# Android SDK

von Simon Kostede
für das Seminar „Android: Plattform für mobile Geräte“
an der Universität Hamburg beim DKRZ

# Inhalt

- Einführung
- Bestandteile des SDKs
  - Tools
- Debugging
  - Performance
- NDK

# Ziel

- Übersicht über das SDK
  - Wo bekomme ich es?
  - Wie funktioniert es?
  - Was ist enthalten?
- Debugging
  - Was bietet das SDK zum Debuggen?
  - Wie überprüfe ich die Performance meins Programms?

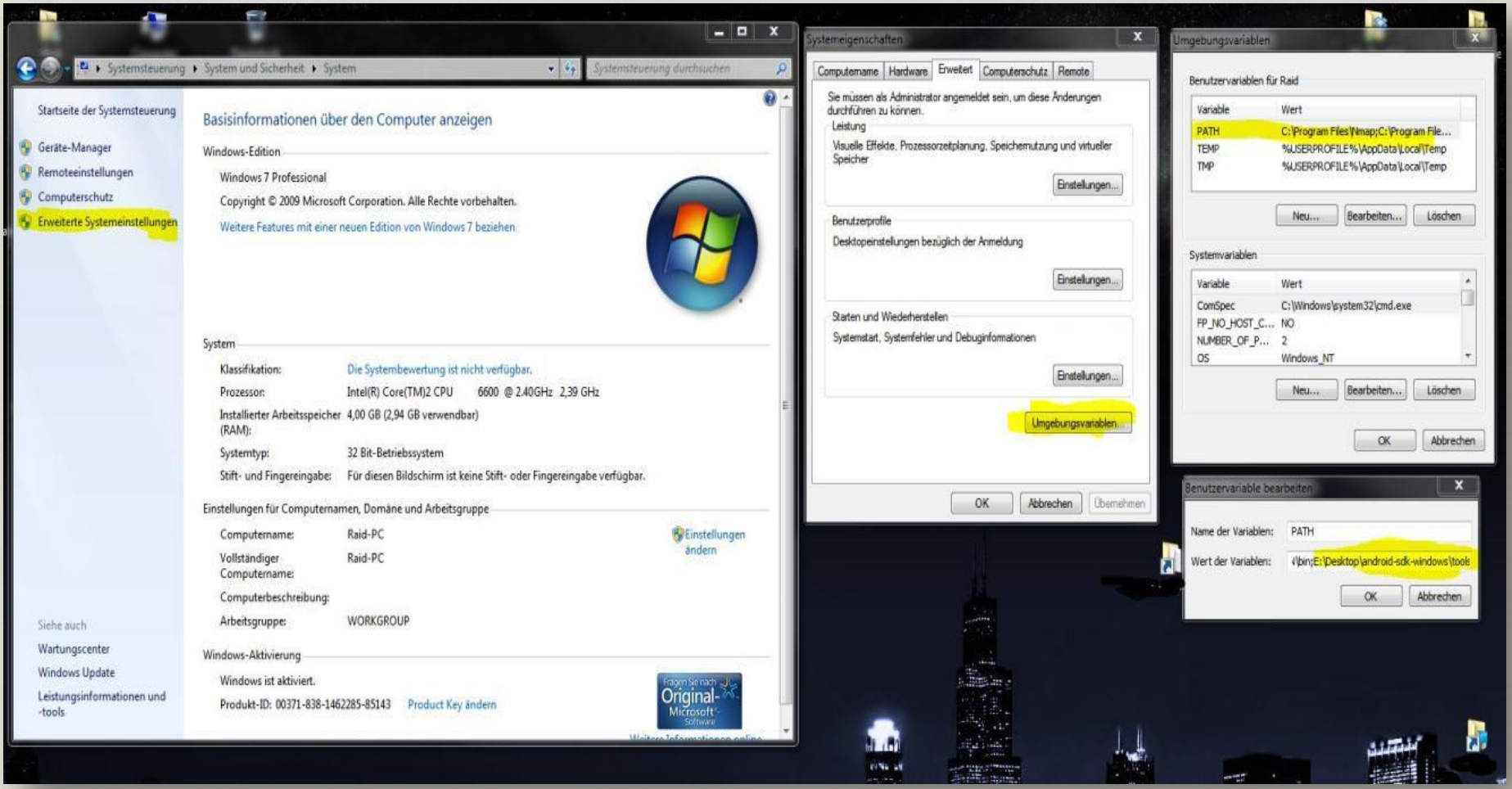
# Inhalt


- Einführung
  - **Installation**
    - Windows
    - Linux
    - IDE/Eclipse
  - Grundlegende Funktionen
- Bestandteile des SDKs
- Debugging
- NDK

# Installation

- Download von http://developer.android.com/sdk/index.html
- Entpacken (Irgendwo)
  - android-sdk-<Plattform>
    - android-sdk-linux
- JDK überprüfen (>= 5)
  - android-sdk-<Plattform>/tools zu PATH hinzufügen

# Installation: Windows



# Installation: Linux

- In /home/<Benutzer>
  - ~/.bash_profile oder ~/.bashrc ändern
  - Nach PATH suchen
    - Oder selber machen: export

      PATH=${PATH}:<*dein_sdk_Pfad*>/tools
  - tools/ hinzufügen
- 32bit Kompatibilität sicherstellen
  - Ubuntu z.B. apt-get install ia32-libs

# Installation: IDE

- Android macht's mit allen
  - (auch Emacs)
- Am besten: Eclipse
  - Eigenes Plugin von Google
    - Kontrolle des Emulators/Device
    - Syntaxvervollständigung
    - Etc.

# Installation: IDE

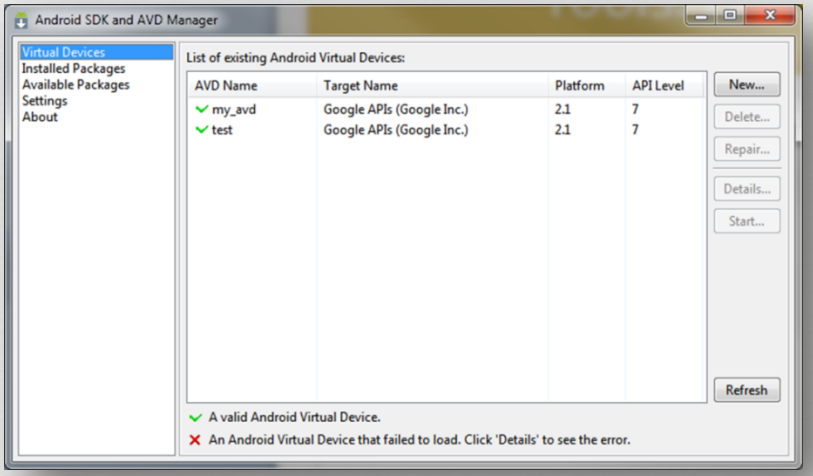
- ADT für Eclipse
  - Neu: 0.9.6 – März 2010
  - Help -> New Software
  - https://dl-ssl.google.com/android/eclipse/
    - Bei Problemen http nutzen
- Pfad zum SDK
  - Window -> Preferences -> Android -> SDK Location

# Inhalt

- Einführung
  - Installation
  - **Grundlegende Funktionen**
- Bestandteile des SDKs
- Debugging
- NDK

# Android Plattformen

- Erstellen von AVDs
- API-Level aussuchen
  - z.Z. Android 2.1 Eclair
  - Dokumentation
  - Updates
  - Beispiele
  - API Addons
    - z.B. Google APIs



# Inhalt

- Einführung
- Bestandteile des SDKs
  - **Tools**
- Debugging
- NDK

# Inhalt

- Tools
  - **Erstellen eines AVDs**
  - **Erstellen eines Projekts**
  - **Anwendung starten**
  - **SD-Karte erstellen**
  - Skalierbares UI
  - Aufbau des UIs
  - Automatische UI Analyse

# Erstellen eines AVDs

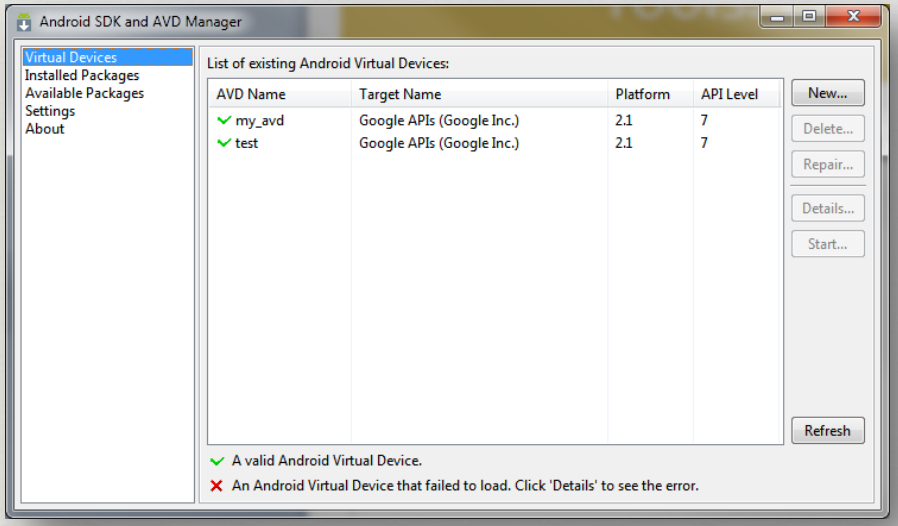
- Tool: android oder GUI

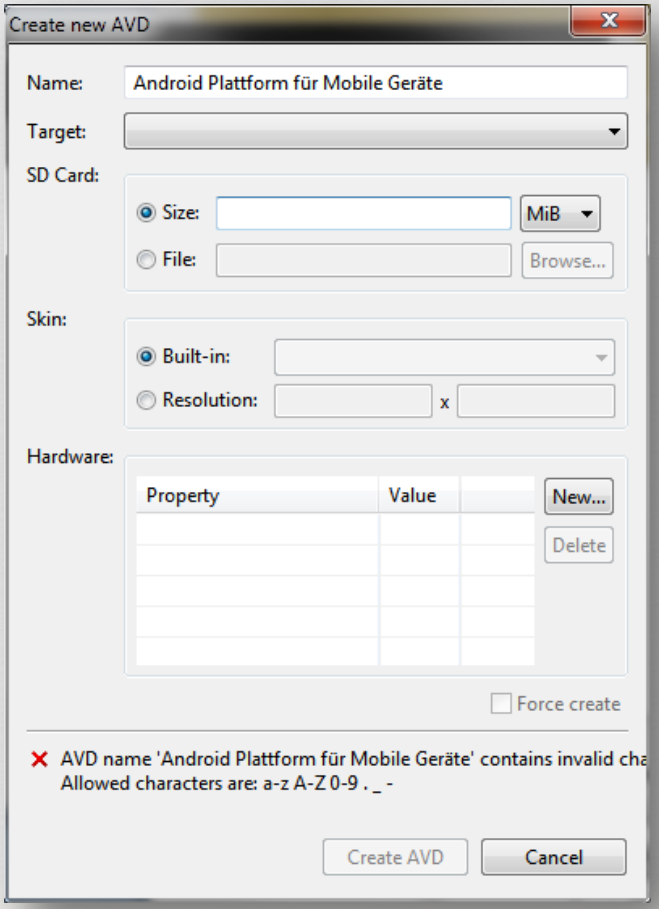
```
$ android create avd -n <Name> -t <targetID>
```

- Welche Targets?

```
$ android list targets
Available Android targets:
id:1
    Name: Android 1.1
    Type: platform
    API level: 2
    Skins: HVGA (default), HVGA-L, HVGA-P, QVGA-L, QVGA-P
```

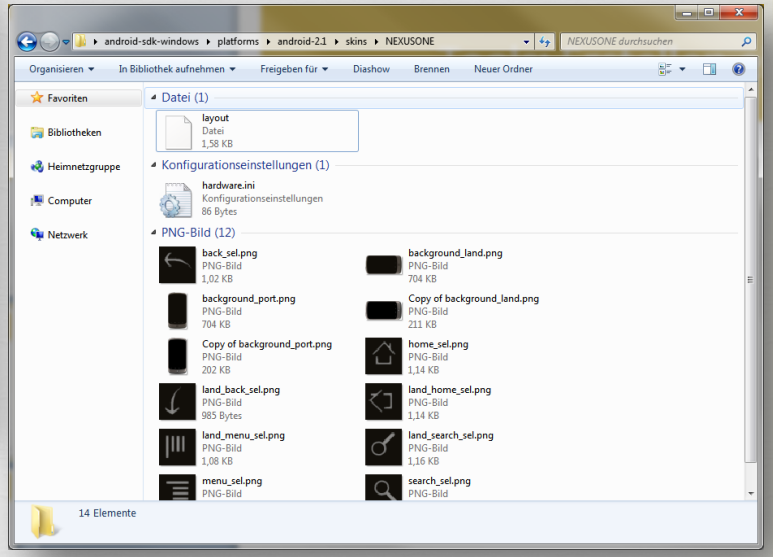
# Erstellen eines AVDs



# Erstellen eines AVDs

- Möglichkeit Skins zu benutzen
  - Nexus One Skin:



# Projekt erstellen

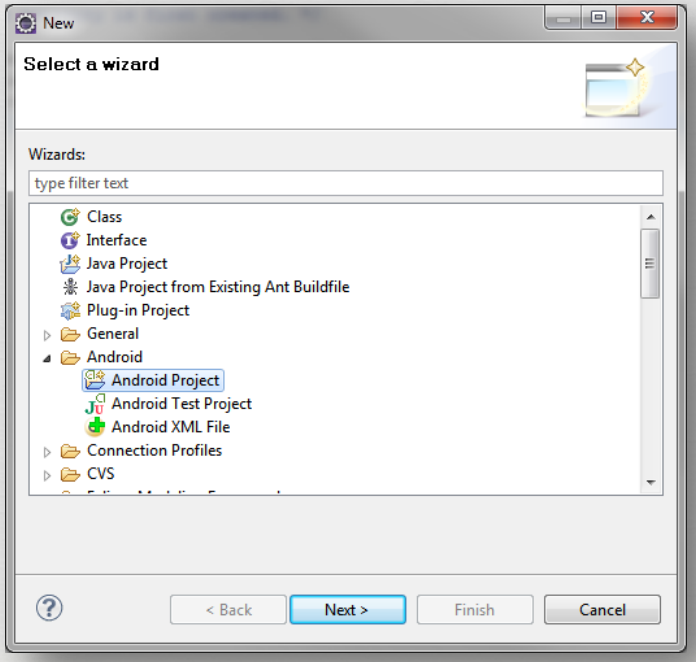
```
$ android create project \
        --target <target_ID> \
        --name <dein_Projekt_name> \
        --path /pfad/zu/deinem/Projekt\
        --activity <dein_activity_Name> \
        --package <dein_package_Namensraum>
```

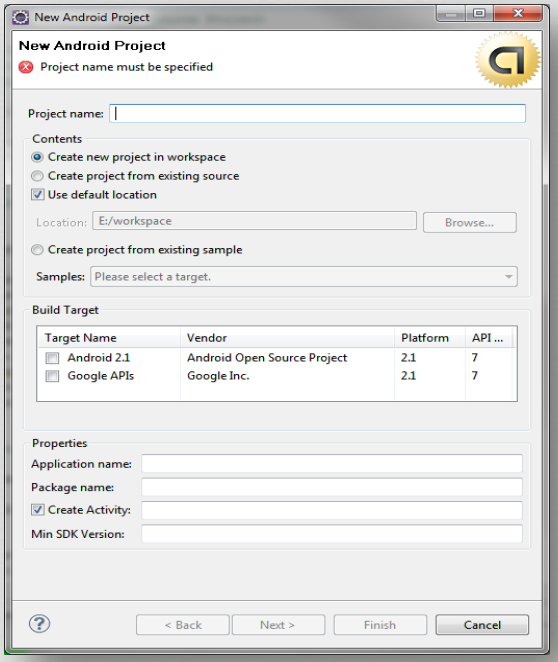
- Update

```
$ android update project
$ android update project --name <Mein_Programm>
        --target 8  – path <PfadZumProjekt>
```

# Projekt erstellen

- In Eclipse:
  - File->New->Other->Android->Android Project



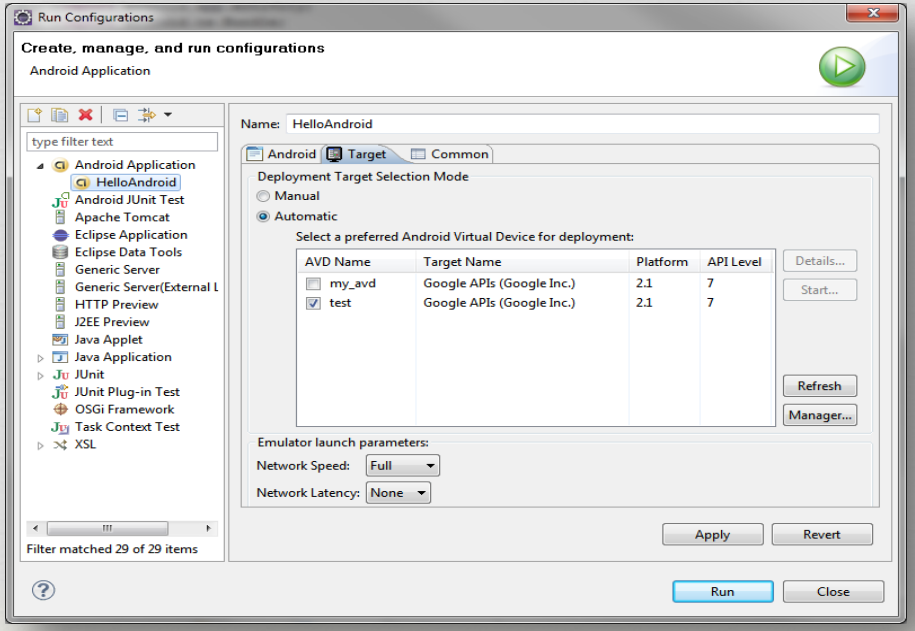
# Anwendung starten

```
$ adb install /Pfad/zu/deinem/Programm.apk
```

- Installiert die Anwendung im Emulator
- Aus dem normalen „App-Menü“ starten

# Anwendung starten

- In Eclipse:
- Rechtsklick auf Projekt: Run As->Run Configuration
  - Anwendung startet automatisch



# SD-Karte erstellen

- Wenn nicht beim Erstellen des AVDs geschehen
- Virtuelle SD-Karte

```
$ mksdcard –l <Label> <Größe>[K|M] <Datei>
```

  - FAT32
  - SDHC – Standard
    - SDXC – Vielleicht irgendwann
- Mount

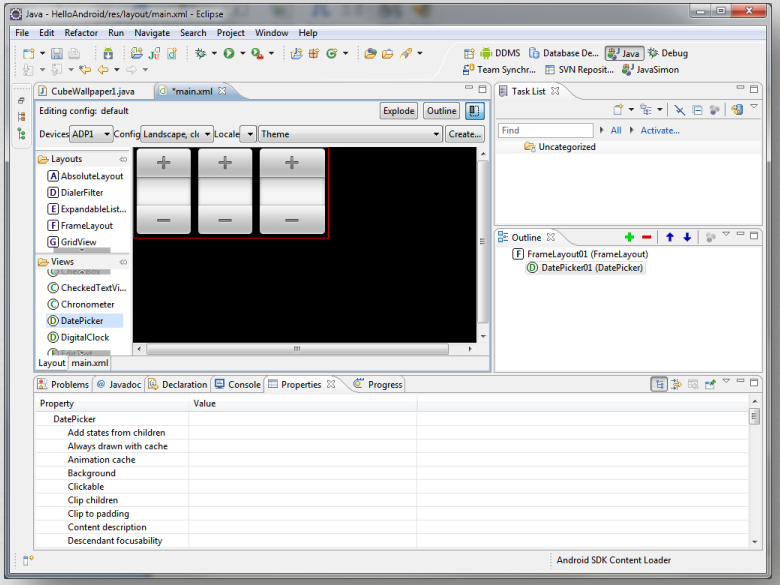
```
$ emulator -sdcard <Datei> -avd <VirtuellesDevice>
```

# Inhalt

- Tools
  - Erstellen eines AVDs
  - Erstellen eins Projekts
  - Anwendung starten
  - SD-Karte erstellen
  - Performance durch Alignement
  - Interprozesskommunikation
  - **Skalierbares UI**
  - **Aufbau des UIs**
  - **Automatische UI Analyse**

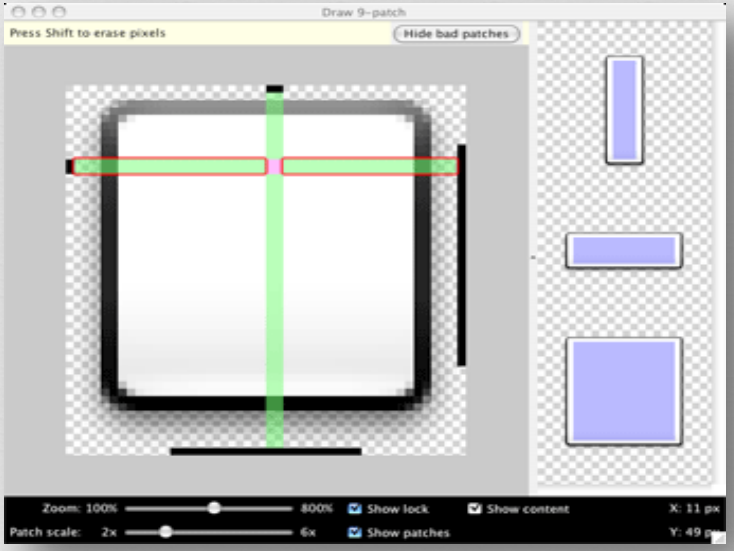
# Skalierbares UI

- WYSIWYG oder XML
- Rudimentäres Drag & Drop
- XML vergleichbar mit XAML oder XHTML



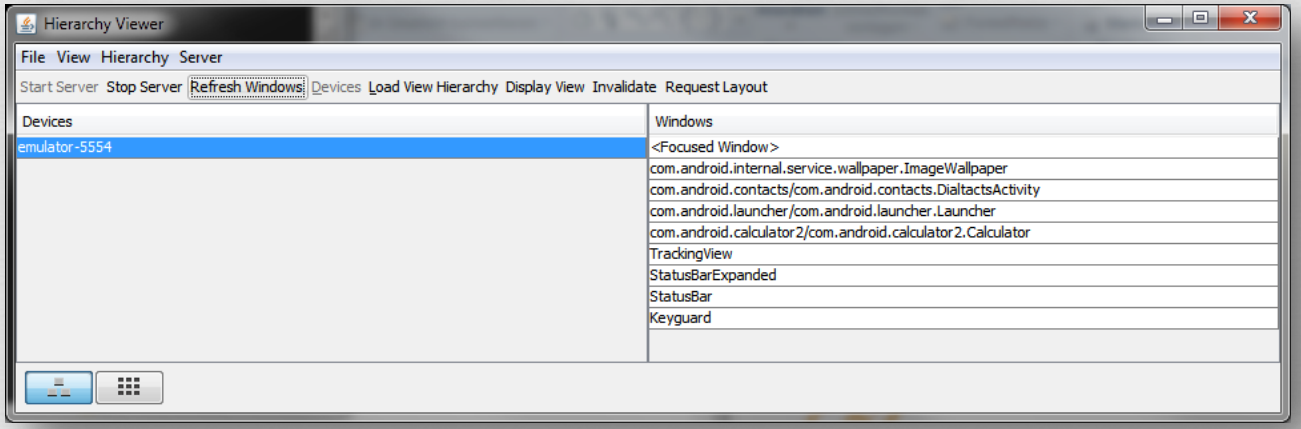
# Skalierbares UI

- Draw9
  - WISIWYG Editor für NinePatch Grafiken
    - NinePatch -> skalierbare UI Elemente auf PNG basis



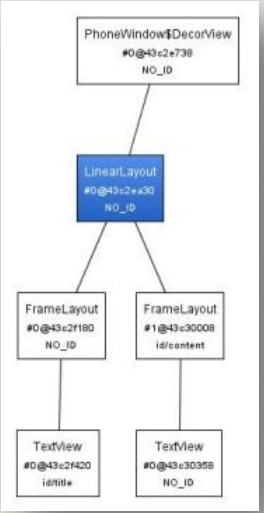
# Aufbau des UIs

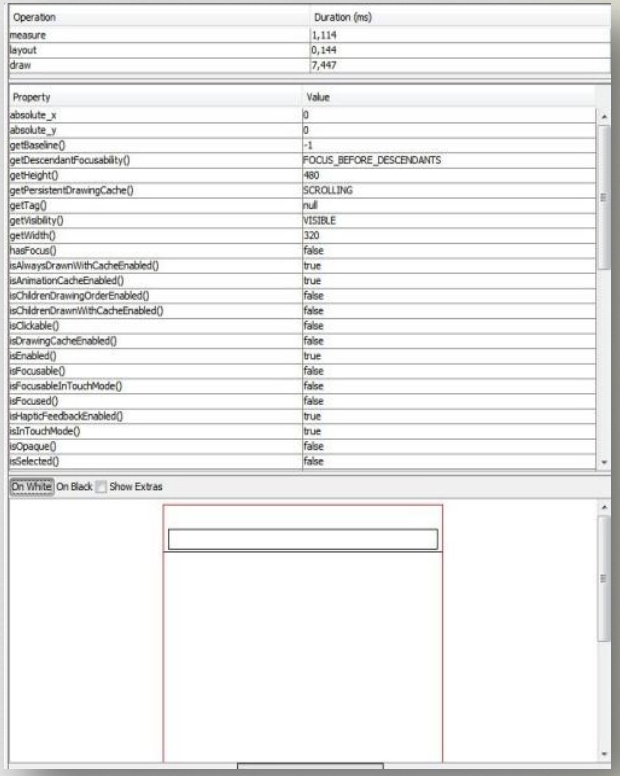
- Hierarchy Viewer
  - GUI – Tools
    - Zugriff auf den Emulator
    - Ermöglicht Layout View und Display View bzw. Pixel Perfect View



# Aufbau des UIs

- Layout View
  - 3 Ansichten
    - „Tree View“
      - Hierachisches Diagramm der Ansicht
    - „Properties View“
      - Eigenschaften der ausgewählten Elemente
    - „Wire-frame View“
      - Skizze des UIs



| Operation | Duration (ms) |
|---|---|
| measure | 1,114 |
| layout | 0,144 |
| draw | 7,447 |

| Property | Value |
|---|---|
| absolute_x | 0 |
| absolute_y | 0 |
| getBaseline() | -1 |
| getDescendantFocusability() | FOCUS_BEFORE_DESCENDANTS |
| getHeight() | 480 |
| getPersistentDrawingCache() | SCROLLING |
| getTag() | null |
| getVisibility() | VISIBLE |
| getWidth() | 320 |
| hasFocus() | false |
| isAlwaysDrawnWithCacheEnabled() | true |
| isAnimationCacheEnabled() | true |
| isChildrenDrawingOrderEnabled() | false |
| isChildrenDrawnWithCacheEnabled() | false |
| isClickable() | false |
| isDrawingCacheEnabled() | false |
| isEnabled() | true |
| isFocusable() | false |
| isFocusableInTouchMode() | false |
| isFocused() | false |
| isHapticFeedbackEnabled() | true |
| isInTouchMode() | true |
| isOpaque() | false |
| isSelected() | false |

On White On Black Show Extras

# Aufbau des UIs

- Pixel Perfect View
  - 3 Ansichten
    - „Explorer View“ zeigt die Elementhierarchie als Liste
    - „Normal View“ zeigt die normale Ansicht des Geräts
      - Zeigt die Grenzen der einzelnen UI-Elemente an
    - „Loupe View“ zeigt die selbe Ansicht pixelgenau
  - Möglichkeit Bilder als „Overlay“ zu laden

# Automatische UI Analyse

- layoutopt
  - Analysiert die dem UI zugrundeliegenden XML-Dateien
  - Testet das UI gegen einen Regelsatz
    - Findet Mängel und gibt Ratschläge

```
$ layoutopt samples /
samples/compound.xml
    7:23 The root-level <FrameLayout/> can be replaced with <merge>
    11:21 This LinearLayout layout or its FrameLayout parent is useless
samples/simple.xml
    7:7 The root-level <FrameLayout/> can be replaced with <merge>
samples/too_deep.xml
    -1:-1 This layout has too many nested layouts: 13 levels, it should have <=10
    28:81 This LinearLayout layout or its LinearLayout parent is useless
```

# Inhalt


- Einführung
- Bestandteile des SDKs
- **Debugging**
  - **Syslog Anzeigen**
  - The Monkey
  - Android Debug Bridge
  - Dalvik Debug Monitor
  - Performance
- NDK

# Syslog Anzeigen

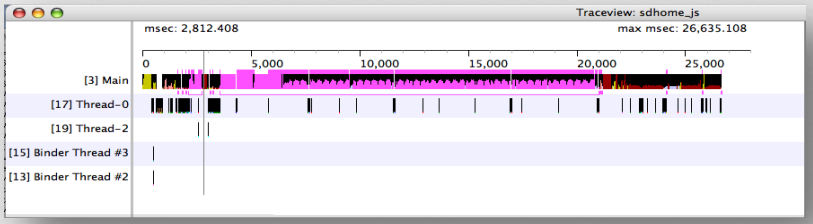
- Traceview: Grafische Anzeige des Logs

```
//startet tracing nach „/sdcard/cal.trace“
Debug.startMethodTracing(„calc“);
```

```
//stop tracing
Debug.stopMethodTracing();
```

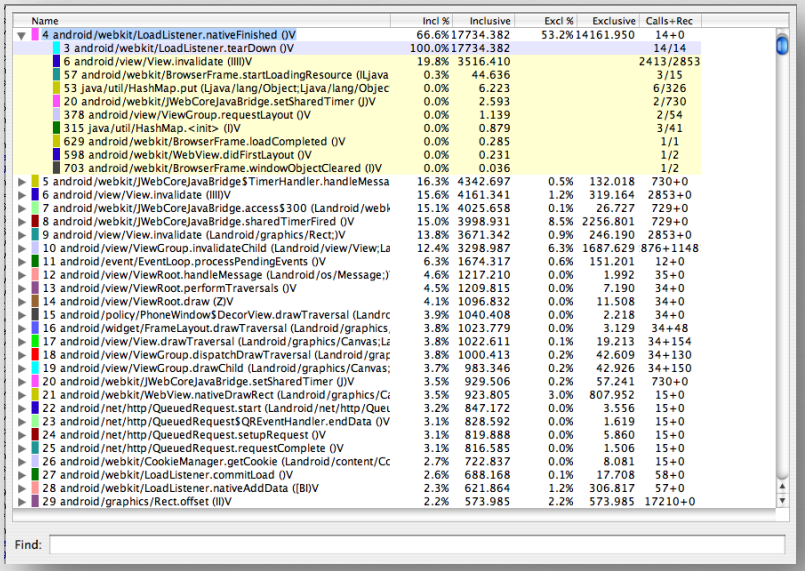
  - SD-Karten Image notwendig
  - Datei herunterladen
    - adb pull /sdcard/calc.trace <Pfad>
  - Datei anzeigen
    - traceview <Pfad>

# Syslog Anzeigen




- Zeigt das ausführen von Threads

- Profile Pane
  - Zeit in Methoden
    - Inclusive
      - Mit Subfunktionen
    - Exclusie
      - Nur die Methode

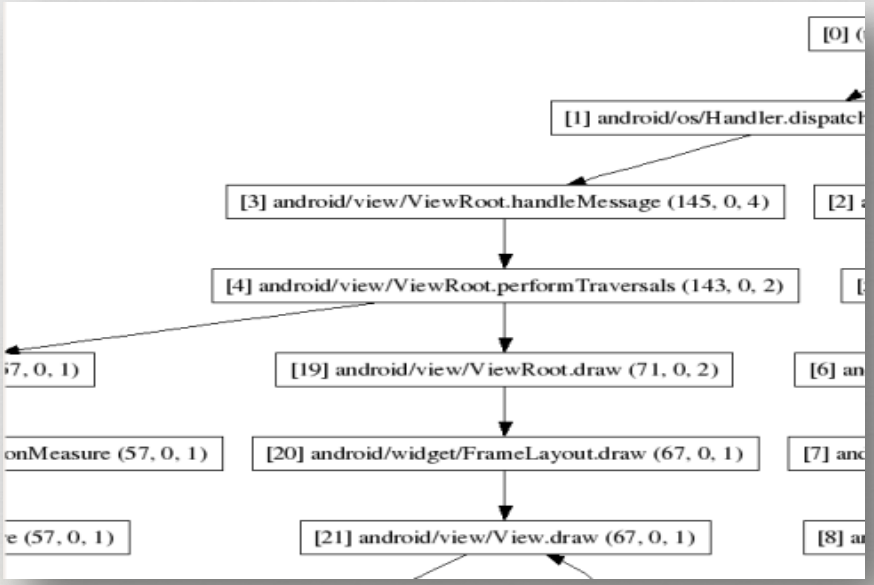


| Name | Incl % | Inclusive | Excl % | Exclusive | Calls+Rec |
|---|---|---|---|---|---|
| 4 android/webkit/LoadListener.nativeFinished ()V | 66.6% | 17734.382 | 53.2% | 14161.950 | 14+0 |
| 3 android/webkit/LoadListener.tearDown ()V | 100.0% | 17734.382 | | | 14/14 |
| 6 android/view/View.invalidate (IIII)V | 19.8% | 3516.410 | | | 2413/2853 |
| 57 android/webkit/BrowserFrame.startLoadingResource (ILjava | 0.3% | 44.636 | | | 3/15 |
| 53 java/util/HashMap.put (Ljava/lang/Object;Ljava/lang/Objec | 0.0% | 6.223 | | | 6/326 |
| 20 android/webkit/JWebCoreJavaBridge.setSharedTimer (J)V | 0.0% | 2.593 | | | 2/730 |
| 378 android/view/ViewGroup.requestLayout ()V | 0.0% | 1.139 | | | 2/54 |
| 315 java/util/HashMap.<init> ()V | 0.0% | 0.879 | | | 3/41 |
| 629 android/webkit/BrowserFrame.loadCompleted ()V | 0.0% | 0.285 | | | 1/1 |
| 598 android/webkit/WebView.didFirstLayout ()V | 0.0% | 0.231 | | | 1/2 |
| 703 android/webkit/BrowserFrame.windowObjectCleared (I)V | 0.0% | 0.036 | | | 1/2 |
| 5 android/webkit/JWebCoreJavaBridge$TimerHandler.handleMessa | 16.3% | 4342.697 | 0.5% | 132.018 | 730+0 |
| 6 android/view/View.invalidate (IIII)V | 15.6% | 4161.341 | 1.2% | 319.164 | 2853+0 |
| 7 android/webkit/JWebCoreJavaBridge.access$300 (Landroid/webk | 15.1% | 4025.658 | 0.1% | 26.727 | 729+0 |
| 8 android/webkit/JWebCoreJavaBridge.sharedTimerFired ()V | 15.0% | 3998.931 | 8.5% | 2256.801 | 729+0 |
| 9 android/view/View.invalidate (Landroid/graphics/Rect;)V | 13.8% | 3671.342 | 0.9% | 246.190 | 2853+0 |
| 10 android/view/ViewGroup.invalidateChild (Landroid/view/View;La | 12.4% | 3298.987 | 6.3% | 1687.629 | 876+1148 |
| 11 android/event/EventLoop.processPendingEvents ()V | 6.3% | 1674.317 | 0.6% | 151.201 | 12+0 |
| 12 android/view/ViewRoot.handleMessage (Landroid/os/Message;) | 4.6% | 1217.210 | 0.0% | 1.992 | 35+0 |
| 13 android/view/ViewRoot.performTraversals ()V | 4.5% | 1209.815 | 0.0% | 7.190 | 34+0 |
| 14 android/view/ViewRoot.draw (Z)V | 4.1% | 1096.832 | 0.0% | 11.508 | 34+0 |
| 15 android/policy/PhoneWindow$DecorView.drawTraversal (Landrc | 3.9% | 1040.408 | 0.0% | 2.218 | 34+0 |
| 16 android/widget/FrameLayout.drawTraversal (Landroid/graphics, | 3.8% | 1023.779 | 0.0% | 3.129 | 34+48 |
| 17 android/view/View.drawTraversal (Landroid/graphics/Canvas;La | 3.8% | 1022.611 | 0.1% | 19.213 | 34+154 |
| 18 android/view/ViewGroup.dispatchDrawTraversal (Landroid/grap | 3.8% | 1000.413 | 0.2% | 42.609 | 34+130 |
| 19 android/view/ViewGroup.drawChild (Landroid/graphics/Canvas; | 3.7% | 983.346 | 0.2% | 42.926 | 34+150 |
| 20 android/webkit/JWebCoreJavaBridge.setSharedTimer (J)V | 3.5% | 929.506 | 0.2% | 57.241 | 730+0 |
| 21 android/webkit/WebView.nativeDrawRect (Landroid/graphics/Ca | 3.5% | 923.805 | 3.0% | 807.952 | 15+0 |
| 22 android/net/http/QueuedRequest.start (Landroid/net/http/Queu | 3.2% | 847.172 | 0.0% | 3.556 | 15+0 |
| 23 android/net/http/QueuedRequest$QREventHandler.endData ()V | 3.1% | 828.592 | 0.0% | 1.619 | 15+0 |
| 24 android/net/http/QueuedRequest.setupRequest ()V | 3.1% | 819.888 | 0.0% | 5.860 | 15+0 |
| 25 android/net/http/QueuedRequest.requestComplete ()V | 3.1% | 816.585 | 0.0% | 1.506 | 15+0 |
| 26 android/webkit/CookieManager.getCookie (Landroid/content/Cc | 2.7% | 722.837 | 0.0% | 8.081 | 15+0 |
| 27 android/webkit/LoadListener.commitLoad ()V | 2.6% | 688.168 | 0.1% | 17.708 | 58+0 |
| 28 android/webkit/LoadListener.nativeAddData ([B)V | 2.3% | 621.864 | 1.2% | 306.817 | 57+0 |
| 29 android/graphics/Rect.offset (II)V | 2.2% | 573.985 | 2.2% | 573.985 | 17210+0 |

Find:

# Syslog Anzeigen

- dmtracedump
  - Erzeugt ein Graphen, der den Aufrufstack repräsentiert
    - Benötigt Graphviz



# Inhalt

- Einführung
- Bestandteile des SDKs
- **Debugging**
  - Syslog Anzeigen
  - **The Monkey**
  - Android Debug Bridge
  - Dalvik Debug Monitor
  - Performance
- NDK

# The Monkey

- Stresstest des UIs
- Erzeugt zufällige Ereignisse
  - Klicks
  - Berührungen
  - Gesten
  - Systemereignisse
- Ereignisstyp einschränkbar

# The Monkey

- Bearbeitet von einem Package

```
$ adb shell monkey -p dein.package.Name -v 500
```

- Stoppt bei Fehlern
  - Unbehandelten Exceptions
  - Crash
  - *„application not responding“* Error

# Inhalt

- Einführung
- Bestandteile des SDKs
- **Debugging**
  - Syslog Anzeigen
  - The Monkey
  - **Android Debug Bridge**
  - Dalvik Debug Monitor
  - Performance
- NDK

# Android Debug Bridge

- Abkürzung: adb
- Vielseitig
  - Debugging
  - Datenübertragung
  - Shell zugriff
- Besteht aus Client, Server und Daemon
  - PC: Client, Server
  - Emulator oder Device: Daemon

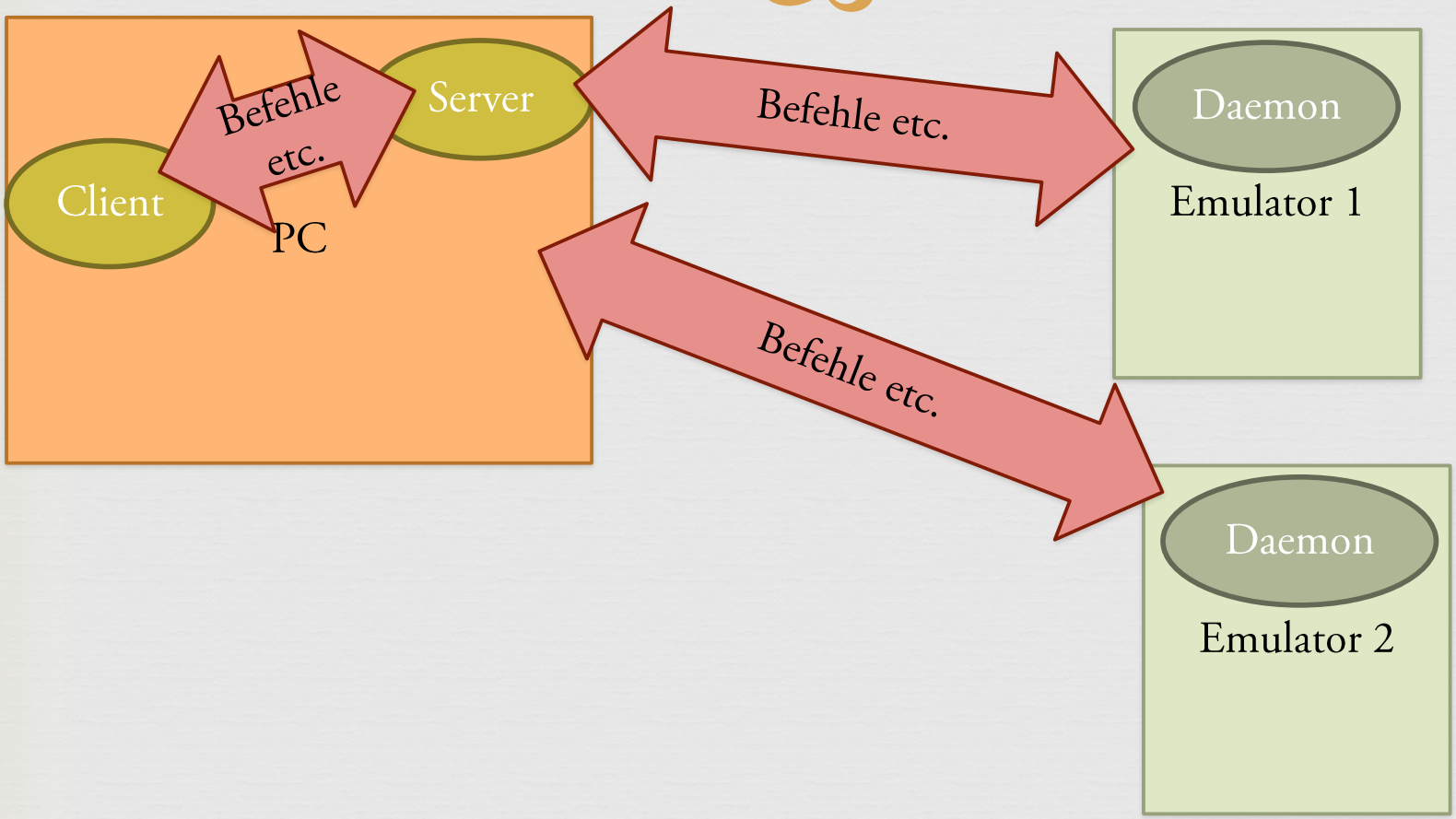
# Android Debug Bridge

- Adb Server läuft auf Port 5037
- Emulator / Device auf Port 5555-5585 (ungeraden)
  - Android Console auf den geraden Ports (Port – 1)
    - adb braucht beide
- Client-Server Struktur
    - Normal: ein Rechner; ausbaufähig
  - einfache Kommunikation
  - Bei mehreren Adb-Daemons :

```
$ adb devices
```

    - Gibt Liste der Daemons aus

# Android Debug Bridge



# Android Debug Bridge

- Kann Programme installieren

```
$ adb install <Datei>
```

- Kann mit mehreren Daemons umgehen

```
$ adb -s emulator-5556 install <Datei>
```

- Möglichkeit zum Port Forwarding

```
$ adb forward tcp:6100 tcp:7100
```

- Datenübertragung

```
$ adb push / adb pull
```

# Android Debug Bridge

- Logdatei anzeigen

```
$ adb logcat <Filter>
```

- Bugreport erstellen

```
$ adb bugreport
```

  - Gibt kompletten Gerätestatus aus
    - Kombination aus:
      - Logcat
      - Dumpsys
      - Dumpsate
  - Emulator ohne laufende Programme 1,32MB an Text

# Android Debug Bridge

- Logcat
  - Zeigt System Debug Informationen an
- Filter
  - V – Verbose (niedrigste Priorität: alles wird ausgegeben)
  - D - Debug
  - I - Info
  - W - Warning
  - E - Error
  - F - Fatal
  - S – Silent (höchste Priorität: nichts wird ausgegeben)

# Android Debug Bridge

- Logcat Filter:

```
$ adb logcat Activity:Level
$ adb logcat ActivityManager:I *:S
```

  - Info zu ActivityManager
  - Silent zu Rest
- Logcat Format:
  - -v <eineOption>
    - brief, process, tag, thread, raw, time, long
- Logcat andere Logs
  - Radio, events, main (default)

# Android Debug Bridge

- Shell

```
$ adb shell
```

- Binärdateien in /system/bin
  - Einige *nix Tools vorhanden

```
$ adb shell top /reboot / dd / mount /…
```

# Android Debug Bridge

- Sqlite3 Database

```
$ adb shell sqlite3
```

- .help für Hilfe
- Grundlegende SQL Syntax
  - Create table
  - Insert
  - SELECT FROM WHERE

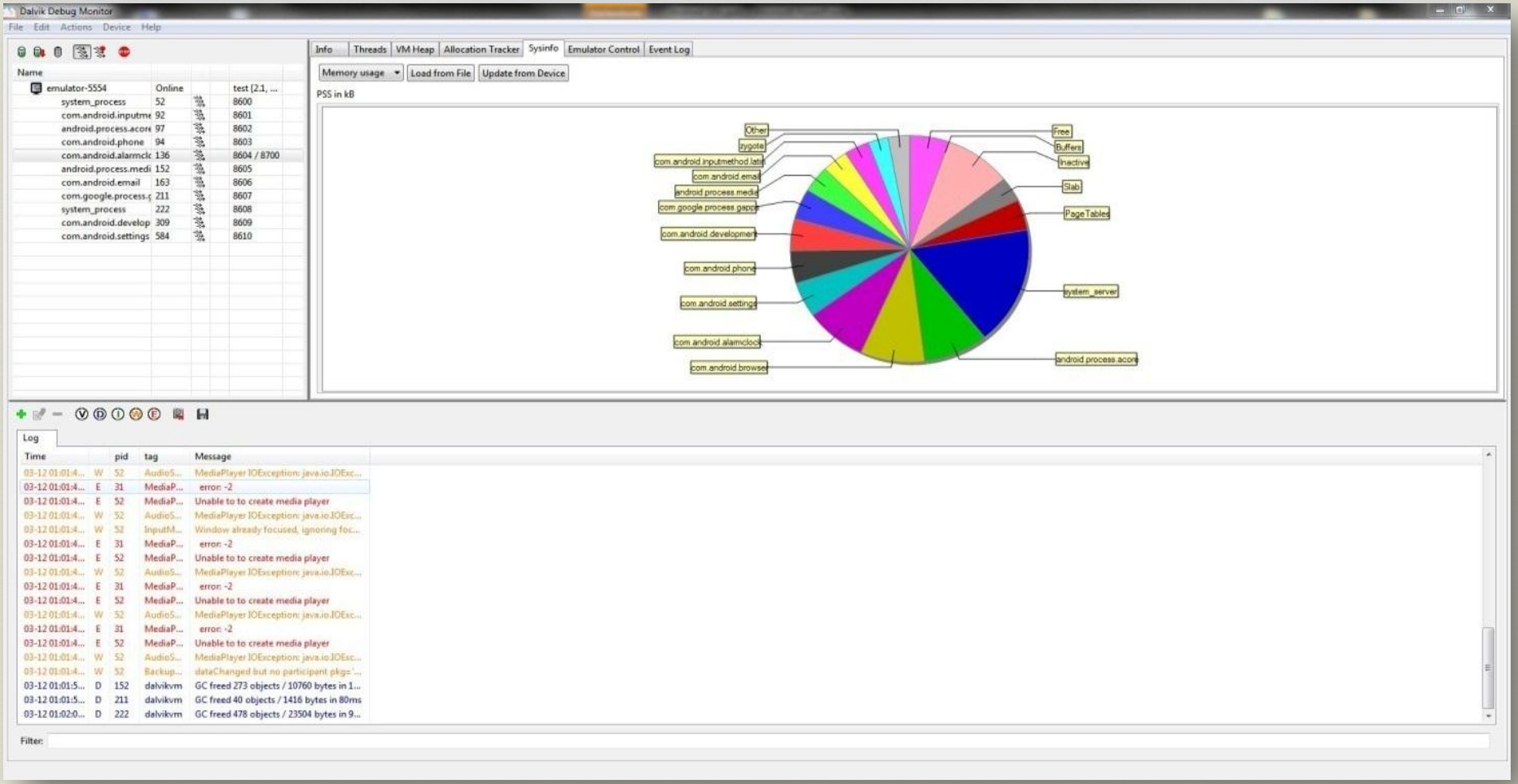
# Inhalt

- Einführung
- Bestandteile des SDKs
- **Debugging**
  - Syslog Anzeigen
  - The Monkey
  - Android Debug Bridge
  - **Dalvik Debug Monitor**
  - Performance
- NDK

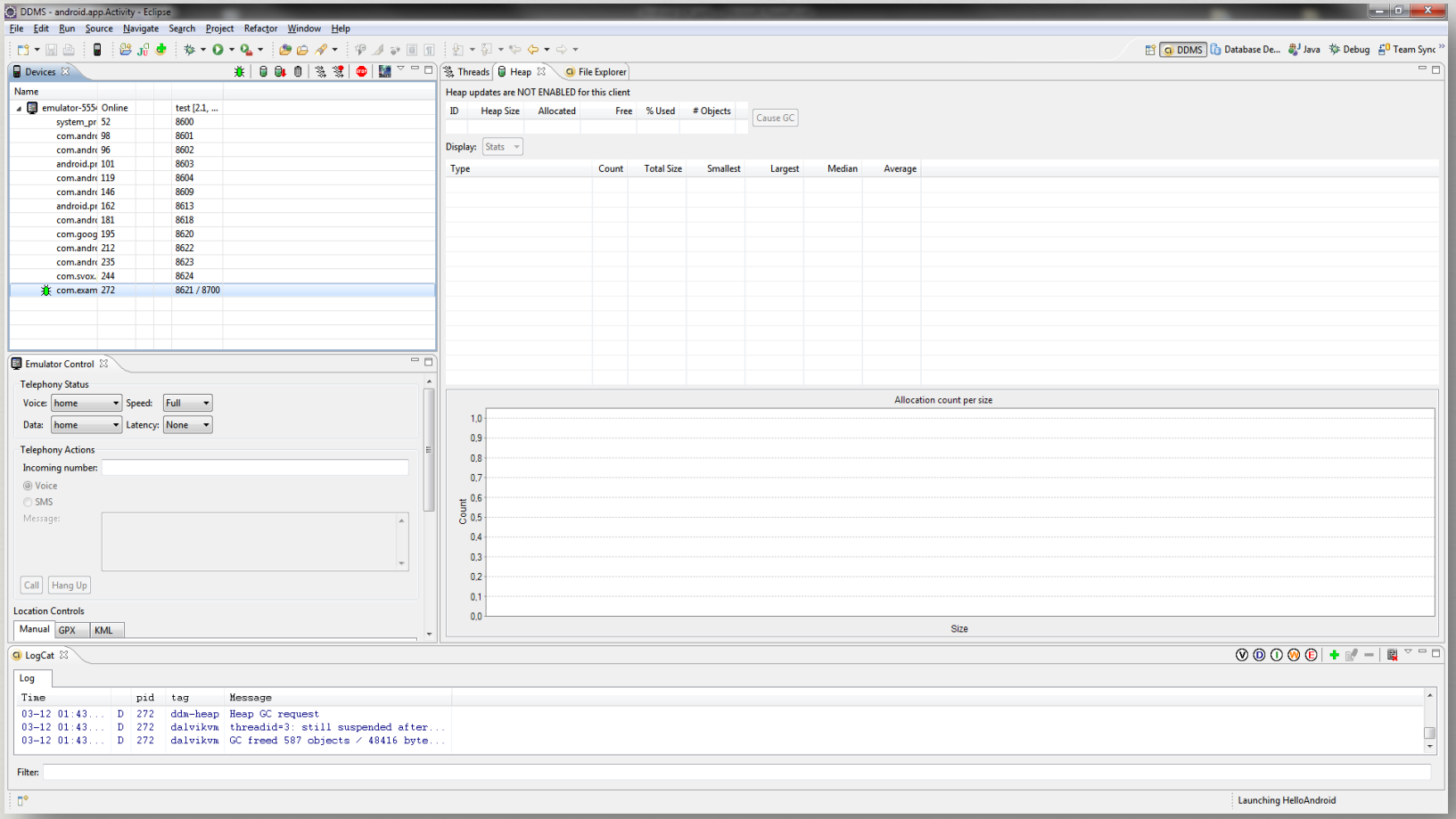
# Dalvik Debug Monitor

- Anzeige des Heaps und Stacks
- Schrittbasiertes Debugging mit Breakpoints
- Übersicht über Garbage Collection
- Möglichkeit das Gerät „anzurufen“
  - Oder SMS
  - GPS Position bereitstellen
    - Manuell (default ist irgentwo vor Afrika ☺)
    - Per GPX oder KML Datei

# Dalvik Debug Monitor



# Dalvik Debug Monitor



# Dalvik Debug Monitor

Debug - HelloAndroid/src/com/example/helloandroid/HelloAndroid.java - Eclipse

File Edit Run Source Navigate Search Project Refactor Window Help

Debug

Thread [<3> main] (Suspended (breakpoint at line 11 in HelloAndroid))
- HelloAndroid.onCreate(Bundle) line: 11
- Instrumentation.callActivityOnCreate(Activity, Bundle) line: 1047
- ActivityThread.performLaunchActivity(ActivityThread$ActivityRecord, Intent) line: 2459
- ActivityThread.handleLaunchActivity(ActivityThread$ActivityRecord, Intent) line: 2512
- ActivityThread.access$2200(ActivityThread, ActivityThread$ActivityRecord, Intent) line: 119
- ActivityThread$H.handleMessage(Message) line: 1863
- ActivityThread$H(Handler).dispatchMessage(Message) line: 99
- Looper.loop() line: 123
- ActivityThread.main(String[]) line: 4363
- Method.invokeNative(Object, Object[], Class, Class[], Class, int, boolean) line: not available [native method]
- Method.invoke(Object, Object...) line: 521
- ZygoteInit$MethodAndArgsCaller.run() line: 860
- ZygoteInit.main(String[]) line: 618
- NativeStart.main(String[]) line: not available [native method]

Variables | Breakpoints

| Name | Value |
|---|---|
| this | HelloAndroid (id=830084772488) |
| ACCESSIBILITY_SERVICE | "accessibility" (id=830002519480) |
| ACCOUNT_SERVICE | "account" (id=830002513864) |
| ACTIVITY_SERVICE | "activity" (id=830002513936) |
| ALARM_SERVICE | "alarm" (id=830002514008) |
| APPWIDGET_SERVICE | "appwidget" (id=830002514072) |
| AUDIO_SERVICE | "audio" (id=830002514144) |
| BACKUP_SERVICE | "backup" (id=830002514208) |

com.example.helloandroid.HelloAndroid@44e87288

HelloAndroid.java

```
1 package com.example.helloandroid;
2
3 import android.app.Activity;
4 import android.os.Bundle;
5
6
7 public class HelloAndroid extends Activity {
8     /** Called when the activity is first created. */
9     @Override
10    public void onCreate(Bundle savedInstanceState) {
11        super.onCreate(savedInstanceState);
12        this.setContentView(R.layout.main);
13    }
14 }
```

Outline
- com.example.helloandroid
- import declarations
- HelloAndroid
  - onCreate(Bundle) : void

Console | Tasks

Android

```
[2010-03-12 02:36:45 - HelloAndroid]Android Launch!
[2010-03-12 02:36:45 - HelloAndroid]adb is running normally.
[2010-03-12 02:36:45 - HelloAndroid]Performing com.example.helloandroid.HelloAndroid activity launch
[2010-03-12 02:36:45 - HelloAndroid]Automatic Target Mode: Preferred AVD 'test' is available on emulator 'emulator-5554'
[2010-03-12 02:36:45 - HelloAndroid]Application already deployed. No need to reinstall.
[2010-03-12 02:36:45 - HelloAndroid]Starting activity com.example.helloandroid.HelloAndroid on device
[2010-03-12 02:36:47 - HelloAndroid]ActivityManager: Starting: Intent { cmp=com.example.helloandroid/.HelloAndroid }
[2010-03-12 02:36:48 - HelloAndroid]Attempting to connect debugger to 'com.example.helloandroid' on port 8621
```

Writable | Smart Insert | 11 : 1 | Launching HelloAndroid

# Dalvik Debug Monitor

- File Explorer
- Bildschirmfoto
- Prozessanzeige (Basiert auf: ps –x)
- Zugriff auf logcat / dumpsys / dumpstate aus dem UI

# Inhalt

- Einführung
- Bestandteile des SDKs
- **Debugging**
  - Syslog Anzeigen
  - The Monkey
  - Android Debug Bridge
  - Dalvik Debug Monitor
  - **Performance**
- NDK

# Performance d. Alignment

- Tool: Zipalign

```
$ zipalign [-f] [-v] [-c] <Alignment> Input.apk Output.apk
```

  - Alignment muss immer **4** sein!
    - bei 32bitWörtern (die es in Dalvik sind)
- Aus Eclipse automatisch
- Warum?
  - Direkter Zugriff mit mmap()
  - Weniger RAM

# Performance

- Mehrere Möglichkeiten
  - Tracing
  - TimingLogger
- MethodTracing
  - Via Traceview
  - Laufzeit der einzelnen Methoden betrachten
- TimingLogger
  - Via logcat
  - Erzeugt Meldungen im „base“ log

# Performance

- MethodTracing
  - Start des Tracings (z.B. Programmstart)

```
Debug.startMethodTracing("<Name>");
```

  - Explizites Beenden des Tracings

```
Debug.stopMethodTracing();
```

| Name | Incl % |
|---|---|
| 0 (toplevel) | 100.0% |
| 1 android/os/Handler.dispatchMessage (Landroid/os/Message;)V | 96.0% |
| 2 android/view/ViewRoot.handleMessage (Landroid/os/Message;)V | 71.7% |
| 3 android/widget/ListView.makeAndAddView (IIZIZ)Landroid/view/View; | 70.5% |
| 4 android/widget/AbsListView.trackMotionScroll (II)V | 52.5% |
| 5 android/widget/ListView.fillGap (Z)V | 52.2% |
| 6 android/widget/AbsListView.obtainView (I)Landroid/view/View; | 52.2% |
| 7 [redacted]Adapter.getView (ILa | 52.1% |

# Performance

- TimingLogger
  - Erzeugt Meldungen unter Topic "TopicLogTag“

```
TimingLogger timings = new TimingLogger("TopicLogTag","preparePicturesFromList");
<zu_loggender_Code>
timings.dumpToLog();
```

  - Bessere Unterteilung mit `adsplits(„<Label>“)`
- Logcat: `$ adb logcat -v time TopicLogTag:V  *:E`

```
D/TopicLogTag(14857): preparePicturesFromList: begin
D/TopicLogTag(14857): preparePicturesFromList:      1999 ms, creating picture object
D/TopicLogTag(14857): preparePicturesFromList:      712 ms, generating thumbnails
D/TopicLogTag(14857): preparePicturesFromList: end, 2711 ms
```

# Performance

- Testen der Geräteleistung
  - Langsame Hardware / Langsamer Emulator
- Linpack für Android
  - Im Android Market oder im Netz
  - Android 2.1 auf Core2Duo @ 2,4GHz = 1,131 Mflops
    - Übertaktetes Droid bis zu 25Mflops @1,3GHz
    - Nexus One 8-10Mflops

# Inhalt

- Einführung
- Bestandteile des SDKs
- Debugging
  - Syslog Anzeigen
  - The Monkey
  - Android Debug Bridge
  - Dalvik Debug Monitor
  - Performance
- **NDK**

# NDK

- Native Developement Kit
  - Grundsätzlich eine Toolchain die ARM-Bytecode erzeugt
  - Erlaubt Teile der Anwendung in „native“ Code zu bauen
    - Keine kompletten Anwendungen
    - Unterstützt C / C++
    - Nicht unbedingt schneller, aber komplexer
    - Wiederbenutzung von bestehendem
- Enthält OpenGL ES Bibliotheken
  - Und weitere

# Zusammenfassung

- Installation und grundlegende Konfiguration
- Übersicht über die meisten Tools
  - Emulator
  - Avds erstellen
- Übersicht über die Debugging Funktionen
  - DDM – Dalvik Debug Monitor
  - ADB – Android Debug Brindge