CS 696 Mobile Phone Application Development
Fall Semester, 2009
Doc 19 Screen Sizes
Nov 19, 2009

References

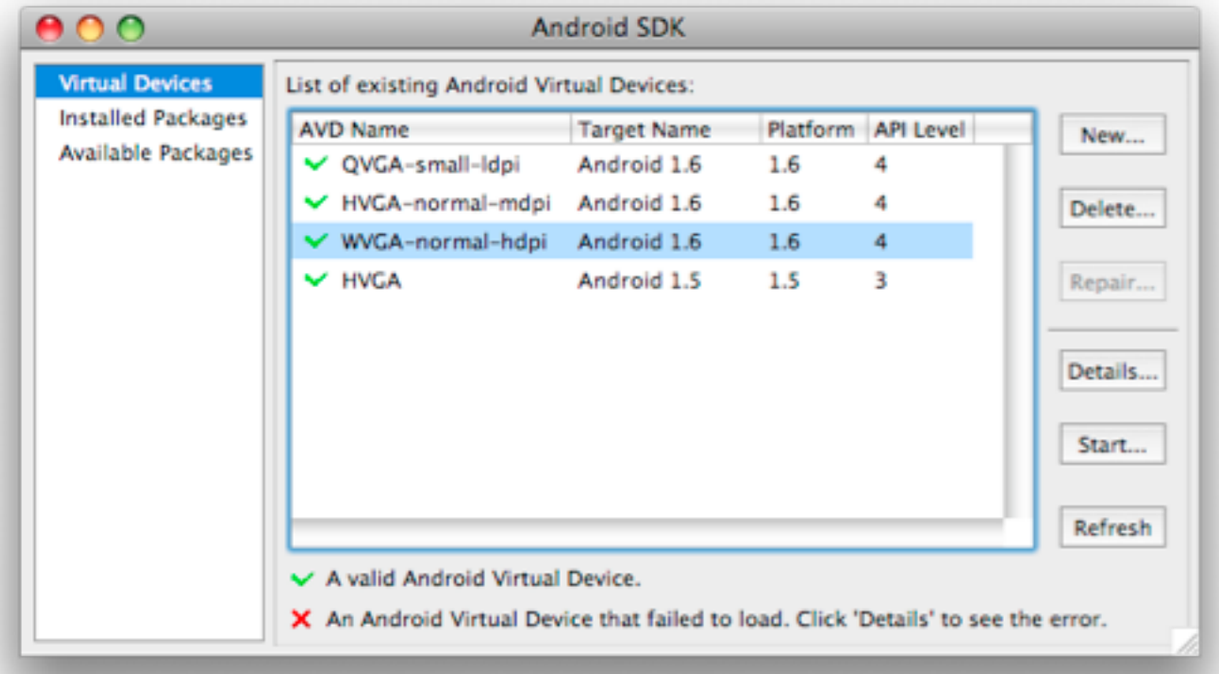
Supporting Multiple Screens,
http://developer.android.com/guide/practices/screens_support.html

Screen Size/Density

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

Test Your Application on Multiple Screens

QVGA (240x320, low density, small screen)
HVGA (320x480, medium density, normal screen)
WVGA800 (480x800, high density, normal screen)
WVGA854 (480x854 high density, normal screen)
Android 2.0
WQVGA400 (240x400, low density, normal screen)
WQVGA432 (240x432, low density, normal screen)



Getting correct size screen

Create an AVD for the target screen size & density

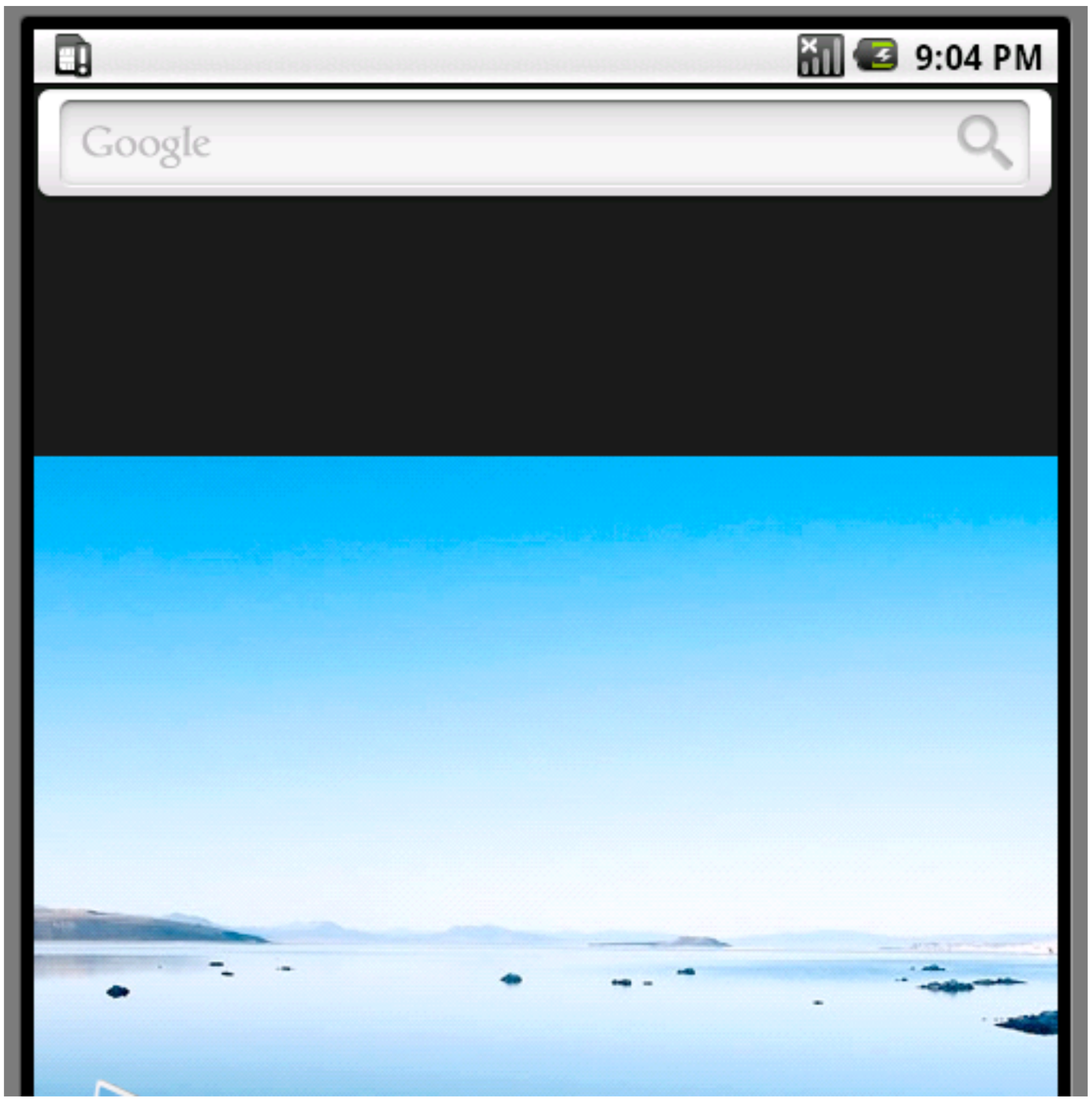
Tell the emulator your monitors density

emulator -avd normal -scale 96dpi

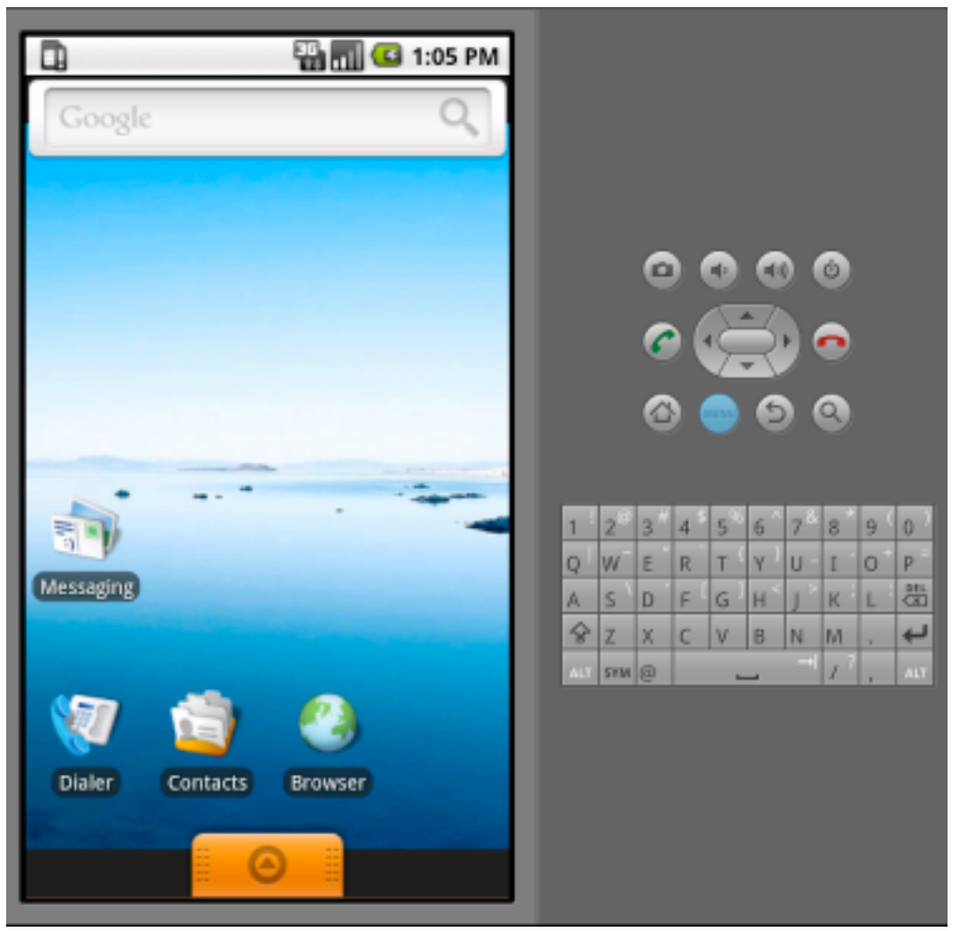
Difference Emulator Sizes

WVGA854

default emulator size



Corrected for monitor density



Supporting Multiple Screens

Resource Files for different size/density

Specify the supported Screen

Specify the supported Screen

android:smallScreens
android:normalScreens
android:largeScreens
android:anyDensity

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android">

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

Resource Files for different size/density

Screen characteristic	Qualifier
Size	small
	normal
	large
Density	ldpi
	mdpi
	hdpi
	nodpi
Aspect ratio	long
	notlong
Platform version	v<api-level>

res directory

res/layout/my_layout.xml // layout for normal screen size
res/layout-small/my_layout.xml // layout for small screen size
res/layout-large/my_layout.xml // layout for large screen size
res/layout-large-land/my_layout.xml // layout for large screen size in landscape mode

res/drawable-ldpi/my_icon.png // icon image for low density
res/drawable-mdpi/dpi/my_icon.png // icon for medium density
res/drawable-hdpi/my_icon.png // icon image for high density

res/drawable-nodpi/composite.xml // density independent resource

Runtime support

Pre-scaling of resources

Load size- or density-specific resources

If not available scale resource to needed size

Runtime support

Auto-scaling of pixel dimensions and coordinates

If application does not support different screen densities

Android auto-scales

- absolute pixel coordinates,
- pixel dimension values, and
- pixel math

Runtime support

Compatibility-mode display on larger screen-sizes

If current screen's size is larger than your application supports

Application is shown in supported size surrounded by black

Density independence

Pre-scaling of drawable resources

Auto-scaling of device-independent pixel (dip) values

Auto-scaling of absolute pixel values used in the application
when android:anyDensity="false"

Best practices for Screen Independence

Prefer wrap_content, fill_parent and the dip unit to px in XML layout files

Avoid AbsoluteLayout

Do not use hard coded pixel values in your code

Use density and/or resolution specific resources

Prefer wrap_content etc

wrap_content, fill_parent & dip
layout_width="100dip"

Font sizes

sp (scale-independent pixel)
Allows the user to scale the text

dip
User can't scale the text

AbsoluteLayout

deprecated in Android 1.5

Use FrameLayout

set layout_margin of children

Don't hard-coded pixel values in your code

But you may need to know the actual pixel

scroll gesture -
triggered when move by at least 16 pixels

HVGA
16 pixels / 160 dpi = 1/10th of an inch

high (240) density display
16 pixels / 240 dpi = 1/15th of an inch

getContext().getResources().getDisplayMetrics().density

scaling factor for the Density Independent Pixel

160dpi screen - 1.0
120dpi screen - 0.75
240dpi screen - 1.5

```
private static final float GESTURE_THRESHOLD_DIP = 16.0f;

// Convert the dips to pixels
final float scale = getContext().getResources().getDisplayMetrics().density;
mGestureThreshold = (int) (GESTURE_THRESHOLD_DIP * scale + 0.5f);
```

Use pre-scaled configuration value

android.view.ViewConfiguration

ViewConfiguration.get(aContext).getScaledTouchSlop()

getScaledXXX()

returns values in pixels that will display properly

Pre-scaling, auto-scaling

resources

pre-scaled before displayed

value of android:anyDensity does not affect pre-scaling

android:anyDensity=true

resources are pre-scaled

original sizes are reported

If you modify the resource odd things may happen

nodpi resources are not scaled

BitmapFactory.Options

gives control over pre-scaling

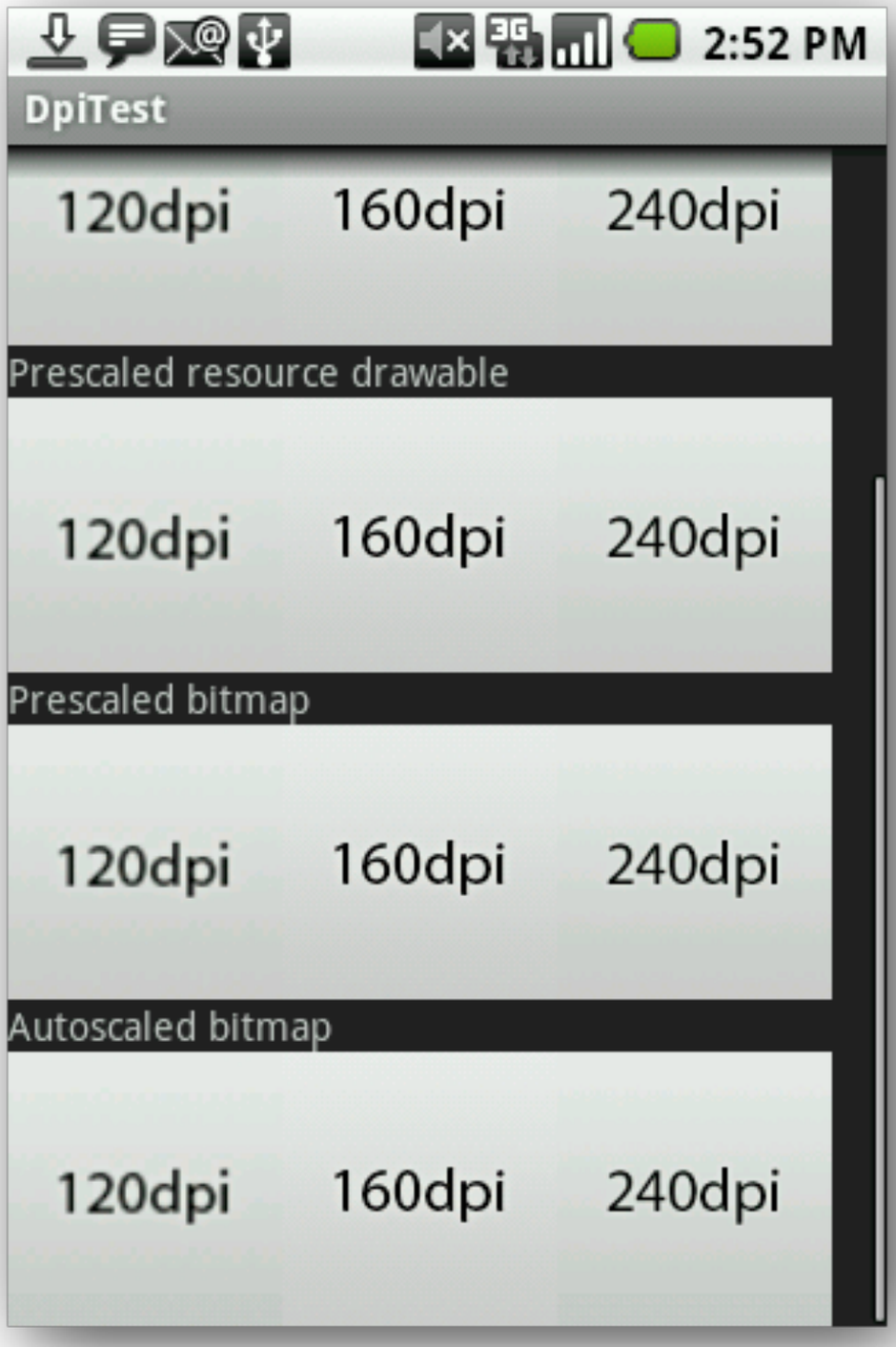
Pre-scaling, auto-scaling

auto-scaling

Done when creating bitmaps in memory and drawing on them

Drawing on the larger bitmap is different than scaling after drawing

Pre-scaling versus auto-scaling



Effects of loading low(120dpi) medium(160)dpi & high(240 dpi) bitmaps on a medium screen