 만들기 위해서는 Direct3D 자원의 픽셀 포맷을 지정해야 함.

픽셀 포맷은 D3DFORMAT 열거형 멤버로 지정되며, 자주 이용되는 포맷은 아래와 같음.

* **D3DFMT\_R8G8B8** : 24비트 픽셀 포팻을 지정. 가장 왼쪽의 8비트는 적색, 다음 8 비트는 녹색, 마지막 8비트는 청색에 할당.
* **D3DFMT\_X8R8G8B8** : 32비트 픽셀 포맷을 지정. 가장 왼쪽의 8비트는 이용되지 않으며, 다음 8비트는 적색, 다음 8 비트는 녹색, 마지막 8비트는 청색에 할당.
* **D3DFMT\_A8R8G8B8** : 32비트 픽셀 포맷을 지정하고, 가장 왼쪽 8비트는 투명도(알파)에 해당됨.
* **D3DFMT\_A16R16G16B16** : 64비트 부동소수점 픽셀 포맷.(투명도 포함)
* **D3DFMT\_A32R32G32B32** : 128비트 부동소수점 픽셀 포맷.(투명도 포함)

첫 세 개의 포맷은 대부분 하드웨어에서 지원하지만, 부동소수점 픽셀 포맷이나 나머지 포맷은 그리 널리 지원되지 않음.

**메모리 풀**

표면이나 그 밖의 다양한 Direct3D자원들은 여러가지 종류의 메모리 풀에 보관할 수 있음. 메모리 풀은 D3DPOOL 열거형의 멤버로 지정되며, 이용할 수 있는 메모리 풀에는 아래와 같은 것들이 있음.

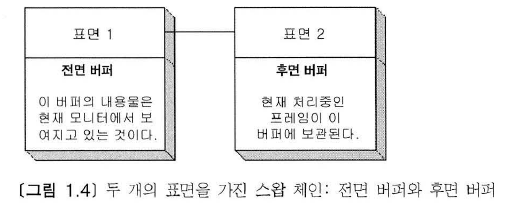
* **D3DPOOL\_DEFAULT** : 디폴트 메모리 풀은 자원의 타입과 이용방식에 가장 적합한 자원들을 메모리에 보관하도록 Direct3D에 요청함. 여기서 말하는 자원이란, 비디오 메모리나 AGP 메모리, 혹은 시스템 메모리 등을 말하는 것으로, 디폴트 풀 내의 자원은 반드시 IDirect3DDevice9::Reset호출 이전에 파괴(해제)되어야 하며, reset 호출 이후에 다시 초기화 되어야 한다.
* **D3DPOOL\_MANAGED** : 관리 풀에 보관된 자원은 Direct3D에 의해 관리된다.(즉 필요에 따라 자동으로 비디오 메모리나 AGP메모리로 옮겨진다.) 부가적으로 자원의 백업 복사본이 시스템 메모리 내에 보관되는데, 응용프로그램이 자원에 접근하고 수정할 때는 시스템 복사본을 이용하며, Direct3D는 필요에 따라 자동으로 이를 비디오 메모리에 갱신함.
* **D3DPOOL\_SYSTEMMEM** : 시스템 메모리 내에 보관될 자원을 지정함.
* **D3DPOOL\_SCRATCH** : 시스템 메모리 내에 보관될 자원을 지정함. 앞서의 D3DPOOL\_SYSTEMMEM과는 달리 이 풀의 자원은 그래픽 장치의 제한을 따라서는 안 된다. 따라서 장치는 이 풀 내의 자원에 직접 접근할 수 없지만, 자원을 두 풀 사이에 서로 복사하는 것은 가능함.

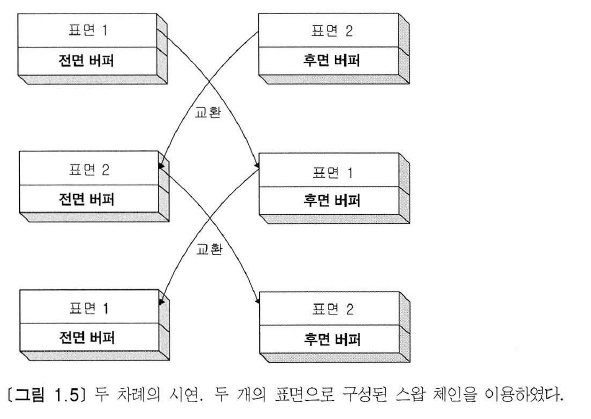
**스왑 체인과 페이지 플리핑**

Direct3D는 보통 두 개나 세 개의 표면을 하나의 컬렉션으로 관리하며, 이를 스왑 체인이라고 부름.

스왑 체인은 IDirect3DSwapChain9 인터페이스를 통해 이용할 수 있지만, 대부분의 작업은 Direct3D가 직접 관리하므로 우리가 이 인터페이스를 이용하는 경우는 거의 없음.

스왑 체인과 페이지 플리핑 기술은 프레임 간의 부드러운 애니메이션을 제공하기 위한 것임.





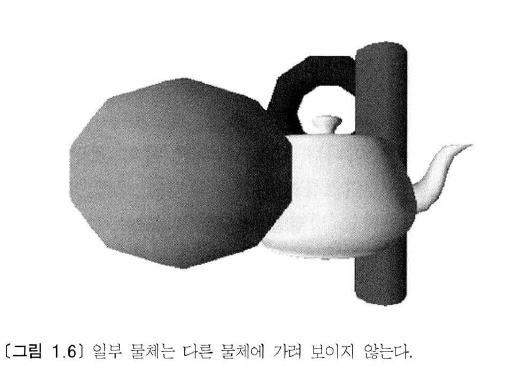
화면 밖의 표면(후면 버퍼)에서 렌더링을 수행하고, 전면 버퍼 표면의 디스플레이가 완료되면 스왑 체인의 끝으로 돌아가 후면 버퍼를 전면 버퍼로 전환하는 방법을 이용. 이와 같은 과정을 시연(Presenting)이라고 함.

**깊이버퍼**

이미지 데이터가 아닌 특정 픽셀의 깊이 정보를 포함하는 표면을 말함.

깊이 버퍼 내에는 최종 렌더링된 이미지의 각 픽셀에 해당하는 항목들을 포함.

예를 들어 렌더링된 이미지가 640 \* 480 해상도를 가진다면, 640 \* 480 개의 깊이 항목이 존재함.



위 그림은 물체의 일부가 다른 물체에 가려 보이지 않는 간단한 상황을 보여주고 있음. Direct3D는 물체의 픽셀이 다른 픽셀을 가리는지의 여부를 판단하기 위해 깊이 버퍼링 혹은 z-버퍼링이라는 테크닉을 이용함.

깊이 버퍼링은 각 픽셀의 깊이 값을 계산하고 깊이 테스트를 수행함으로써 이루어지며, 깊이 테스트는 특정 픽셀의 위치에서 경쟁하는 픽셀의 깊이를 비교하는 과정임. 카메라와 가장 가까운 깊이 값을 가지는 픽셀이 경쟁에서 승리하며, 승리한 픽셀이 이미지에 그려진다. 카메라와 가까운 픽셀이 뒤쪽의 픽셀을 가리는 간단한 원리임.

깊이 버퍼의 포맷은 깊이 테스트의 정확도를 결정함. 즉, 24 비트 깊이 버퍼는 16비트 깊이 버퍼에 비해 더욱 정확함. Direct3D는 32비트 깊이 버퍼를 제공하고 있지만, 대부분의 응용프로그램에서는 24비트 깊이 버퍼로도 충분함.

* **D3DFMT\_D32** : 32비트 깊이 버퍼를 지정.
* **D3DFMT\_D24S8** : 24비트 깊이 버퍼를 지정하고 스텐실 버퍼로 8비트를 예약.
* **D3DFMT\_D24X8** : 24비트 깊이 버퍼를 지정
* **D3DFMT\_D24X4S4** : 24비트 깊이 버퍼를 지정하고 스텐실 버퍼로 4비트를 예약.
* **D3DFMT\_D16** : 16비트 깊이 버퍼를 지정.

**버텍스 프로세싱**

버텍스는 3D 기하 물체를 구성하는 기본 단위로, 소프트웨어나 하드웨어 의 두 가지 방법으로 처리할 수 있음. 소프트웨어 버텍스 프로세싱은 언제나 지원되며 항상 이용 가능한 반면, 하드웨어 버텍스 프로세싱은 그래픽 카드가 버텍스 프로세싱을 지원하는 경우에만 이용할 수 있음.

소프트웨어에 비해 전용 하드웨어의 성능이 우수하므로 하드웨어 버텍스 프로세싱을 이용하는 것이 유리하다. 게다가 하드웨어로 버텍스를 처리하면 CPU를 다른 계산에 할당할 수 있다.

**텍스트, 테이블이(가) 표시된 사진

자동 생성된 설명**

**장치특성**

Direct3D가 제공하는 모든 기능들은 D3DCAPS9 구조체 내의 비트와 데이터 멤버에 대응됨.

기본적인 방식은 특정 하드웨어의 특성에 따라 D3DCAPS9 인스턴스의 멤버를 초기화하는 것으로, 우리의 응용프로그램에서는 D3DCAPS9 인스턴스의 대응되는 비트나 데이터 멤버를 확인하여 장치가 특정한 기능을 제공하는 지를 확인할 수 있다. (SDK 문서 참고)

텍스트이(가) 표시된 사진

자동 생성된 설명

**<Direct3D 초기화>**

1. IDirect3D9 인터페이스로의 포인터를 얻는다. 이 인터페이스는 시스템의 물리적 하드웨어 장치에 대한 정보를 얻고 3D 그래픽을 디스플레이 하는데 이용하는 물리적 하드웨어 장치를 나타내는 C++객체인 IDirect3DDevice9 인터페이스를 만드는 데 이용된다.

2. 기본 디스플레이 어댑터(기본 그래픽 카드)가 하드웨어 버텍스 프로세싱을 지원하는지 알아보기 위해 장치특성(D3DCAPS9)을 확인한다.

3. D3DPRESENT\_PARAMETERS 구조체 인스턴스를 초기화한다. 이 구조체는 우리가 만들고자 하는 IDirect3DDevice9 인스턴스의 특성을 지정하기 위한 몇 가지 데이터 멤버들을 포함한다.

4. 초기화된 D3DPRESENT\_PARAMETERS에 따라 IDirect3DDevice9 객체를 만들어 낸다. 앞서 언급했듯이 IDirect3DDevice9 객체는 3D 그래픽을 디스플레이 하는데 이용될 물리 하드웨어 장치를 나타내는 C++ 객체이다.