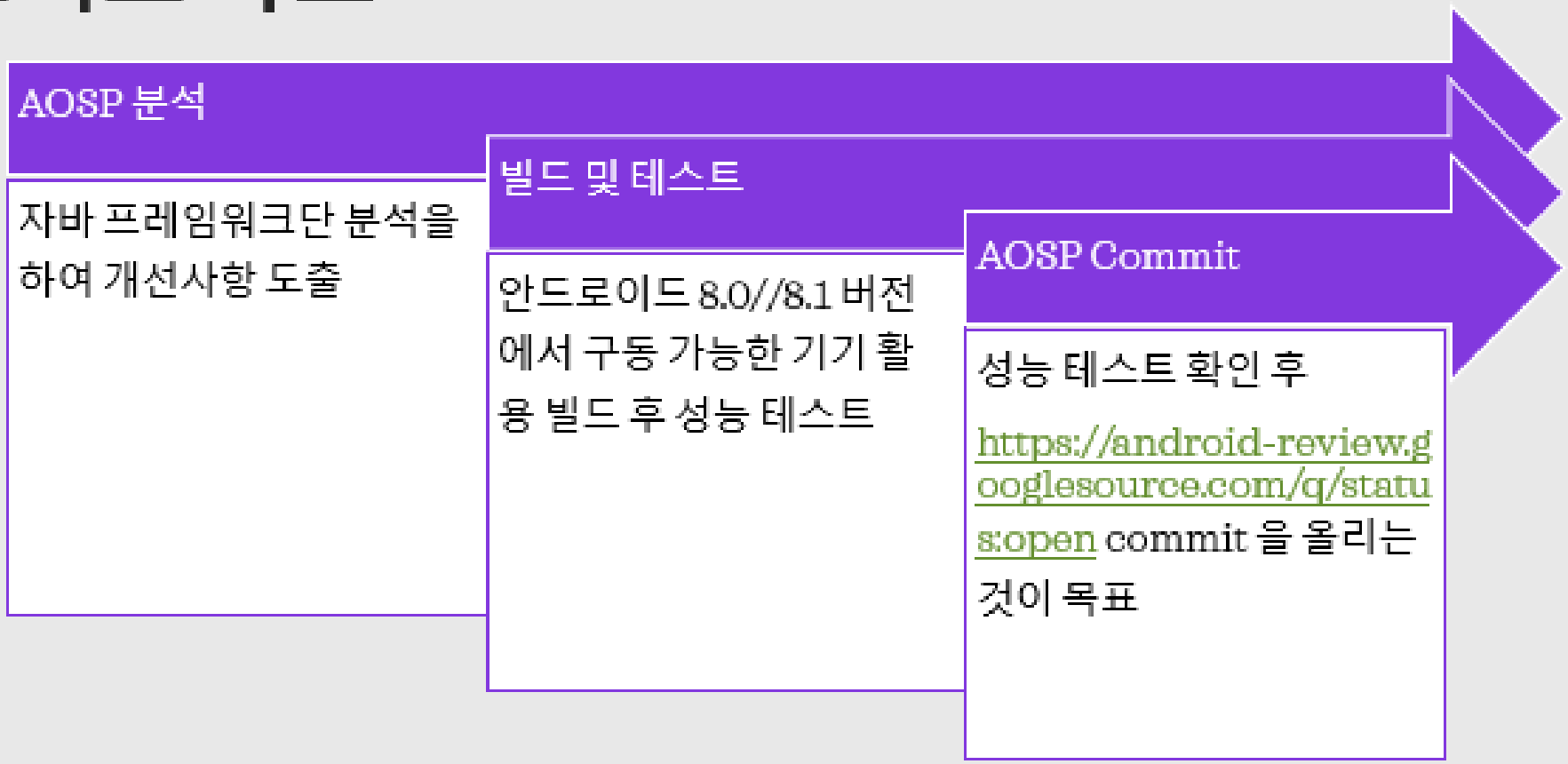
종합설계 3주차
(최종보고서)

AOSP - 201502088 이송무

프로젝트에 대한 설명

- 안드로이드 프레임워크 개선으로 안드로이드 성능을 향상시키기 위한 프로젝트입니다.
- AOSP : Android Open Source Project

프로젝트 목표


AOSP 분석
자바 프레임워크단 분석을 하여 개선사항 도출
빌드 및 테스트
안드로이드 8.0//8.1 버전에서 구동 가능한 기기 활용 빌드 후 성능 테스트
AOSP Commit
성능 테스트 확인 후
https://android-review.googlesource.com/q/status:open commit 을 올리는 것이 목표

1 일차 (MAP & HMW)

MAP

행위자와 결말을 정하고
 접근방법 및 방안들을 생각해 보는 과정입니다.

행위자를 사용자로 설정하고, 결말은 향상된 성능으로 설정을 했습니다.

HMW

어떻게 하면 부팅 시간을 개선할 수 있을까?	어떻게 하면 배터리 사용량을 줄일 수 있을까?	어떻게 하면 필요없는 기능들을 개선해 성능을 향상시킬 수 있을까?
어떻게 하면 RAM을 효율적으로 관리할 수 있을까?	어떻게 하면 전체 프레임워크에서 코드를 개선해 성능을 향상시킬 수 있을까?	어떻게 하면 하드웨어를 조정해서 성능향상을 할 수 있을까?

"어떻게 하면 ~~ 할 수 있을까?"
방식으로 전에 작성한 MAP을 참조해, 어떻게 하면 프로젝트를 개선할지에 대해 의견을 종합하는 과정입니다.

이렇게 MAP과 HMW를 정리하면서 문제의 지도와 문제에 대한 질문을 생각해볼 수 있었습니다.

2 일차 (라이트닝 데모 & Crazy 8`s)

- 라이트닝 데모는 HMW에서 생각했던 문제들에 대한 질문들의 기존방안들을 조사하고 다른 방안으로는 무엇이 있을까 생각하는 과정입니다.
- 이에 기존에 있던 안드로이드의 성능개선 방안들을 조사하고 다른 해결방안들을 생각해봤습니다.

- Crazy 8`s 는 라이트닝 데모에 나왔던 의견의 직관적인 해결방안을 종이를 8등분 해 직접 스케치를 하면서 생각의 구체화 과정을 가졌습니다.

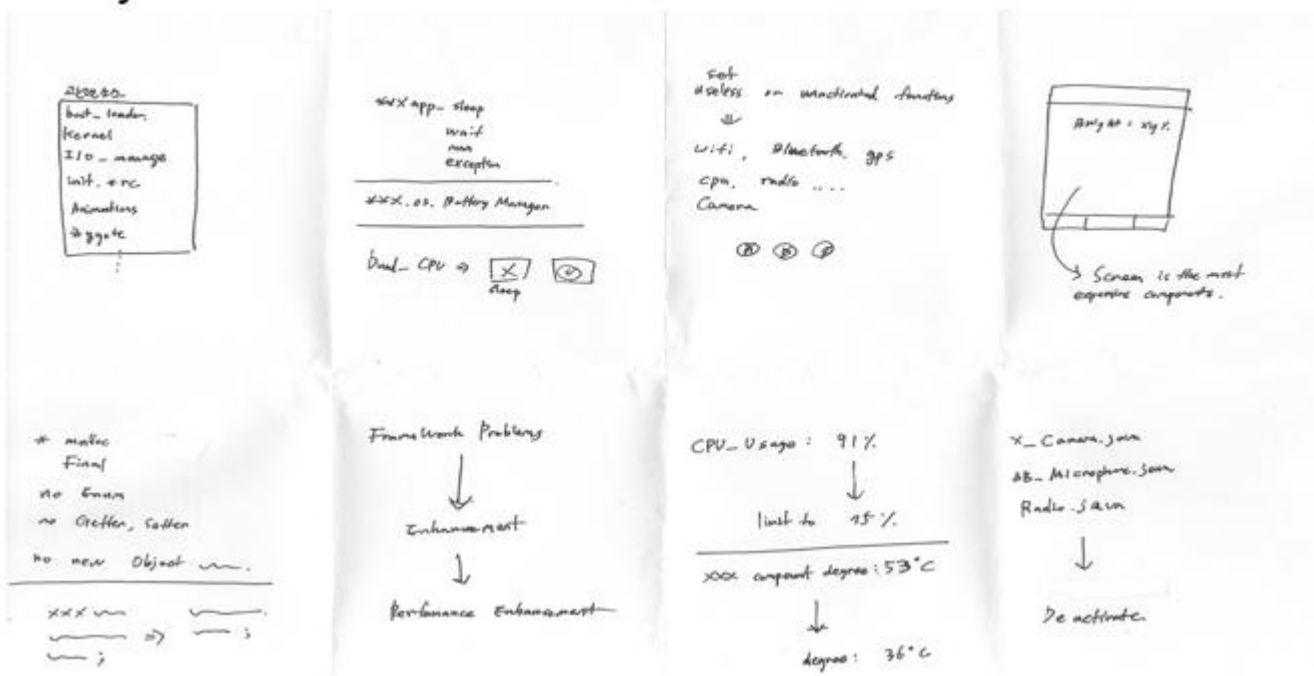
라이트닝 데모

1. 어떻게 하면 부팅시간을 개선할 수 있을까?
   기존방안에는 부트로더 최적화, 커널 최적화 방법 등이 있다.
   프레임워크에서 부팅과 관련된 디렉터리나 파일을 분석해 개선여지를 찾는 방법이 있다.

2. 어떻게 하면 배터리 사용량을 줄일 수 있을까?
   기존 방안에는 앱의 상태를 조절하는 방법 및 기기 온도상승을 모니터링해 limit를 주면서 배터리 사용량을 줄이는 방법 등이 있다.
   프레임워크에서 배터리와 관련된 디렉터리나 파일을 분석해 개선여지

3. 어떻게 하면 RAM을 효율적으로 관리할 수 있을까?
   기존 방안에는 네이티브 코드의 개선방법 및 시스템 메모리 할당 방법
   프레임워크에서 전체적으로 메모리를 많이 쓰는 파일이나, 램과 관련해 개선여지를 찾는 방법이 있다.

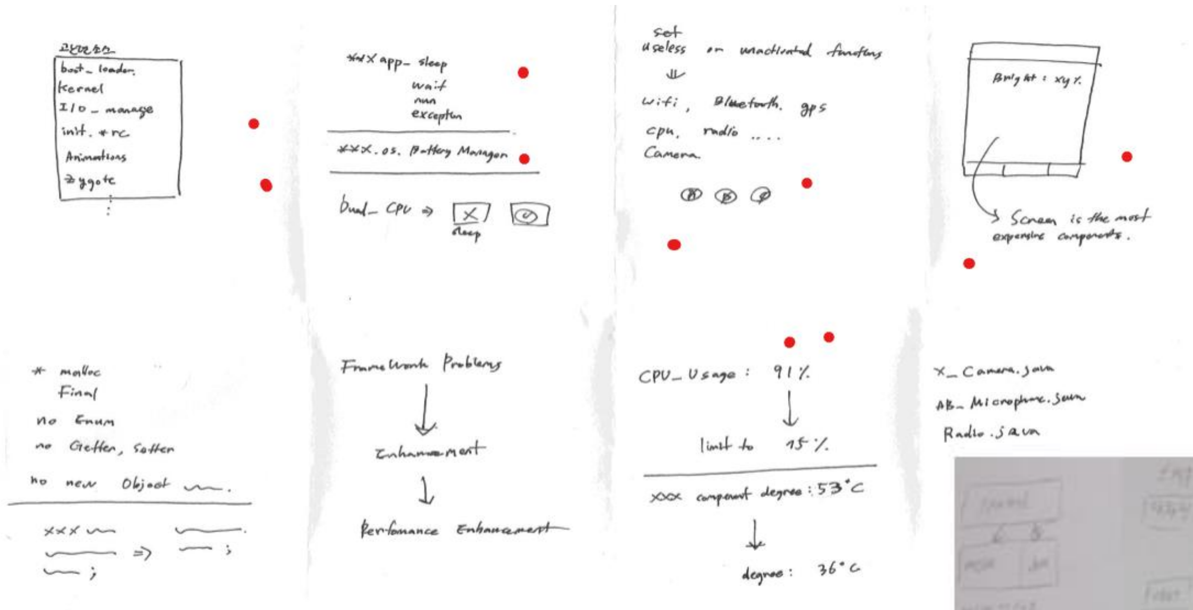
Crazy 8`s

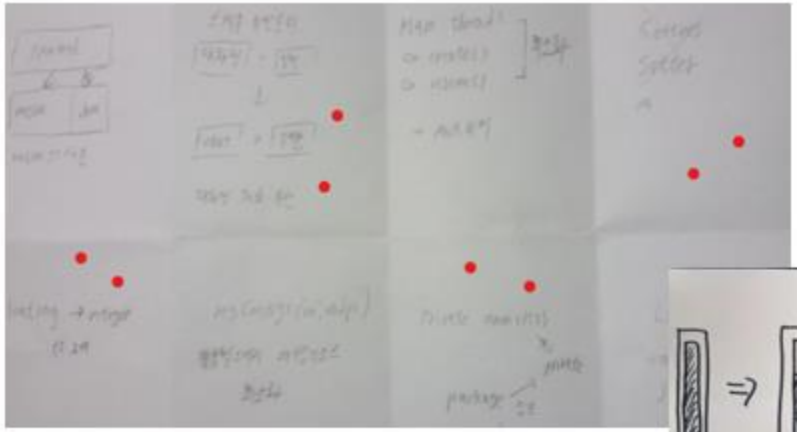


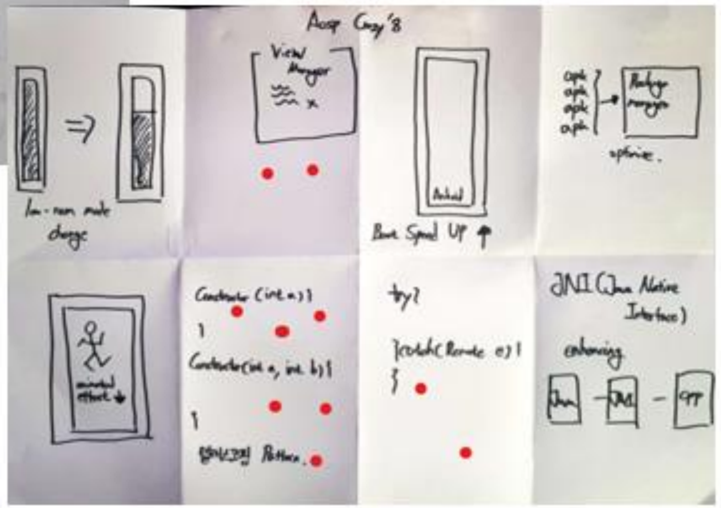
3 일차 (투표 및 스토리보드)

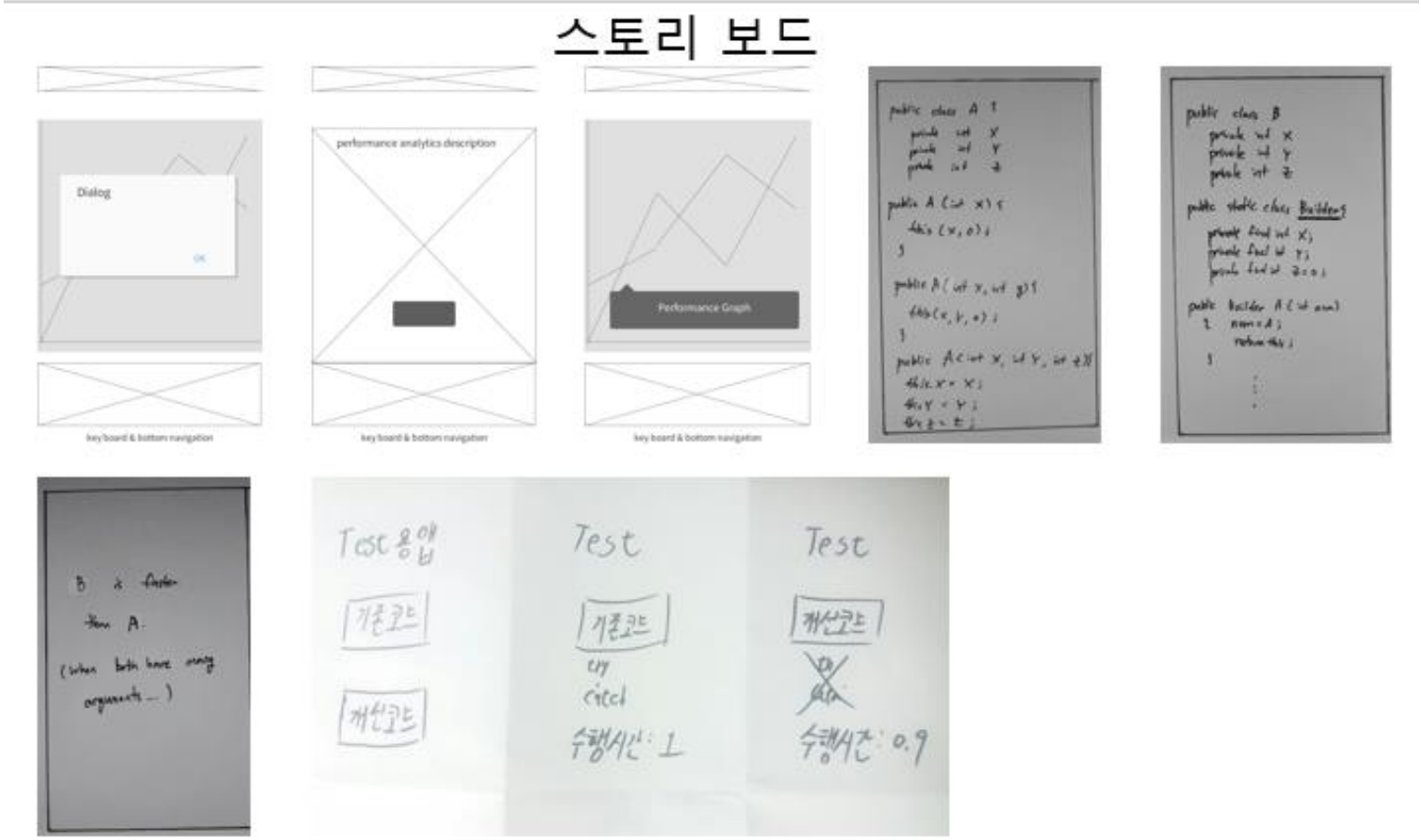
전에 작성했던 Crazy 8`s 과 그에 대한 설명을 하는 영상으로 다른 조들에게 구글 설문지로 투표를 받아 투표를 받은 솔루션에 해당하는 스토리보드(솔루션)를 작성하는 과정입니다.

(투표결과)


Framework Problems
Enhancement
Performance Enhancement
CPU_Usage : 91%
limit to 15%
degree : 36°C
Screen is the most expensive components.
Radio.java




View Manager
Boot Speed Up
JNI (Java Native Interface)



투표 결과로
텔레스코핑 패턴을 수정하여 성능을 개선하는 솔루션,
Try catch 문을 수정하여 성능을 개선하는 솔루션,
성능 측정 프로파일 도구를 직접 만드는 솔루션

들이 채택되어 그에 해당하는 스토리 보드를 팀원들과 분업하여 작성했습니다.

4 일차 (프로토타입 만들기)

- 팀원들과 분업하여 자신이 맡은 스토리보드의 프로토타입을 만드는 시간을 가졌습니다.

대략적인 순서입니다

- 1. 텔레스코핑 패턴을 수정하여 성능을 개선
- 2. Try-catch 문을 개선하여 성능을 개선
- 3. 성능측정 프로파일 도구

텔레스코핑 패턴을 수정하여 성능 개선 방법

안드로이드 프레임 워크 파일안에 있는 WindowManager.java 파일이다. 실제로 Telescoping Constructor Pattern이 LayoutParams 클래스에서 확인이 가능하다.

```
public LayoutParams(int _type) {
    super(LayoutParams.MATCH_PARENT, LayoutParams.MATCH_PARENT);
    type = _type;
    format = PixelFormat.OPAQUE;
}

public LayoutParams(int _type, int _flags) {
    super(LayoutParams.MATCH_PARENT, LayoutParams.MATCH_PARENT);
    type = _type;
    flags = _flags;
    format = PixelFormat.OPAQUE;
}

public LayoutParams(int _type, int _flags, int _format) {
    super(LayoutParams.MATCH_PARENT, LayoutParams.MATCH_PARENT);
    type = _type;
    flags = _flags;
    format = _format;
}

public LayoutParams(int w, int h, int _type
    super(w, h);
    type = _type;
    flags = _flags;
    format = _format;
}

public LayoutParams(int w, int h, int xpos,
        int _flags, int _format) {
    super(w, h);
    x = xpos;
    y = ypos;
    type = _type;
    flags = _flags;
    format = _format;
}
```

Telescoping Constructor Pattern

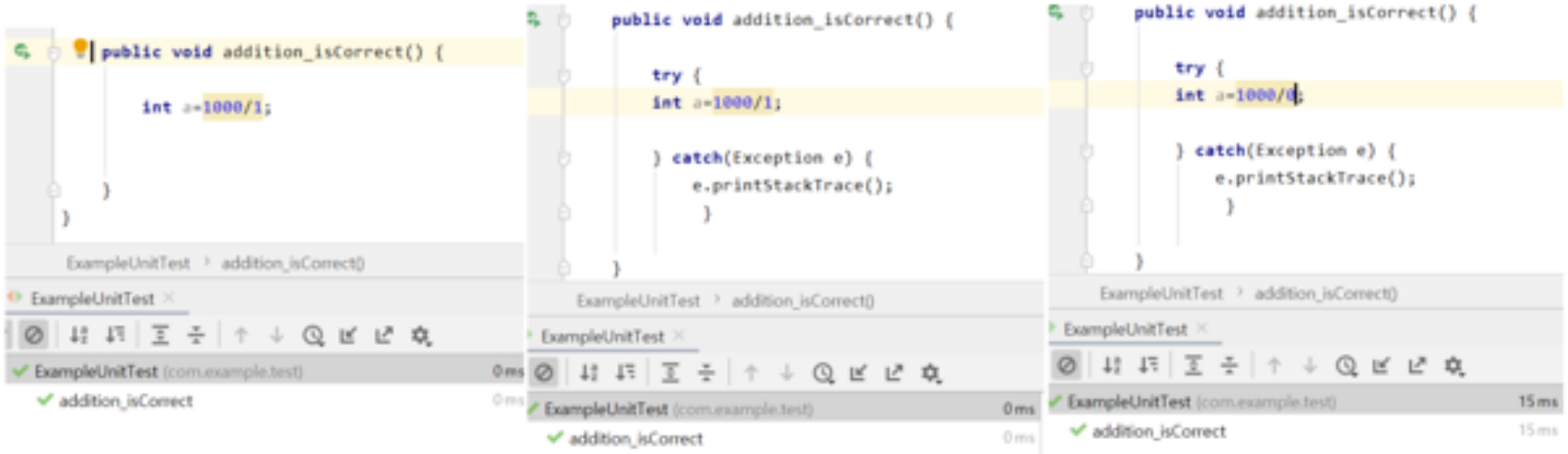
우선, 텔레스코핑 패턴을 이용한 결과 화면이다.

8개의 인자를 주고, 8개의 인자를 호출하는 함수의 실행시간을 측정하였다.

실행시간은 506 * 10 ^ -6 초 인것을 확인 가능하다.

(프로토타입의 일부입니다)

Try-Catch 문 수정하여 성능 개선



try문에서 예외 발생시 추가연산발생

(프로토타입의 일부입니다)

성능측정도구 제작

안드로이드의 성능을 개선한 빌드 버전에서 부팅 속도 및 측정할 수 있는 테스트 앱을 설치를 하여 수정된 부분에서 성능 개선이 되었는지를 확인하는 솔루션이다.

(프로토타입의 일부입니다)

만들어진 스토리보드를 기반으로
텔레스코핑 패턴을 빌더 패턴으로 개선 -> 예외처리문 개선 -> 테스팅 앱 구현
을 통하여 안드로이드 오픈 소스 프로젝트에서의 코드 패턴들을 개선함으로써 성능개선을 기획하고, 성능측정도구로 그 정도를 측정하도록 프로토타입을 구현했습니다.

5 일차 (설문 분석 및 최종 발표)

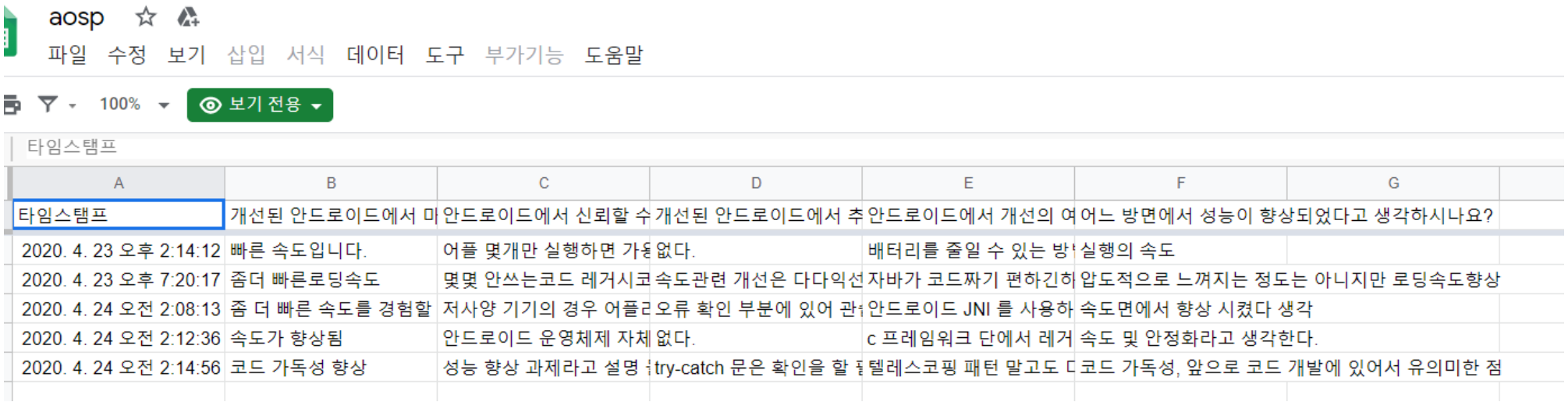
4일 차에서 작성한 프로토타입을 가지고 5명 이상의
구글 설문조사를 통하여 디자인 스프린트 프로세스를 통해서
기능들에 대한 설문결과를 확인했습니다.

설문조사 질문들

- 개선된 안드로이드에서 마음에 들었던 점이 있다면 어느 점에서 장점을 느꼈는가?
- 안드로이드에서 신뢰할 수 없는 기능이나 단점이 있다면 어떤 것이며 그 이유는 무엇인가?
- 개선된 안드로이드에서 추가된 기능이 필요 없다고 생각하는 부분이 있는가?
- 안드로이드에서 개선의 여지가 있는 부분이 있다면 어떤 것이며 그 이유는 무엇인가?
- 어느 방면에서 성능이 향상되었다고 생각하는가?

설문조사 결과

- 구글 설문지를 이용해 설문조사



aosp

파일 수정 보기 삽입 서식 데이터 도구 부가기능 도움말

100% 보기 전용

타임스탬프

	A	B	C	D	E	F	G
	타임스탬프	개선된 안드로이드에서 마	안드로이드에서 신뢰할 수	개선된 안드로이드에서 추	안드로이드에서 개선의 여	어느 방면에서 성능이 향상되었다고 생각하시나요?	
	2020. 4. 23 오후 2:14:12	빠른 속도입니다.	어플 몇개만 실행하면 가	없다.	배터리를 줄일 수 있는 방	실행의 속도	
	2020. 4. 23 오후 7:20:17	좀더 빠른로딩속도	몇몇 안쓰는코드 레거시코	속도관련 개선은 다다익선	자바가 코드짜기 편하긴하	압도적으로 느껴지는 정도는 아니지만 로딩속도향상	
	2020. 4. 24 오전 2:08:13	좀 더 빠른 속도를 경험할	저사양 기기의 경우 어플리	오류 확인 부분에 있어 관	안드로이드 JNI 를 사용하	속도면에서 향상 시켰다 생각	
	2020. 4. 24 오전 2:12:36	속도가 향상됨	안드로이드 운영체제 자체	없다.	c 프레임워크 단에서 레거	속도 및 안정화라고 생각한다.	
	2020. 4. 24 오전 2:14:56	코드 가독성 향상	성능 향상 과제라고 설명	try-catch 문은 확인을 할	텔레스코핑 패턴 말고도 다	코드 가독성, 앞으로 코드 개발에 있어서 유의미한 점	

설문조사 결과 분석

- **개선된 안드로이드에서 마음에 들었던 점이 있다면 어느 점에서 장점을 느꼈는가?**

: 전보다 빠른 속도 (4명), 코드 가독성 향상 (1명)

- **안드로이드에서 신뢰할 수 없는 기능이나 단점이 있다면 어떤 것이며 그 이유는 무엇인가?**

: 하드웨어의 성능만 믿고 통으로 짠 코드가 많다, 가용램이 적다, 레거시 코드가 많다.

- **개선된 안드로이드에서 추가된 기능이 필요 없다고 생각하는 부분이 있는가?**

: try / catch 문을 재확인 할 필요성이 있다.

- **안드로이드에서 개선의 여지가 있는 부분이 있다면 어떤 것이며 그 이유는 무엇인가?**

: 안드로이드 JNI를 사용하기 때문에 관습적인 코드가 있을 수 밖에 없다고 생각,
텔레스코핑 패턴 말고도 다른 레거시 패턴들이 있을 것이다.,
C코드를 쓰는 것이 더 빠르지 않을까?

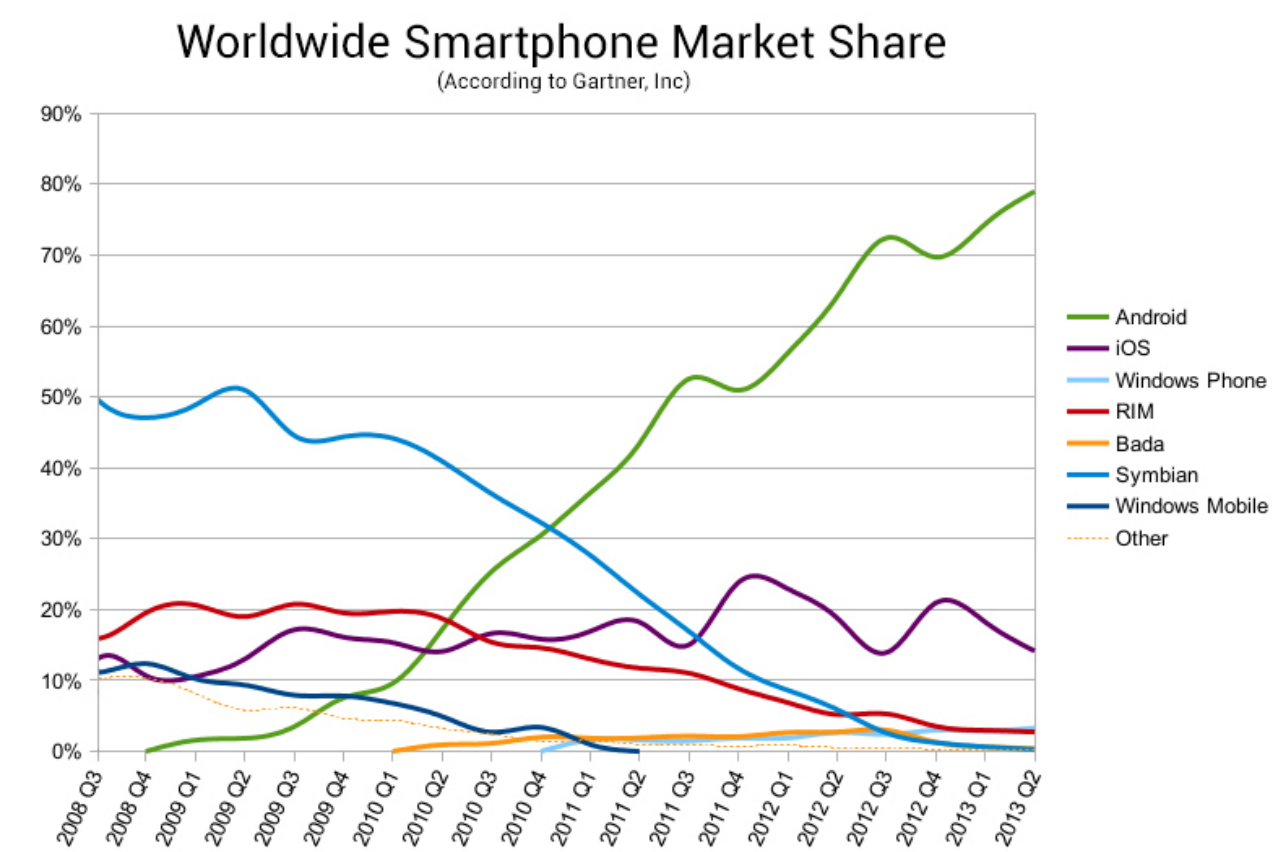
- **어느 방면에서 성능이 향상되었다고 생각하는가?**

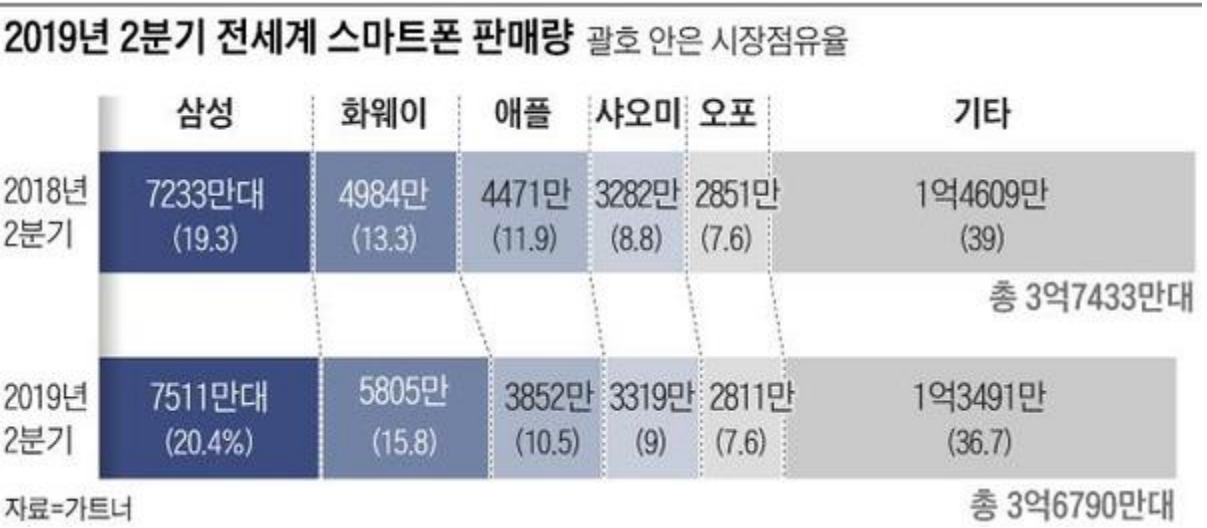
: 대부분 실행 속도(4명), 코드의 가독성 및 앞으로의 코드 개발 여부

- AOSP조는 저번의 경험을 되살려, 졸업프로젝트에 디자인스프린트를 적용시켜 프로젝트 진행을 했다. 안드로이드의 성능개선과 같은 모호한 주제의 프로젝트를 진행시키는데 디자인스프린트 방식은, 졸업 프로젝트의 방향을 잡아주고, 명확하게 무엇을 진행하는지를 확인시켜주는데 도움을 줬다.

- 안드로이드의 성능을 개선시키기 위해 우리 조는 안드로이드 프레임워크의 소스를 뜯어보아 성능저하를 초래할 여지, 예를 들어 레거시 코드나, 전에 언급했던 텔레스코핑 패턴 등을 개선시켜 성능향상 방향을 모색하고 있는 중이다.

- 구현 및 실현 고려사항으로는 아직 프로젝트가 진행중 이므로, 우리 조가 더욱더 연구를 심도있게 하고, 많은 자료를 참고하고, 코드를 열심히 뜯어보면 안드로이드의 성능개선의 여지는 반드시 있을 것이다.





시장 등을 보면 안드로이드의 점유율이 점점 증가하고 있다. 이는 안드로이드가 애플을 견제하기 위한 AOSP 결과의 일부이기도 하다.

또한 소비자는 스펙 상향평준화가 되어버린 스마트폰 시장에서, 더 이상, 다른 제품보다 비싼, 고가의 아이폰을 살필요가 없다고 여기고 있다.

AOSP 기반 안드로이드 OS는 구글의 인증을 받지 않았기 때문에 구글 모바일 서비스를 공식적으로 이용할 수 없지만 안드로이드 스마트폰에서는 정상적으로 작동한다.
AOSP를 통해 OS를 개발한 업체가 구글 서비스를 대체할 자체 서비스와 앱 마켓을 운용 할 수 있다면 독자적 생존이 가능한 것이다. AOSP를 가장 적극적으로 이용한 업체가 바로 중국의 샤오미다. 샤오미 또한 빠른 속도로 스마트폰 시장 점유율을 차지하고 있다.

이를 통해 만약 우리 조가 눈에 띄는 안드로이드 성능개선을 이루어내면, 시장에서 많이 유리하지 않을까 싶다.

후기

- 한번 디자인스프린트를 경험해 보니, 다시 졸업프로젝트에 적용시키기 쉬운 것 같다.

- 디자인스프린트를 이용하니, 진행이 유동적이며, 나아갈 방향을 잡을 수 있었고, 서로 조원들끼리 무엇을 생각하는지 알아볼 수 있는 기회를 가질 수 있었다.

링크들

- 깃허브 : https://github.com/keelim/project_aosp
- 유투브 : https://www.youtube.com/channel/UC05ce0W79PQWj6H86cN4_LQ