Do's and Don'ts on Android

Experiences from a successful project

software pilots
TRIFORK.

Overview

- Introduction
- Trifork & Me
- The project
- Live Demo
- Our Setup
- High level architecture
- The technical stuff
  - Camera
  - Animation / Graphics
- Wrap-up
- Questions

Trifork & Me

- Danish, living in CH for 3 years.
- I work for Trifork GmbH (The Swiss Branch) and this project since sept. 2010.
- Background in embedded programming
  - C/C++
  - Optimizing for speed and memory
  - Very small platforms, few resources
  - Brought up with Java at University

The project

- Provide mobile banking on smart phones for the biggest bank in northern Europe.

- For release as independent packages in 5 different countries:

  DK, SE, FI, NO, EN(*).

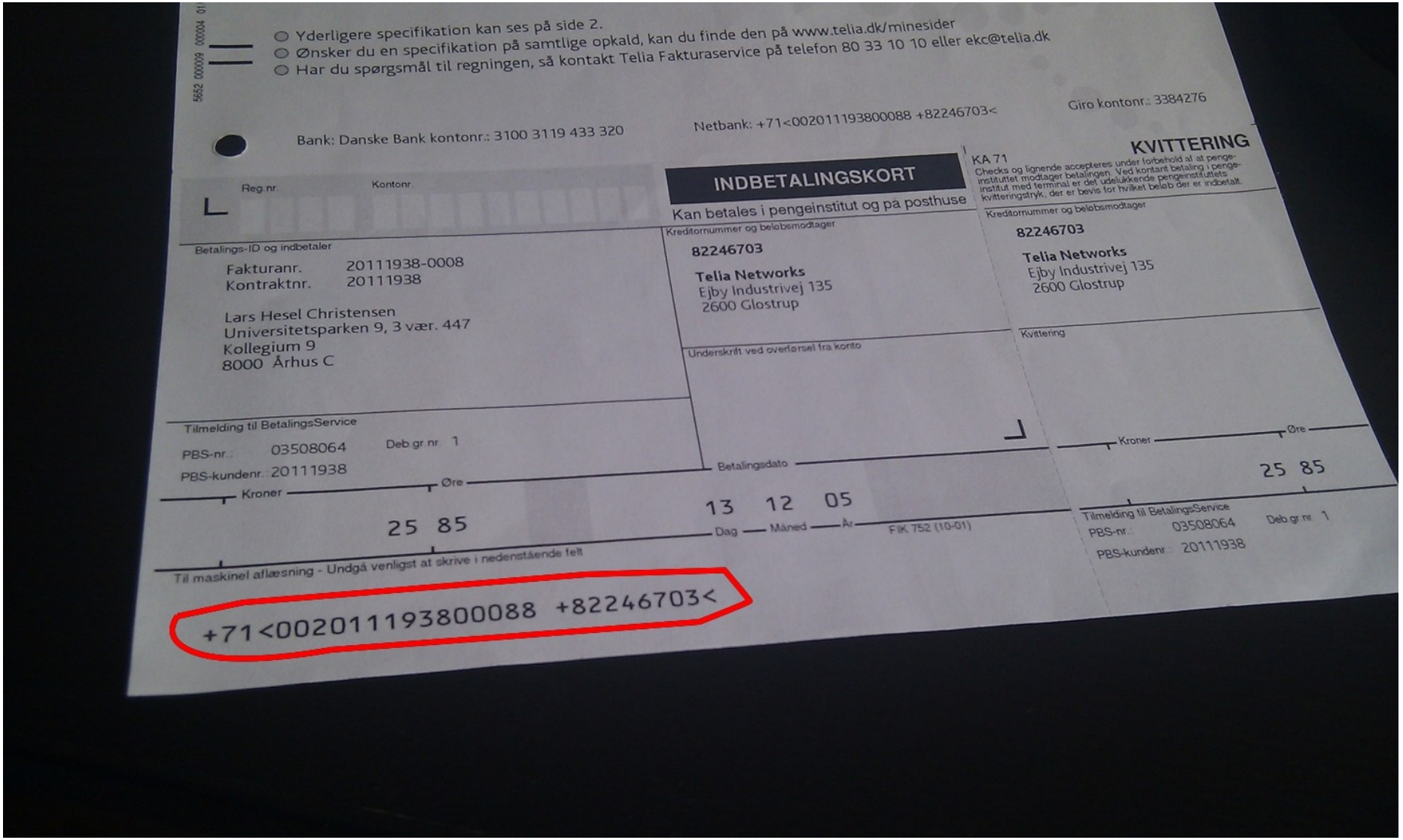
What we achieved

- To be first movers
- To deliver an extremely popular mobile banking solution.
- The most advertised piece of software (in Denmark at least).
- Was top download from the Android Market for weeks.
- A very happy customer.

Features

- View accounts
- Transfers (domestic)
- Payments (Camera/OCR/Barcodes)
- Locate ATMs and branches
- Currency calculator

Live Demo...



Yderligere specifikation kan ses på side 2.
Ønsker du en specifikation på samtlige opkald, kan du finde den på www.telia.dk/minesider
Har du spørgsmål til regningen, så kontakt Telia Fakturaservice på telefon 80 33 10 10 eller ekc@telia.dk

5562 000009 000004 01

Bank: Danske Bank kontonr.: 3100 3119 433 320

Netbank: +71<002011193800088 +82246703<

Giro kontonr.: 3384276

Reg.nr. Kontonr.

INDBETALINGSKORT

Kan betales i pengeinstitut og på posthuse

Betalings-ID og indbetaler

Fakturanr. 20111938-0008
Kontraktnr. 20111938

Lars Hesel Christensen
Universitetsparken 9, 3 vær. 447
Kollegium 9
8000 Århus C

Kreditornummer og beløbsmodtager

82246703

Telia Networks
Ejby Industrivej 135
2600 Glostrup

Underskrift ved overførsel fra konto

Tilmelding til BetalingsService

PBS-nr.: 03508064 Deb.gr.nr. 1
PBS-kundenr.: 20111938

Kroner Øre

25 85

Betalingsdato

Dag 13 Måned 12 År 05

FIK 752 (10-01)

Til maskinel aflæsning - Undgå venligst at skrive i nedenstående felt

+71<002011193800088 +82246703<

KA 71

KVITTERING

Checks og lignende accepteres under forbehold af at pengeinstituttet modtager betalingen. Ved kontant betaling i pengeinstitut med terminal er det udelukkende pengeinstituttets kvitteringstryk, der er bevis for hvilket beløb der er indbetalt.

Kreditornummer og beløbsmodtager

82246703

Telia Networks
Ejby Industrivej 135
2600 Glostrup

Kvittering

Kroner Øre

25 85

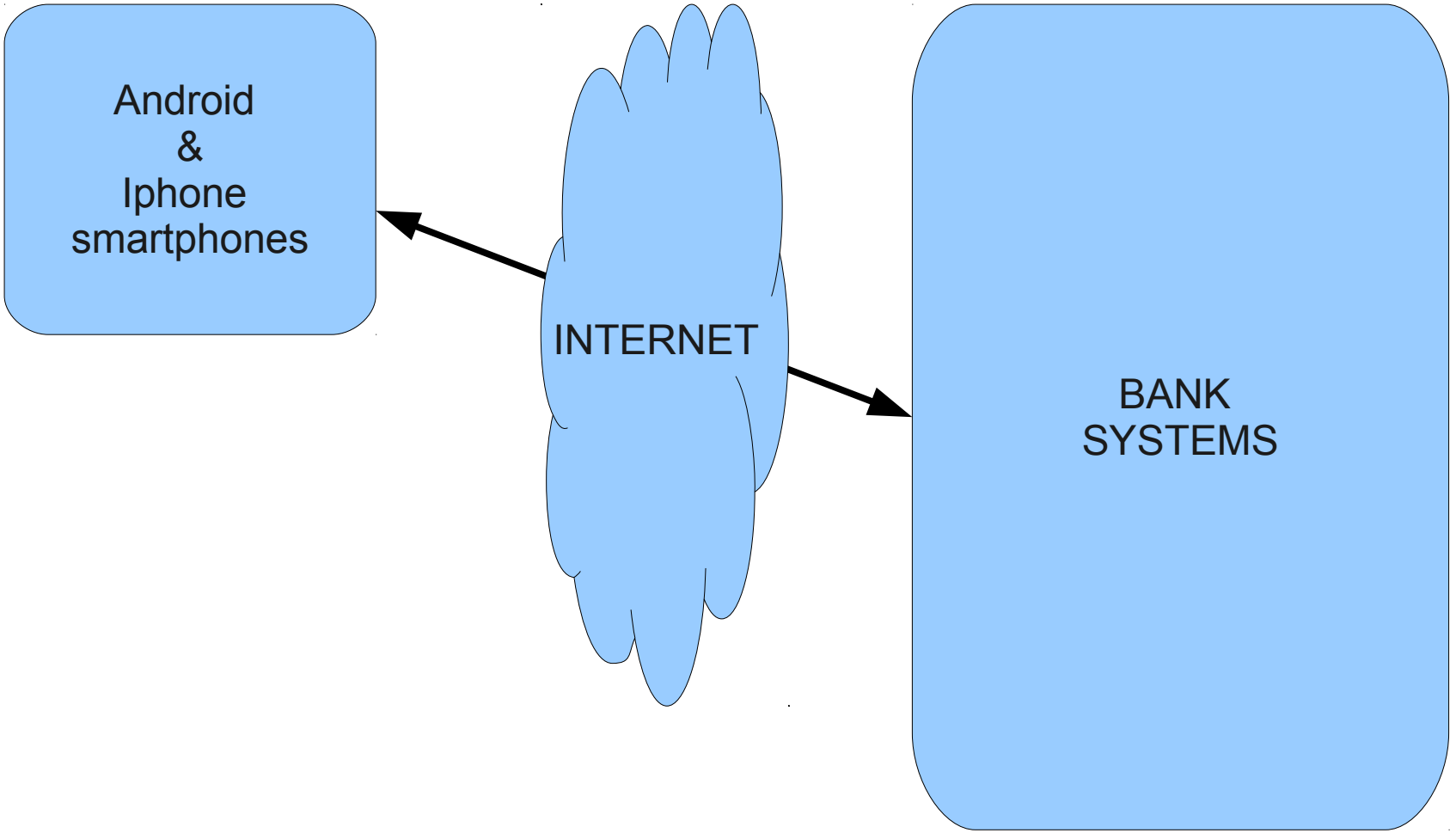
Tilmelding til BetalingsService

PBS-nr.: 03508064 Deb.gr.nr. 1
PBS-kundenr.: 20111938

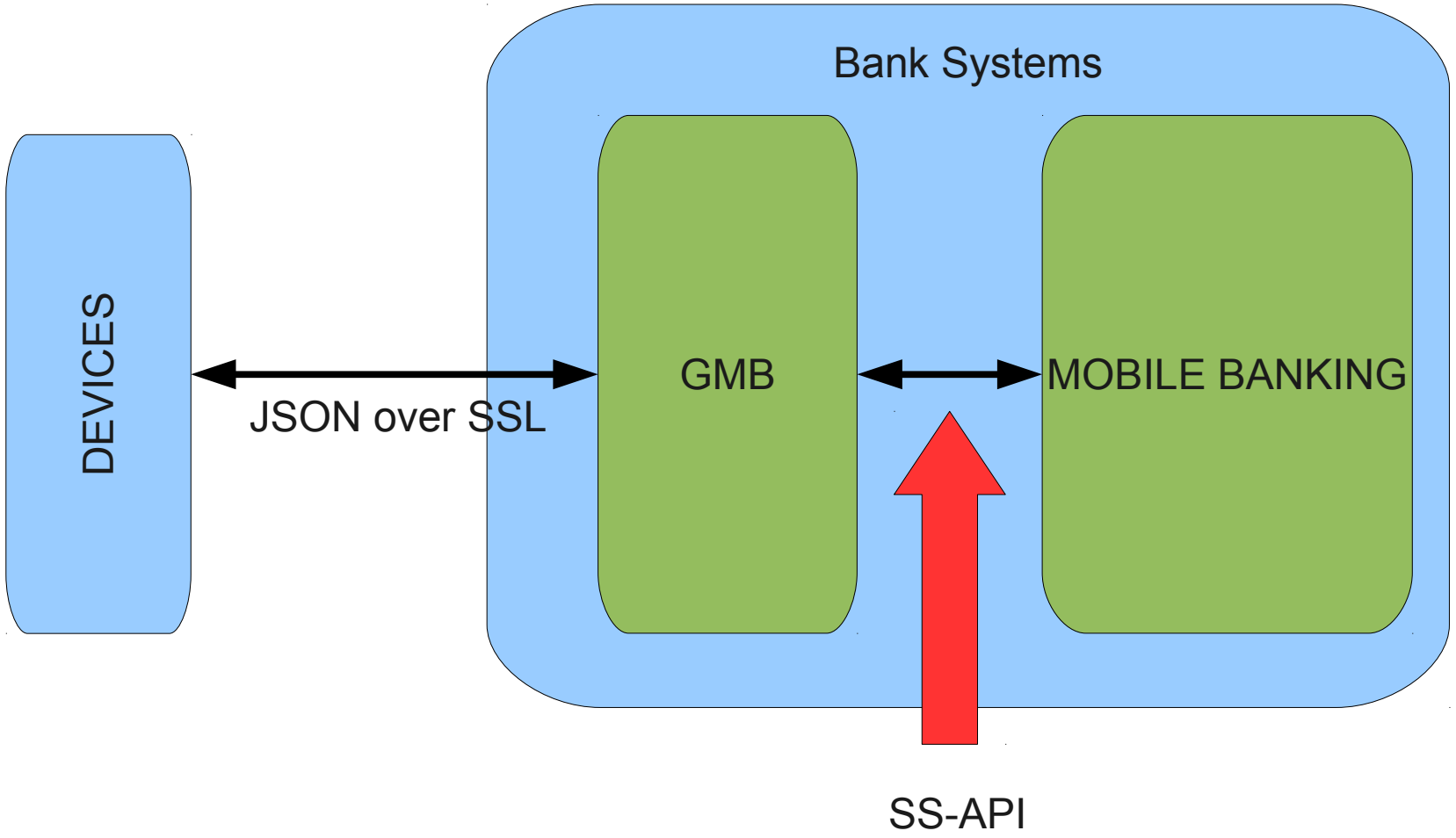
The Setup

- Eclipse (OS X, Linux, Windows)
- Android SDKr9
- Mercurial
- Build server: Jenkins (formerly known as Hudson)
- Custom ant build scripts
- Scrum
- Good people (placed in CH and DK)

High level architecture



Architecture



Cameras

- Are difficult:
  - Must work on Android 1.6 (API Lvl4)
    - which has bad camera support
  - Different devices have very different cameras (or behavior)
  - Need to minimize network traffic
    - Minimize picture resolution

Camera API

- API Lvl 4:
  - Does not support

    public List<Integer> getSupportedPictureFormats ()
  - Possible solution Cameras.Parameters.flatten():
    - sharpness-max=30;zoom=0;taking-picture-zoom=0;zoom-supported=true;sharpness-min=0;…;picture-size-values=2592x1952,2592x1728,2592x1552,2560x1920,2560x1712,2048x1536,2048x1360,2048x1216,2016x1344,1600x1200,1584x1056,1280x960,1280x848,1280x768,1248x832,1024x768,640x480,640x416,640x384,624x416,512x384,400x400,272x272;....;contrast-min=0;picture-size=1024x768;max-zoom=5;effect=none;saturation=5;whitebalance-values=auto,incandescent,fluorescent,...
  - Together with: public void set (String key, String value)

Camera API

- On phones API > 4: try reflection to get / set picture sizes...
- Server architecture to the rescue:
  - It is possible to send 'overrides' form the server.
  - As a last resort: disable functionality.

- This is a general solution to the problem of a very diverse device eco-system! (If you use a server architecture)

Animations

- Initially we did this wrong:
  - Layers
  - Many pieces of graphics
  - Slow onDraw impl.
  - Nice Object Oriented Coding

- Result: 10-12 FPS.

Animations
Slow onDraw impl

```
Matrix mtx = new Matrix();
mtx.postRotate(bitmapAngle);
Bitmap.createBitmap(bitmap, 0, 0,
bitmap.getWidth(), bitmap.getHeight() , mtx, true);
```

- Replaced with rotating canvas (eliminating creation + rotating bitmaps): +5-6 FPS.

Animations
Do not do: Nice OO

```
protected void OnDraw() {
    ...
    drawInCenter(canvas, rotatedBitmap);
    ...
}
```

- A extends B extends View; drawInCenter() implemented in B.
- Getting rid of B and implementing method locally: +4 FPS.

Animations
Do not do: Nice OO

- An experiment:
  - A extends B; B extends C.
  - A and B have method

    ```
    public void exec() { super.exec(); }
    ```
  - C has method

    ```
    public void exec() { a++;}
    ```

Animations
Do not DO: Nice OO

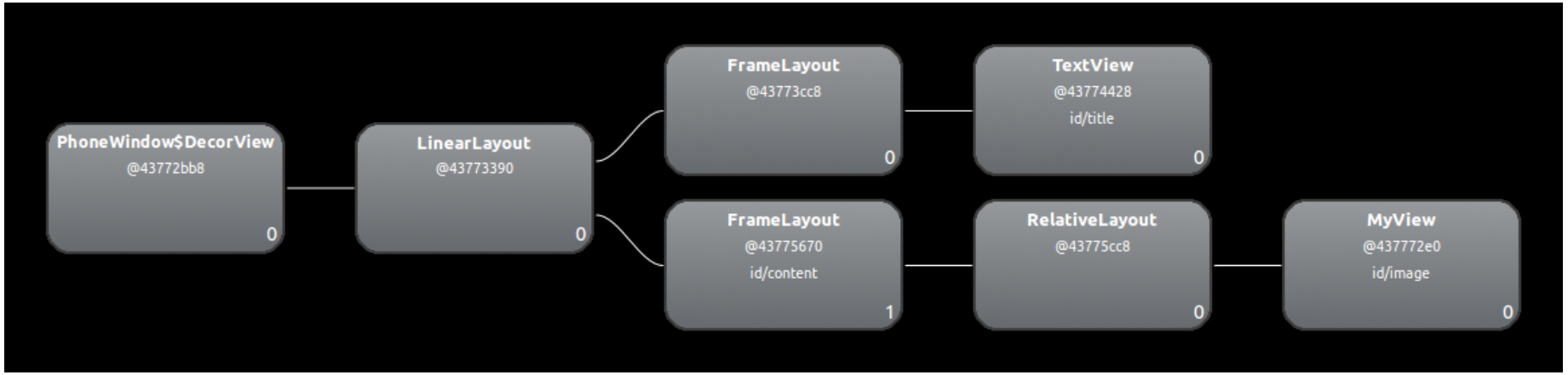
- A a = new A(); B b = new B();

Hierarchy height	2 (b.exec();)	3 (a.exec();)
Execution time	~11.3s	~16.5s
Relative exec. time	+0%	+46%

- Do do nice OO – just not where it hurts!
- Do learn where it hurts!

An (easy & small) optimization

- This one described on Google-dev:
  - Remove the DecorView background image on opaque & full screen applications.



  - This gave me a 6% performance boost, going from 44 to 47 FPS in an application.

An (easy & small) optimization

- Create theme (res/values):

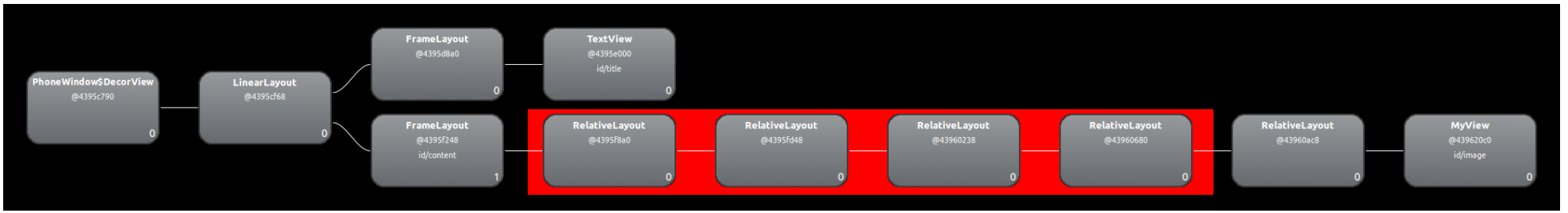
```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="Theme.NoBackground" parent="android:Theme">
        <item name="android:windowBackground">@null</item>
    </style>
</resources>
```

- Use it (AndroidManifest.xml):

```
<application android:icon="@drawable/icon" android:label="@string/app_name"
android:theme="@style/Theme.NoBackground">
    ...
</application>
```

Another (“easy & small”) optimization

- What's the cost of complicated view hierarchies?



- Wrapped view in 4 RelativeLayouts
- Each RL had a cost of 2-4 FPS!

Crash reports

- Android market gives us a view of the health of our products.
  - Exception traces
  - Versions
  - Statistics.
- Does not tell us:
  - Device type,
  - Make & model,
  - os version,
  - changeset number etc.

Crash reports

- That is easy:
  - Create your own UncaughtExceptionHandler
  - Make it put whatever your heart desires in the crash report
- BUT:

  Make sure it does not crash!

Crash handler example

```
public class MyUncaughtExceptionHandler extends Object implements
UncaughtExceptionHandler {

    private final UncaughtExceptionHandler defaultHandler;

    public MyUncaughtExceptionHandler(Activity app) {
        defaultHandler = Thread.getDefaultUncaughtExceptionHandler();
    }

    public void uncaughtException(Thread thread, Throwable e) {

        StringBuilder report = new StringBuilder(e.toString());

        // add stuff to the report.

        Throwable t = new Throwable(report.toString(), e);
        defaultHandler.uncaughtException(thread, t);
    }
}
```

CHANGESET: 29616dab2e52

BOARD: bravo
BRAND: htc_wwe
CPU_ABI: armeabi-v7a
DEVICE: bravo
DISPLAY: bravo
FINGERPRINT: htc_wwe/htc_bravo/bravo/bravo:2.2/FRF91/293415:user/release-keys
HOST: AA108
ID: FRF91
MANUFACTURER: HTC
MODEL: HTC Desire
PRODUCT: htc_bravo
VERSION.CODENAME: REL
VERSION.INCREMENTAL: 293415
VERSION.RELEASE: 2.2
VERSION.SDK: 8
VERSION.SDK_INT: 8

---

at com.trifork.android.CrashHandler2.MyUncaughtExceptionHandler.uncaughtException(MyUncaughtExceptionHandler.java:50)

at java.lang.ThreadGroup.uncaughtException(ThreadGroup.java:887)

at java.lang.ThreadGroup.uncaughtException(ThreadGroup.java:884)

at dalvik.system.NativeStart.main(Native Method)

Caused by: java.lang.OutOfMemoryError

...

Feedback

- Read the comments on your Google market account:
  - Get clues on weird behavior.
  - Get great ideas for improvements.
  - Get good and bad criticism.
  - Have a laugh :) (some comments are really funny)

- More importantly:
  - *Do take the feedback seriously!*

For the nerds!

- The source is available.
- You can build the entire Android OS yourself!
- You can read the source code.
- Can be helpful if documentation is lacking / insufficient.
- Can give you deep understanding of the system (OS Framework / Dalvik / Camera handling …)
- Debug / step through OS Code...

Know your tools

- Eclipse (Through DDMS)is bundled with some strong tools:
  - Profiler
  - Memory statistics
  - Hierarchy Viewer
  - logcat
  - Much more...

Summary

- Do use the server! (if you have one)
- Do create annotated crash reports. (But KISS)
- Do: Google has an great collection of technical articles. All great reads. Read them!
- Do lots of experiments. Get real experience! Experiment to know what really matters and what not.
- Do: Use the tools! Inspect layouts, use the profiler, use DDMS etc.
- Do take feedback seriously!

Summary

- Prevent any unnecessary memory allocations.
- Particularly in your onDraw methods:
  - precalculate whatever you can
  - Remove memory allocs. The GC will come for you.
- Do sanitize your layouts. Keep the hierarchies low!
- Do keep the number of graphics down.
- Do not try to do nice OO (where it matters!)

Questions?

THANK YOU!