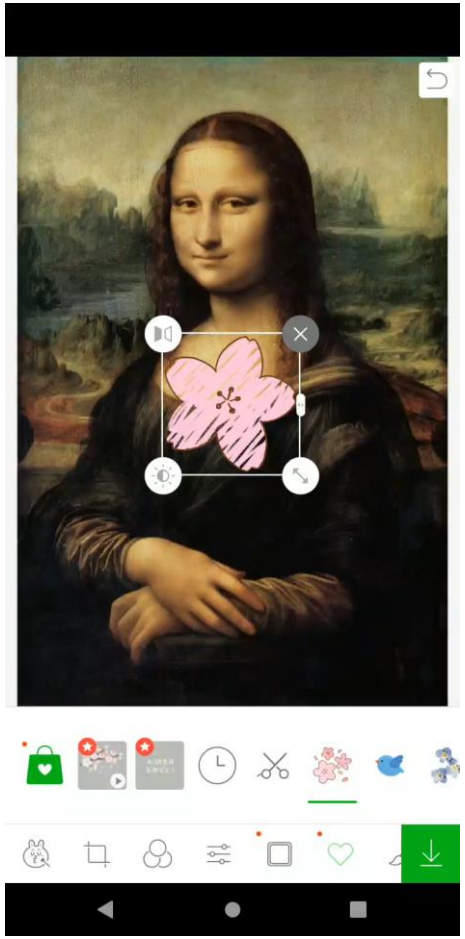
스마트폰에서 커스텀 뷰의 이동으로 발생하는 전력 소모를 줄이는 렌더링 최적화 기법

연구 동기

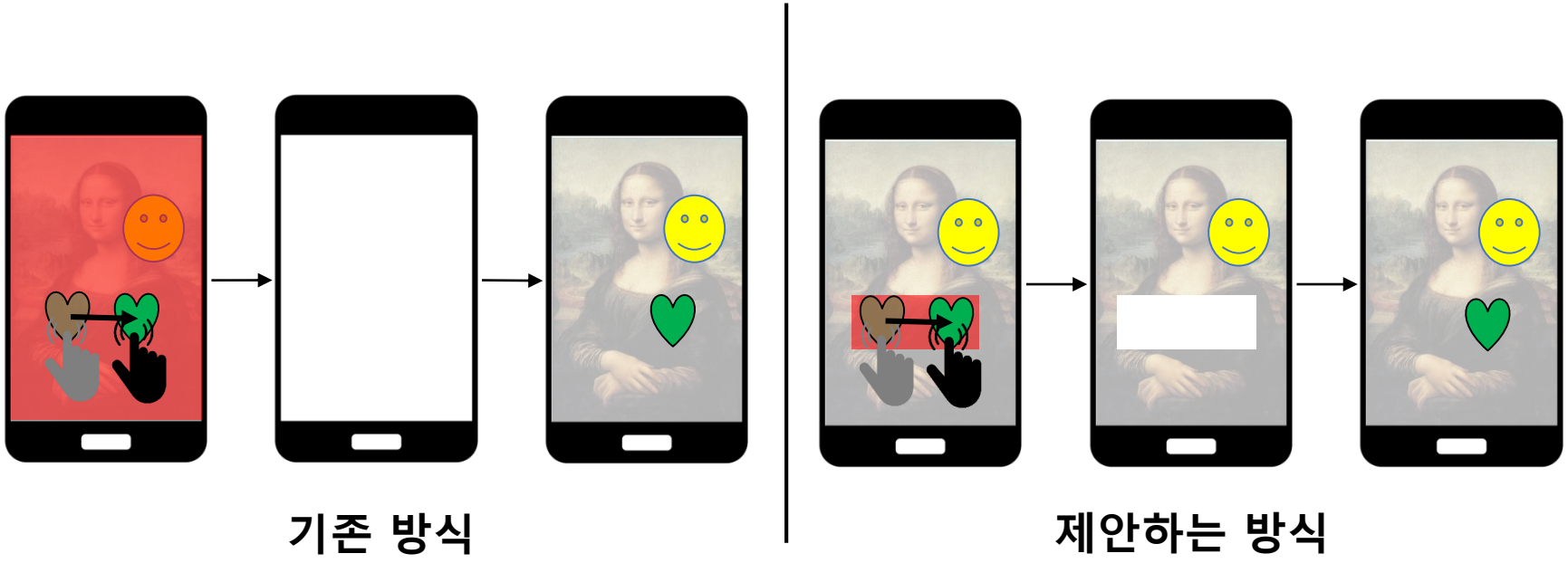
- 최근 스마트폰은 고성능의 CPU와 GPU를 탑재하여 많은 전력을 소모하고 있다.
- SNS의 대중화로 스마트폰을 이용한 사진 편집이 많아졌다.
- 사진을 편집하는 중 불필요한 Rendering이 발생한다.



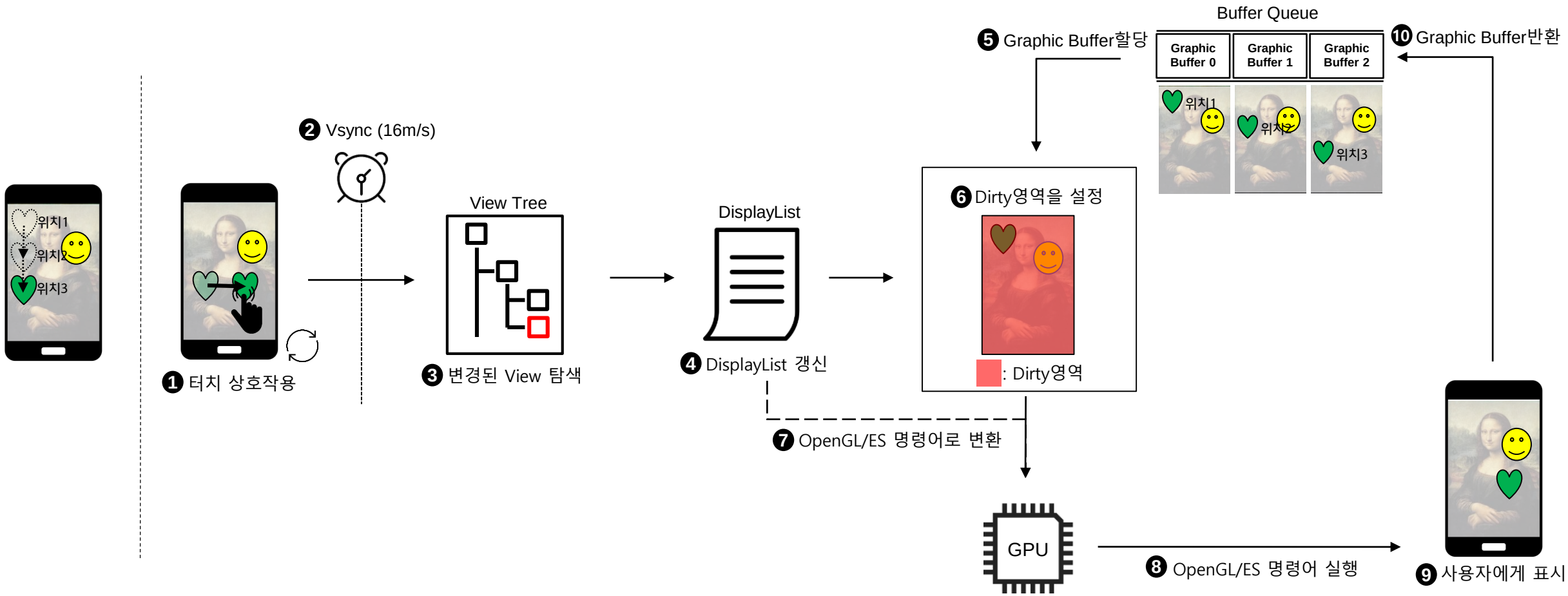
Line camera application

문제점, 해결 방안

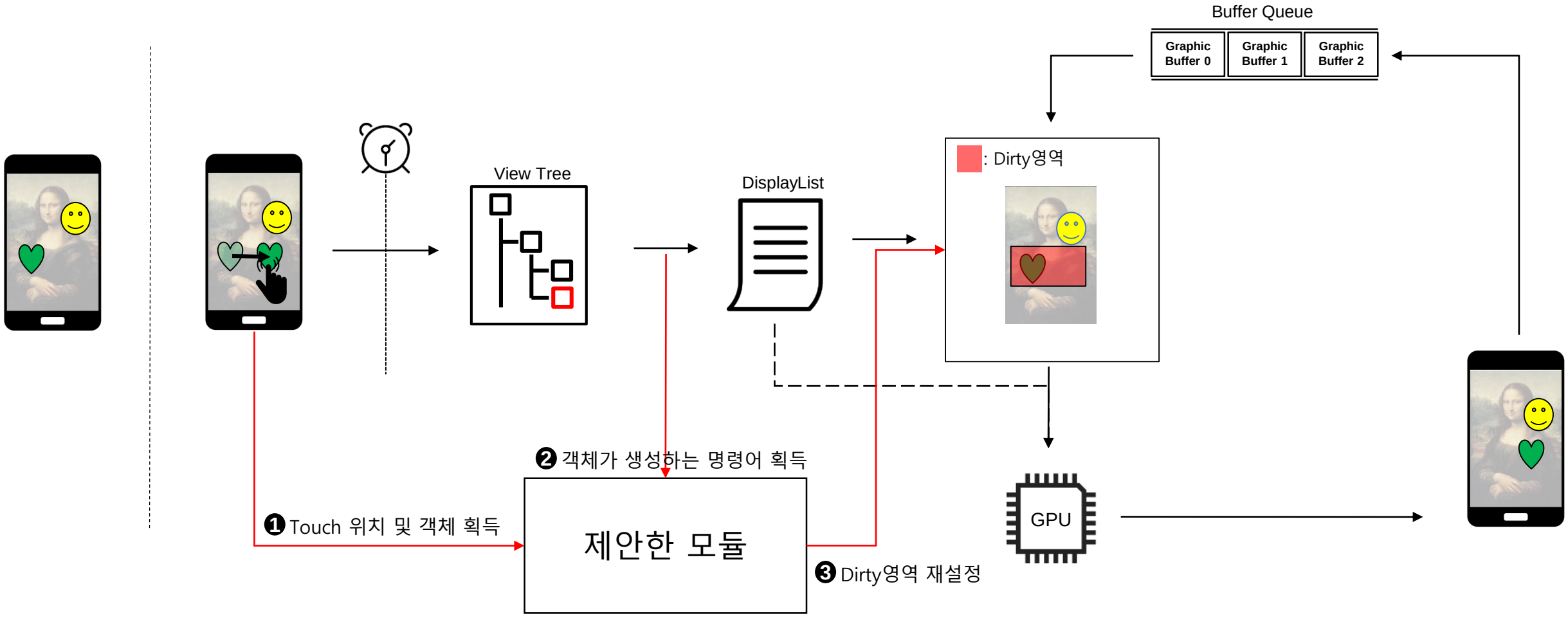
- 문제점
  - 불필요한 영역의 rendering으로 배터리 전력의 낭비가 발생
  - 사용자가 관심을 가지는 한정적인 영역에 비해 큰 rendering 영역이 설정됨
  - 시스템에서 Dirty 영역에 속한 모든 UI의 Rendering을 요구
- 해결책
  - 사용자가 관심있는 **스티커의 영역**만을 Rendering하여 **배터리 소모를 최소화**



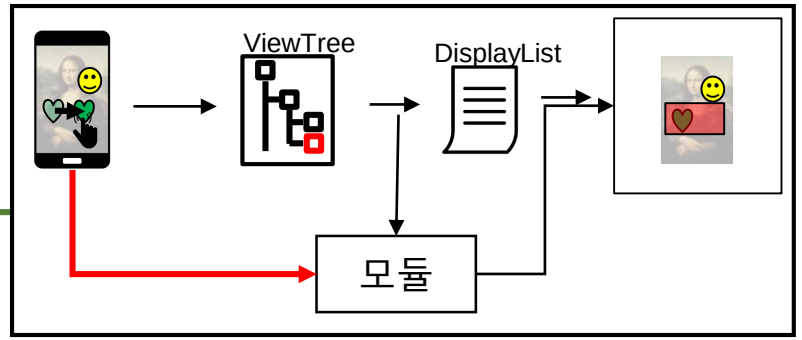
배경 지식


위치1
위치2
위치3
❶ 터치 상호작용
❷ Vsync (16m/s)
View Tree
❸ 변경된 View 탐색
DisplayList
❹ DisplayList 갱신
❺ Graphic Buffer할당
Buffer Queue
Graphic Buffer 0
Graphic Buffer 1
Graphic Buffer 2
❻ Dirty영역을 설정
: Dirty영역
❼ OpenGL/ES 명령어로 변환
GPU
❽ OpenGL/ES 명령어 실행
❾ 사용자에게 표시
❿ Graphic Buffer반환

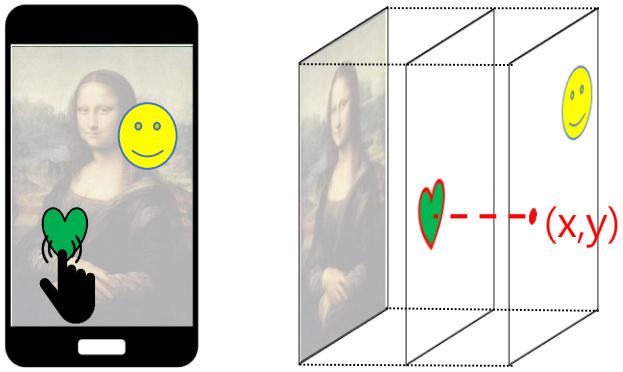
제안하는 방식


Buffer Queue
Graphic Buffer 0
Graphic Buffer 1
Graphic Buffer 2
View Tree
DisplayList
: Dirty영역
❷ 객체가 생성하는 명령어 획득
❶ Touch 위치 및 객체 획득
제안한 모듈
❸ Dirty영역 재설정
GPU

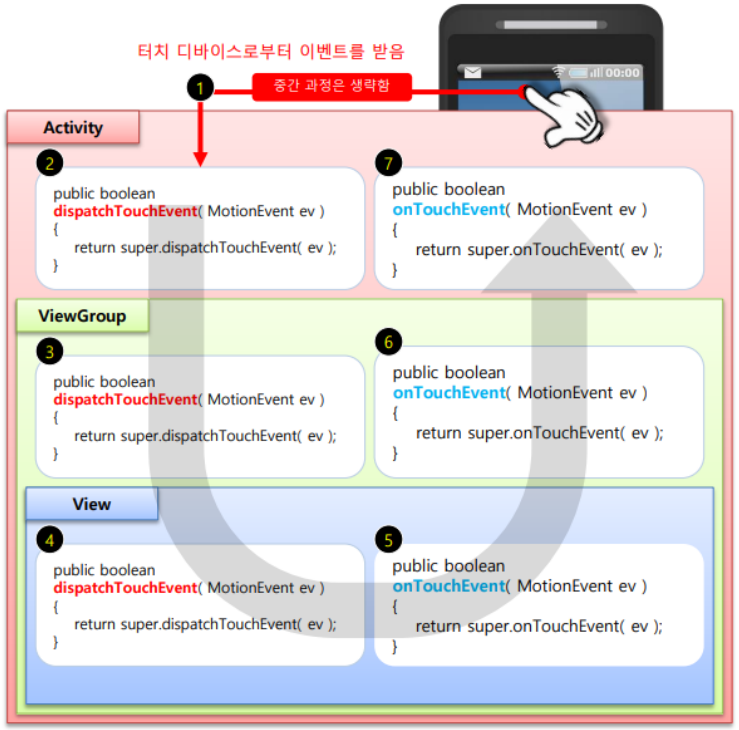
구현1



- Touch 정보 및 객체 획득
  - Touch Event가 동작하는 과정에서 방법을 착안
  - Touch의 위치 및 사용자의 Touch 상태(드래그)를 획득
  - 획득한 정보를 제안하는 모듈로 전달



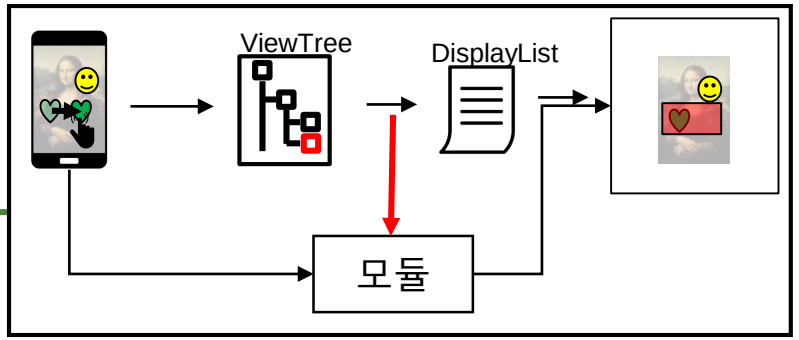
<터치 위치를 이용한 View 특정>



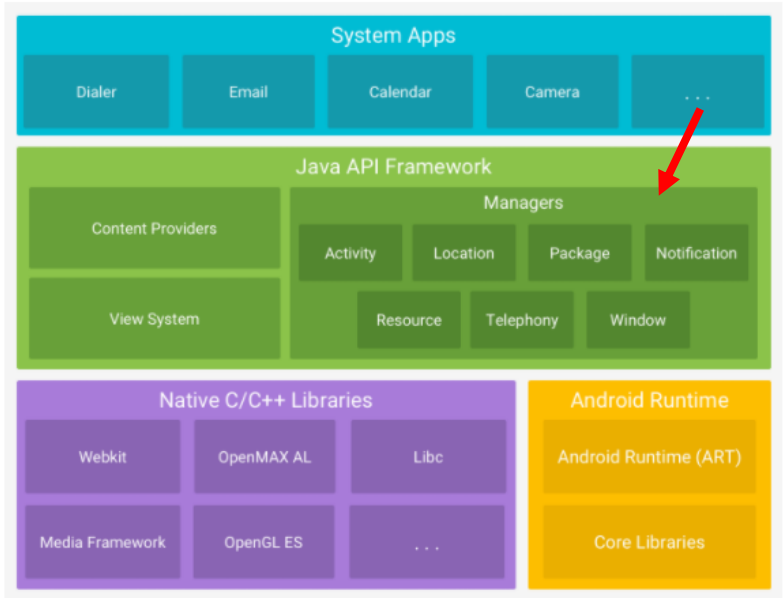
[그림] 안드로이드 Touch Event 전달 방식

출처: "이것이 안드로이드다" (책)

구현2



- 객체가 생성하는 명령어 획득
  - 애플리케이션 마다 다른 객체(View)를 사용하여 접근이 제한
  - 객체가 호출하는 Android Drawing API에서 argument를 통해 정보를 획득
  - 획득한 정보로 객체의 크기 및 기울기를 계산



drawBitmap(...)

drawRect (...)

drawPicture (...)

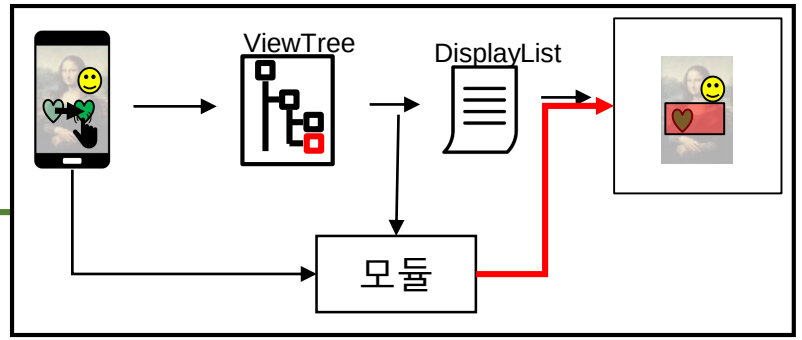
⋮

Android Drawing API

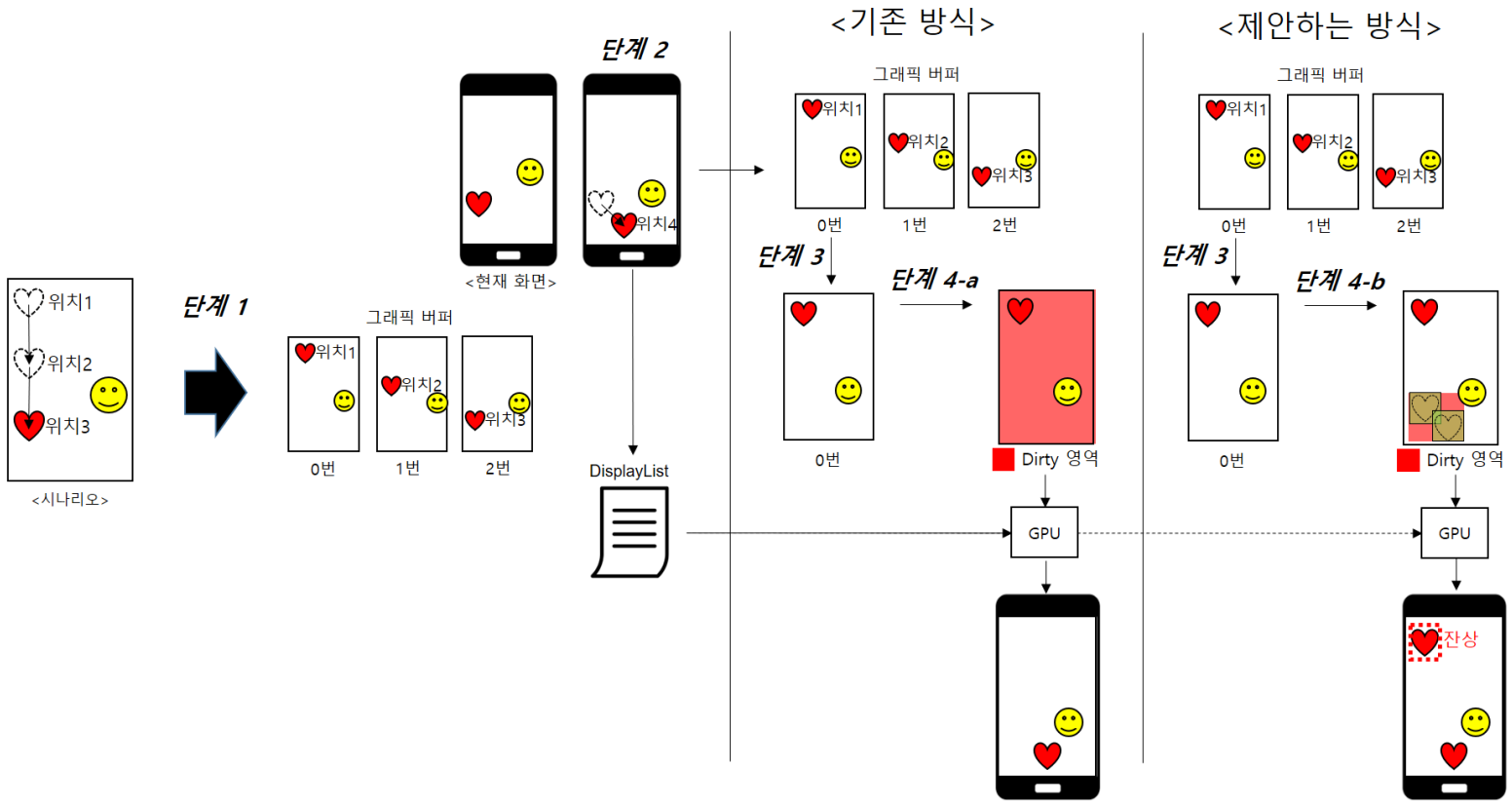
[그림] android 플랫폼 아키텍처 일부

출처: android developers

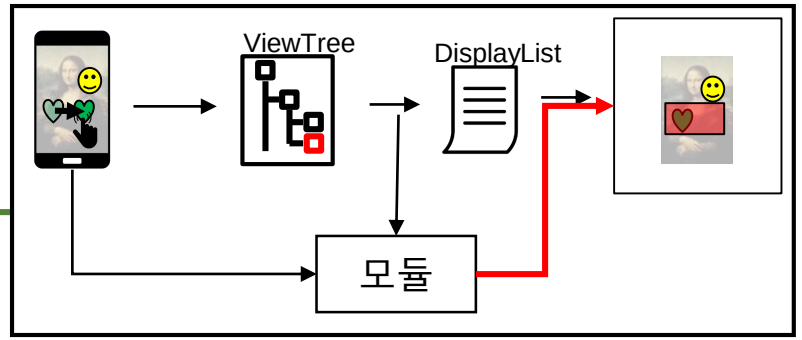
구현3



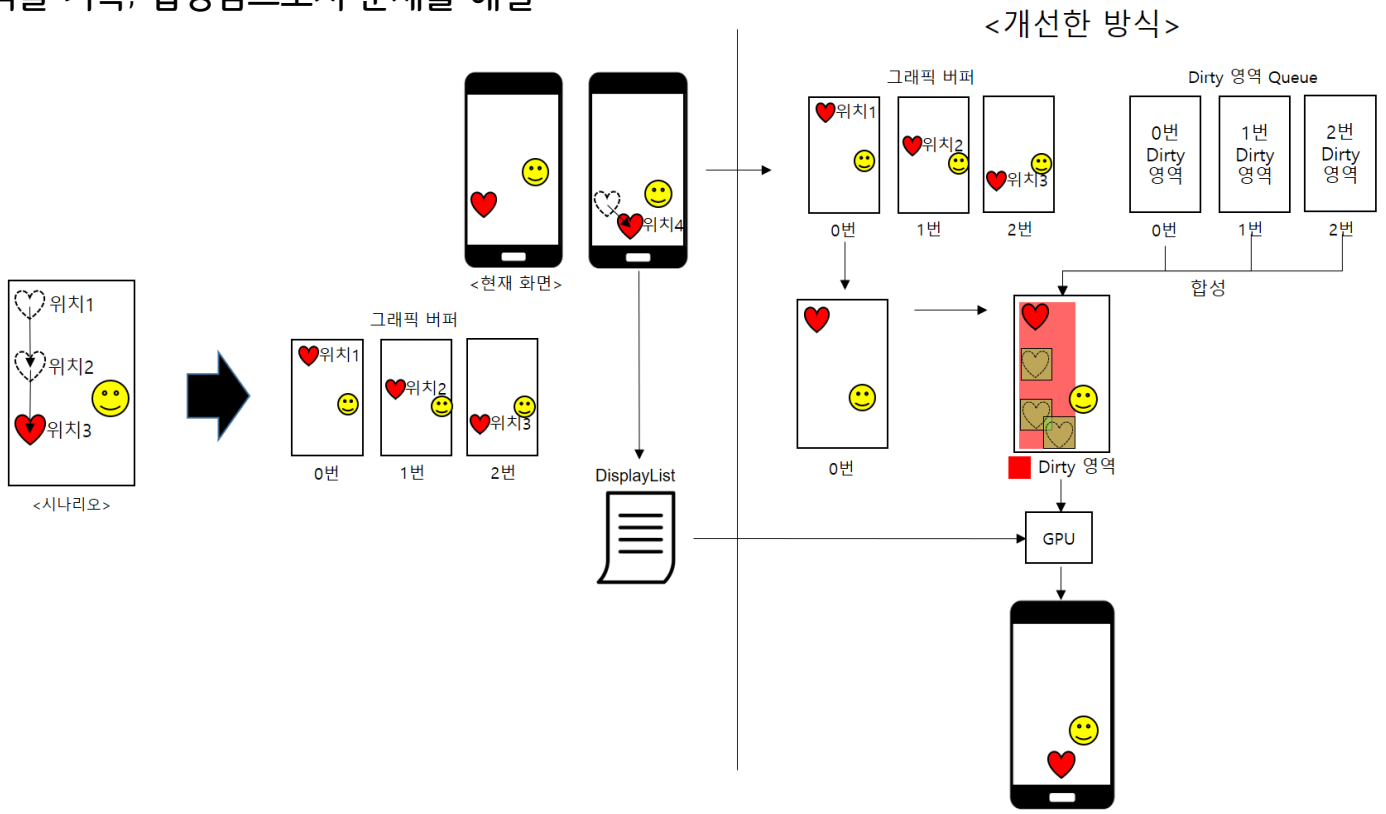
- Dirty 영역 재설정
  - 이전 단계에서 계산한 영역으로 Dirty 영역을 재설정
  - 하지만 Triple Buffering 정책으로 잔상이 생기는 문제가 발생



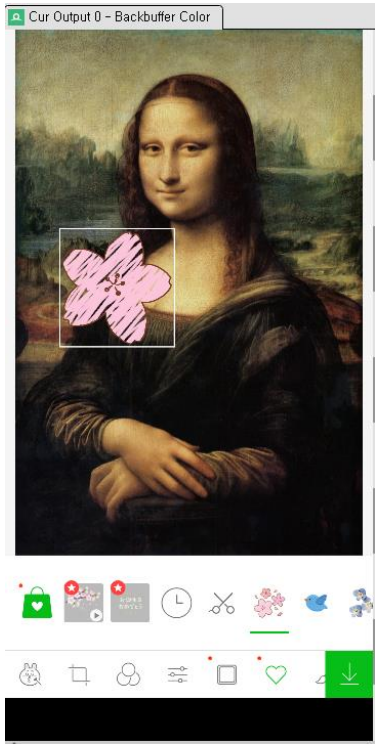
구현3



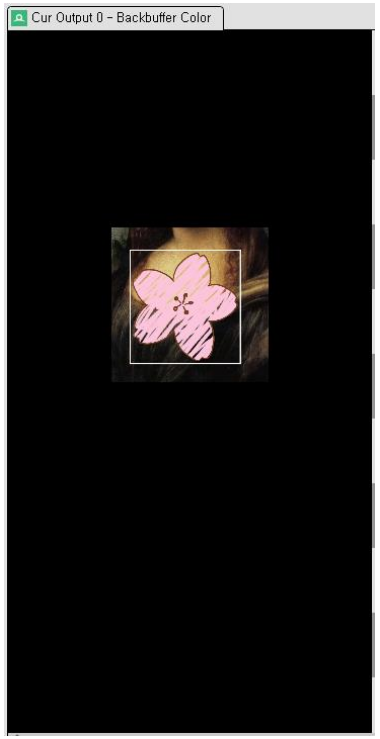
- Dirty 영역 재설정
  - 이전 단계에서 계산한 영역으로 Dirty 영역을 재설정
  - 하지만 Triple Buffering 정책으로 잔상이 생기는 문제가 발생
  - 이전 Dirty 영역을 기록, 합성함으로서 문제를 해결



구현 결과



일반적인 Rendering방식



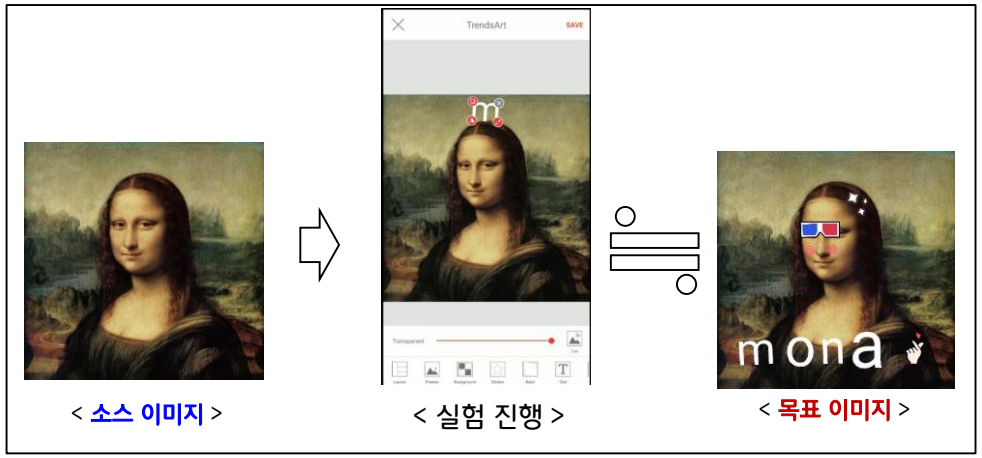
제안한 Rendering방식

Renderdoc사용 : https://renderdoc.org/

실험

- 실험 방식
  - Youtube에서 조회수가 많은 꾸미기 방식을 차용
  - 소스 이미지에 꾸미기가 적용된 목표 이미지를 정의
  - 목표 이미지와 똑같이 만드는 과정에서 측정

실험 방법

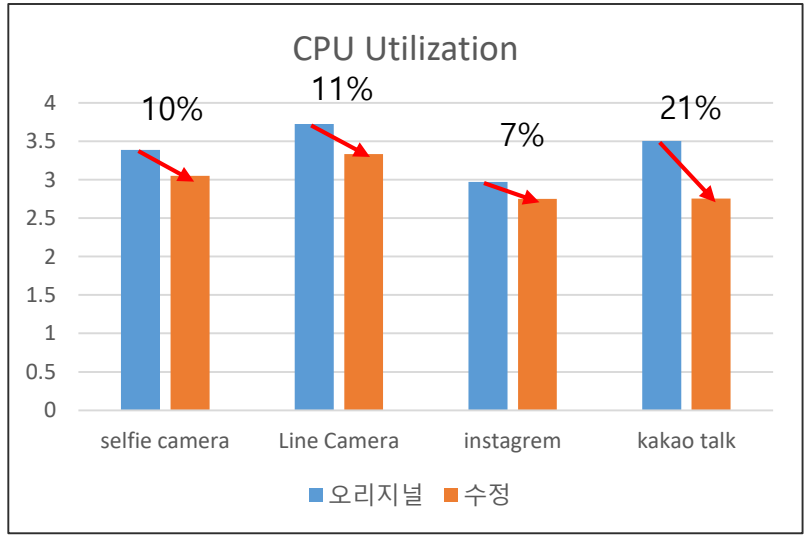


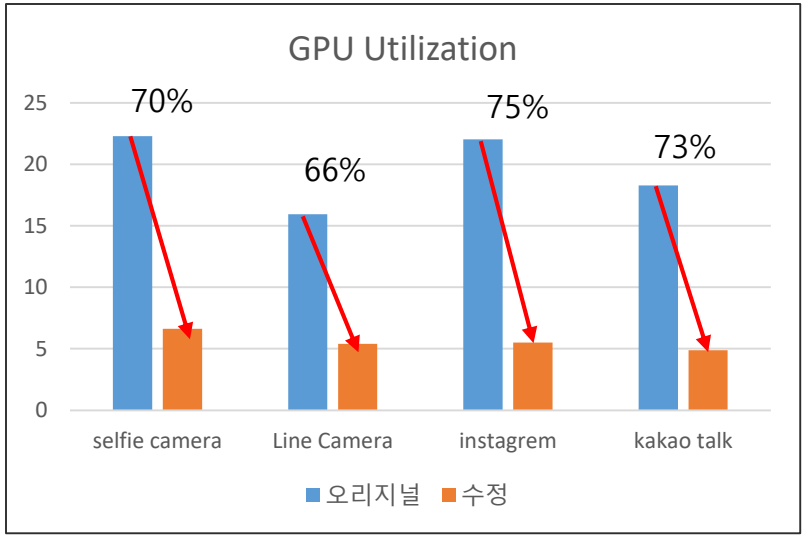
< 소스 이미지 > < 실험 진행 > < 목표 이미지 >

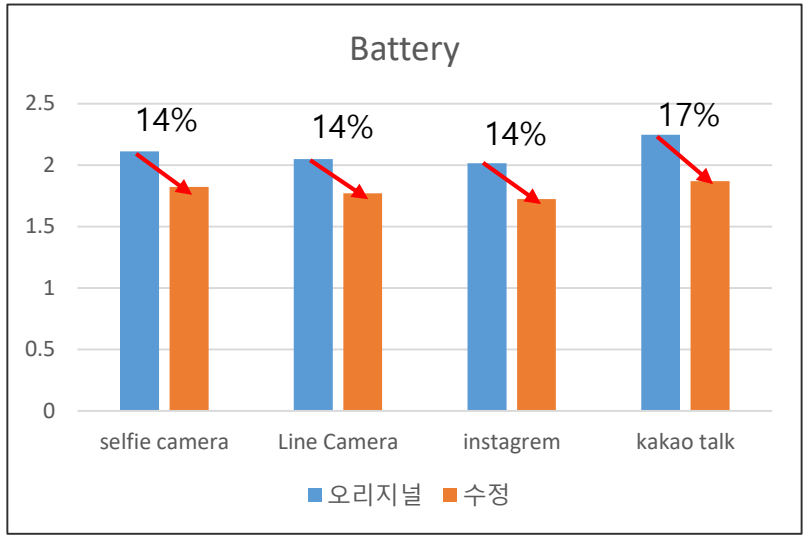
youtube “Popular decorating methods applications”
https://www.youtube.com/watch?v=CpaLdAq2h18 (1421m view)
https://www.youtube.com/watch?v=CVzh7szKiCM&t=132s (500m view)
https://www.youtube.com/watch?v=nAN-HJc4mwQ (147m view)
https://www.youtube.com/watch?v=p629joymqs4 (215m view)
https://www.youtube.com/watch?v=peptzWIFlsI (235m view)

실험

- 실험 결과







Snapdragon Profiler사용: https://developer.qualcomm.com/software/snapdragon-profiler

결론

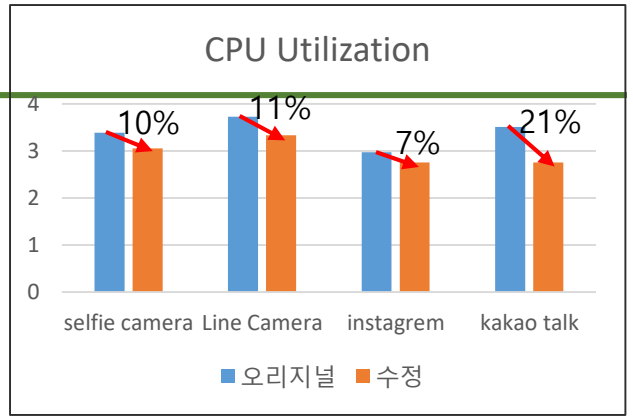
- CPU Utilization 7~21%, GPU Utilization 66~75%, Battery 14~17% 감소

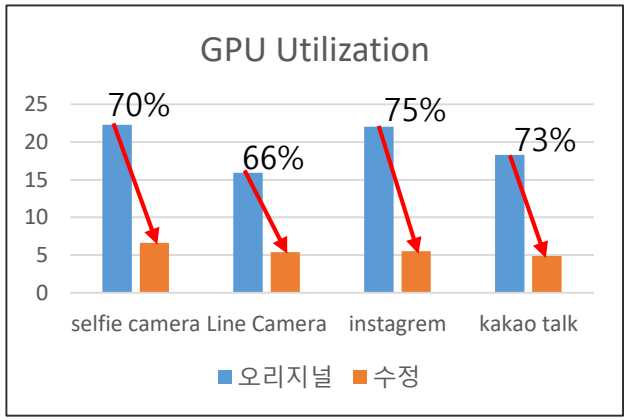
  => GPU 감소에 비해 Battery의 감소가 크게 변화하지 않음

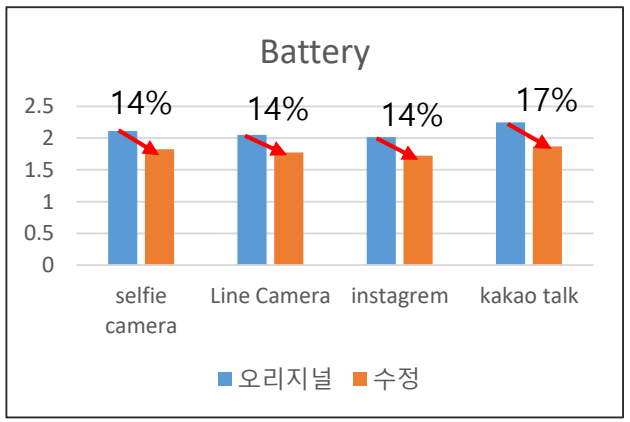
  => GPU에 보다 CPU가 Battery에 많은 영향을 준다고 생각

- CPU Utilization이 많이 줄어들지 않은 이유

  => 안드로이드 시스템은 rendering 외에 많은 작업을 처리하고 있어 rendering에서 사용되는 CPU는 줄어든 만큼 일부라고 생각







추가 실험(함수 별 수행 시간 측정)

- Rendering은 크게 3가지 함수로 구성됨(getFrame, draw, swapBuffer)