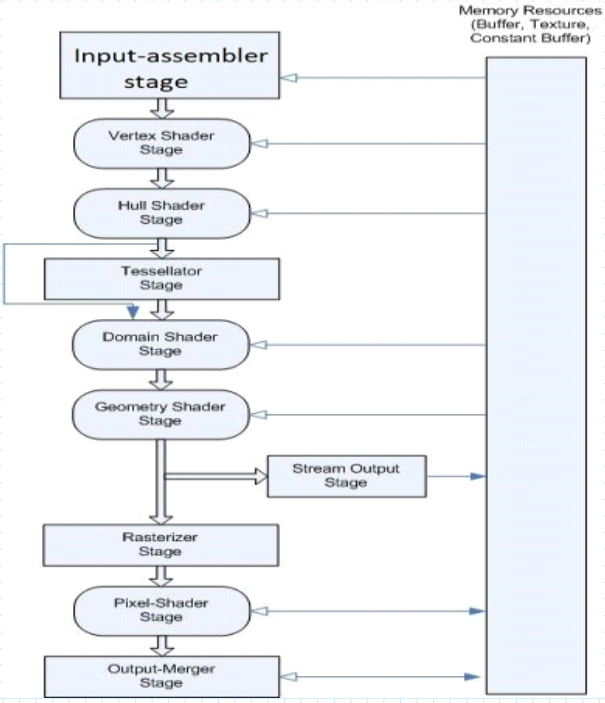
2023년 5월 17일 수요일 오후 8:27

렌더링 파이프라인

렌더링 파이프라인 또는 그래픽스 렌더링 파이프라인이라 부르는 과정은 3차원으로 만들어진 3D 모델 데이트들을 2차원(모니터)에 투영(바꾸다)하는 과정의 프로세를 자세하게 표현한 것이다.

크게 4가지 단계로 나뉜다.
1. 정점 데이터 처리 단계
2. 레스터라이저 단계
3. 픽셀(pixel) 단계 또는 프래그먼트(fragment) 단계
4. 출력 또는 병합 단계

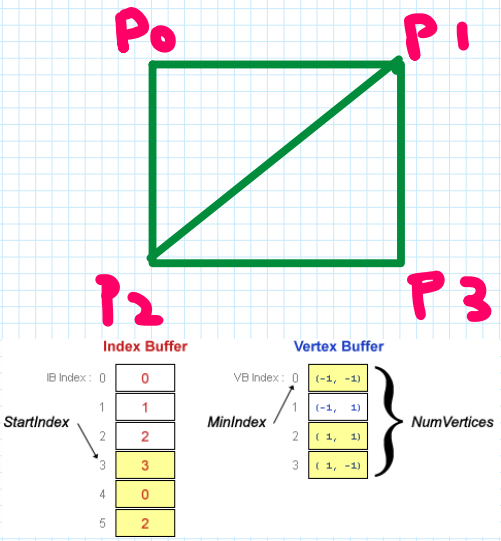


[Input Assembler]

3차원 모델 하나를 3차원 세상에 나타내기 위해서는 가장 먼저 해줘야 할것이 무엇일까?
우선 모델정보를 GPU로 전달해야한다.
여기서 모델은 점의 집합 (폴리곤) 이다. 주로 삼각형을 가지고 3D 물체를 정의한다. 이때 정점들을 운반하는 자료구조를 vertex buffer 라고 합니다.

정점버퍼와 같이 등장하는 용어로 Index buffer라는것이 있다.

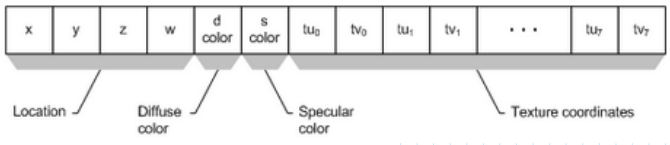
인덱스 버퍼는 쉽게 생각하면 정점들의 인덱스를 저장하고 있는 버퍼라고 할 수 있는데 사각형을 예를들어서 생각해보면 사각형을 그리기 위해서는 2*3 = 6 개의 정점이 필요한데 실제로 사각형을 구성하는데 4개의 정점만 있으면 된다.



6개의 정점으로 표현한다면 2개의 정점이 중복되어 메모리 낭비가 발생할수 있다. 만약에 사각형 하나가 아니라 사각형 10,000개로 구성된 모델이라고 가정하면 20000개의 정점이 낭비된다. 이것은 메모리적으로 엄청난 손해이다. 또한 같은 정점작업도 추가적으로 20000번의 작업이 더 이루어져야 한다. 그럼으로 계산량도 늘어난다.

이러한 중복되는 정저믕ㄹ 해결하기 위한 방법이 Index Buffer이다.
정점은 vertex buffer[] = {p0, p1, p2, p3}; 4개만 보내주고
Indxe buffer[] = { 0, 1, 2, 1, 3, 2};
요런식으로 정점을 어떤 순서로 그려야 하는지 알려주는 목록을 제공하면 위에서 설명한 두가지 문제 (메모리낭비, 연산양 증가) 를 해결할수 있다.

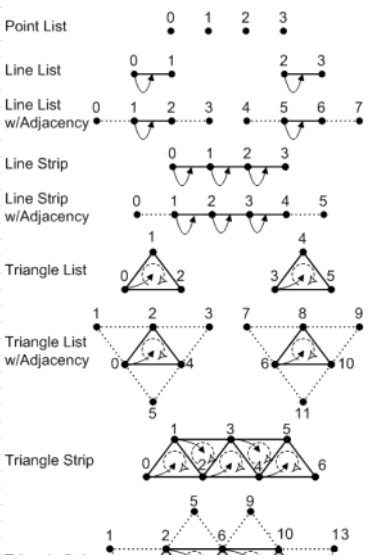
정점데이터가 위치데이터뿐만 아니라 다양한 데이터를 전달하는 예시



그럼 추가적으로 인덱스버퍼를 사용하는데 그것도 결국 메모리를 사용하니까 똑같은게 아닌가 생각할수 있지만 정점은 위치데이터 말고도 색깔, 법선(Normal), UV 등등 여러가지 프로그래머 원하는 데이터를 추가하여 사용이 가능하기 때문에 단순하게 정수만 저장하는 인덱스버퍼가 훨씬 메모리적으로 효율적이다.

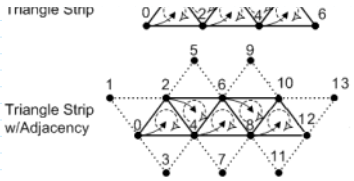
https://learn.microsoft.com/ko-kr/windows/win32/direct3d9/rendering-from-vertex-and-index-buffers

[정점 버퍼]
다시 정점버퍼로 돌아와서
정점버퍼는 그냥 정점들의 연속적인 메모리에 저장하는자료구조에 불과하기 때문에 실제로 gpu에서는 이러한 정점들을 이용하여 어떤 도형을 만들어야 하는지 정보가 필요하다. 해당 도형 정보를 Directx3D에서는 Primitive Topology라고 한다.
대표적으로 point list, line list, triangle strip, trinangle list등이 있습니다.



Input Assembler(입력 조립기)에서는 이러한 정점들 읽어서 삼각형과

대표적으로 point list, line list, triangle strip, trinangle list등이 있습니다.
Input Assembler(입력 조립기)에서는 이러한 정점들 읽어서 삼각형과 같은 도형으로 조립하는 단계의 일을 합니다. 그래서 Input Assembler(입력 조립기)라고 합니다.



[Vertex Shader]
Input Assembler 에서 받은 정점 정보들의 정보로 도형은 생성이 되었지만 로컬좌표계에 있기때문에 해당 데이터를 화면에 그대로 출력 해버리면 여러가지의 도형 전부 같은 위치에 생성이 된다. 공간좌표계(World)로 변환할 필요가 있다. Local Space에서 World Space로 변환하고, 실제 플레이어가 바라보는 카메라 중심이되는 공간 View Space변환해준다. 그리고 마지막으로 Projection 을 거쳐서 최종적으로 정의된 Clip Space 공간으로 변환해준다.