**언리얼 엔진 모바일 최적화**

**모바일 최적화를 통해 메모리 사용량을 3.5GB 이하로 유지하고, 30FPS의 안정적인 프레임을 목표로 합니다.**

**\* 메모리 최적화**

**1 . ASTC (Adaptive Scalable Texture Compression) 적용**

- 캐릭터 , UI 는 ASTC 6X6 적용

- 월드 텍스처 는 ASTC 8X8 적용

**-> 텍스처 메모리 사용량 최대 40% 절감, VRAM 최적화**

**2 . 텍스처 리사이징**

- 4K/2K 텍스처 제거, 1024px 이하로 리사이징

**-> 메모리 사용량 절감 & 로딩 속도 향상**

**3 . 텍스처 스트리밍 설정 최적화**

- 한 번에 로드되는 텍스처 개수 제한 , 텍스처 풀 크기 조정

**-> 렌더링 부하 감소**

**4 . Platform Memory Bucket 설정**

- DefaultMemoryBucket 설정

**-> 기기별 메모리 최적화, 저사양 기기에서도 안정적인 실행**

**5. Level Streaming 적용**

- UGameplayStatics::LoadStreamLevel() / UnloadStreamLevel() 활용

- 맵 로드 최적화 : 필요 없는 맵 자동 언로드

**-> 초기 로딩 속도 단축 및 메모리 사용량 절감**

**6 . 오브젝트 풀링 & 동적 로딩 적용**

- 자주 생성되는 액터 (몬스터, 투사체, 이펙트) 는 오브젝트 풀링 적용

- LoadAssetAsync() 를 활용하여 비동기 에셋 로딩 적용

**-> CPU, 메모리 사용 최적화로 게임 플레이 중 끊김 현상 방지**

**\* 프레임 최적화**

**1 . 프레임 측정 및 분석 시스템 구축**

- Profiler & Debug UI 를 활용하여 프레임 성능 데이터를 수집

**-> 프레임 드랍 원인을 명확히 분석하여 최적화 작업 방향 설정 가능**

**2 . 파티클 이펙트 렌더링 부하 해결**

- GPU 부하가 큰 파티클은 모바일용 저사양 버전 제작

- Opacity 연산을 줄임 : 반투명 파티클 사용 최소화 , 반투명 대신 Clip Mask 사용

- Niagara Particle Count 를 줄여서 파티클 수를 제한하고 파티클의 Lifetime 조정 , LOD 적용

**-> 렌더링 부하 감소로 평균 FPS 향상 및 배터리 소모 절감**

**3 . GC (Garbage Collection) 퍼지 최적화**

- GC 실행 간격을 늘려 CPU 부하 분산

- 오브젝트 풀링 적용 : 자주 생성& 삭제되는 액터 (몬스터, 투사체 등) 미리 생성하여 재사용

**-> GC 실행 부하 감소로 갑작스러운 프레임 드랍을 방지**

**4 . 플레이 중 에셋 로딩 & 액터 스폰 최적화**

- 사전 로딩 적용 : UAssetManager::LoadPrimaryAsset() 활용하여 에셋을 미리 로드 , Level Streaming 사용하여 필요한 리소스를 미리 로드하고 언로드

- 오브젝트 풀링 적용 : 매번 SpawnActor() 호출하여 새 오브젝트를 생성하는 게 아닌 미리 생성한 오브젝트를 활성화/비활성화하여 사용

**-> 불필요한 메모리 사용 방지 및 CPU 연산 부담 감소**

**5 . Opacity 렌더링 부하 해결**

- Opacity 사용 최소화 , Depth Prepass 활성화

**-> Opacity 연산을 미리 처리하여 렌더링 부하 감소**