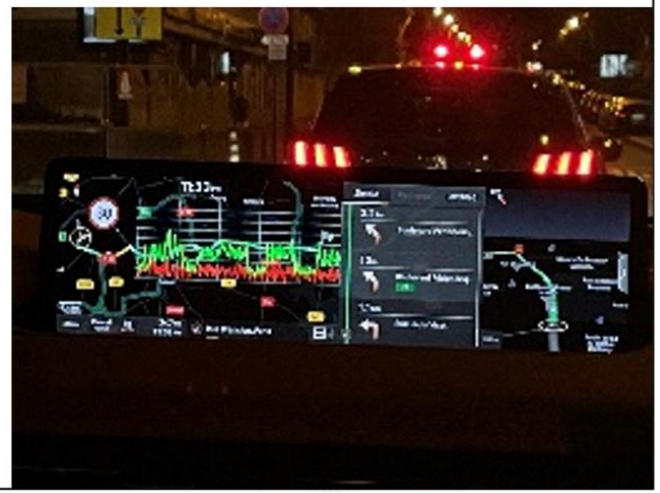
State Machine
(in Rendering Engine)

Jeseon Lee

Background

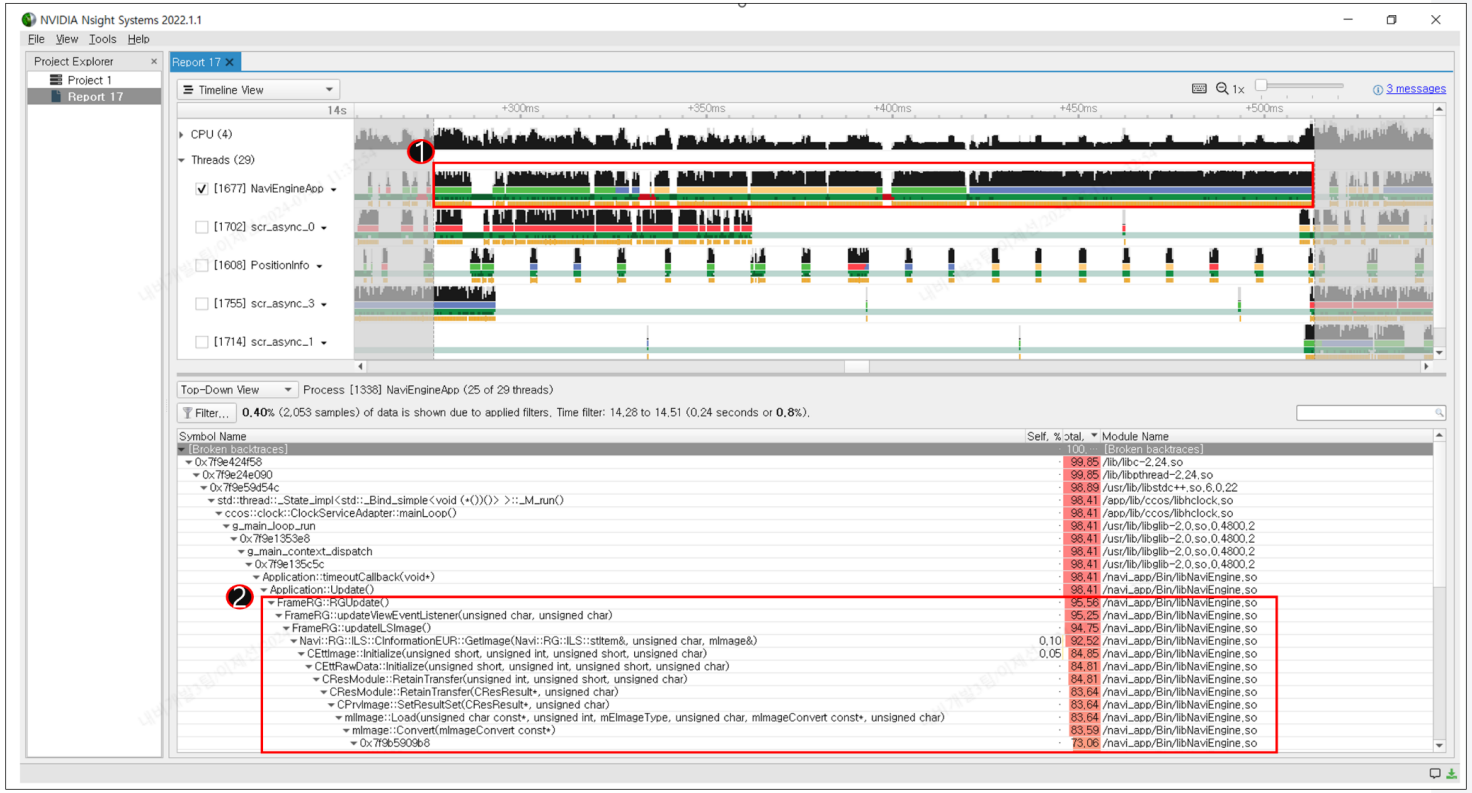
- **ILS 구조 개선(RG→ MD) 이 필요했던 이유**
  - **1) 성능 문제**
    - **1. 해외 평가에서 사용자 반응성이 사용 어려운 수준의 반응 현상 발생**

- 문제 발생
  - 러시아 22년 1월 평가. 특정 구간에서 사용 불가 수준의 반응성 문제 발생
  - 파리 현지 평가 중 해당 현상 재현 시도하였으나 동일 현상 재현 불가함
  - 파리에서 유사 현상(부하량은 상대적 낮음) 발생하여 프로파일링 진행
- 분석 계획
  - 도심 주행 중 불특정 구간/시점에서 사용자 이벤트 반응성 급격히 저하. 실시간 데이터 등으로 인하여 문제 위치 특정 어려움
  - 영국 일정 취소 → 파리 도심에서 추가 모니터링
  - 이력을 실시간으로 파악 불가 → 문제 파악을 위한 기능 개발 → 내비 점유율 그래프로 출력(향후 고도화 필요)
- 분석
  - 시스템 전체 CPU 사용량이 95% 이상 유지. 내비 동작(스크롤, 화면 전환)에 필요한 자원 점유 불가
  - 다양한 메인쓰레드 점유 동작 발생. 부하 상황에서 지연 동작 유발
- 개선(육성)
  - 메인쓰레드 장시간 점유 동작의 분산/이동
    - ILS 이미지 생성 부하
    - 경로 정보 생성 부하
    - JCT 데이터 생성 부하
    - Turn 정보 리스트 생성 빈번
    - 주변 검색 부하
    - 센터 동작 부하
  - 장시간 동작 쓰레드 부하 개선
    - MCP 디코딩 부하 개선
  - 필수 동작에 대한 우선 순위 조정



Background

- 프로파일링하여 분석한 결과, RG로부터 ILS 이미지를 업데이트하고 표시하는 과정에서 지연이 생긴다는 것을 확인



①은 내비 엔진앱이 CPU를 사용한 시간이 많다는 것을 확인

②은 ①에서의 CPU 점유 시간 비율이 높은 API를 확인할 수 있음.

=> 결과적으로, FrameRG::RGUpdate()시 95.56%의 높은 CPU 점유 시간 비율을 가져 지연 원인이 될 수 있음.

Background

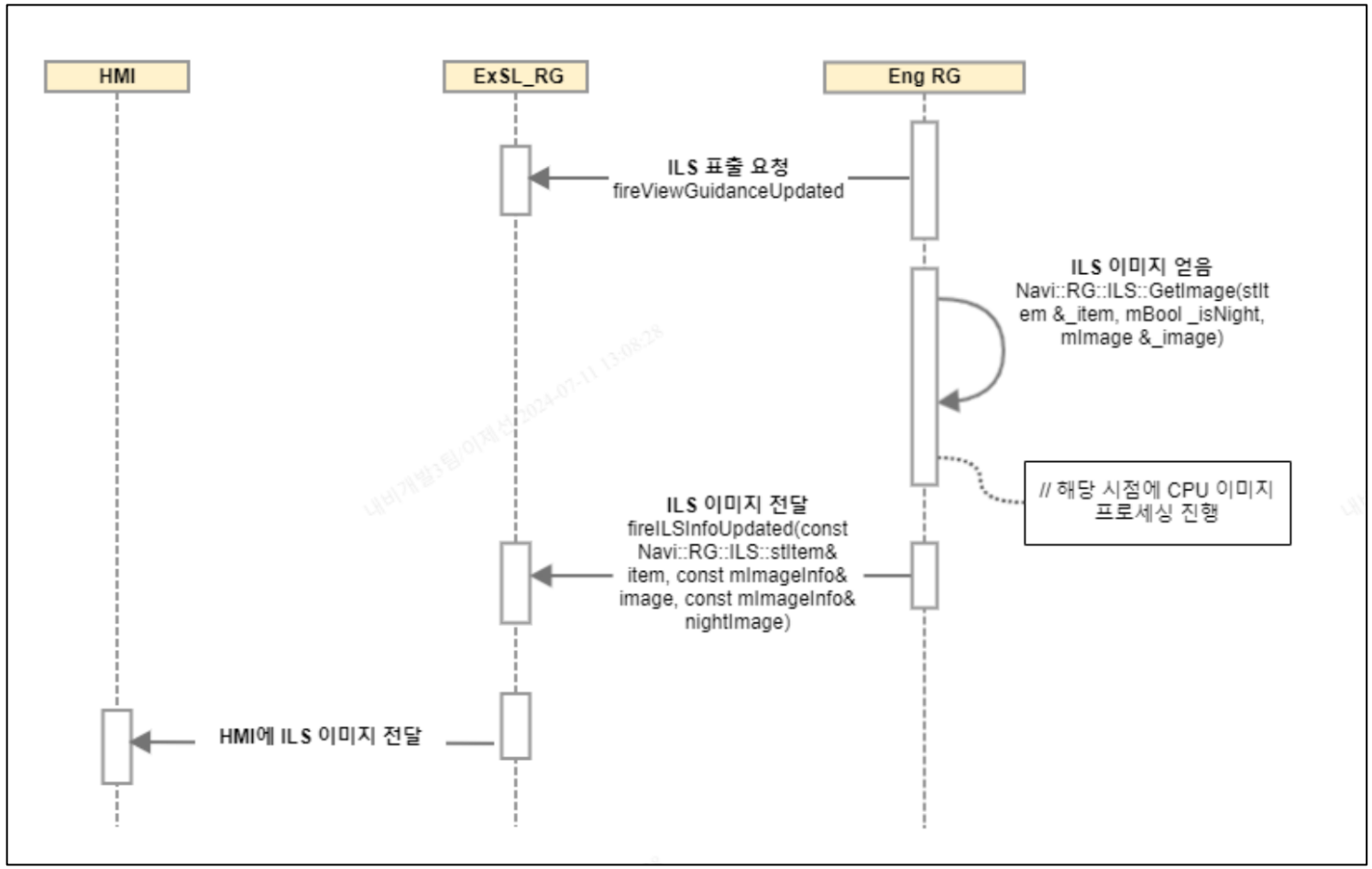
- RG의 ILS 표시 방식 확인 : 픽셀 단위 image Processing을 CPU에서 실행하여 HMI에 전달하는 방식
- 지연 timeline
  - → RG에서 ILS를 렌더링 하기 위해, CPU에서 image Processing을 진행하여 부하 발생(FrameRG::RGUpdate)
  - → Navi Engine 쓰레드의 지연이 발생
  - → 지연 시간 동안 새로운 지도 화면 갱신 수행이 불가
  - → 지도 화면 프레임 끊김 발생

- 위와 같은 성능 문제로 표시 주체 변경 필요성이 제기됨 (RG → MD)

- CPU(Image Processing) → GPU

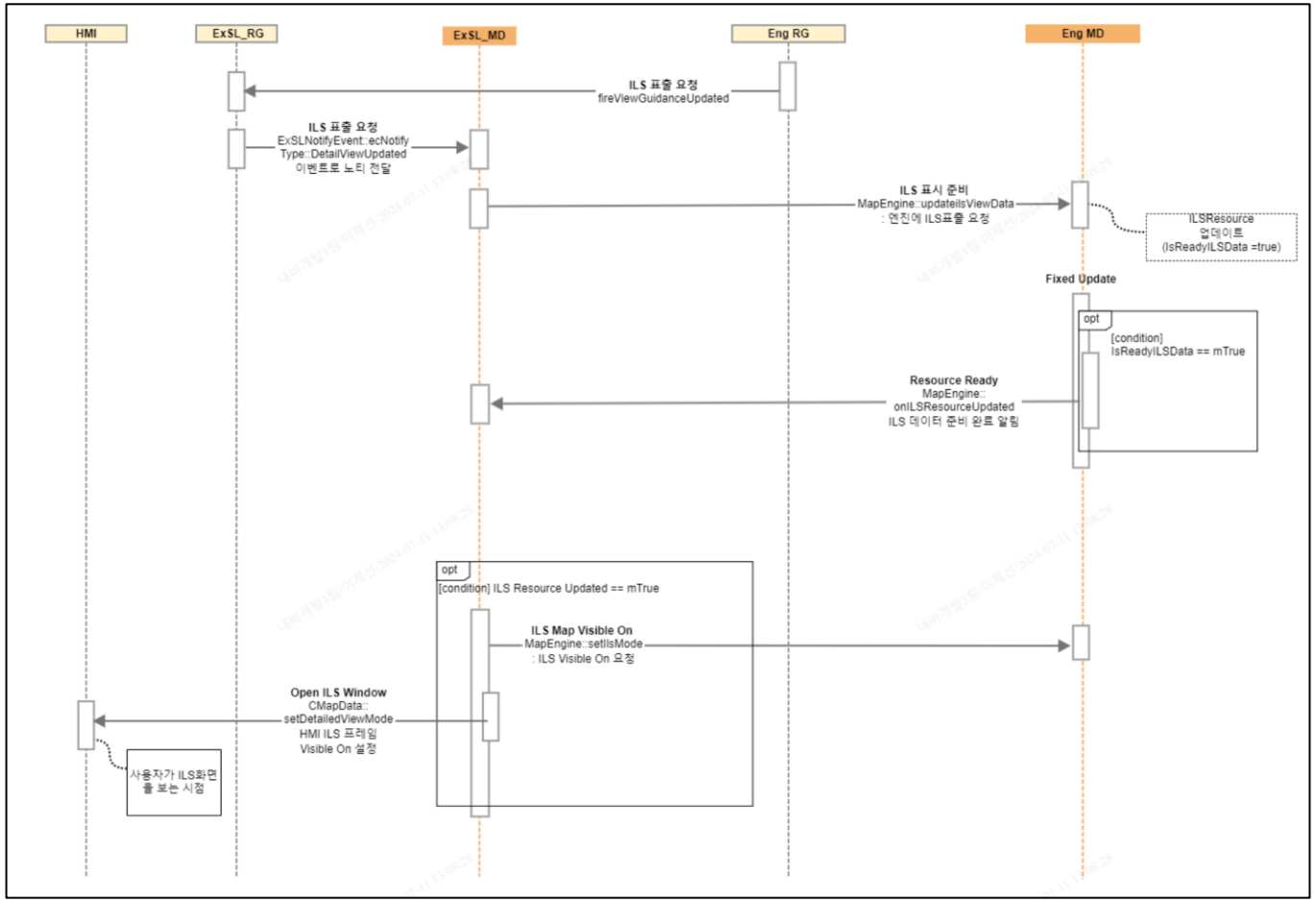
Background

- ILS 구조 개선
  - 기존 ILS
    - 표시 주체가 RG이였으므로 RG 모듈에서만 동작.



Background

- 구조 개선 후 ILS 시퀀스
  - ILS 전체 시퀀스
    - (표시 주체가 MD로 바뀌었기 때문에) RG에서 표출 요청을 하지만 실제 렌더링은 MD에서 진행



Problem

- 이슈: SS에 ILS / 벡터 확대도 블랙스크린

내비SW 자동화 검증 관리 (FPT ATOMP) / HAEQECT-8515

[HAEQE][국내][PreMaster] 확대도 블랙스크린 표출 또는 미표출 현상

Edit | Add comment | Assign | More | Reopen Issue

Details

Type:	Bug	Status:	CLOSED (View Workflow)
Priority:	B	Resolution:	Duplicate
Affects Version/s:	None	Fix Version/s:	N/A
Component/s:	pre_master_QE_자동화평가_Issue, ... (1)	Security Level:	Public(공개) (Everyone can see this issue)
Labels:	MD_BUG MD_F_JUNCTION MD_NEW_240221 MD_REVIEW_Z MD_X_REVIEW_1 QE_국내 ccIC_국내_PRE_MASTER_240220_DAILY_CI_VAILDATION pre_master_QE_자동화평가_Issue_auto 국내_CICD_ccIC_pre_master_일반_차종_기본기능평가		

Field Tab 결함 유형 분류

Problem

- Log 확인
  - 이슈에선 state machine 관련 로그가 출력
  - 자세히 확인해보면, "Can't change state" 이라는 로그가 출력. initial 단계에서 draw를 trigger 하고있었음. 실패하여 Rollback 하고 있음.

  - => state machine 개념을 알아야 이슈 해결이 가능.

```
159701 2024/02/21 13:07:36.333052 74.4384 42 CCIC Navi NENG 1710 log error verbose 1 E [2103][Draw : 996] Can't change state(0x01010500) from(Initial). trigger(Draw)
159702 2024/02/21 13:07:36.333120 74.4384 43 CCIC Navi NENG 1710 log error verbose 1 E [2103][Draw : 996] Can't change state(0x01010500) from(Initial). trigger(Rollback)
```

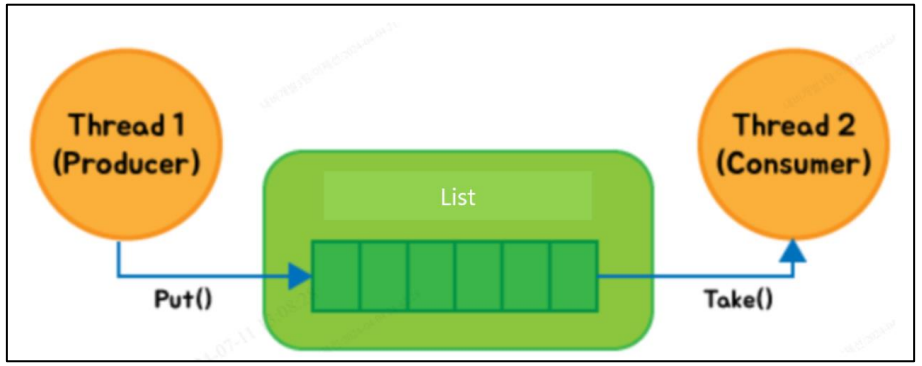
State Machine

- 생산자/소비자 패턴
- 멀티 쓰레드 환경에서 패턴 자주 사용
- 생산해내는 주체와 작업을 처리하는 주체를 분리시키는 설계 방법

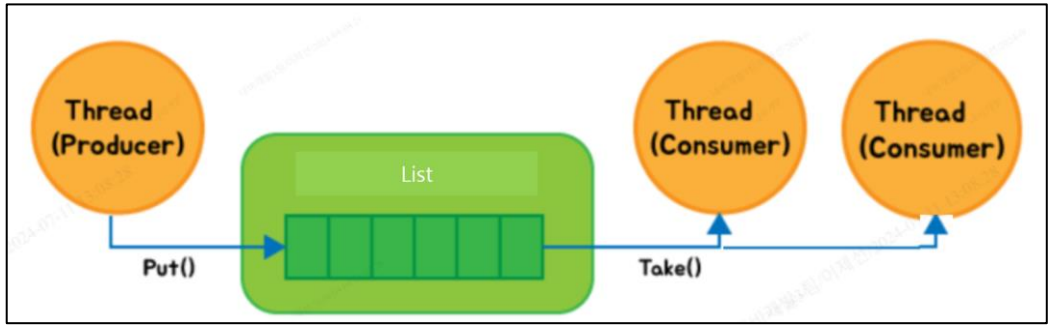
- 패턴 사용 이유
  - 성능 향상 목적, 작업을 '생성하는 부분'과 '소비하는 부분'을 분리한다면 부하를 줄일 수 있음

State Machine

- mdTaskDispatcher
- mdTaskDispatcherSingle
  - 순서가 보장되야할 때 1개의 Consumer
  - coNA의 DisplayControl과 CoreContorl 관계는 렌더링 순서가 보장되어야 하기 때문에 Single Dispatcher사용

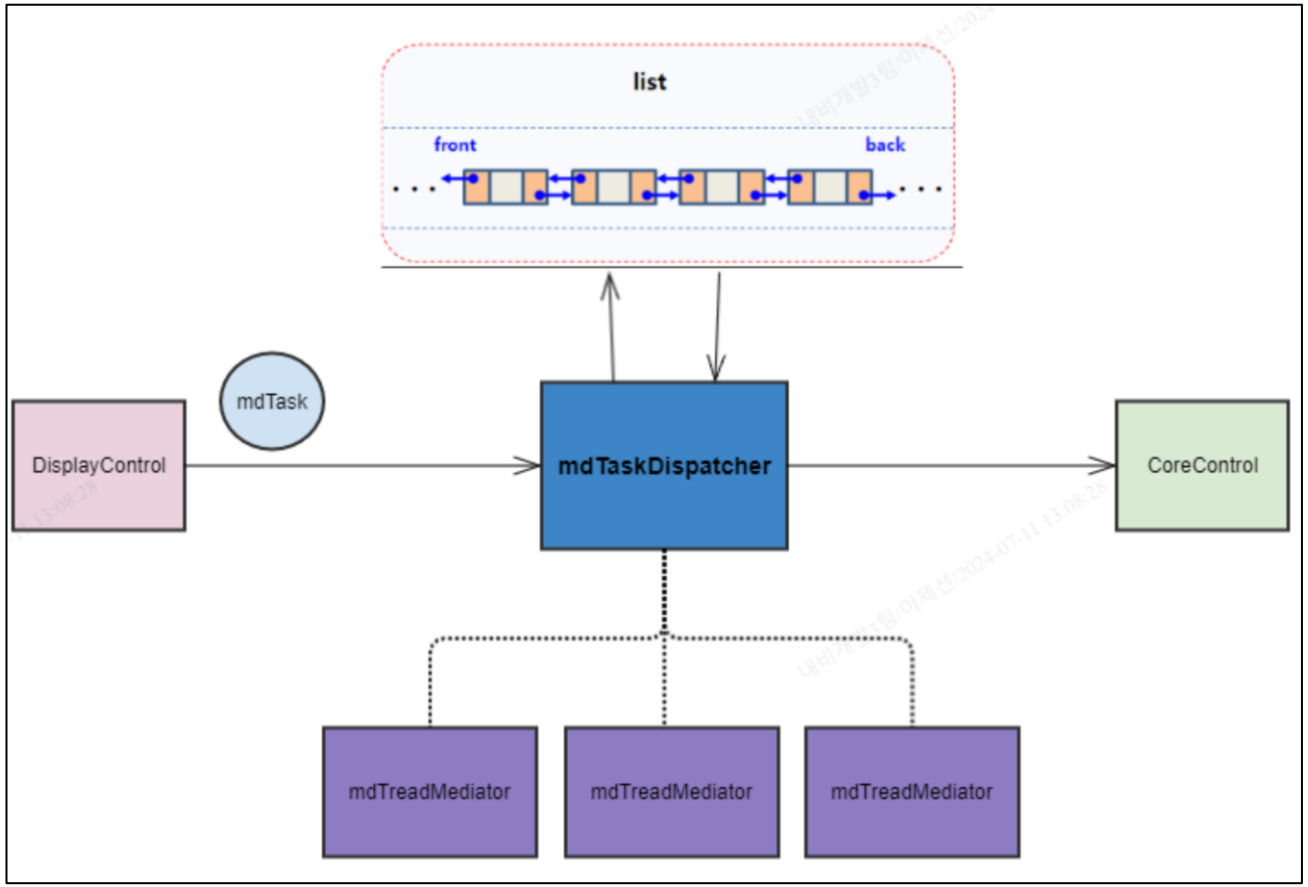


- mdTaskDispatcherParallel
  - 순서가 보장되지 않아도 될 때는 n개의 Consumer



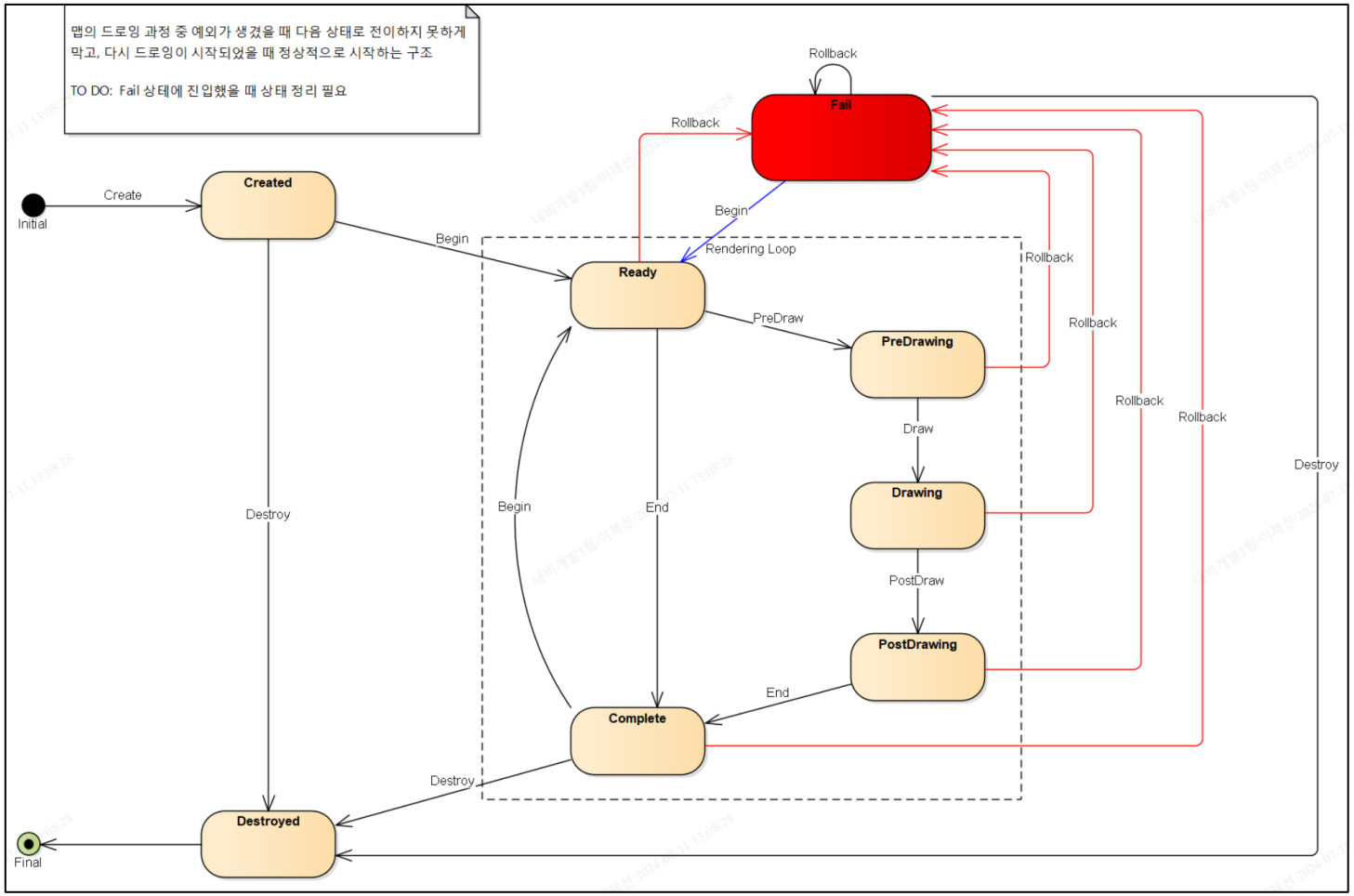
State Machine

- coNA의 생산자 /소비자
  - 메인 Thread의 Producer가 List에 push함
  - 렌더 Thread의 Consumer가 List에 저장된 Task를 꺼내서 실행



State Machine 개념

- **CoreControl - Life Cycle 과정**
  - 부팅 초기 시점에 initial함. 이때 지도 인스턴스는 렌더링 라이프 사이클을 탈 수 있게 생성
  - 특이 케이스로 실패시 Rollback되어 Fail Container로 보냄



Troble Shooting

- 위 State Machine 개념을 익혔으니,, 다시 로그를 봐보자.
  - 현재 실패 단계가 Initial 임. 즉, 부팅 초기에 실패 했다는 것을 추론 가능.
  - 그래서 초기 시점 부팅 로그를 보았더니, [MD] Error! Task Queue cleared!!! 로그가 존재.

  - [MD] Error! Task Queue cleared!!! 코드를 확인해보니 아래 이미지와 같이 task가 5000개 이상일 경우는 에러로 Task를 clear하는 로직.

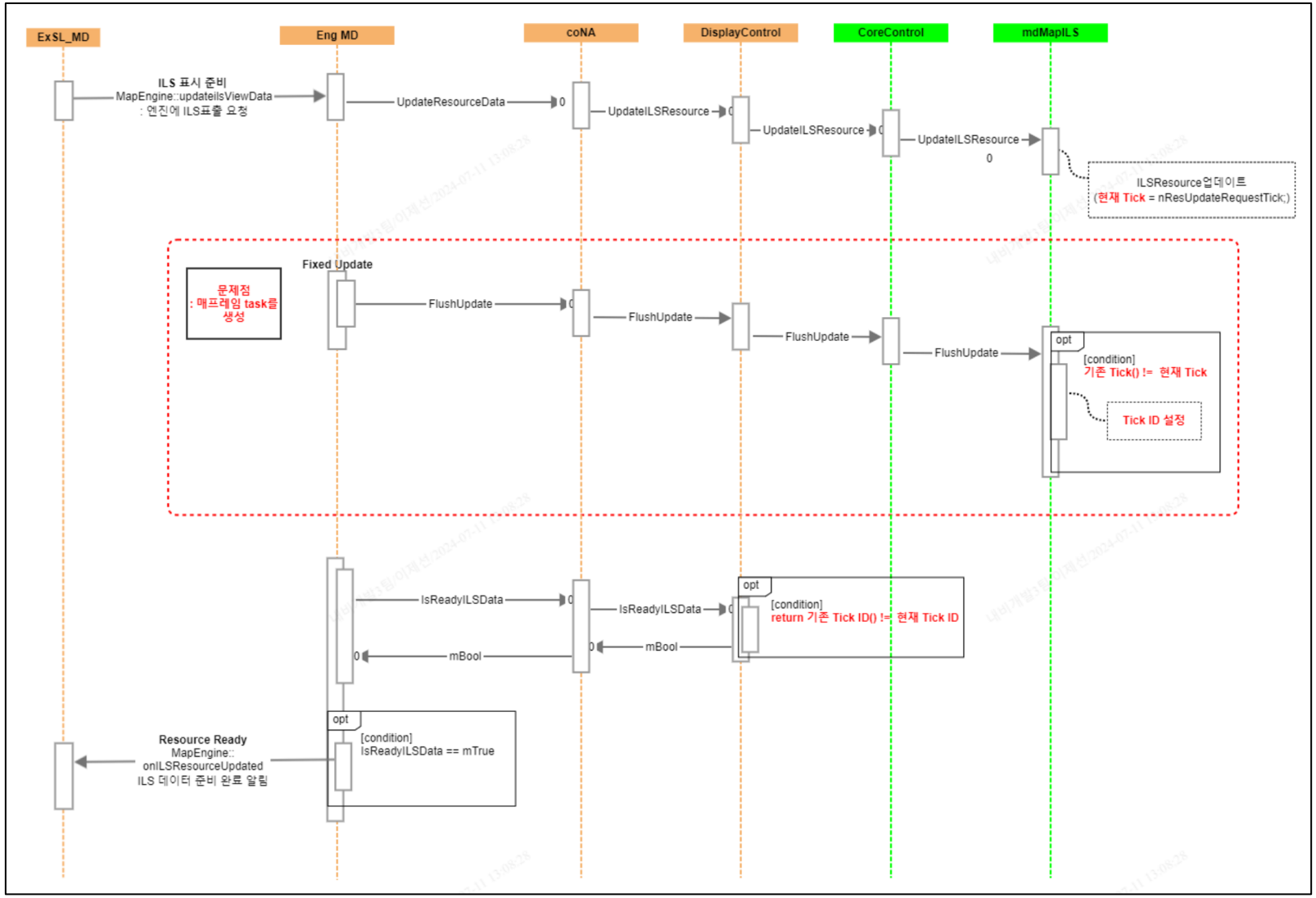
```
void mdTaskDispatcherSingle::PushTaskQueue()
{
    mdLockGuard<mdMutex> lock(m_targetMutex);
    m_targetQueue.splice(m_targetQueue.end(), m_inputQueue);

#ifdef FORCE_CUTOFF_TASK
    mDword now = mGetTickCountCoarse();
    if (m_targetLastFlushedTick < now - CUTOFFTIME && m_targetQueue.size() > CUTOFFCOUNT)
    {
        MLOG_E(M_T("[MD] Error! Task Queue cleared!!! m_targetQueue.size(%d) threadName(%s)"), m_targetQueue.size(), m_threadName);
        m_targetQueue.clear();
    }
#endif
}
```

- =====>>> 그렇다면? ILS 에서 task를 너무 많이 생성

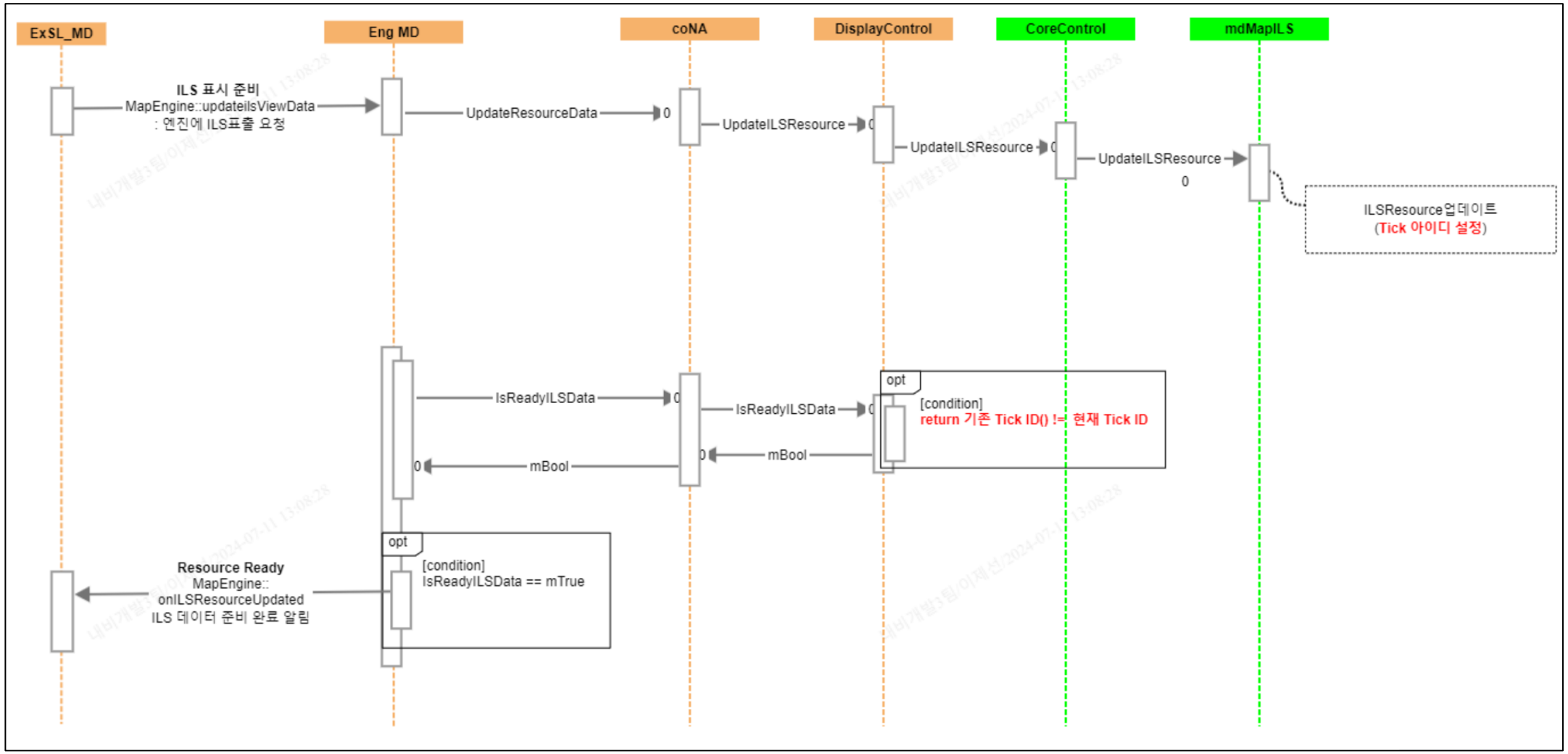
Troble Shooting

- 실제 문제 시퀀스 확인



Troble Shooting

- 실제 문제 시퀀스 확인



Result

- Rendering Engine(coNA)의 Task나 Life cycle을 제대로 이해하지 못하면 해결할 수 없는 이슈였음.

- 이번 기회로 Task 개념을 익혔고, 앞으로 블랙 out 하는 이슈가 발생한다면 Task 주변도 검토할 수 있는 시야가 트이게 되었음.

- 앞으로 이러한 세미나 공유를 통해 많이 하면 좋을 것 같음

Q n A