От теорията към практиката

Калин Кадиев

Astea Solutions

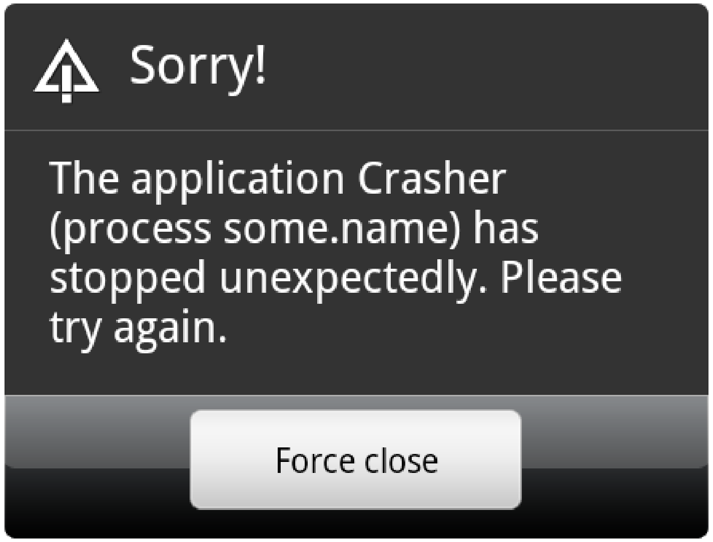
What we'll talk about

- Debugging
- Optimizing
- Compatibility

Reusing android code

- Android software stack license – Apache 2.0
- Download Android source - Repo and Git, source.android.com

Dialog of Death



Sorry!

The application Crasher (process some.name) has stopped unexpectedly. Please try again.

Logcat

- Provides a mechanism for collecting and viewing system debug output
- System.out and System.err go to “/dev/null”
- Log only important messages - avoid flood

Logcat logging

- Log.<method>(tag, message);
- Log.<method>(tag, message, error);
- tag: getClass().getSimpleName();
- message: any String (not null)

Logcat usage

- [adb] logcat [<option>] ... [<filter-spec>] ...
- option:
  - f <filename>
  - v <format>
  - ...
- filter-spec: tag:priority

Logcat filters

Level	**Log method**	**Logcat priority**
Verbose	v	V
Debug	d	D
Info	i	I
Warn	w	W
Error	e	E
Fatal / What a terrible Failure	wtf	F
Silent	-	S

Demo

Dalvik Debug Monitor Server (DDMS)

- screen capture on the device
- thread and heap information on the device
- Logcat
- File explorer

DDMS

- process and radio state information
- incoming call and SMS spoofing
- location data spoofing
- more.

Traceview

Priceless for performance optimizations!

Creating Trace Files

```
// start tracing to "/sdcard/bu.trace"
Debug.startMethodTracing("bu");
// ...
// stop tracing
Debug.stopMethodTracing();
```

Viewing Trace Files

```
adb pull /sdcard/<name>.trace /<dir>

traceview <dir>/<name>.trace
```

Demo

Traceview hints

- Always remove trace calls
- Don't try to generate absolute timings from the profiler results

Hierarchy Viewer

- Connect your device or launch an emulator.
- Launch hierarchyviewer.
- Select a device.
- Select the window that you'd like to inspect and click Load View Hierarchy.

Demo

Layout Hints

- The fewer layouts – the better
- No point of layout with a single children
  - another layout
- Avoid LinearLayout.

layoutopt

- Helps you optimize the layouts and layout hierarchies of your applications.
- Usage: *layoutopt <resources>*

Android NDK

- A toolset that lets you embed components that make use of native code in your Android applications
- Used mainly for code reusage.
- “using native code does not result in an automatic performance increase, but does always increase application complexity”

Supporting multiple screens

Terms

- **Screen size**: Actual physical size, measured as the screen's diagonal. Large, normal, small.
- **Resolution**: The total number of physical pixels on a screen.
- **Density**: The spread of pixels across the physical width and height of the screen.

Terms

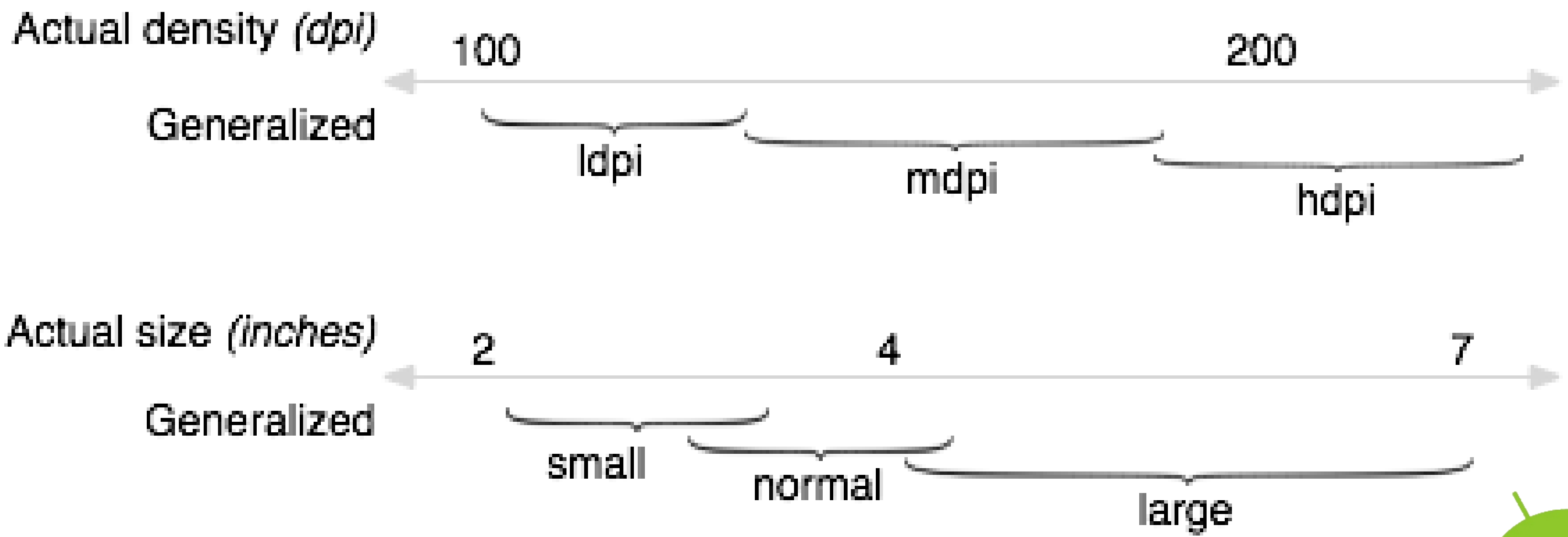
- **Density-independent pixel (dp)**: A virtual pixel unit that expresses layout dimensions or position in a density-independent way.

➢ *pixels = dips * (density / 160)*

Terms

- **Scale-independent Pixel (sp):** Like the dp unit, but also scaled by the user's font size preference. Recommended for font sizes.

Generalization of densities and screen sizes



The <support-screens> tag

- *android:smallScreens*
- *android:normalScreens*
- *android:largeScreens*
- *android:anyDensity*
- Default values : true to all since 1.6

Resource directory qualifiers

Size

- *small*
- *normal*
- *large*

Resource directory qualifiers

Density

- n*odpi*: not auto-scaled by platform, regardless of the current screen's density
- *mdpi*
- *hdpi*
- *ldpi*

Resource directory qualifiers

Aspect ratio

- *long:* significantly taller or wider than the baseline configuration
- notlong

Platform version

- v<api-level>

Best practices

- 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

Q & A