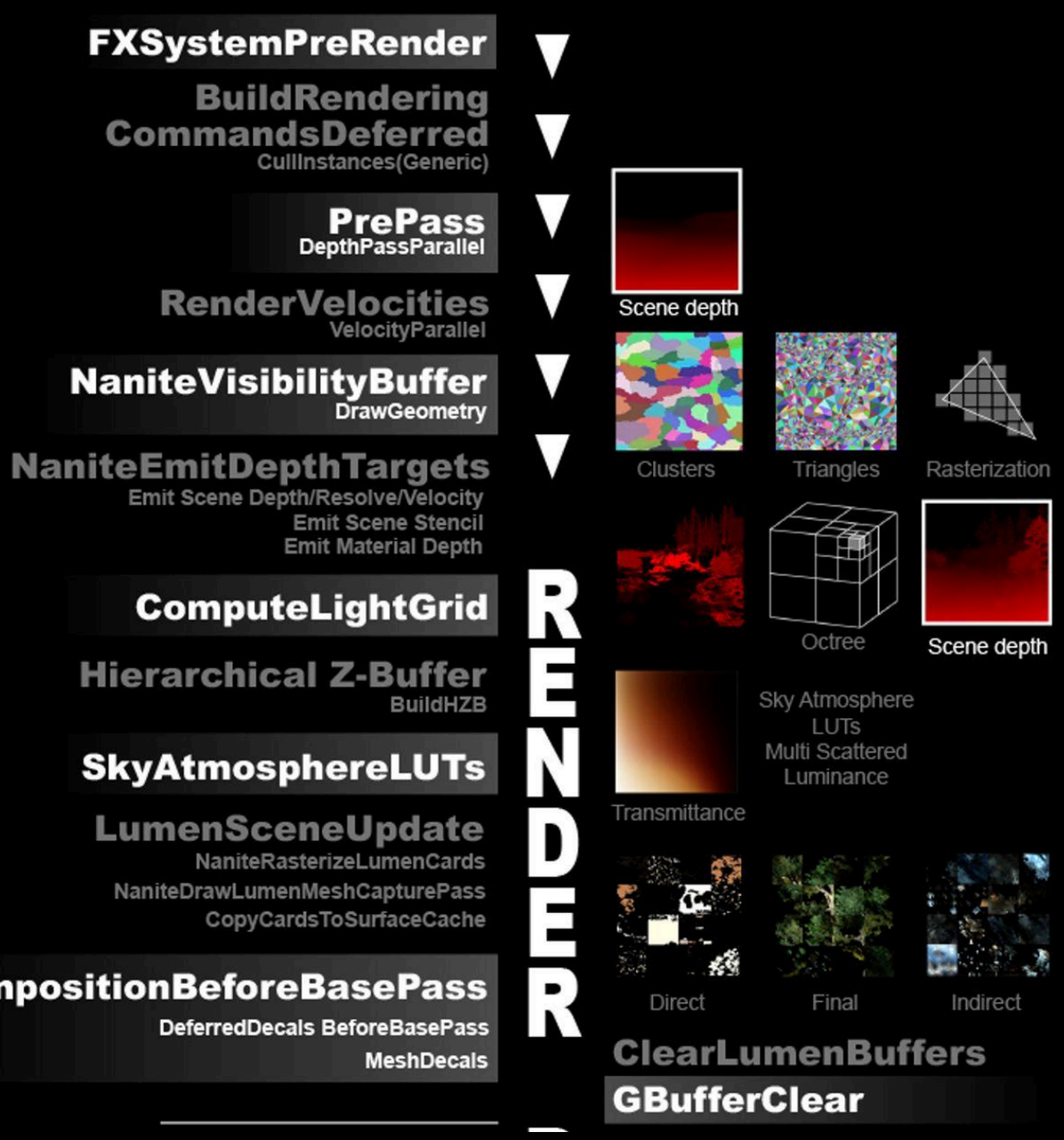

FX 시스템 사전 렌더링
시퀀스나 게임 플레이가 시작되기 전에 특정 효과(이펙트)를 미리 시뮬레이션하고 준비시키는 과정

Build Rendering Commands Deferred (지연된 렌더링을 위한 렌더링 명령)

Game Thread : 다음 프레임의 게임 로직(물리, 애니메이션, AI)을 계산
Rendering Thread : 현재 프레임의 렌더링 명령을 구성

게임 스레드 → 렌더 스레드: 장면 스냅샷과 드로우 요청 전달
GPU : 이전 프레임에서 Rendering Thread가 제출한 명령을 실행

PrePass

- DepthPassParallel

깊이 버퍼 생성 및 깊이 테스트 : 불투명 오브젝트 대상
폐색 테스트(Occlusion Culling) : HZB의 입력으로 사용
깊이 값 기록 / Early Z-Test 를 위한 초석

Parallel (병렬)

다수의 명령 리스트를 생성하고
병렬로 실행하여
멀티코어 CPU 환경에서 렌더링
명령 제출 속도를 높임

RenderVelocities (속도 렌더링)

- VelocityParallel

화면의 각 픽셀에 대한 모션 벡터(Motion Vector)를 계산
속도 버퍼(Velocity Buffer)라는 별도의 텍스처에 저장
TAA/TSR 안티엘리어싱과 모션 블러가 속도 버퍼 기반으로 작동됨

Motion Vector (모션 벡터)

2D 벡터이며 현재 프레임의 한
픽셀이 바로 이전 프레임에서는
화면의 어느 위치에 있었는지를
알려주는 '움직임 정보'

NaniteVisibilityBuffer (나나이트 가시성 버퍼)

가시 프리미티브를 식별할 최소 정보 + 깊이를 저장하는 버퍼
클러스터 + 프리미티브 식별자 (및 머티리얼 조회용 인덱스 포함) + 깊이 형태
이 버퍼의 정보는 나중에 BasePass에서 G-Buffer를 채운다

- DrawGeometry

GPU 컴퓨트 쉐이더가 인스턴스/클러스터 컬링과 LOD 선택
하드웨어, 소프트웨어 라스터 단계에서 픽셀마다 가시성 정보를 Visibility Buffer에 기록
결과는 BasePass에서 머티리얼 셰이딩의 입력이 된다

- Nanite Emit Depth Targets

VisibilityBuffer만으로 깊이·스텐실·속도 같은 깊이 계열 출력을 먼저 만들어두는 단계
전체 머티리얼 셰이딩 없이, 후속 패스가 필요로 하는 최소 데이터를 확보한다

Compute Light Grid (Tiled Rendering)

포워드 렌더링(Forward Rendering)에서 빛 연산의 효율성을 높이기 위한 최적화 기법
3D 공간을 그리드(Grid)로 분할하고 각 그리드 셀에 영향을 미치는 광원 목록을 저장

- Compute Shader를 사용하여 뷰 프러스텀을 균일한 3D 그리드로 분할
- 각 그리드 셀에 대해 해당 영역에 영향을 미치는 광원들을 식별
- 렌더링 시 픽셀의 위치에 해당하는 그리드 셀에서만 광원 목록을 참조

Compute Light Grid (Tiled Rendering)

포워드 렌더링(Forward Rendering)에서 빛 연산의 효율성을 높이기 위한 최적화 기법
3D 공간을 그리드(Grid)로 분할하고 각 그리드 셀에 영향을 미치는 광원 목록을 저장

- Compute Shader를 사용하여 뷰 프러스텀을 균일한 3D 그리드로 분할
- 각 그리드 셀에 대해 해당 영역에 영향을 미치는 광원들을 식별
- 렌더링 시 픽셀의 위치에 해당하는 그리드 셀에서만 광원 목록을 참조

Hierarchical Z-Buffer (계층적 Z-버퍼)

Z-Buffer(Depth Buffer)를 MIP맵처럼 계층적 피라미드로 만들고 비교하여
불필요한 연산을 줄임.

- **buildHZB**

언리얼 엔진에서 HZB 피라미드를 생성하는 렌더 패스의 명칭

- PrePass 과정의 깊이 버퍼를 입력으로 받아 씬 깊이 텍스쳐 생성
- 다운샘플링으로 2×2 깊이 집계로 다음 밉 생성 (집계 연산은 깊이 규약에 따라 달라짐)
- 이 과정을 계속 반복하여 1×1 텍스처가 될 때까지 계층 구조 밉맵 생성
- 이렇게 완성된 밉맵 체인이 바로 Hierarchical Z-Buffer

만들어진 Z-Buffer는 다른 렌더링 패스(오클루전 컬링, SSR, TAA 등)에서 참조

Sky Atmosphere LUTs (하늘 대기 Look-Up Tables)

물리 기반의 사실적인 하늘과 대기 효과를 실시간으로 렌더링하기 위해 미리 계산된 텍스처들의 집합

- 핵심은 미리 계산(Pre-computation) 하는 것
- 대기 산란 계산을 사전에 한 번만 수행하여 그 결과를 텍스처(LUT)에 저장
- 실제 렌더링 시 이 텍스처를 참조 (Look-Up) 만 함

주요 LUT의 종류와 용도

1. Transmittance LUT (투과율 LUT)
- 특정 지점에서 특정 방향으로 빛이 대기층을 통과할 때 얼마나 빛이 감소(흡수/산란)되는지를 저장

2. Multi-Scattering LUT (다중 산란 LUT)
- 다중 산란광의 평균 기여량을 고도와 태양 고도(태양광 경로 각도) 별로 저장

3. Sky View LUT (하늘 뷰 LUT)
- 카메라 시점에서 시선 각도, 태양 각도에 따른 하늘의 최종 색상을 미리 계산

4. Aerial Perspective LUT (공기 원근 LUT)
- 카메라와 물체 사이의 대기가 빛을 산란시켜 발생하는 "공기 원근감" 효과를 저장
- 먼 물체가 흐려지고 푸르스름해지는 효과

SkyAtmosphereLUT 계산 과정에 Volumetric Fog LUT는 포함되지 않는다.

- SkyAtmosphere LUT: 2D, 3D LUT (고도·시선·태양 각 기반)
- Volumetric Fog LUT: 3D Froxel Volume (뷰 공간 XYZ 분할)

Lumen Scene Update (루멘 씬 업데이트)

원본 씬을 기하(Nanite/Distance Field) + 표면 속성(Surface Cache Atlas)으로 구성된 광선 추적용 데
이터로 변환하고 업데이트 하는 과정을 의미한다

루멘 씬 생성 (초기 빌드)

1. Nanite Rasterize LumenCards (나나이트 지오메트리를 루멘 카드로 래스터화)

씬의 메시들을 분석해 표면을 루멘 카드(Lumen Cards)라는 수많은 사각형 판으로 씬을 재구성
각 카드는 월드 위치·방향·크기를 가지며, 카드의 공간적 배치는 한 번 정해지면 계속 유지된다

2. Nanite Draw Lumen Mesh Capture Pass (루멘 메시 캡처 패스)

각 카드에 대해 대응하는 나나이트 메시를 렌더링하여 메시의 재질 속성(Material Attributes)을 캡처
알베도·노멀·러프니스·이미시브 등 재질 정보를 갖게 됨

3. Copy Cards To Surface Cache (서피스 캐시에 카드 데이터 복사)

캡처한 재질 데이터를 효율적으로 관리하기 위해 GPU 메모리 내 저장소에 정리하는 단계
서피스 캐시(Surface Cache)라고 불리는 거대한 텍스처 아틀라스(Atlas)에 차곡차곡 복사하여 저장

- **루멘 씬의 각 카드는 "내 표면 정보는 아틀라스의 (U, V) 좌표에 저장되어 있어" 라고 쓴 주소값만 가리킴**
- **루멘이 레이 트레이싱 중 특정 카드와 충돌하면, 해당 카드가 알려주는 주소(좌표)를 이용해 아틀라스를 딱 한 번 조회함으로써 표면의 색상과 재질을 매우 빠르게 알아냄**
- **즉 히트 지점 → 카드 인덱스 → 아틀라스 샘플로 표면 속성을 얻는다**

Lumen Scene Update 과정

dirty 카드 : 메시가 새로 생기거나 머티리얼이 변경되어 다시 캡처해야 하는 카드
dirty 카드의 머티리얼 속성을 다시 렌더해 Surface Cache Atlas에 덮어씀

1. 장면 변화나 시야 이동으로 카드가 더러워졌는지 (dirty) 판단
2. dirty 카드가 있으면 Nanite Rasterize LumenCards 로 캡처 타겟 준비
3. Nanite Draw Lumen Mesh Capture Pass 로 해당 카드의 머티리얼 속성을 다시 렌더링
4. 캡처된 결과 (알베도·노멀·러프니스 등)를 기존 Surface Cache Atlas 카드 위치에 덮어 씀

Composition Before Base Pass (베이스 패스 시작 전 합성 단계)

베이스 패스가 시작되기 전에 G-Buffer의 내용을 미리 수정하거나 덮어쓰는 합성 단계
G-Buffer에 기록될 재질 자체를 미리 바꿀 때 사용한다

1. PrePass
- 캔버스에 스케치 (깊이 정보)를 그리는 단계

2. Composition Before Base Pass
- 스케치 위 특정 영역에 먼저 특수한 밑칠 (데칼 속성)을 칠하는 단계

3. Base Pass
- 모든 픽셀에 기본 머티리얼 속성을 덮어쓰며 채색하는 단계
- 미리 밑칠된 부분은 최종 색에 반영된다

Deferred Decals, Mesh Decals 정보를 미리 G-buffer에 기록해둠