안드로이드 UI에서의 다양한 스크린 사이즈 지원

안드로이드펍
박성서

발표자

- **박성서**(회색)
  - 2008 안드로이드 개발자 챌린지 Ⅰ 입상
  - 2009 안드로이드 개발자 챌린지 Ⅱ TOP 20
  - 안드로이드펍 운영자
    - http://www.androidpub.com
  - 회색의 구글 안드로이드 개발 블로그
    - http://graynote.tistory.com

UI 레이아웃

레이아웃 디자인 어떻게 하시나요?

안드로이드의 다양성

- 현재의 스크린
  - QVGA(240x320), 120dpi : HTC Tatoo
  - HVGA(320x480), 160dpi : 안드로원, HTC G1
  - WVGA(480x800), 240dpi : 넥서스원, 갤럭시A
  - FWVGA(480x854), 240dpi : 모토로이
- 앞으로는 더 종류가 많아짐
  - WQVGA(240x400)
  - FWQVGA(320x432)
  - ..
- 스크린 사이즈 처리방법을 모르면 제대로 된 안드로이드 앱개발을 할 수가 없음

용어 정리

- **스크린 사이즈 (Screen Size)**
  - 스크린 사이즈는 스크린의 대각선 크기로 재어지는 물리적인 크기를 나타낸다. 안드로이드는 스크린 사이즈를 크게 3가지로 분류하는데 large, normal, small로 나눈다.

- 가로세로 비 (Aspect ratio)
  - 가로세로 비는 스크린의 물리적인 넓이와 높이의 비율로 결정된다. 안드로이드는 가로세로 비를 long과 notlong 으로 나눈다.

- 해상도 (Resolution)
  - 스크린이 가지고 있는 전체 픽셀의 수. 해상도는 종종 넓이 x 높이로 표현되지만 해상도가 특정 가로세로 비를 의미하지는 않는다. 안드로이드에서는 해상도를 직접 처리하지 않는다.

용어 정리

- **밀도 (Density)**
  - 스크린 해상도를 기반으로 물리적인 넓이와 높이 안에 얼마나 많은 픽셀이 들어있는가를 나타낸다. Lower density의 스크린에서는 같은 넓이와 높이 안에 더 적은 수의 픽셀이 있고, higher density의 스크린에서는 같은 넓이와 높이 안에 더 많은 수의 픽셀이 있다. 안드로이드는 밀도를 high, medium, low 세가지 분류로 나눈다. 플랫폼에서는 실제 스크린 밀도에 맞게 리소스들의 사이즈를 조정한다.

- **Density-independent pixel (dip)**
  - 밀도와 상관없이 레이아웃의 위치를 표현할 때 사용하는 가상의 pixel 단위. Density-independent pixel 은 기본 밀도인 160dip에서의 물리적인 pixel과 같다.
  - 픽셀 변환 공식 pixels = dips * (density / 160)

지원되는 스크린 타입

	Low density 120 **ldpi**	Medium density 160 **mdpi**	High density 240 **hdpi**
Small screen	• QVGA (240x320) 2.6"-3.0″		
Normal screen	• WQVGA (240x400) 3.2"-3.5″ • FWQVGA (240x432) 3.5"-3.8″	• HVGA (320x480) 3.0"-3.5″	• WVGA (480x800) 3.3"-4.0″ • FWVGA (480x854) 3.5"-4.0″
Large screen		• WVGA (480x800) 4.8"-5.5″ • FWVGA (480x854) 5.0"-5.8″	

- 기본 스크린 (Baseline screen)
  - HVGA, Normal Screen, Medium density
  - DIP와 Pixel 1:1 매치

서로 다른 스크린의 리소스 관리

- 장치 종류마다 별도의 리소스를 사용할 수 있다
  - 스크린 사이즈 (small, normal, large)
  - 밀도 (ldpi, mdpi, hdpi, nodip)
  - 가로세로 비 (long, notlong)
- 리소스 포더 이름으로 구분 처리

res/layout/my_layout.xml	Normal 스크린 사이즈 레이아웃
res/layout-small/my_layout.xml	Small 스크린 사이즈 레이아웃
res/layout-large/my_layout.xml	Large 스크린 사이즈 레이아웃
res/drawable-ldpi/my_icon.png	Low density 를 위한 아이콘
res/drawable-mdpi/dpi/my_icon.png	Medium Density를 위한 아이콘
res/drawable-hdpi/my_icon.png	High Density를 위한 아이콘
res/drawable-nodpi/composite.xml	Density 와 무관한 리소스

리소스 관리

장치마다 별도의 리소스를
모두 생성해야 할까?

레이아웃과 이미지

Pixel

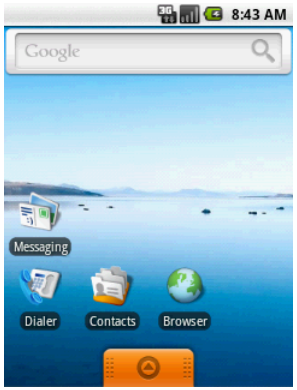
PX

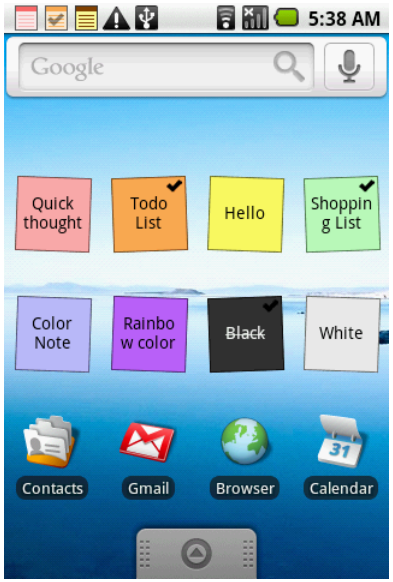
익숙함, 편함

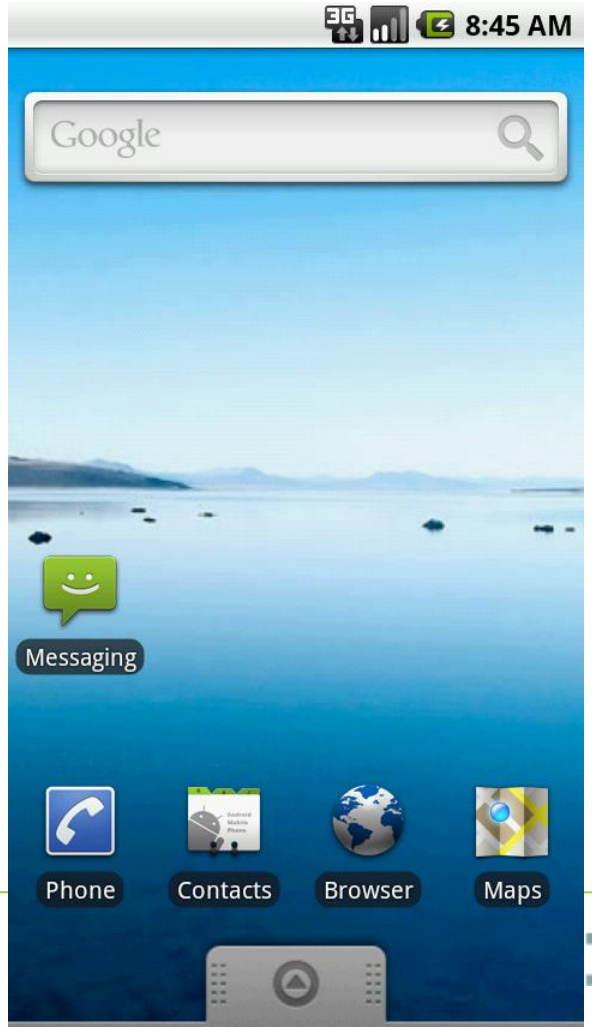
가장 큰 실수

3가지 해상도의 디바이스 (px)

QVGA	HVGA	WVGA854
(240px·320px)	(240px·320px)	(480px·854px)







- 모두 다른 해상도?

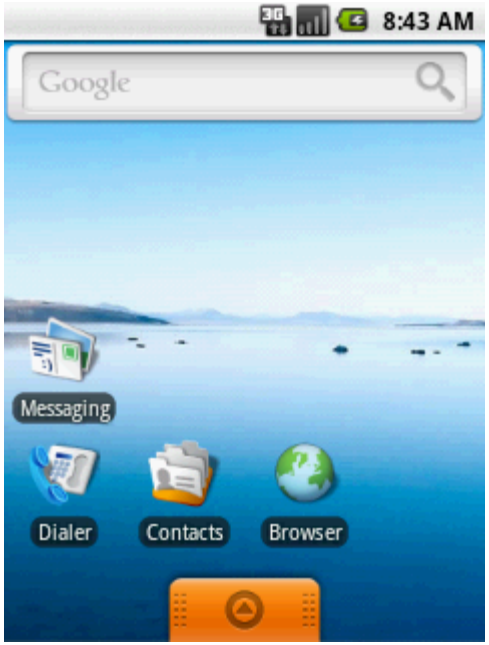
DIP : Density Independent Pixel

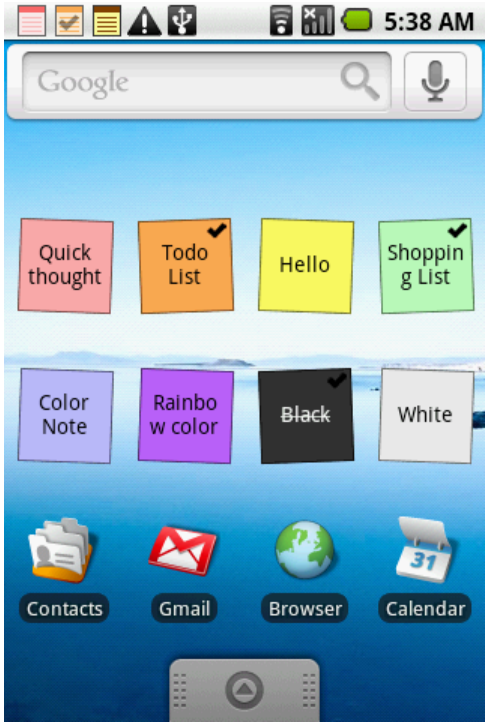
DIP

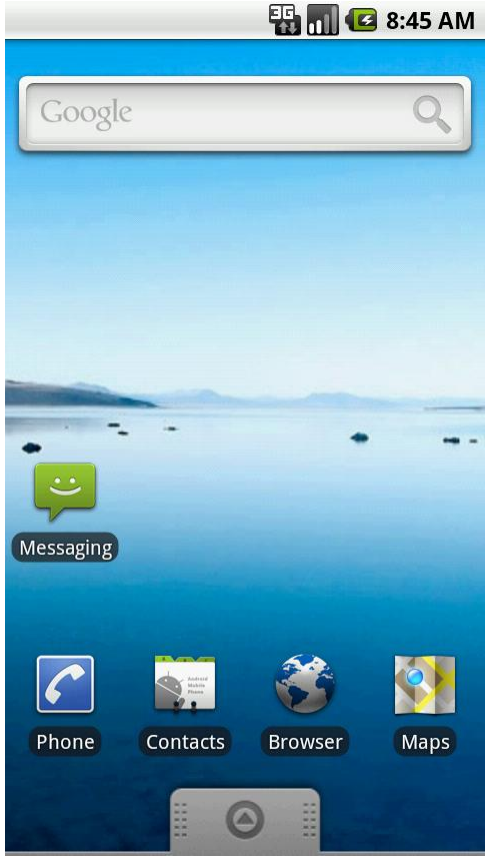
서로 다른 장치에서
호환성 보장

3가지 해상도의 디바이스 (dip)

QVGA	HVGA	WVGA854
(320dip·426dip)	(320dip·480dip)	(320dip·569dip)







- 모두 같은 넓이의 DIP

TIP 1 : 레이아웃 작성

HVGA 기본 스크린에서 DIP만 사용해서 레이아웃 디자인을 한다

px과 dip가 1:1이라 이해 쉽다

코드에서의 DIP 변환

- 그래픽 관련 메소드는 대부분 Pixel을 인자로 받음
- 상수는 항상 DIP로 정의한 후 Pixel로 변환해 사용

```
private static final float GESTURE_THRESHOLD_DIP = 16.0f;    //상수 정의

mGestureThreshold = TypedValue.applyDimension(               //Pixel 변환
            TypedValue.COMPLEX_UNIT_DIP,
             GESTURE_THRESHOLD_DIP,
            getResources().getDisplayMetrics());
```

```
private static final float GESTURE_THRESHOLD_DIP = 16.0f; //상수 정의

final float scale = getContext().getResources().getDisplayMetrics().density;
mGestureThreshold = (int) (GESTURE_THRESHOLD_DIP * scale + 0.5f);
```

Dimensions 리소스 이용

- res/values/dimensions.xml

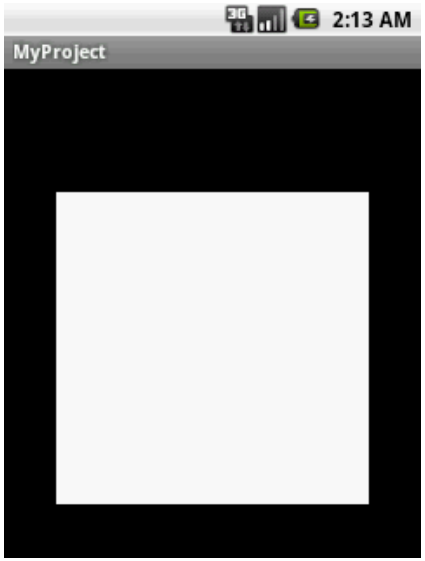
```
<resources>
    <dimen name="length">20dip</dimen>
</resources>
```

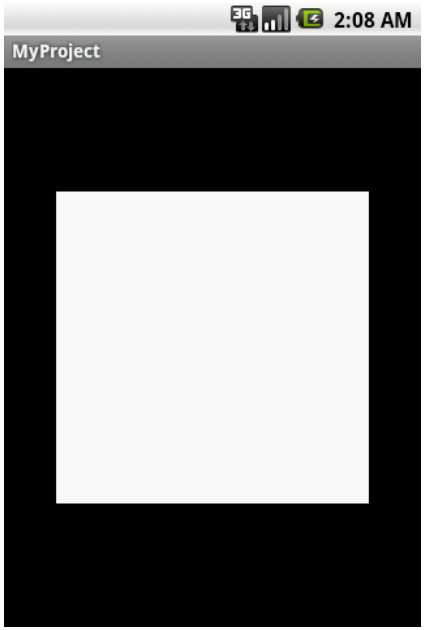
- Java Code

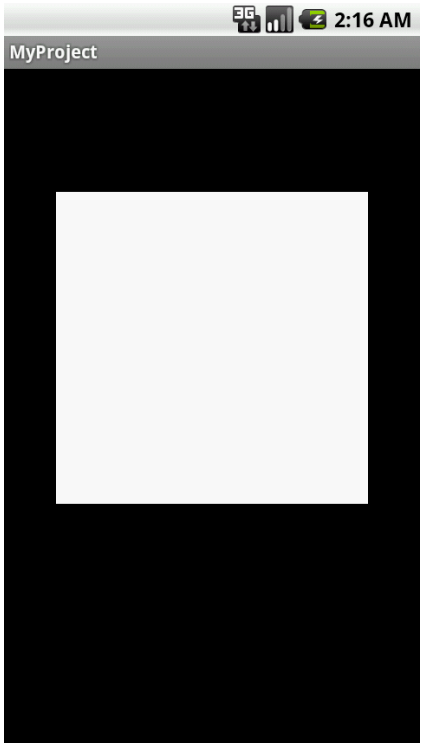
```
int length = getResources().getDimensionPixelSize(R.dimen.length)
```

절대위치

- AbsoluteLayout
  - 절대 위치를 사용하면 높이가 다른 장치에서 원하는 데로 표시가 안될 수 있으므로 사용하지 않는다.

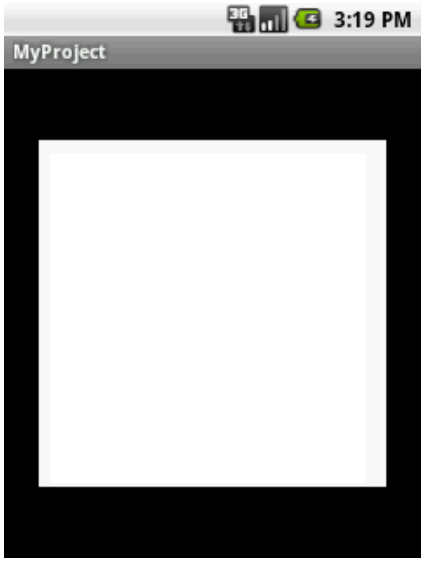


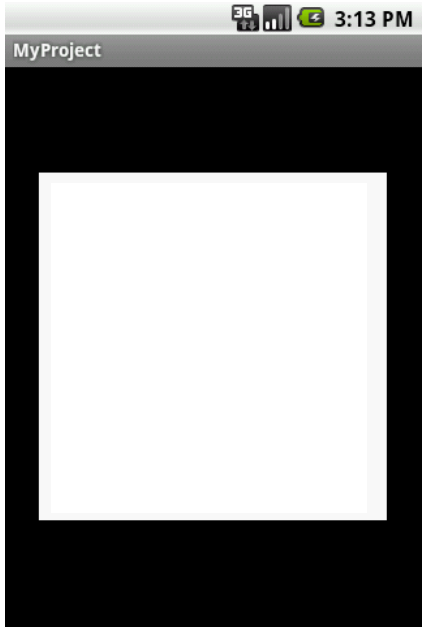


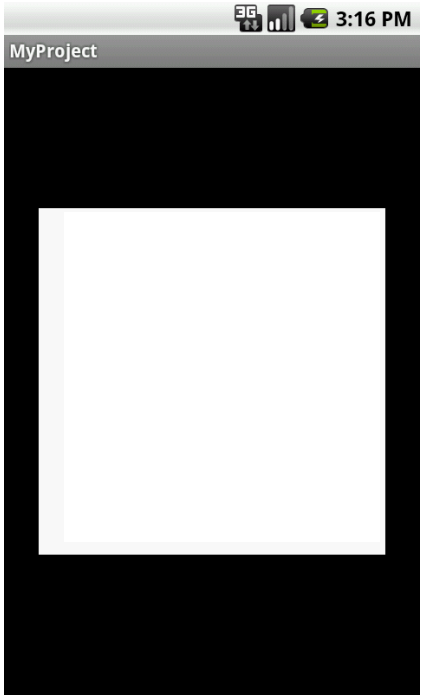


상대위치

- RelativeLayout
  - 상대 위치를 사용하면 높이가 다른 장치에서도 원하는 데로 표시하기 쉽다.







다양한 사이즈 지원 방법

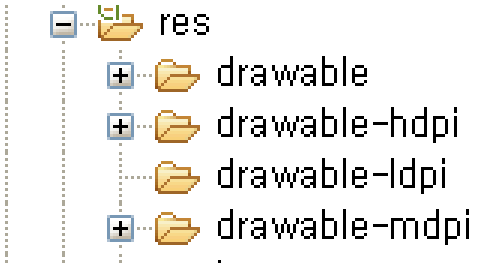
1. 다양한 스크린 사이즈 처리는 안드로이드 1.6 버전에서부터 지원됨

2. DIP상으로는 모두 같은 넓이를 가지므로 Layout XML에서든 Java Code에서든 절대 Pixel 단위를 쓰지 않고 DIP를 쓴다.

3. DIP상으로도 모두 같은 높이를 가지진 않으므로 AbsoluteLayout등으로 절대적인 좌표를 사용하여 View를 배치하지 않는다.

Bitmap 해상도

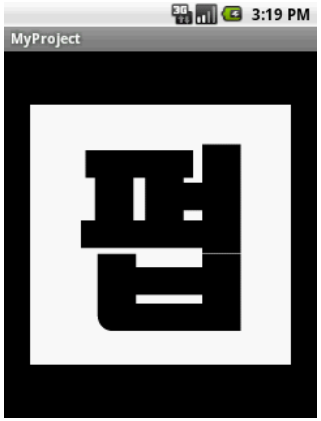
- Resource

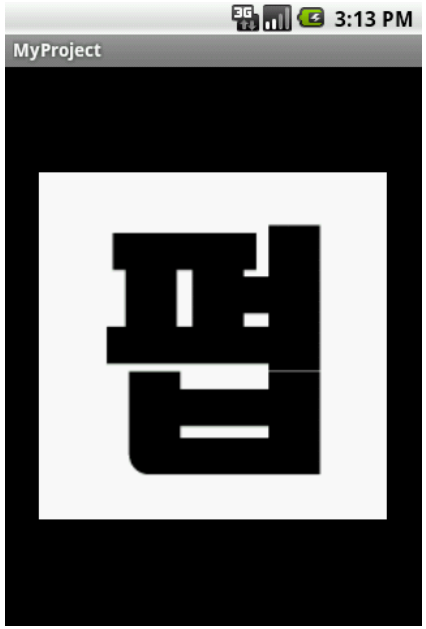


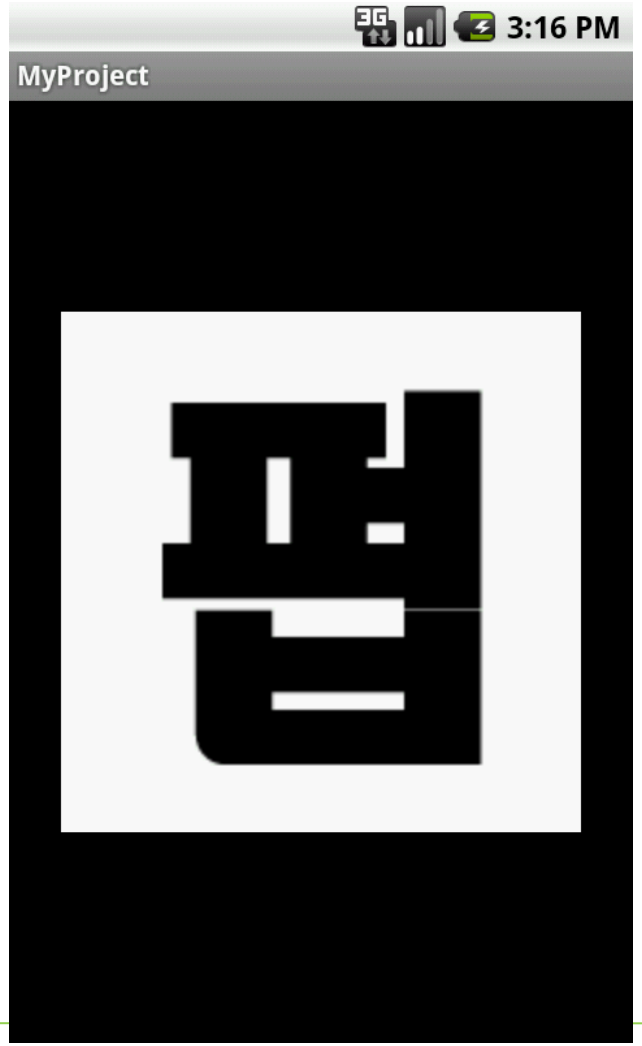
  - drawable : 해상도와 상관없는 xml drawable 파일
  - drawable-ldpi : Low Density를 위한 이미지 파일
  - drawable-mdpi : Medium Density를 위한 이미지 파일
  - drawable-hdpi : High Density를 위한 이미지 파일

Bitmap 해상도 ldpi 폴더

- 200px · 200px
- 모두 다른 픽셀 크기
- 자동 비트맵 크기 조정 (확대)



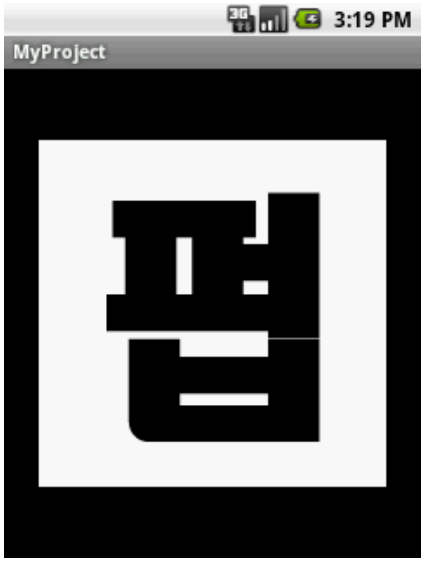


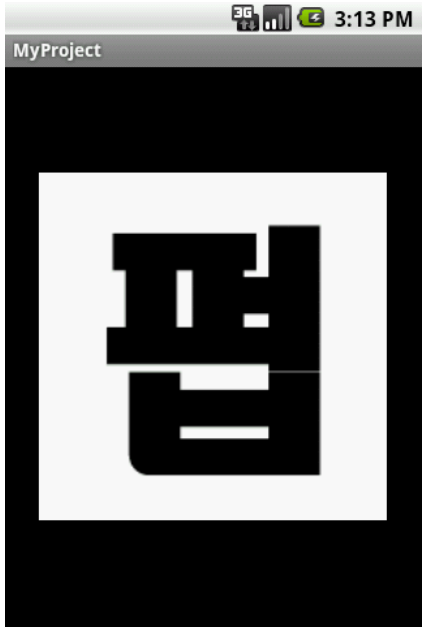


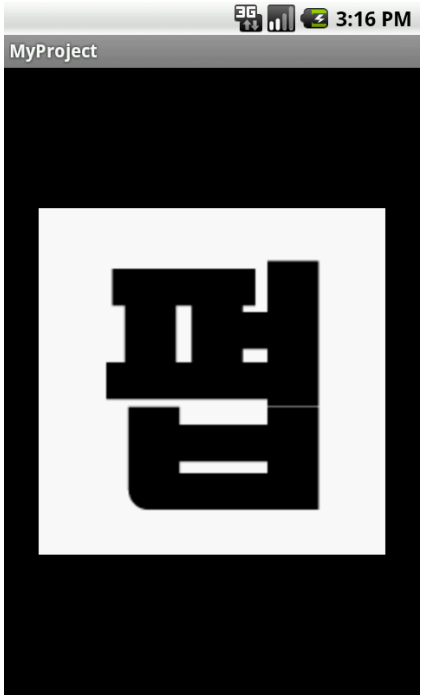
확대를 하게 되므로 뿌옇게 되는 현상 있음

Bitmap 해상도 ldpi 폴더

- 사용자가 보는 물리적인 실제 크기가 화면의 밀도와 상관없이 모두 동일

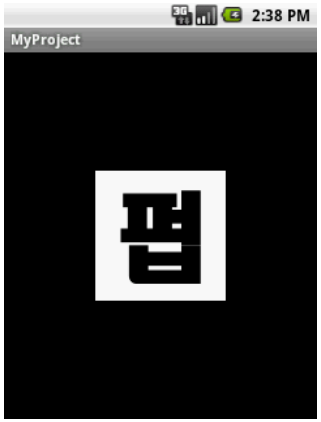


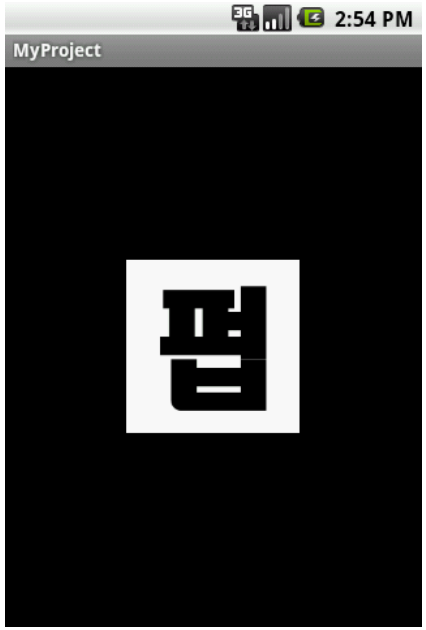


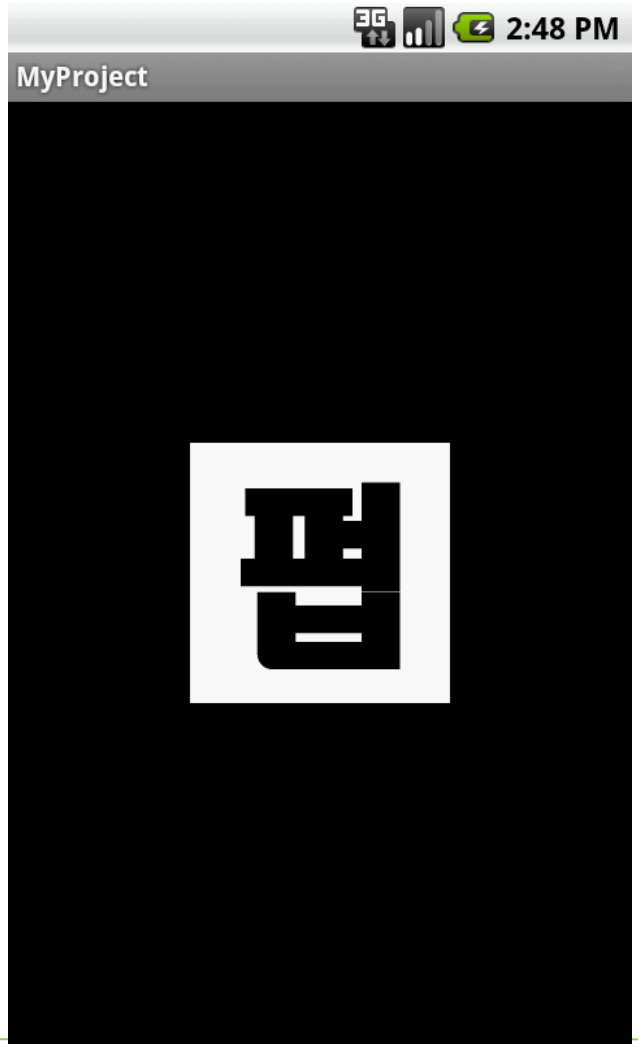


Bitmap 해상도 hdpi 폴더

- 200px · 200px
- 모두 다른 픽셀 크기
- 자동 비트맵 크기 조정 (축소)

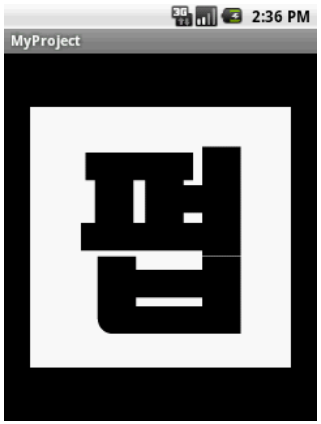


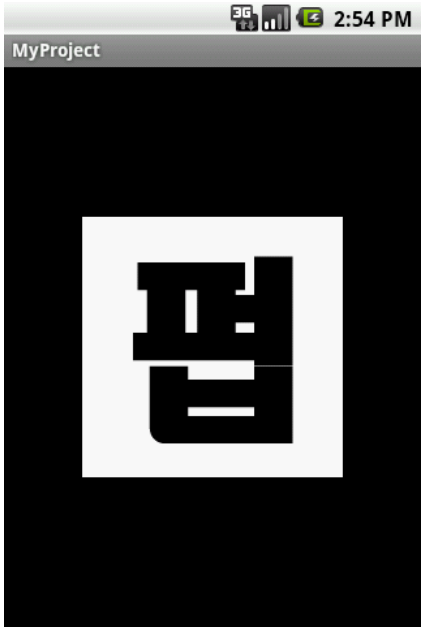


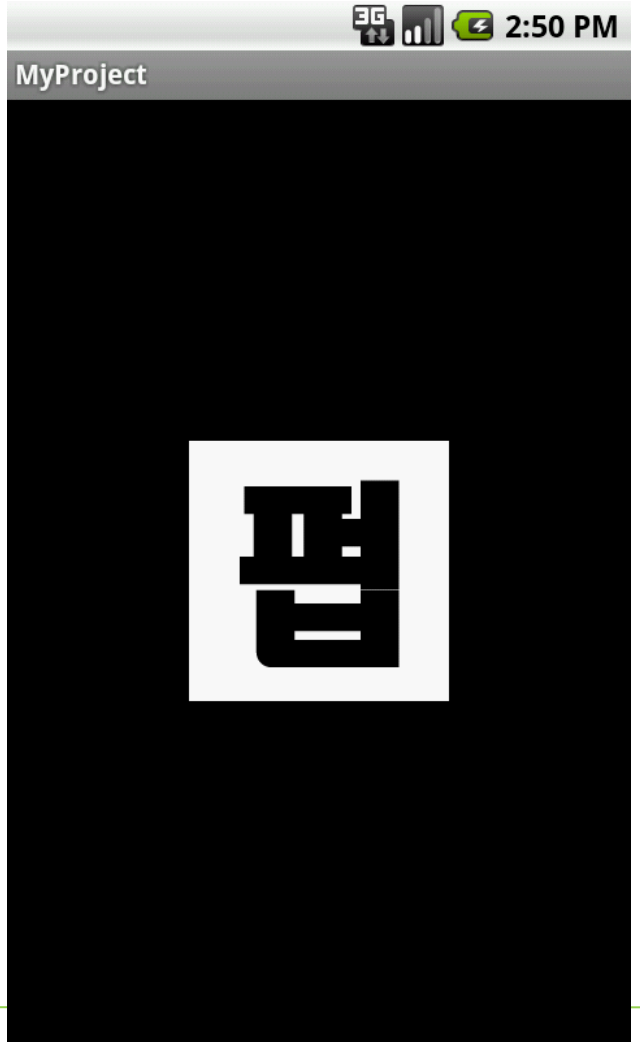


Bitmap 해상도 nodpi 폴더

- 200px · 200px
- 밀도와 관계없이 동일한 픽셀
- 자동 크기 조절 안함

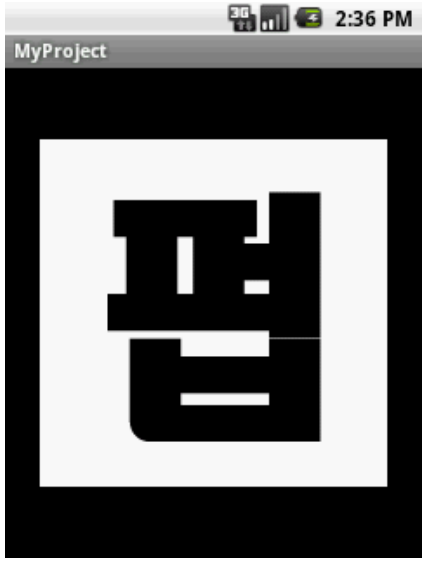


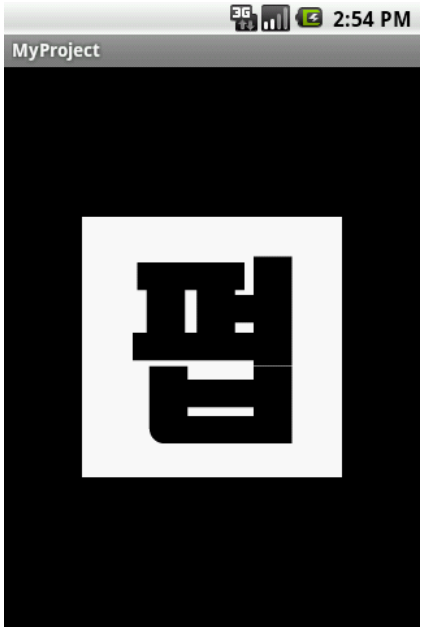


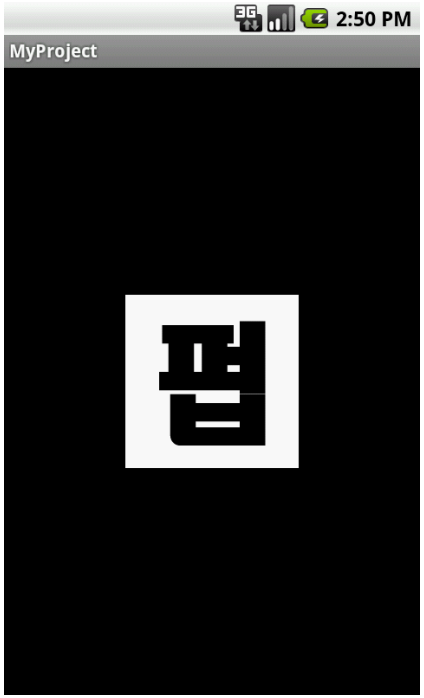


Bitmap 해상도 nodpi 폴더

- 사용자가 보는 물리적인 실제 크기는 화면의 밀도에 따라 차이가 남







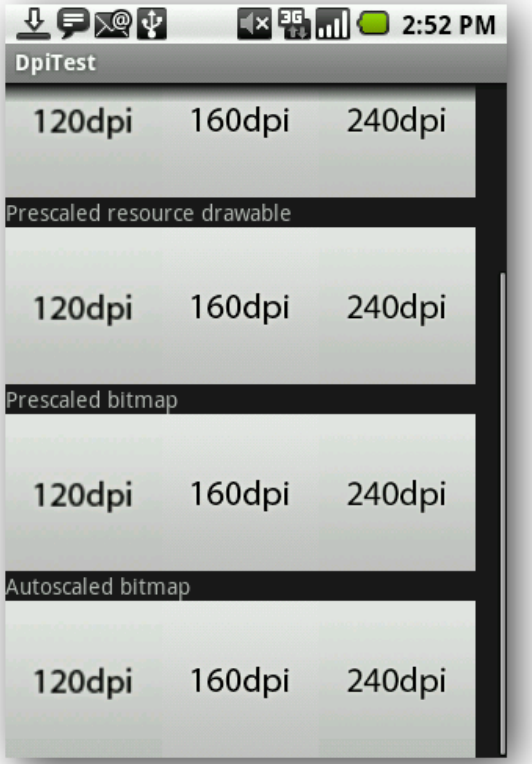
TIP 2 : Bitmap 작성

HDPI를 기준으로
Bitmap 을 작성한다

자동 크기 조정 시 보기 좋다

Pre-Scaling

- 로딩 시간에 크기 조정
- CPU에 이득이 있음



- BitmapFactory.Options
  - inScaled, inDensity, inTargetDensity,

- 예) res/drawable-mdpi/의 100x100 아이콘을 High Density의 스크린에서 로드했을때, 안드로이드는 자동으로 크기를 확대하여 150x150 bitmap을 만든다.

Auto-Scaling

- 그리는 시간에 크기 조절
- 메모리에 이득이 있음

- Bitmap.getDensity()/setDensity()
  - 비트맵에 대한 density 지정
  - 리소스가 아닌 웹, SD카드등에서 데이터를 가져왔을 때

- Bitmap.getScaledHeight()/getScaledWidth()
  - Target Density에 따른 높이와 넓이 구함

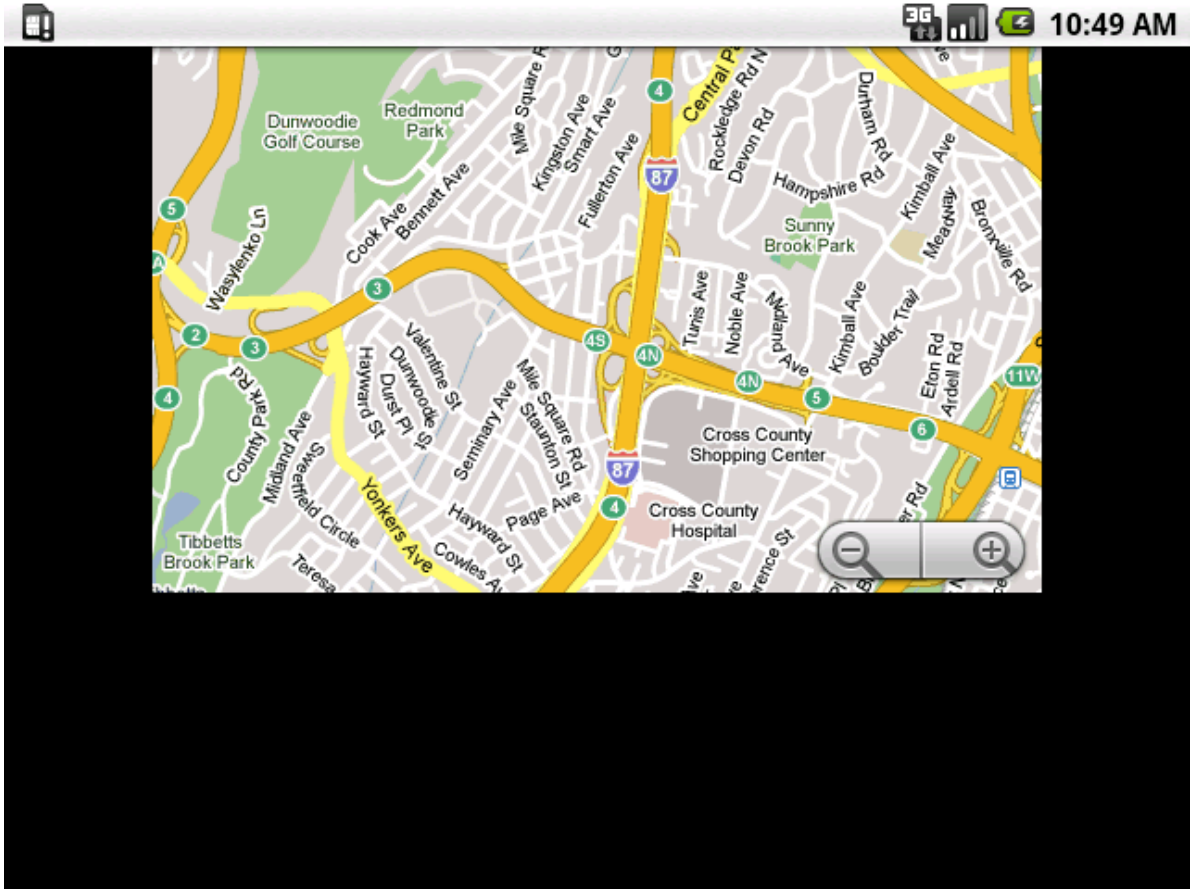
- Bitmap이 Canvas에 그려질때 각각의 Density에 따라 자동으로 크기 조절

Bitmap 의 적용

- Bitmap 리소스는 각 스크린에 맞게 적절히 Resize 되어 적용됨
- 만약 하나의 Bitmap만 만들어 쓴다면? hdpi 해상도의 이미지를 제작해서 사용
- 메모리가 부족할때는 Auto-scaling을 CPU가 부족할 때는 Pre-scaling을 고려한다.

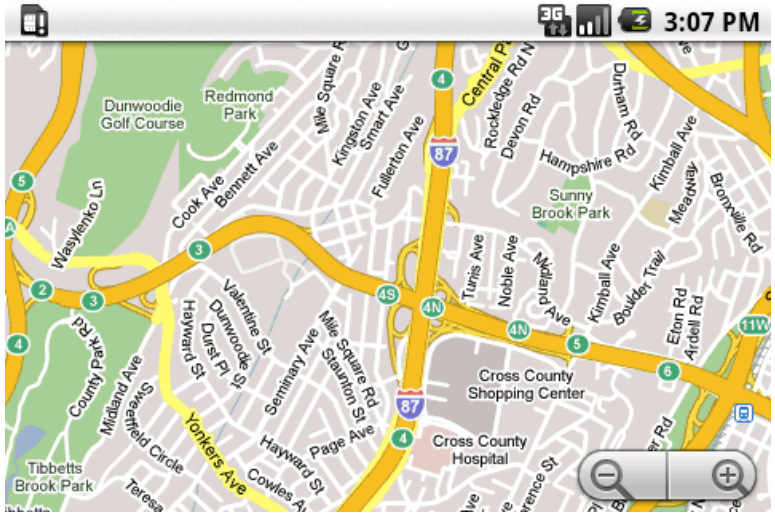
Compatibility Mode (호환 모드)

- Large 스크린을 지원 안하는 앱을 Large 스크린에서 실행하면 검은 배경에 원래 크기만큼의 공간에 표시



자동 픽셀 단위 조절

- 지원안하는 Density에서 실행하는 경우 자동 크기 조절.
  - HVGA Normal Density만 지원하는 앱을 WVGA High Density에서 실행
  - 시스템에서 앱에게 320x533에서 실행되고 있는 것처럼 에뮬레이션을 한다



AndroidManifest.xml

```
<supports-screens
        android:largeScreens="true"
        android:normalScreens="true"
        android:smallScreens="true"
        android:resizable="true"
        android:anyDensity="true"  />
</manifest>
```

다양한 스크린 사이즈 지원

각 장치별로 별도의 레이아웃과 별도의 이미지를 만들면 세밀하게 디자인을 조정할 수 있다. 하지만 관리가 힘들어지므로 안드로이드의 구조를 이해하여 가급적 적은 레이아웃과 이미지로 UI를 구성하는 것이 좋다.

감사합니다.