Tile-Based Rendering
(성능 개선)

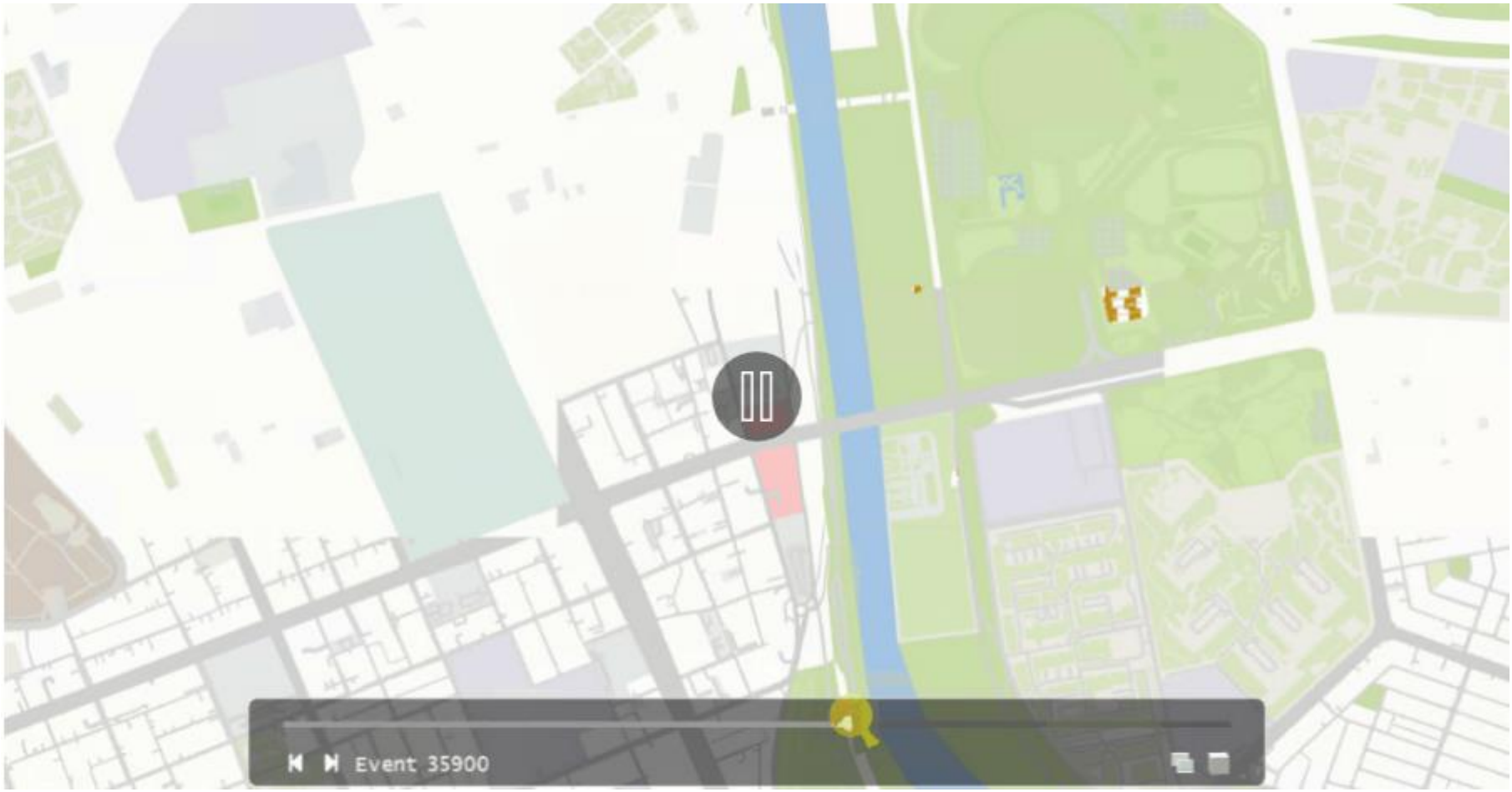
Jeseon Lee

1. Background

- 김한용 모카 유튜버 : 제네시스 차량에서 핀치 줌 인 아웃 시 엄청 느림
  - 주행 테스트 시 "성능 이슈" 발생(아래 영상 참고)
  - → **성능 개선**에 대한 필요성

1. Background

- Direct Rendering(큰 도화지)
  - 전통적으로 이어져 온 렌더링 방법은 draw call 명령에 대해서 오브젝트를 화면(출력 버퍼)에 곧바로 그리는 방식



1. Background

- Tile-Based Rendering(작은 도화지들)
  - FramebufferObject(FBO)를 작은 tile 단위로 나누어서 Rendering 작업을 수행. 이후 FBO들을 출력 버퍼에 Attach 방식

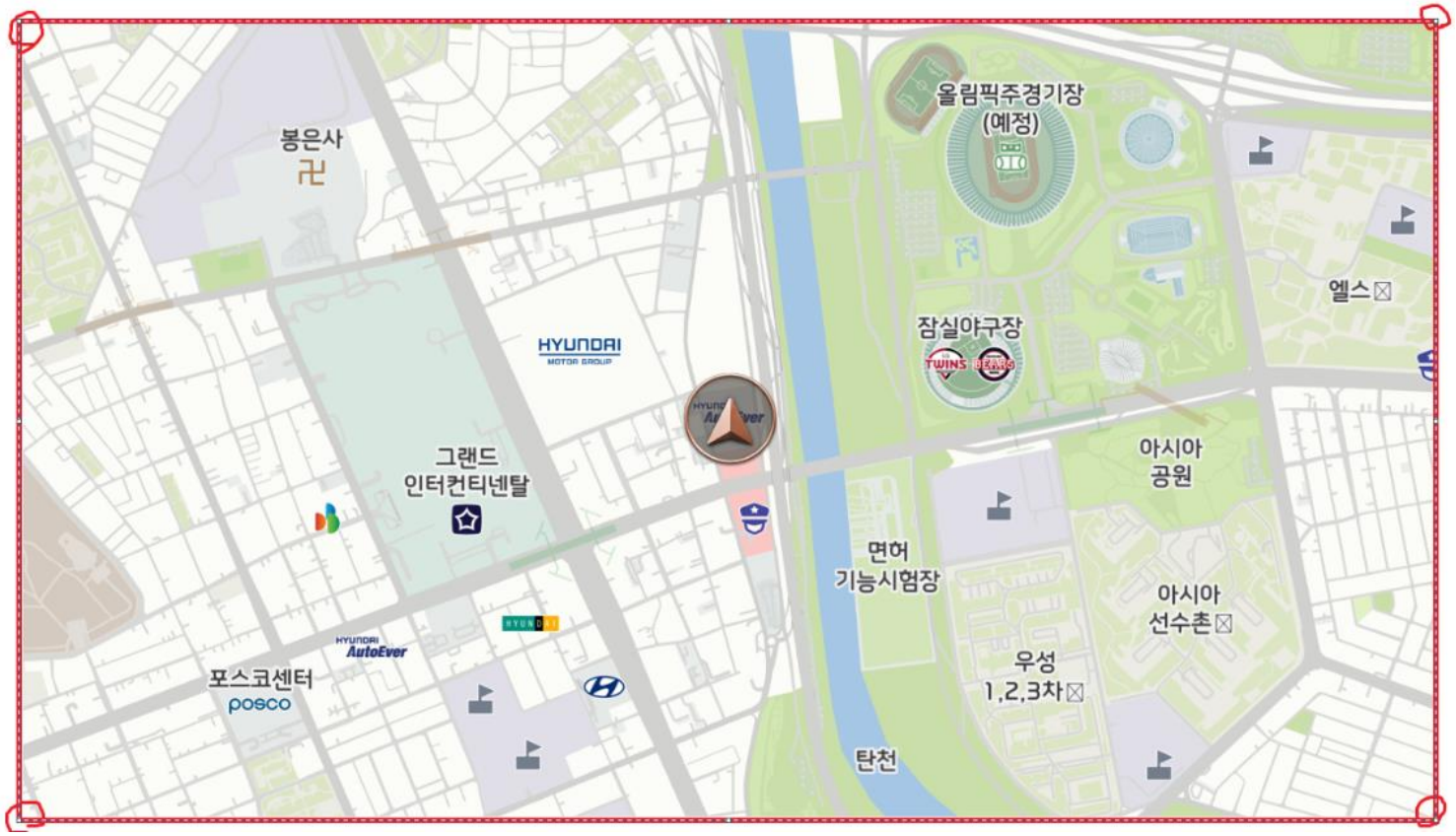
1. Background

- Direct Rendering vs Tile-Based Rendering

	pros	cons
Direct Rendering	전통적인 Rendering Pipeline 방식이라 구현이 간단	다음 프레임에 변화 이벤트가 없어도 frame을 다시 그려야되서 오브젝트별 draw call 발생
Tile-Based Rendering	미리 Tile에 Baking 해두면 다음 frame에 변화 이벤트가 없을 시 bake된 Tile 사용하여 재활용. 즉, draw call 절약 가능.	각 Tile FBO를 처음 생성할 때는 실제로 Direct Rendering이 더 시간이 소요. Tile FBO 텍스쳐가 생성되야하므로 GPU 메모리를 더 사용함.

2. Tile-Based Rendering

- 어떤 기준으로 Tile을 분할할 지
  - 1) 카메라 범위의 사각형 LonLat(위경도) 구함



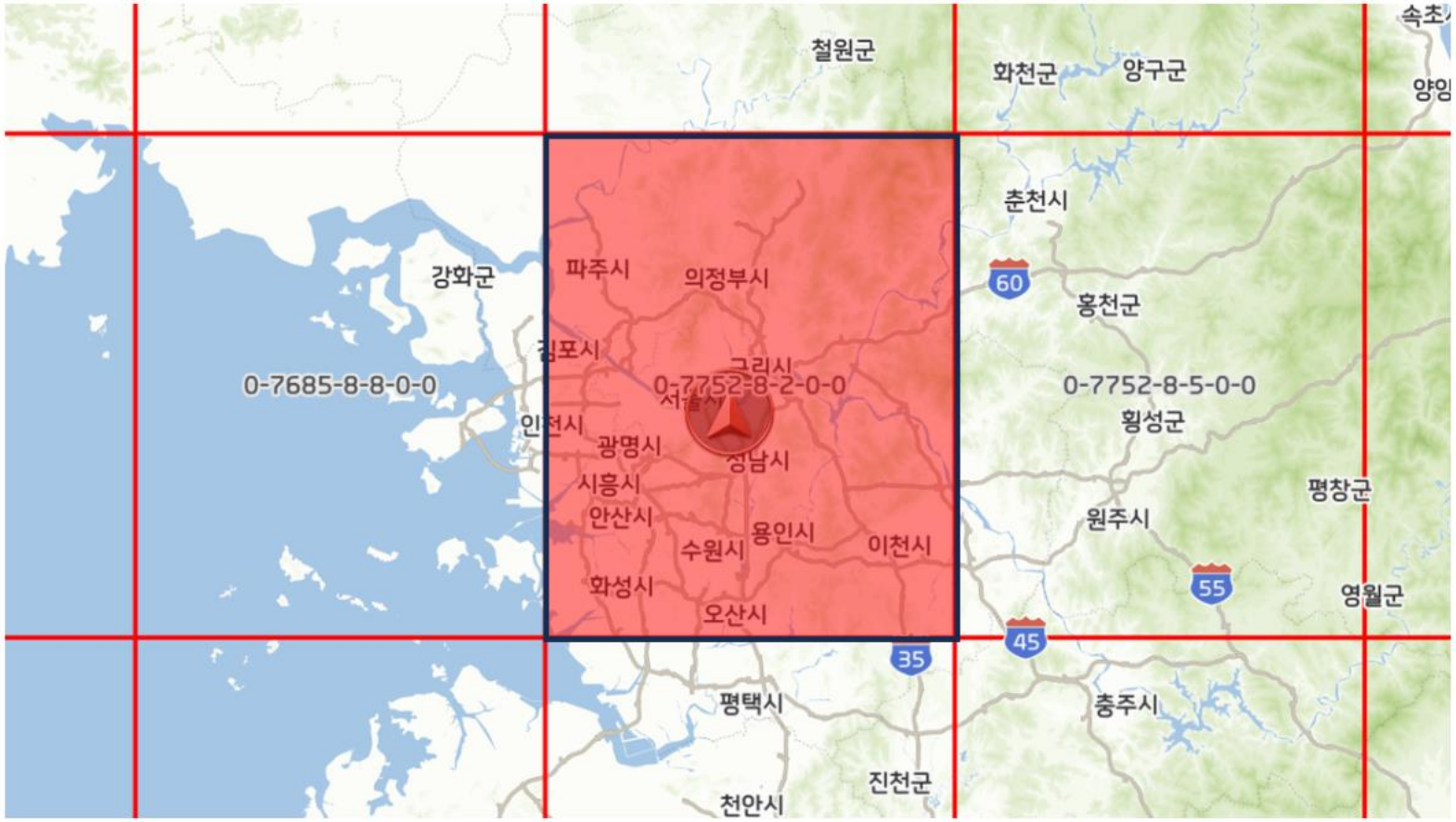
2. Tile-Based Rendering

- 2) 카메라 범위의 LonLat에 자전축(origin)을 더해 도분초 단위로 정규화함. (0.01초 단위 사용)
  - 아래 이미지는 자전축 위치를 나타냄

- ※ Red Line: 등간격 고정 분할 size, 지구의 경위도 정확할 3차 Mesh 크기도 등간격 분할된 사이즈. (1~8 레벨별 도엽 크기 정의)

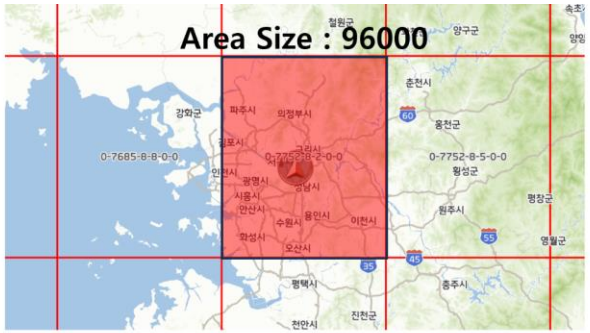
2. Tile-Based Rendering

- 3) 등간격 고정 분할 size으로 정규화된 카메라 범위를 나눔
  - Global Map에서 카메라 범위가 어디에 속하여 있는지에 대한 정보 Global Map ID 얻음.(X축 index , Y축 index)



2. Tile-Based Rendering

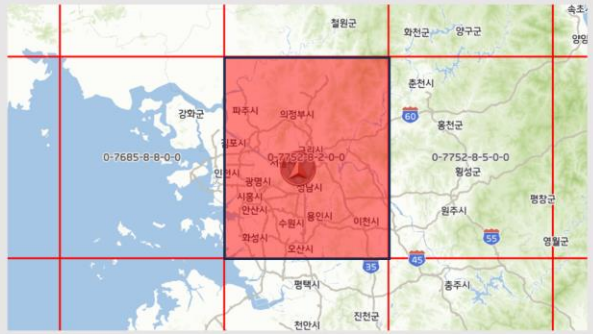
- 4) Global Map ID의 영역에서 Tile 영역을 최종적으로 구함
  - Tile 사이즈 = 등간격 고정분할 size / 분할 사이즈
  - 최종 Tile 영역 = 카메라 화면 범위 / Tile 사이즈



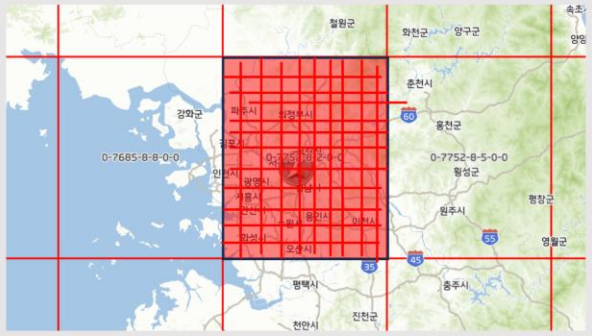
높은 축척
분할 size: 1

낮은 축척
분할 size: 300

Tile 사이즈 = 등간격 고정분할 size / 분할 사이즈(1레벨)
= 960000 / 1
= 960000



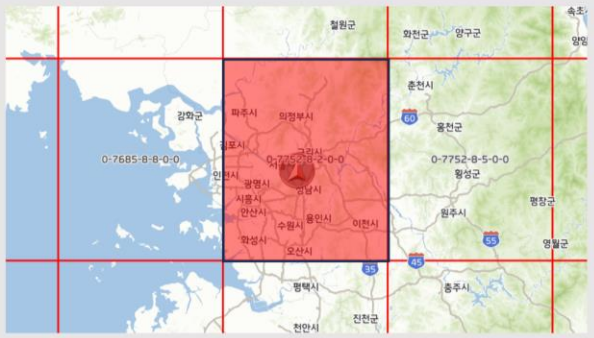
Tile 사이즈 = 등간격 고정분할 size / 분할 사이즈(18 레벨)
= 960000 / 300
= 3200



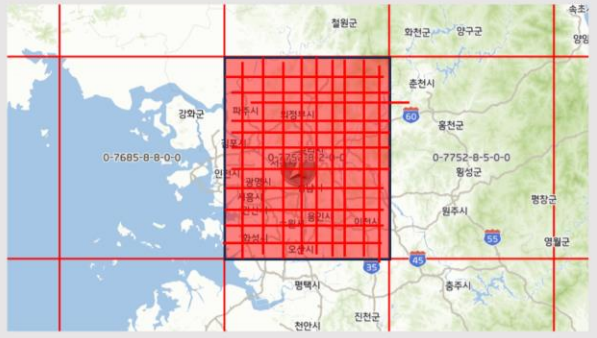
2. Tile-Based Rendering

- 4) Global Map ID의 영역에서 Tile 영역을 최종적으로 구함
  - Tile 사이즈 = 등간격 고정분할 size / 분할 사이즈
  - 최종 Tile 영역 = 카메라 화면 범위 / Tile 사이즈

Tile 사이즈 = 등간격 고정분할 size / 분할 사이즈(1레벨)
= 960000 / 1
= 960000



Tile 사이즈 = 등간격 고정분할 size / 분할 사이즈(18 레벨)
= 960000 / 300
= 3200

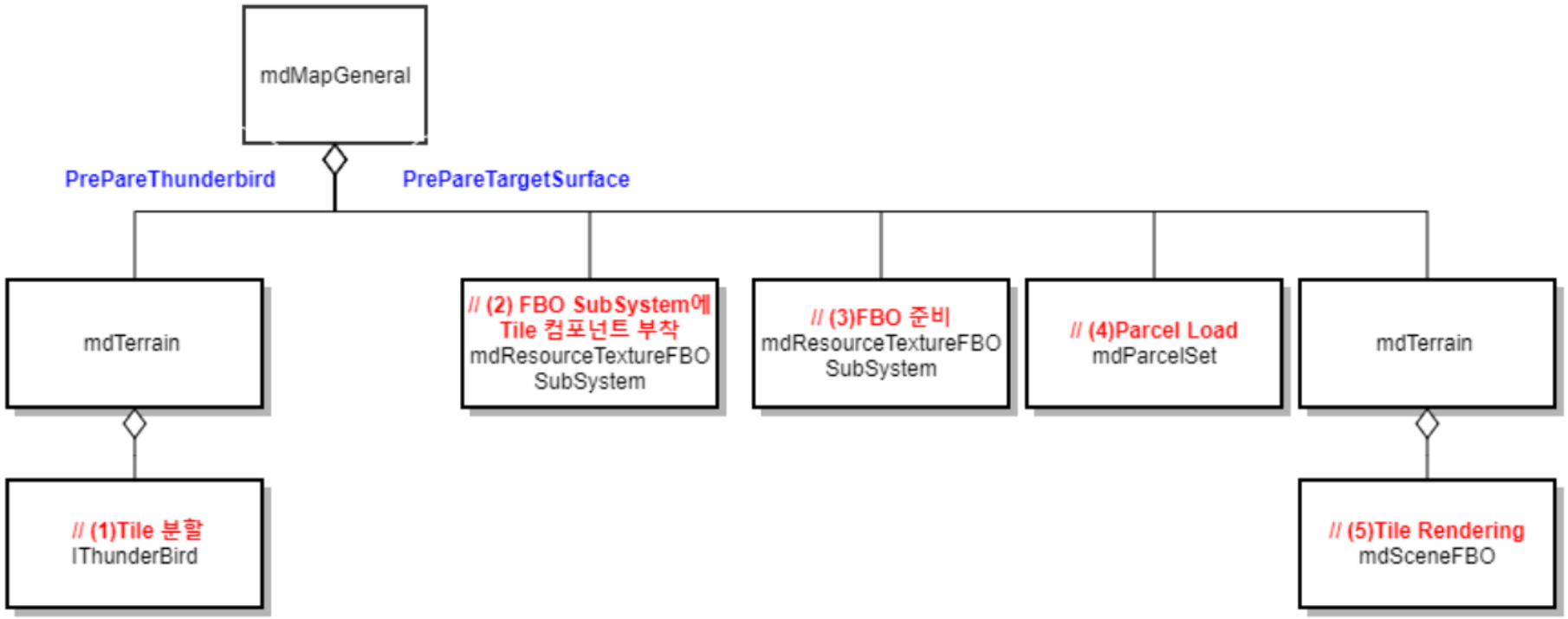


Tile 영역 = 카메라 화면 범위 / Tile 사이즈
= 안나뉨

Tile 영역 = 카메라 화면 범위 / Tile 사이즈
= 나뉨

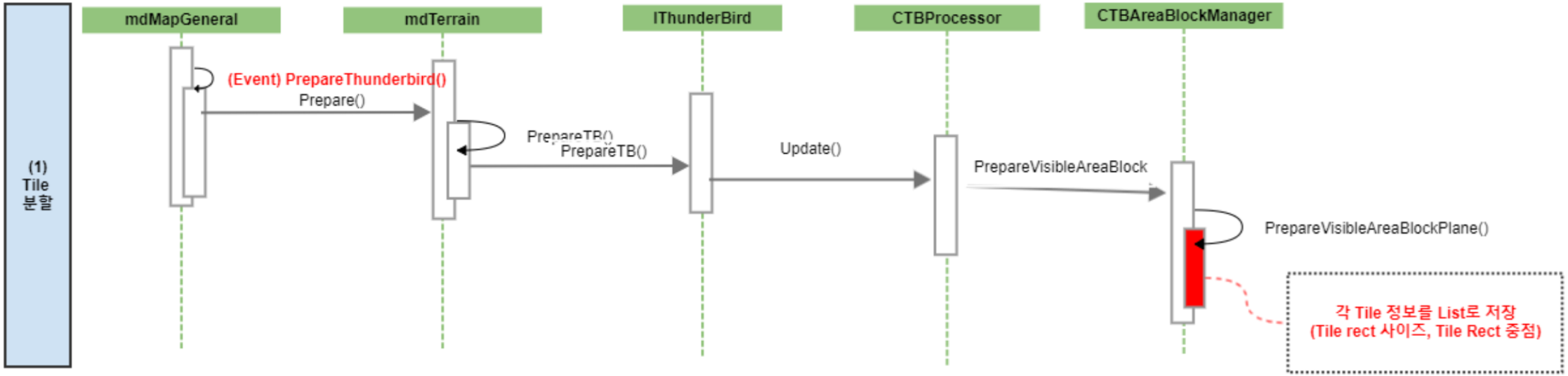
2. Tile-Based Rendering

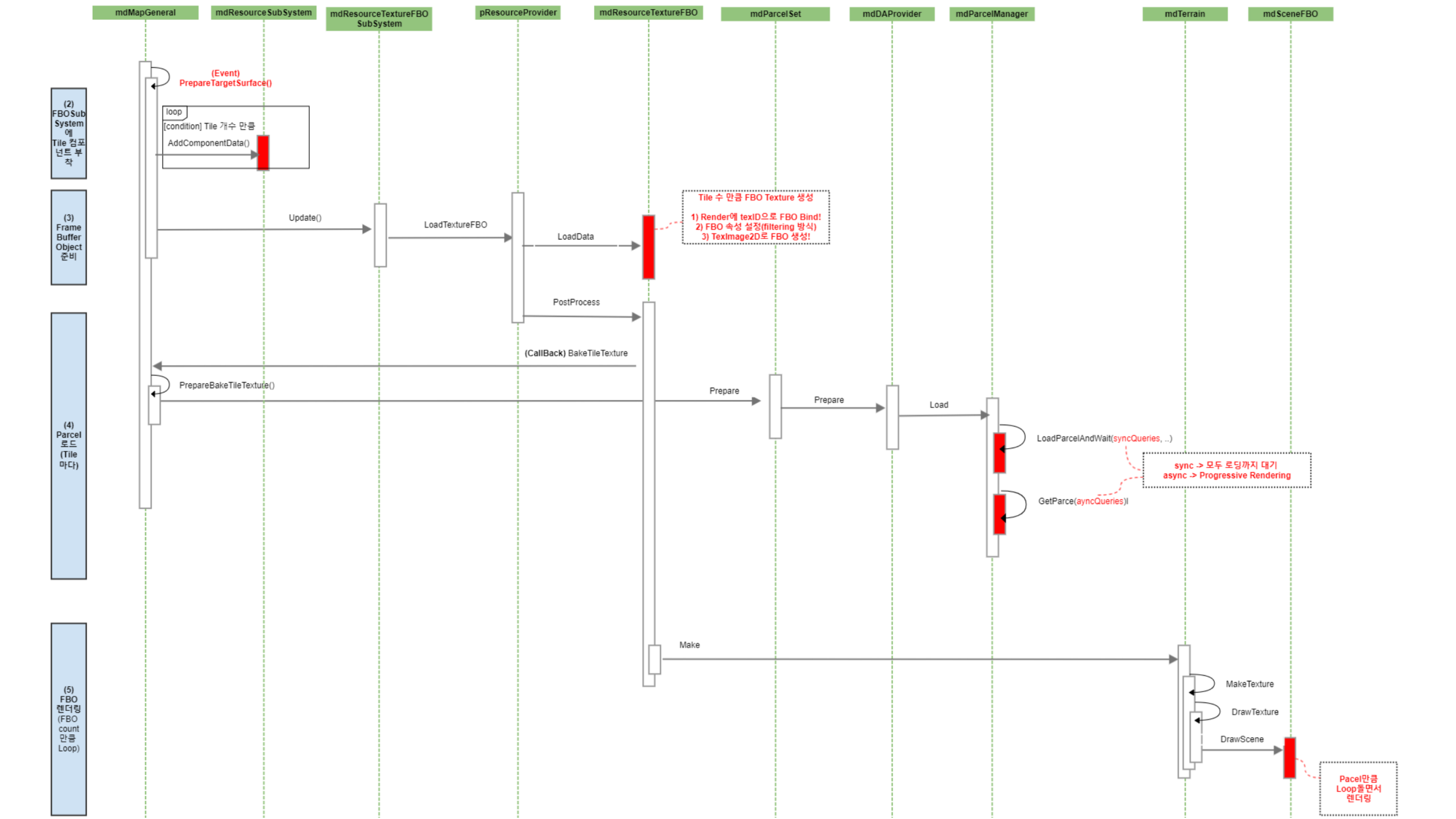
- 타일 렌더링
  - Tile 분할
  - FBO SubSystem에 Tile 개수 만큼 Tile 컴포넌트 부착
  - Tile컴포넌트 만큼 Frame Buffer Object(FBO) 준비 (opengl texture 옵션 설정 등)
  - Tile 별 Parcel 로딩
  - FBO 렌더링



2. Tile-Based Rendering

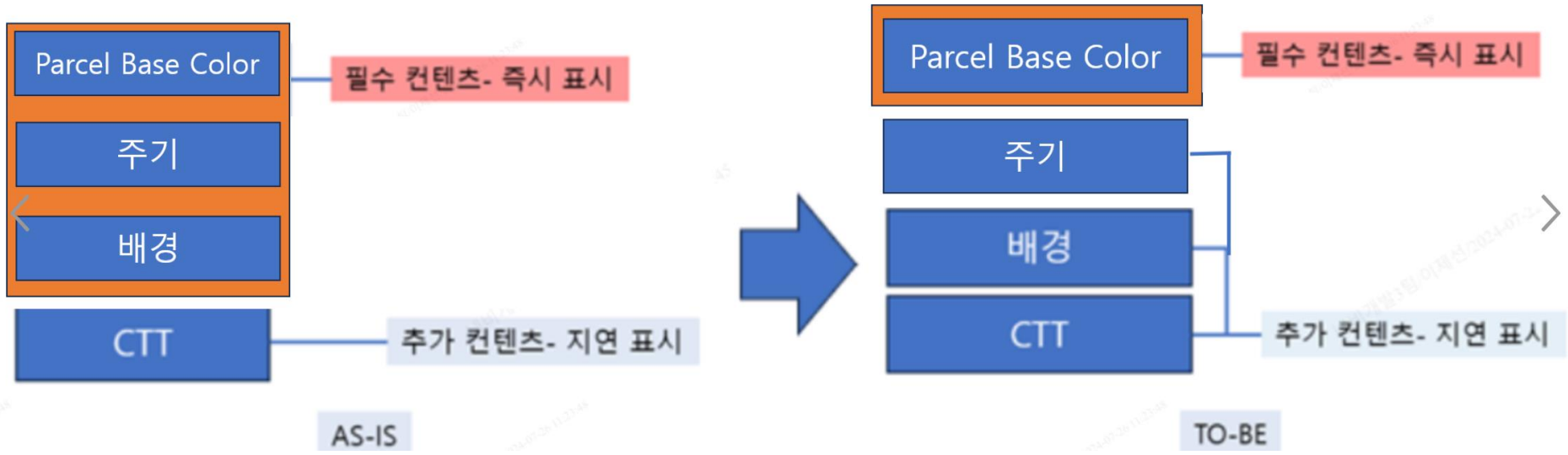
- 타일 렌더링
  - 시퀀스




mdMapGeneral
mdResourceSubSystem
mdResourceTextureFBO SubSystem
pResourceProvider
mdResourceTextureFBO
mdParcelSet
mdDAProvider
mdParcelManager
mdTerrain
mdSceneFBO
(2) FBOSub System 에 Tile 컴포넌트 부착
(Event) PrepareTargetSurface()
loop
[condition] Tile 개수 만큼
AddComponentData()
(3) Frame Buffer Object 준비
Update()
LoadTextureFBO
LoadData
Tile 수 만큼 FBO Texture 생성
1) Render에 texID으로 FBO Bind!
2) FBO 속성 설정(filtering 방식)
3) TexImage2D로 FBO 생성!
PostProcess
(4) Parcel 로드 (Tile 마다)
(CallBack) BakeTileTexture
PrepareBakeTileTexture()
Prepare
Prepare
Load
LoadParcelAndWait(syncQueries, ..)
sync -> 모두 로딩까지 대기
async -> Progressive Rendering
GetParce(ayncQueries)I
(5) FBO 렌더링 (FBO count 만큼 Loop)
Make
MakeTexture
DrawTexture
DrawScene
Pacel만큼 Loop돌면서 렌더링

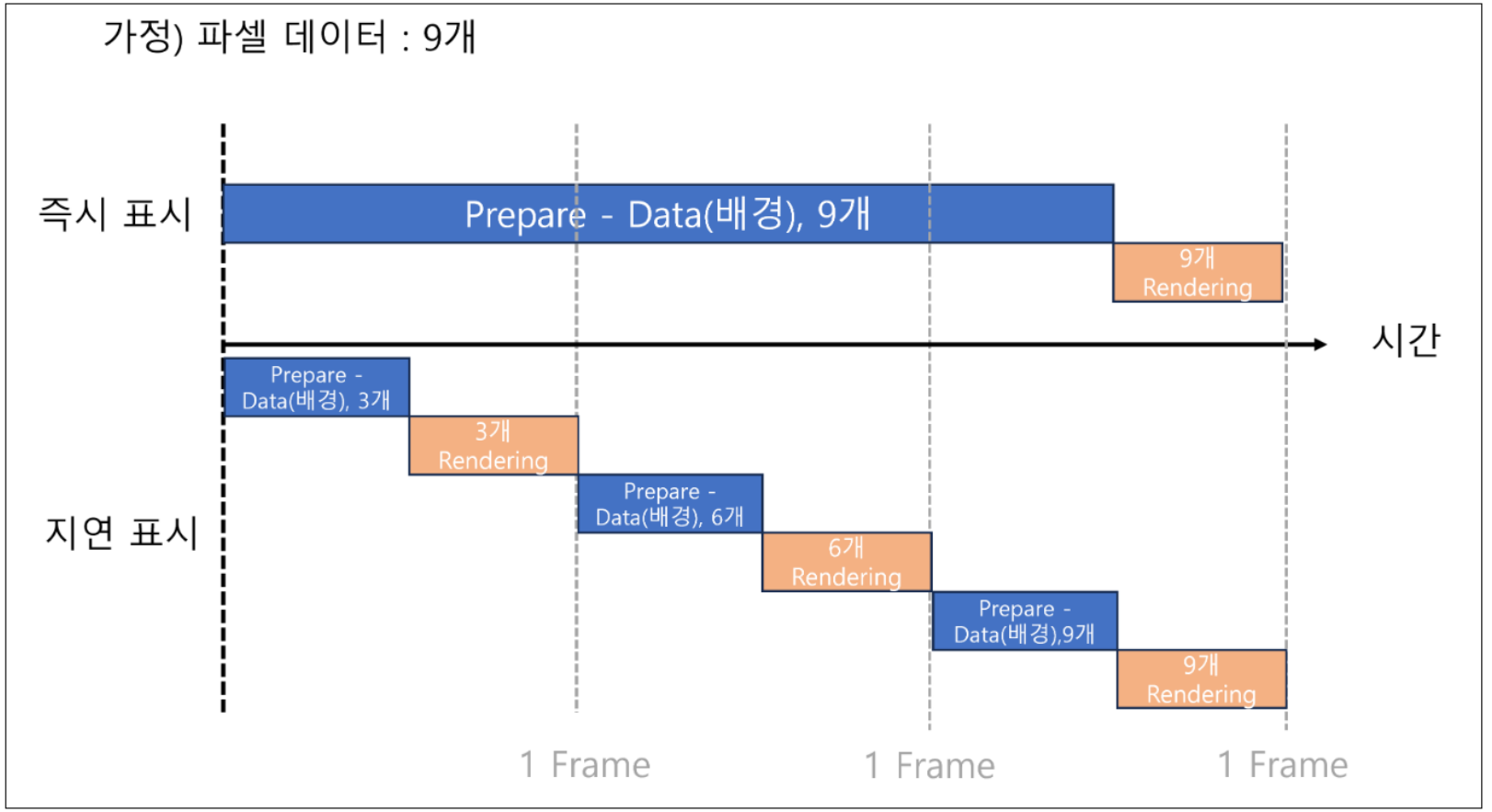
3. Our Approach

- Engine의 배경 데이터 처리
  - "즉시 표시"와 "지연 표시"로 나눠서 렌더링하고 있음
  - 성능 개선을 위해, 지연 표시의 범위를 특히 “배경, 주기” 까지 넓혀서 부하를 개선 시도



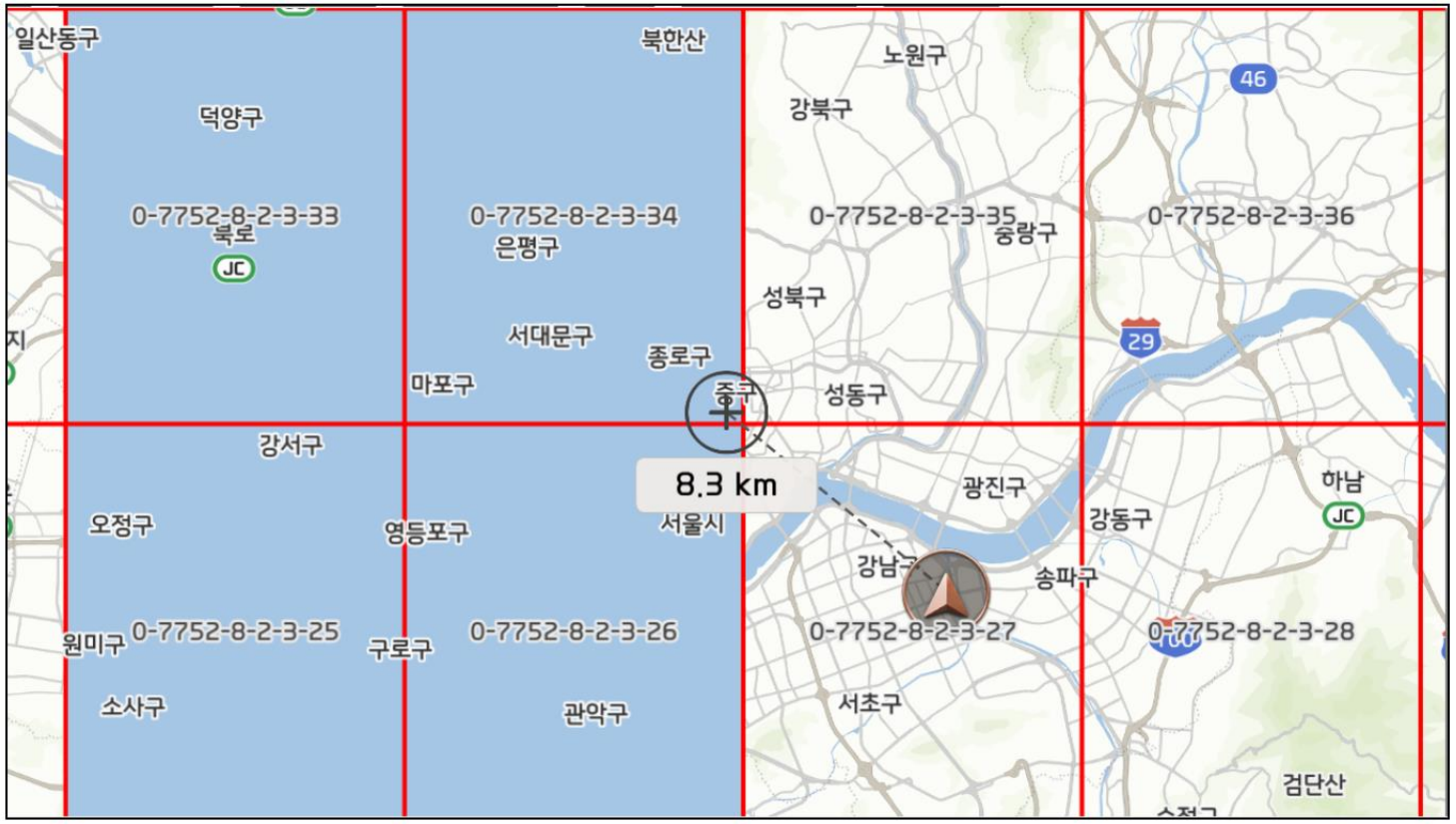
3. Our Approach

- 데이터 처리: 지연 표시
  - 로딩된 만큼 그리고, 나머지를 이후에 로딩하는 방식으로 개선



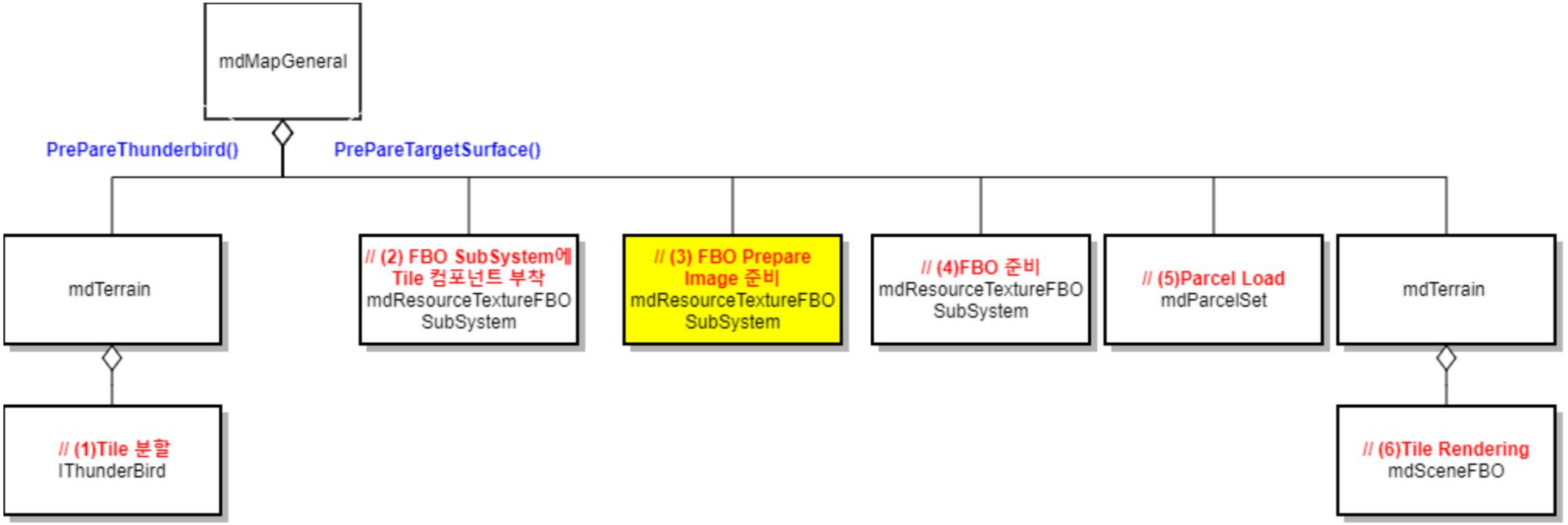
4. Problem

- (문제점 1) 로딩이 되지 않은 배경들에 대해서 빈 공백 영역(Clear color)가 보이는 문제
  - 문제 화면
    - 원인: FBO 렌더링은 배경 파셀 로딩이 완료되지 않는 경우, 표시될 배경 데이터가 없어 공백(clear color)이 보이는게 원인임
    - 개선: FBO 빈 영역을 채워줄 prepare Image를 미리 先 세팅 필요. (임의의 색상값을 사용하기에 로딩도 필요 x)



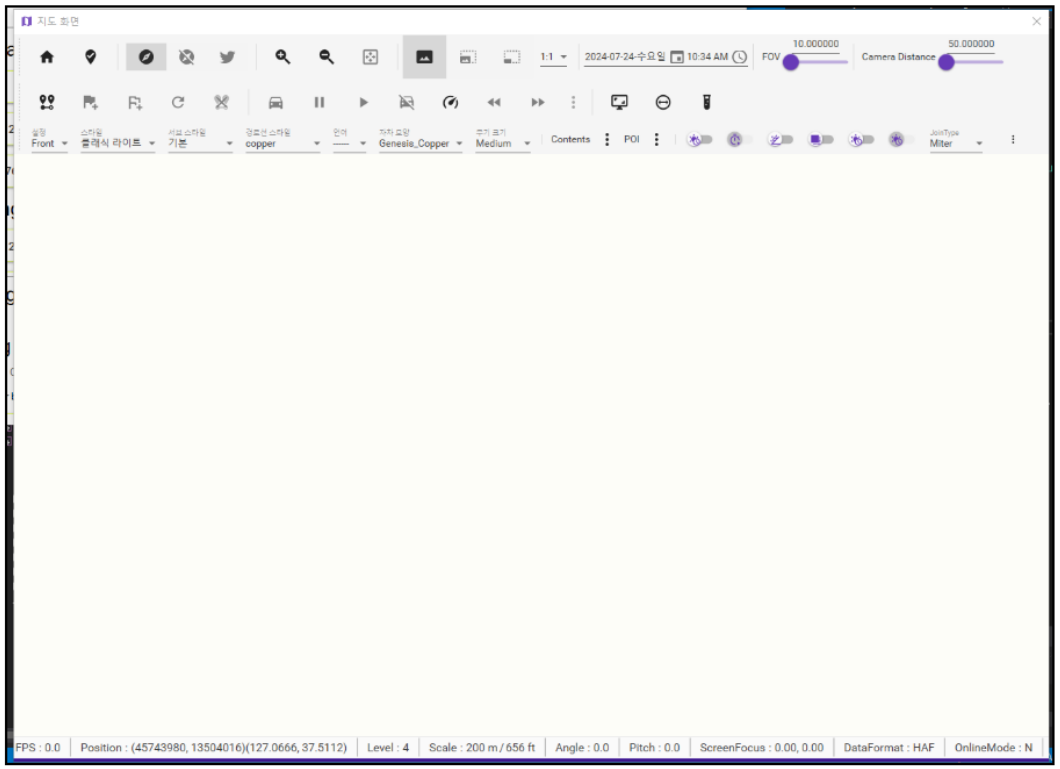
4. Problem

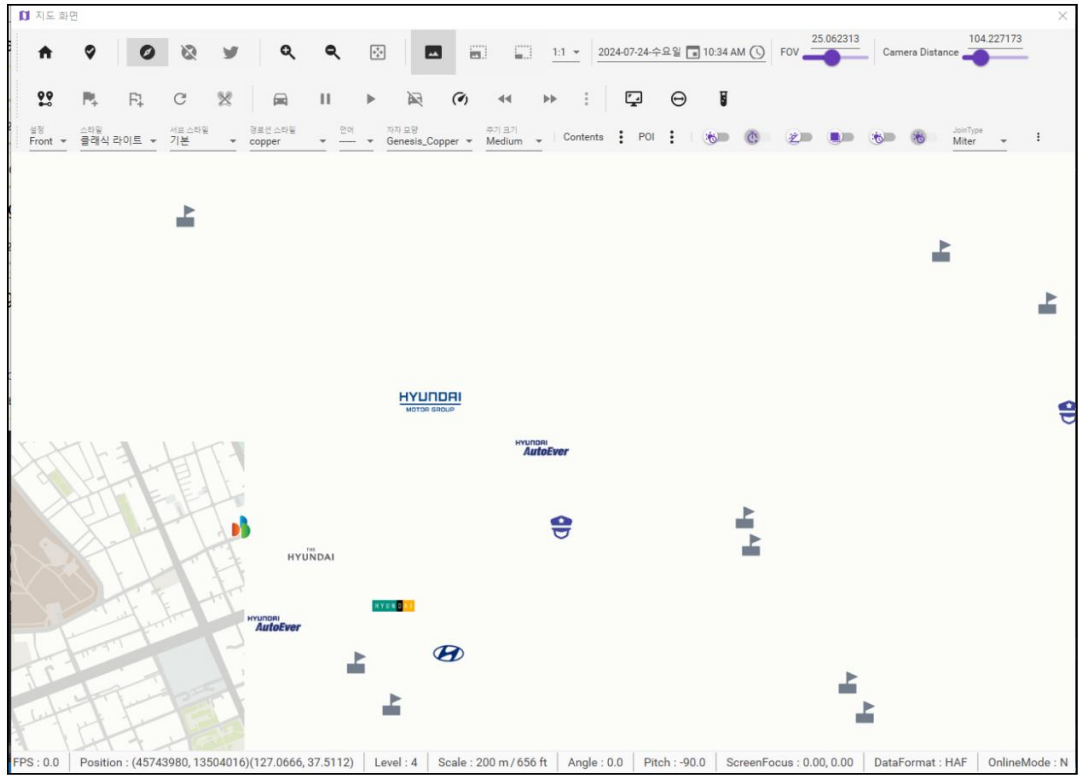
- (문제점 1) 로딩이 되지 않은 배경들에 대해서 빈 공백 영역(Clear color)가 보이는 문제



4. Problem

- 개선 결과
  - step1) Prepare Image 선 표출
  - step2) 나머지 FBO 배경이 순차적으로 출력되어 파란 공백(Clear color)이 사라짐.





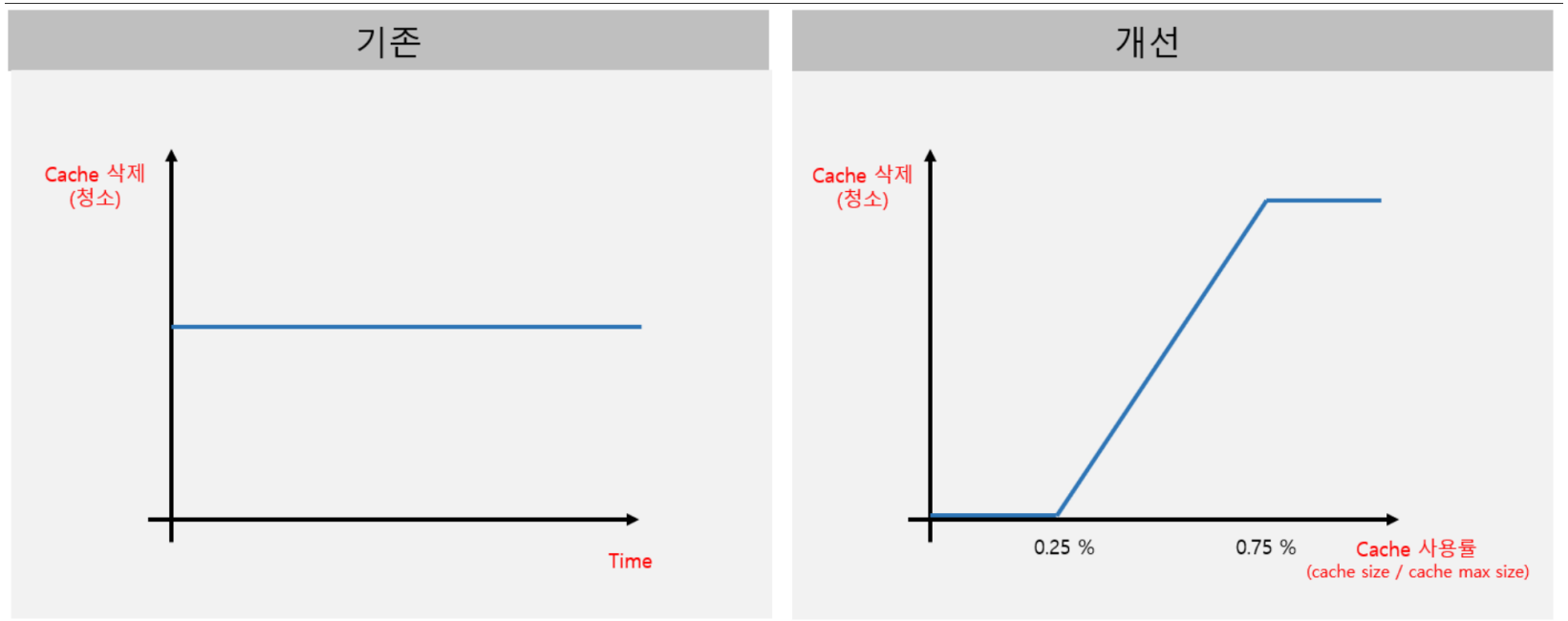
4. Problem

- (문제점 2) 축척 이동시 화면 전체가 화이트 칼라로 깜빡거리는 문제
  - 문제 영상
    - 지구본 축척 → 가장 낮은 축척으로 도착 했을 때 전체적으로 한번 깜빡 화이트 색상이 보이는 문제
    - 원인: FBO Tile Cache 에 저장되어있는 이전 축척의 Tile이 빨리 clear 됨. 따라서 축척 변경시 Tile이 새로 생성이 되어져 깜박이게 보이는 현상이 발생

4. Problem

- 개선 1) Object 마다 Cache 유지시간(cut off) 차별화 정책

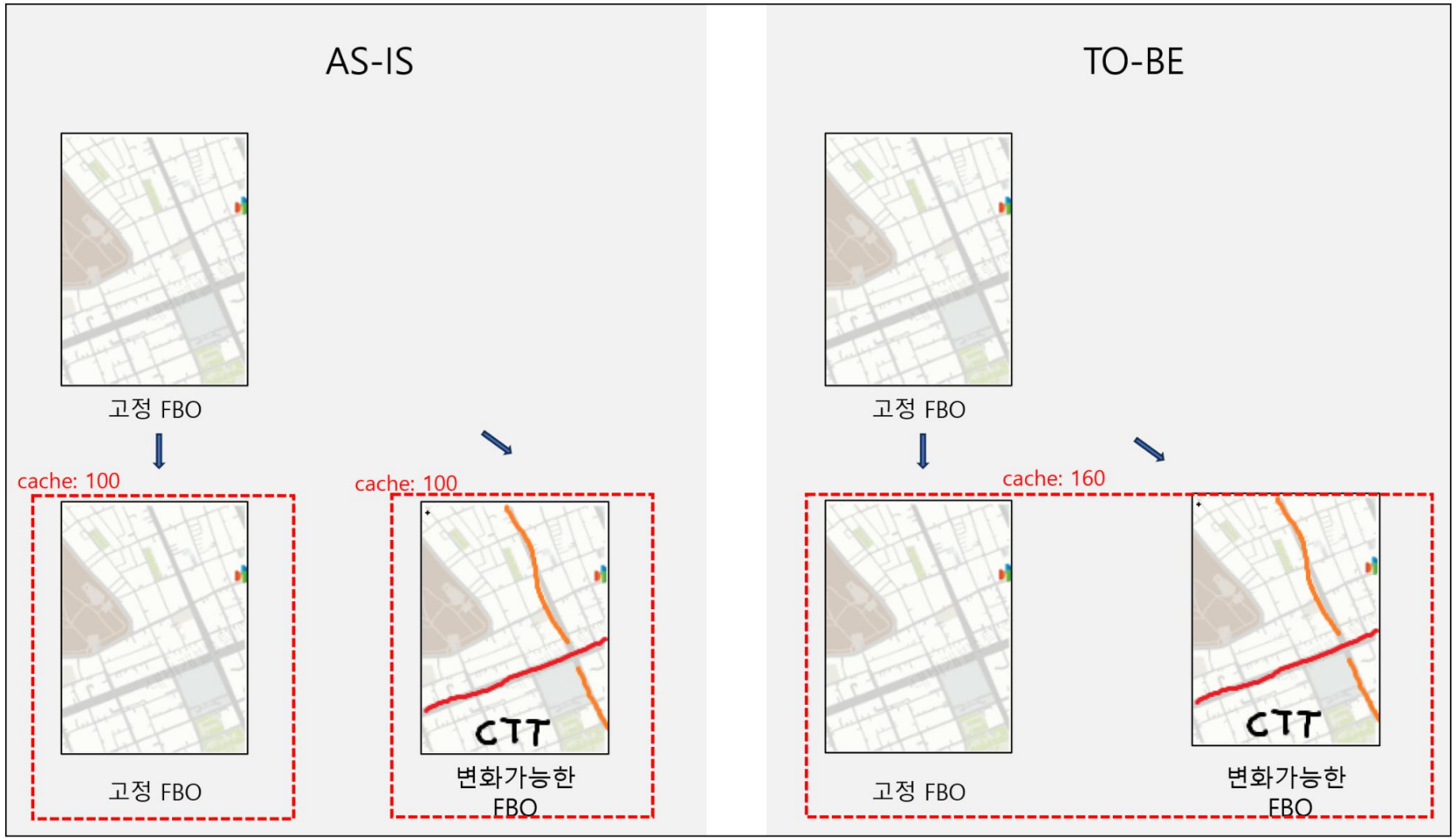
	Cache 유지 시간
기존	• **일반 Object** : 5초 유지 후 Cache에서 삭제 • **Tile FBO** : 5초 유지 후 Cache에서 삭제
개선	• **일반 Object** : 5초 유지 후 Cache에서 삭제 • **Tile FBO** : 현재 Cache 메모리 상황에 맞는 Adaptive한 메모리 관리



4. Problem

- 개선 2) Cache 메모리 통합 관리 정책

	Cache Max Size
기존	• **고정 FBO** : 100개까지 저장 • **변화가능한 FBO** : 100개까지 저장
개선	• **고정 FBO** + **변화가능한 FBO** : 160개까지 저장



4. Problem

- 개선 결과
  - Cache 메모리 개선 후 영상
  - 지구본 축척 → 가장 낮은 축척에 도달하여도 한번 깜빡이는 현상 없음

5. 총평

- 우리 내비게이션의 배경 Rendering에 대해서 알게 되어 좋았음.
- 스터디하면서 생각보다 내용이 많아서 개발하신 분들의 고생함이 느껴졌음.
- 작성한 내용 만큼 Append 가 있는데 나중에 기회가 되면 2탄도 하면 좋을것 같음.

Q n A