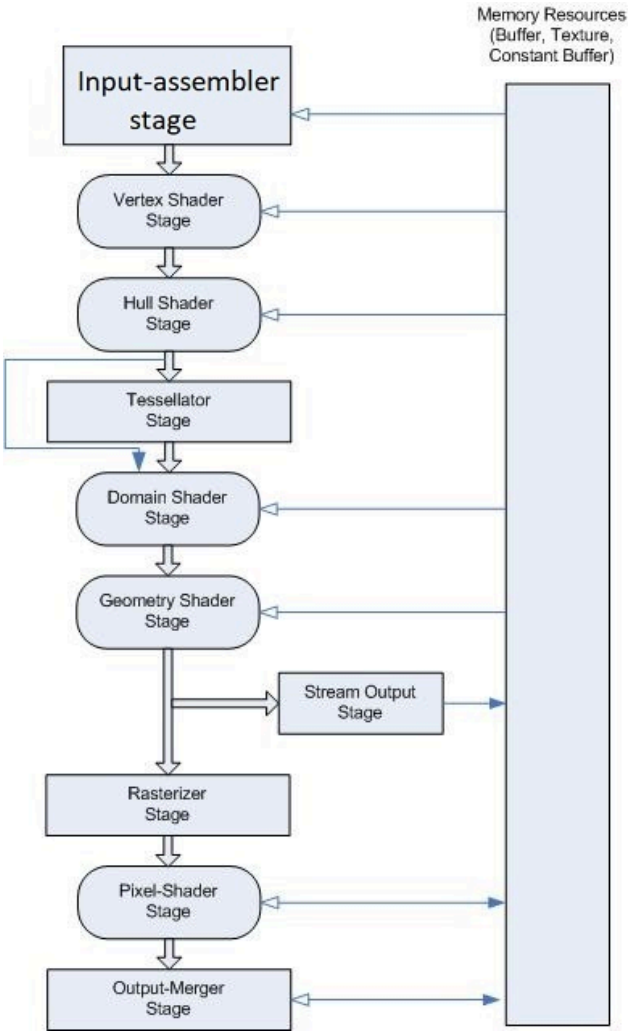
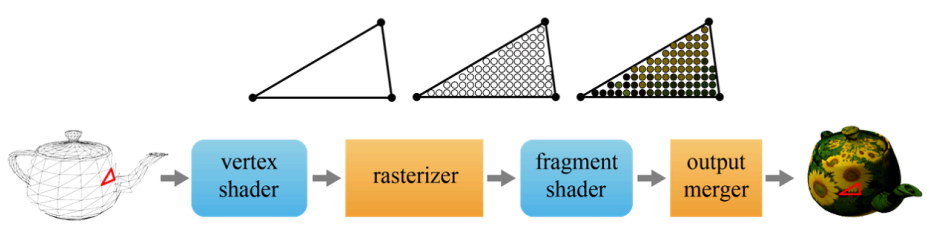
Graphics pipeline - Win32 apps | Microsoft Learn
[면접 준비][Graphics] 렌더링 파이프라인(Rendering Pipeline) (tistory.com)
[Direct3D11] 렌더링 파이프라인? (Rendering Pipeline) :: 오늘의 공부 (tistory.com)
Directx11 렌더링 파이프라인 (tistory.com)
https://velog.io/@hkun_ho/Game-Graphics-DirectX-12
https://eazuooz.tistory.com/62
https://rito15.github.io/posts/rendering-pipeline/

렌더링 파이프라인 9단계에 대해 조사

* 요약:
렌더링 파이프라인은 GPU를 사용해 3D 데이터의 리소스를 2D 이미지로 렌더링 하는 과정입니다.
크게 입력 조립, 정점 셰이더, 래스터화, 픽셀 셰이더, 출력 병합의 단계를 거칩니다.

* 세부설명:





렌더링 파이프라인은 9단계로 구성되어 있습니다.
파이프라인은 Fixed Pipeline, Programmable Pipeline 의 2가지 종류로 분류할 수 있습니다.

Fixed Pipeline은 GPU에서 오직 연산을 수행하고, 프로그래머가 응용 프로그램에서 관여할 수 없는 단계입니다. Input Assembler, Tesselator, Stream Output, Rasterizer, Output-Merger가 해당됩니다.

Programmable Pipeline은 프로그래머가 GPU 연산에 관여할 수 있으며, 이 단계들은 Shader라고 불립니다. Vertex Shader, Hull Shader, Domain Shader, Geometry Shader, Pixel Shader가 해당됩니다.

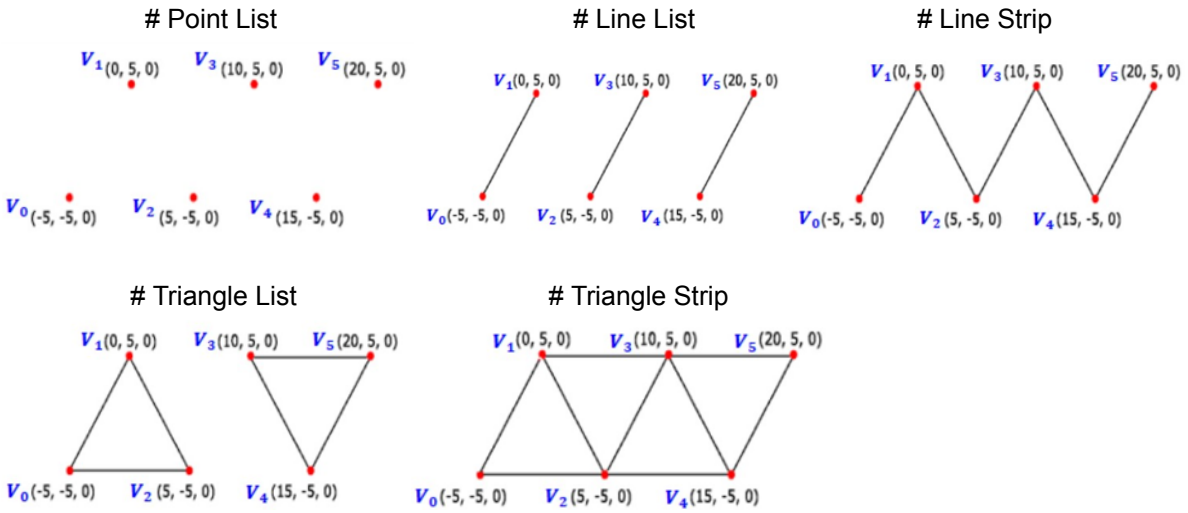
1. Input Assembler (입력 조립**)**

입력 조립 단계는 CPU에서 렌더링을 수행할 도형의 정점 정보들을 정점 버퍼에 담아서 GPU로 운반합니다. 정점 버퍼에 담는 이유는 GPU가 CPU의 자원에 직접적으로 접근할 수 없기 때문입니다.

정점 버퍼에는 렌더링할 물체의 위치, 색상, 텍스쳐의 uv 좌표 값 등이 직렬화된 형태 (배열)로 담겨 있습니다. 텍스쳐의 uv 좌표란 텍스처 이미지를 3차원 공간에 맵핑하기 위한 2차원 공간에서의 좌표를 의미합니다.

GPU가 정점 버퍼를 전달받으면 정점 정보와 Index들을 활용하여
구조체 형식의 정점 데이터로 조립을 합니다.

정점 데이터란 다른 파이프라인 단계에서 사용할 Primitive Type(점, 선, 삼각형 등)을 의미합니다.
Primitive Type은 더 작은 상태로 쪼개거나 분해할 수 없는 기하학적인 형태를 의미합니다.
Primitive Type에는 Point List, Line List, Line Strip, Triangle List, Triangle Strip 등이 있습니다.

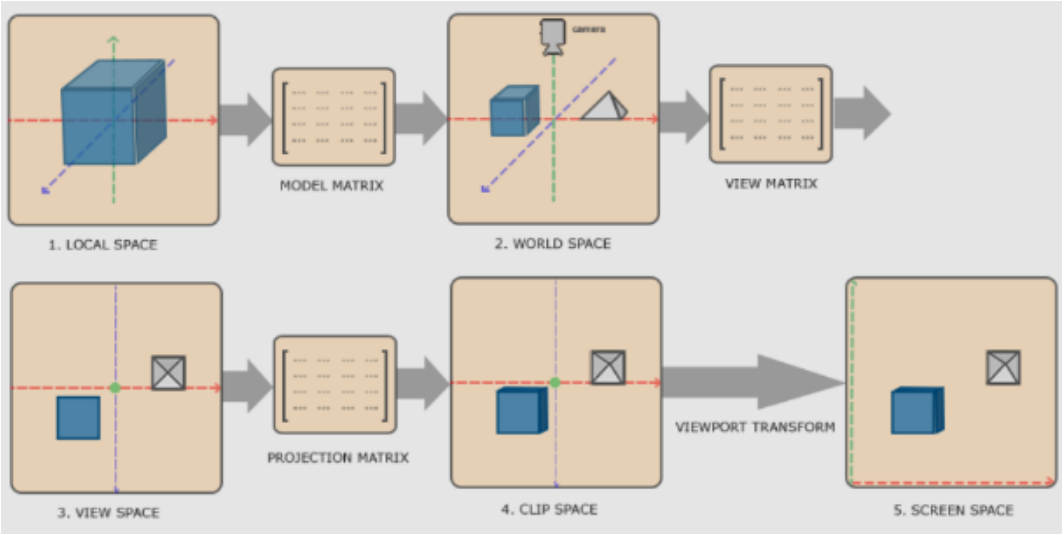


조립이 완료되면 조립된 정점 데이터들은 Vertex Shader에 입력됩니다.

2. Vertex Shader (정점 셰이더**)**

정점 셰이더의 역할은 오브젝트 공간 내에서의 물체의 위치를 화면 좌표로 변환하여 화면에 올바르게 표시하도록 하는 것입니다.

정점 셰이더에서는 조립된 정점 데이터들의 각 정점에 대한 연산을 한번씩 수행합니다.
한 정점에 대해 단일 출력 정점을 생성합니다.

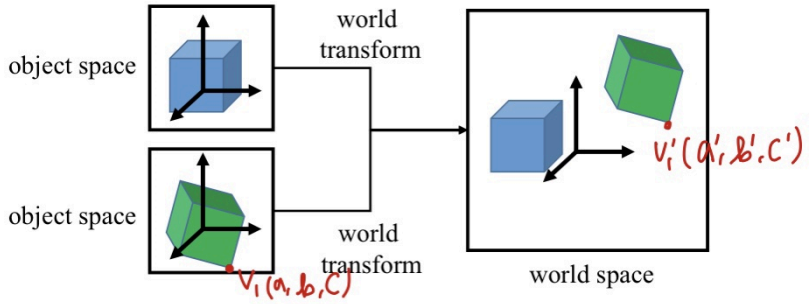


파이프라인이 실행되려면 정점 셰이더 단계가 항상 활성화되어 있어야합니다. 따라서 정점 셰이더 연산 작업이 필요없는 경우에도 임의의 정점 셰이더를 만들어 파이프라인에 적용시킵니다.

프로그래머는 정점 셰이더 함수의 구체적인 내용을 작성할 수 있기 때문에 특수 효과를 수행할 수 있습니다. 특수효과란 Transformation, skinning(정점 혼합), morphing, lighting 등 이 있습니다.

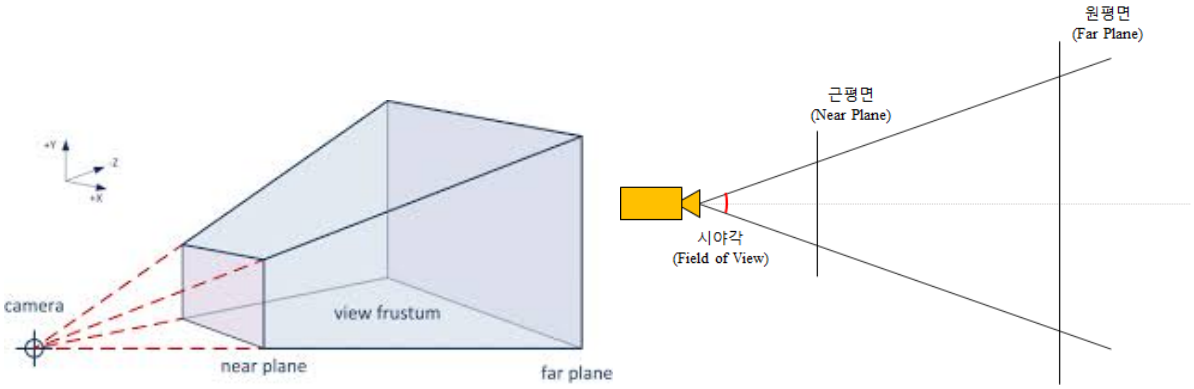
오브젝트 공간 내 물체의 위치를 화면에 올바르게 표시하기 위해서는 공간변환이 일어나야 하고, 이는 행렬 연산을 통해 이루어집니다. Model Matrix, View Matrix, Projection Matrix이 이에 해당합니다.

2.1 World Transform (월드 변환)



물체마다 자신의 로컬 좌표 정보를 가지고 있기 때문에, 이를 월드 공간을 기준으로 통일합니다.

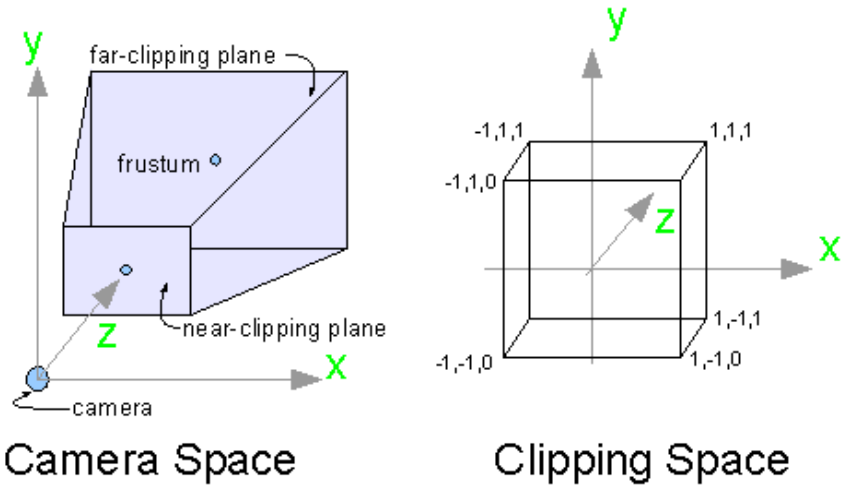
2.2 View Transform (카메라 변환)



View Space는 카메라가 바라보는 공간을 의미합니다. 월드 공간의 모든 정점들은 카메라를 원점으로 하는 좌표계를 사용하여 카메라 공간으로 변환됩니다.

카메라의 시야는 제한이 있을 수 밖에 없으며, 카메라가 볼 수 있는 공간의 부피를 View Space라고 부릅니다. View Space는 절단된 형태로 절두체(Frustum)라고도 합니다.

2.3 Projection Transform (투영 변환)

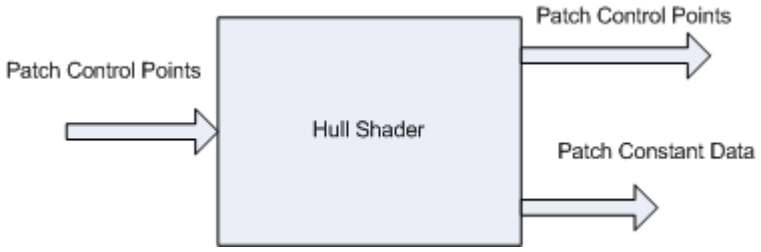


Camera Space Clipping Space

카메라 기준의 정점 위치들을 화면에 보이기 위한 정점 위치로 변환합니다.
카메라의 시야(View Frustum)에 들어오지 않거나, 다른 오브젝트에 가려져서 안보이는 오브젝트들은 렌더링 하지 않도록 합니다. 이를 Culling이라고 합니다.

결과적으로 절두체 형태의 View Space를 행렬(Projection Matrix) 연산을 통해 Clip Space로 변환합니다.

3. Hull Shader (덮개 셰이더)



각 기본 도형(Primitive)에 대한 테셀레이션 계수를 결정하는 단계로, Tessellator 단계를 사용하기 위한 준비과정입니다. Tessellator가 어떻게 쪼개야 하는지 필요한 정보를 넘겨주는 단계입니다.
그리고 기본 도형을 분할하는데 사용할 제어점들을 영역 셰이더(Domain Shader)로 전달해줍니다.

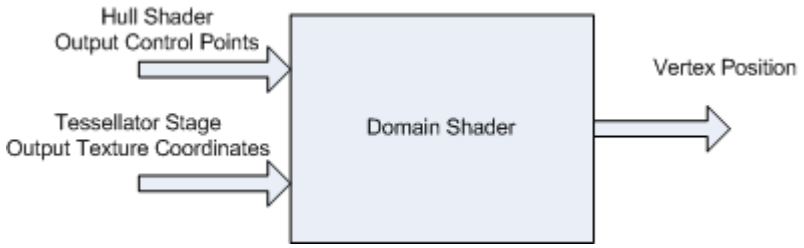
4. Tesselation

테셀레이션은 한 메시의 삼각형들을 잘게 쪼개서 새로운 삼각형들을 만드는 과정으로,
이를 통해 모델의 정점을 더 잘게 쪼개어 디테일한 렌더링이 가능해집니다.

Level Of Detail 메커니즘이란 가까운 거리의 물체는 고해상도(더 많은 삼각형)로, 먼 거리의 물체는 저해상도로 표현하여 성능을 향상시키는 것입니다.

메모리에는 적은 수의 삼각형으로 이루어진 메시(low-poly mesh)를 담아 두고, 필요할 때마다 즉석에서 삼각형들을 추가함으로써 메모리를 절약할 수 있습니다. 또한 애니메이션이나 물리 처리 연산들을 low-poly 메시에서 수행하면 계산량을 줄일 수 있습니다.

5. Domain Shader (영역 셰이더**)**



덮개 셰이더가 생성한 제어점들을 입력으로 받아서 새 정점들을 생성합니다.
도메인 셰이더(Domain Shader)에 대한 입력은 테셀레이터(Tessellator) 단계의 단일 출력 지점이고, 출력은 테셀레이션(세분화)된 정점의 최종 위치를 계산한 결과입니다.

6. Geometry Shader (기하 셰이더**)**

렌더링 파이프라인에서 정점 셰이더 단계와 픽셀 셰이더 단계 사이에 있는 생략 가능한 단계입니다. 기하 셰이더 단계에서는 정점 셰이더 처리를 거친 정점 데이터를 입력받아 응용프로그램의 셰이더 코드를 실행합니다. 입력 데이터는 항상 기본도형을 정의하는 정점들의 배열입니다.

정점 셰이더에서 생성되지 않은 임의의 정점을 추가하거나 삭제하여 모델을 수정할 수 있습니다. 결과적으로 기하 셰이더는 입력받은 정점들을 자신의 목적에 맞게 기본도형(점, 선, 삼각형)들로 재해석해서 출력 스트림을 통해 파이프라인의 다른 단계로 전달합니다.

7. Stream-output stage

스트림 출력 단계는 기하 셰이더 또는 정점 셰이더로부터 지속적으로 정점 데이터를 얻어서 정점 버퍼에 저장하고, 다시 GPU로 운반하는 과정(스트림) 입니다.

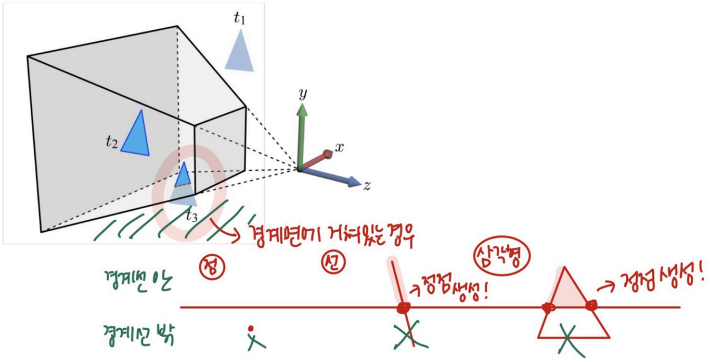
버텍스들은 언제나 완전한 Primitive로 쓰여지며 (하나의 삼각형에는 3개의 정점), 완전하게 만들어지지 않은 Primitive는 출력되지 않습니다.

8. Rasterization (래스터화**)**

래스터화는 Clip Space의 정점 데이터를 활용하여 3D공간을 2D 공간(래스터 이미지)로 변환해주는 단계입니다. 각 Primitive를 구성하는 정점들은 픽셀(Fragment)로 변환되고, Primitive 내부에 해당하는 점들도 보간을 통해 픽셀로 변환합니다.

보간이란 새로운 점들을 만들기 위해 수많은 점들을 평균화시키고, 궤적을 생성해서 연결하는 방법입니다. 그래픽에서 보간이란 원본 이미지의 픽셀 값을 기반으로 새로운 픽셀 값을 생성하는 것을 의미합니다.

8.1 Clipping (클리핑)

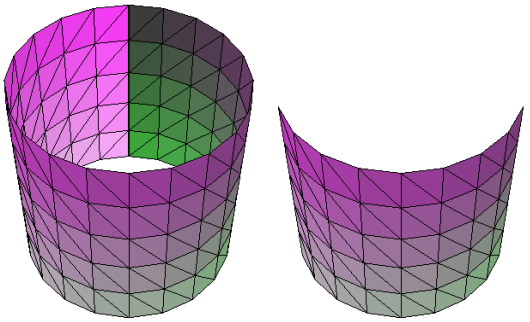


Clipping이란 Primitive가 Clip Space의 경계에 있을 때, 걸쳐있는 부분에 대해 새로운 정점들을 생성하여 분할시키고, Clip Space 안쪽에 위치하고 있는 새로운 Primitive들만 취하는 것입니다.

8.2 Perspective Division (원근 분할)

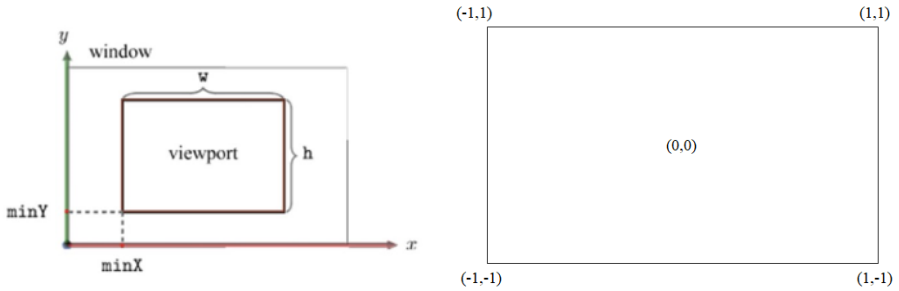
가까운 물체와 먼 거리의 물체를 크게/작게 표현하여 원근감을 부여합니다.
이를 위해서 클립 공간 좌표의 각 항에 대한 연산작업을 해주고, 분할이 완료(w좌표가 1)된 좌표계를 NDC(Normalized Device Coordinates)라고 합니다.

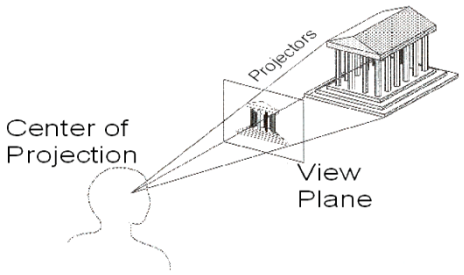
8.3 Back-Face culling (뒷면 제거)



뒷면 제거는 카메라에 등을 돌리고 있어 보이지 않는 폴리곤을 제거하는 작업입니다.
후면은 일반적으로 사용자에게 보이지 않는 면이기 때문에 화면에 그리지 않음으로써 렌더링 성능을 높일 수 있습니다.

8.4 Viewport Transformation (뷰포트 변환)





뷰포트 변환은 NDC 공간 상의 좌표를 2D 스크린 좌표로 변환합니다.
모니터 화면의 해상도는 사용자마다 다르므로 정규화된 좌표계인 Normal Device Coordinate을 이용하여 계산합니다. NDC는 중앙이 원점이며 가로, 세로의 크기가 2인 좌표계입니다.

8.5 Scan Transform (스캔 변환)

래스터화의 마지막 단계로, 삼각형 내부에 차지하는 픽셀(Fragment)을 생성하는 과정입니다.
Screen Space 내 삼각형들의 픽셀 위치를 결정하고, 삼각형의 정점별 속성 (위치, 색상, UV 등)을 보간하여 각 픽셀 위치에 할당합니다.

9. Pixel Shader (픽셀 셰이더**)**

래스터화된 도형들에 대해 렌더링 될 픽셀들의 색을 계산하고, 변화시킬 수 있습니다.
투명도, 조명, 그림자를 처리하거나 텍스쳐에 색상을 입힙니다.
그리고 모든 Primitive의 각 픽셀에 대해 픽셀 셰이더를 한 번씩 호출합니다.

10. Output Merger (출력 병합기**)**

렌더링 파이프라인의 마지막 단계로, 최종적으로 화면에 그려질 픽셀을 정합니다.
여러 물체들의 픽셀이 겹친 상태일 수 있으므로 최종 색상을 판단하기 위한 연산을 수행합니다.
연산의 종류로는 Z-Test, Stencil Test, Alpha Blending 등이 있습니다.

최종적으로 픽셀들의 색상이 결정되면 화면에 출력하게 됩니다.