DX에서 랜더링 파이프라인 이중버퍼링을 진행하며 Direct X를 통해서 필요한 연산을 GPU에 관여하여

출력하는 기능을 의미한다.

DirectX의 파이프라인은 크게 5가지로 나눌 수 있다.

IA,RS,OM 단계는 코딩 불가능 VS,PS 코딩할 수 있다.

1.Input Assembler

2.Vertex Shader

3.Rasterizer

4.Pixel Shader

5.Output Merger

-DX의 공간 구분 Transform

Local 설계상 좌표

World 월드 좌표

View 시선에 보이는 공간

Projection 투영공간 3D데이터->2D데이터 한차원 낮추는것

Clip homogeneous space(1,0,0,0) 으로 좌표 표현 w가 0이면 방향 w가 1 이상이면 위치를 표현

공간은 Matrix(행렬)로 표현되고 아래 그림과 같은 순으로 흘러간다.

1.Input Assembler

정점을 입력하고 버퍼에 데이터를 집어넣는 단계를 의미한다.

이 단계에서는 세팅한 좌표계에 따라 점을 시계방향 혹은 반대 방향으로 넣어줘야하는 점을

가장 중요시 여겨야한다.

2.Vertex Shader

IA에서 입력한 정점을 가지고 Matrix(행렬)를 사용하여 공간 변환을 수행한다.

Vertex를 가지고 Polygon(삼각형)을 만들어 내는 과정을 거치고

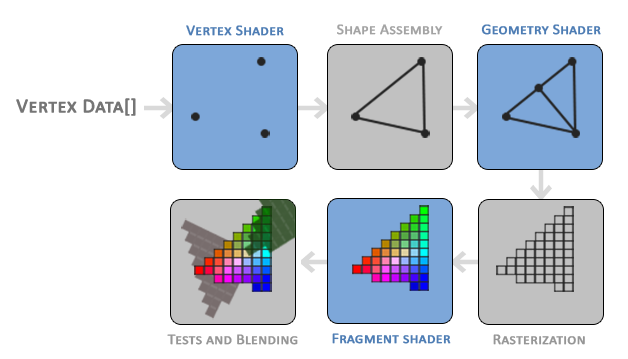


사진 삭제

Vertex Shader의 과정 설명 이미지

만들어진 데이터를 Local->World->View들의 공간에 맞게 Matrix를 사용하여서 정리한다.

이후에 Clip Space에서 원근법을 적용할지 말지 결정하고 NDC좌표계를 통해서 어떤 부분을 사용자에게 보여주고 안보여줄지 결정하는 과정을 거치게 된다.

3.Rasterizer

그리고 이렇게 넘어온 3D 데이터를 화면에 출력하기 위해서

NDC 좌표계 (Normalized Device Coordinate) 정규화된 장치 좌표계 사용하여

Back-Face Culling 과정에서 보이지 않는 면을 제거하고

Viewprot Transform 출력 장치 좌표계로 매핑하여 실제 장치에 출력된 2D 영상을 만들어 내고

Scan Transform 개별 삼각형이 차지하는 스크린 공간의 픽셀 위치를 결정하고, 삼각형의 정점별 속성을 보간하여 이를 각 픽셀 위치에 할당하는 일을 수행하는 단계를 마치고

2D데이터로 바꾸는 것이 Rasterizer 단계의 목적이다.

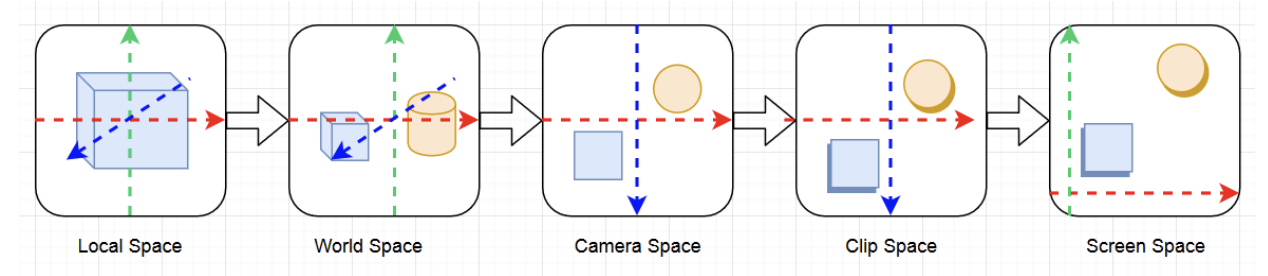


사진 삭제

이 과정을 걷쳐서 사용자에게 출력이 된다

4.Pixel Shader

Pixel Shader 는 Vertex를 통해서 나온 결과를 텍스쳐로 부터 색상을 하나씩 읽어와 Pixel에 적용할 수 있는 연산 프로그램을 지칭한다.

Pixel Shader는 크게 HLSL또는 GLSL로 짜여 지는데 이부분을 제작할때 컴파일러가 오류를 못 잡아 주기 때문에 코드 작성시 확장프로그램을 설치해서 오류를 보여주는 방법 아니면 타이핑시 주의가 필요한 부분이다.

5.Output Merger

다양한 유형의 출력 데이터 (픽셀 셰이더 값, 깊이 및 스텐실 정보)를 렌더링 대상의 내용과 깊이/스텐실 버퍼의 내용과 결합 하여 최종 파이프라인 결과를 생성 합니다.

이런 과정을 거쳐 화면에 출력이 가능해진다.

﻿