netcetera

# Android & iPhone

## A Comparison

*Stefan Tramm*

*JUGS, Jahresevent*

*2008-12-11*

## Agenda

## Situation before 2007

- Three platforms
  - J2ME
  - Symbian
  - Windows Mobile
- all the same problems: incompatibilities between devices
  - screen sizes
  - buttons
  - bugs and optional APIs
  - no easy application deployment and install
- expensive data transfer
- crippled browsers

## 2007/2008 changed everything

- OHA announced Android
  - Google promised a solution

- Apples iPhone entered market as a ‘package’
  - data transfer included
  - full featured browser
  - one screen size and no buttons
  - after opening the platform for 3rd parties: easy deployment (App Store)
  - one set of APIs
  - Apple delivered a solution

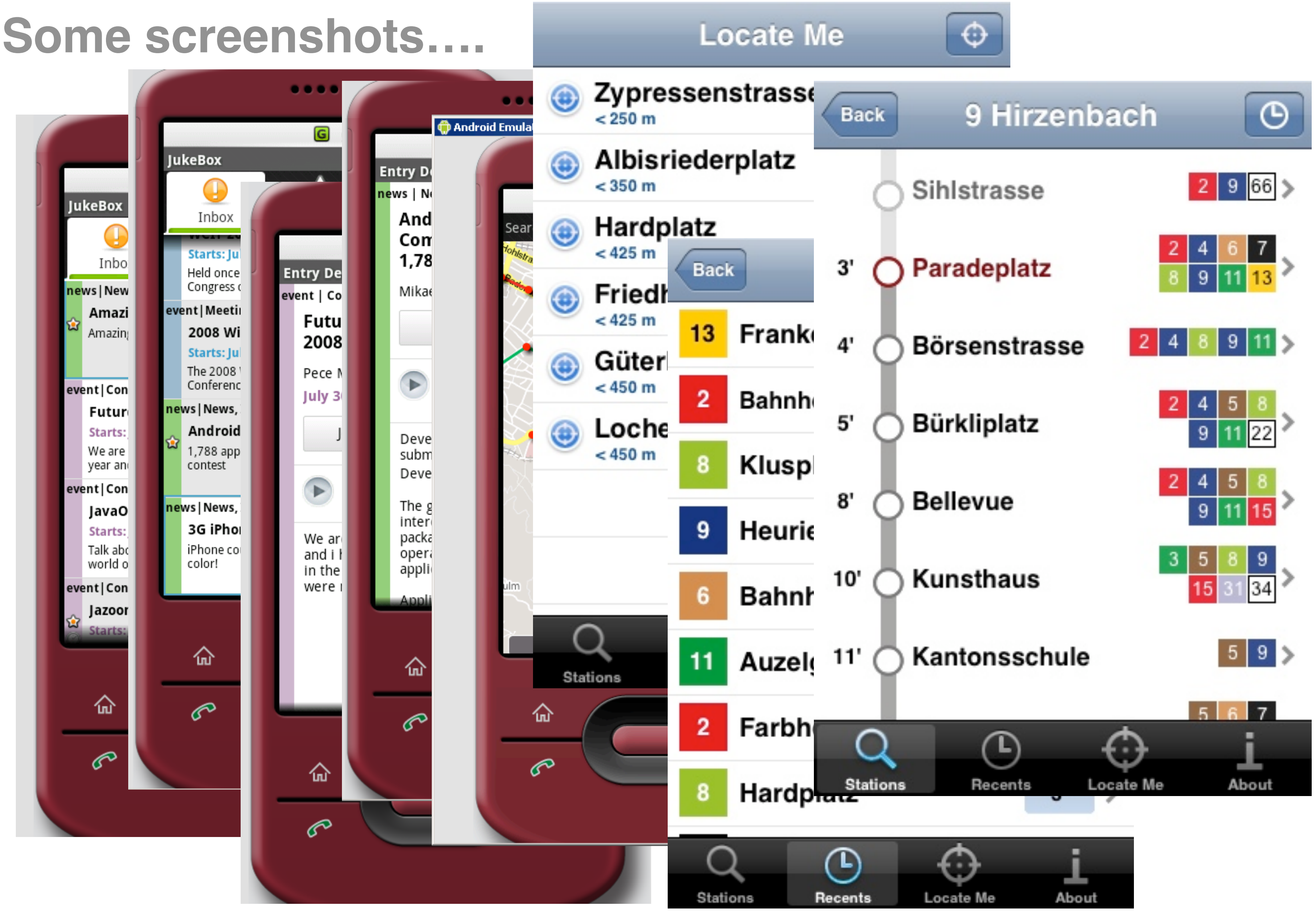
## Why I am here?

- Netcetera did some internal Android based apps:
  - Tramdroid for the ADC
  - ‘Jukebox’ – a RSS reader on steroids

- Netcetera released Tramdroid for iPhone (October 26, 08)
  - no Apple, iPhone, Objective-C knowledge before start
  - initial release took 4 months time, mostly to get the platform and to get it right

- I had the pleasure to be the project lead for all these things

## Some screenshots….



## Part II: Comparison

## Basics iPhone

- touch UI
  - no cursor, no mouse
  - no focus, no highlight
  - no physical buttons
  - gesture based
  - multi-touch (up to five positions)
- Objective-C and Mac OS X based
- worldwide central App distribution
- some numbers:
  - > 10 Mio devices (excl. iPod Touch)
  - > 10’000 Apps

## Basics Android

- hybrid UI
  - 4 directions key (or scroll ball)
  - Back and Menu button
  - touch optional
  - one object has focus
- Java based (Linux Kernel totally hidden)
- worldwide distribution via Android Marketplace
- some number:
  - ~1 Mio devices (US, UK only)
  - <1’000 apps

## Platform Differences

- development environment
  - programming language
- main abstractions
- user interface
- application data
- hardware
  - accelerometer
  - location determination
- digital rights management
- App distribution

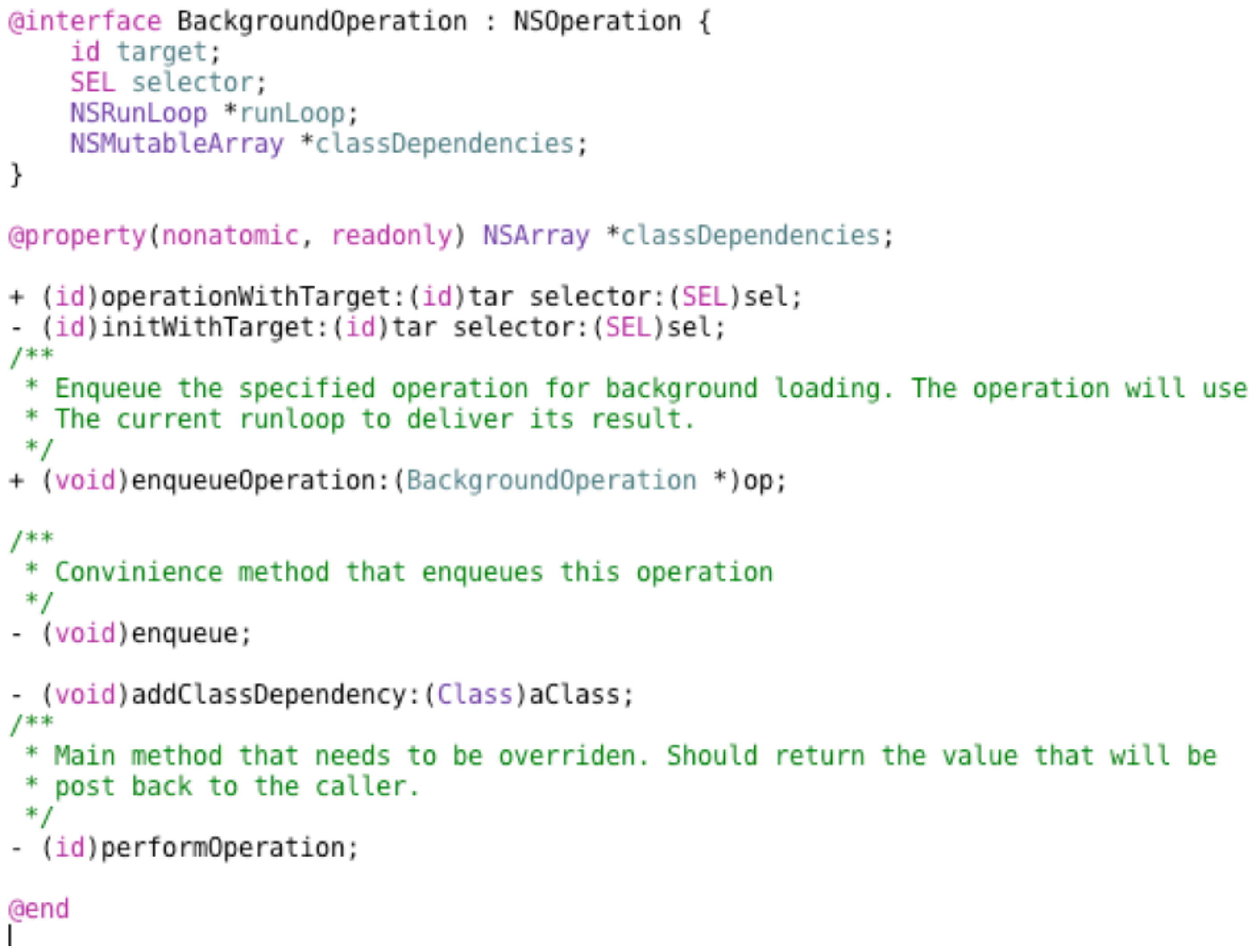
## Development Environment: iPhone

- Objective-C
- Xcode
  - weak Refactoring support
  - good help system
- Interface Builder
- Instruments, dtrace based profiler for memory leaks and performance analysis on the fly, very powerfull
- no automatic test support
- simulator
  - faster turnaround
  - differs from real device
- normal devices can be used (registration required)

## Details Objective-C

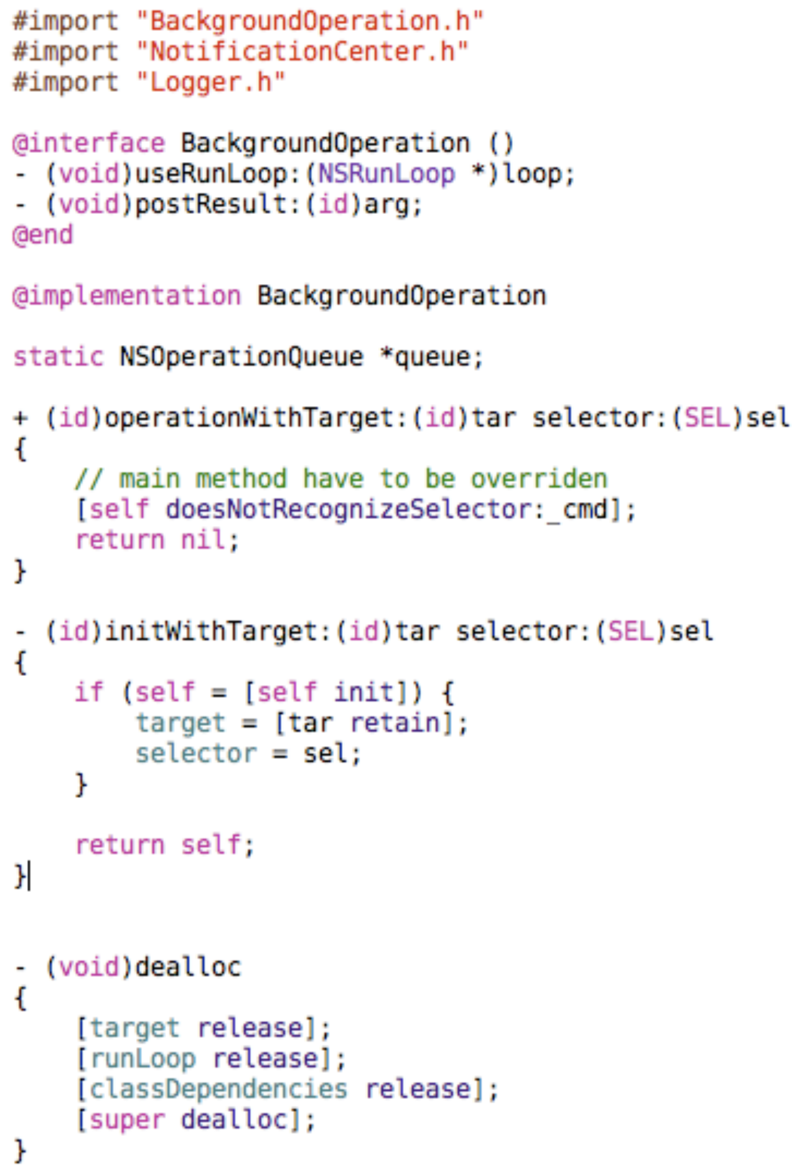
- ANSI C based
- extended with a Smalltalk like OO-Model
  - messages, selectors, implementations
  - classes are objects
  - good introspection at runtime
  - every message can be send to every object id (even nil)
  - [receiver messageselector:parameter]
- no garbage collection
  - semi manual ref-counting: retain and (auto)release
  - several memory leak analyzer available

## Objective-C (Declaration, Header)



```objc
@interface BackgroundOperation : NSOperation {
    id target;
    SEL selector;
    NSRunLoop *runLoop;
    NSMutableArray *classDependencies;
}

@property(nonatomic, readonly) NSArray *classDependencies;

+ (id)operationWithTarget:(id)tar selector:(SEL)sel;
- (id)initWithTarget:(id)tar selector:(SEL)sel;
/**
 * Enqueue the specified operation for background loading. The operation will use
 * The current runloop to deliver its result.
 */
+ (void)enqueueOperation:(BackgroundOperation *)op;

/**
 * Convinience method that enqueues this operation
 */
- (void)enqueue;

- (void)addClassDependency:(Class)aClass;
/**
 * Main method that needs to be overriden. Should return the value that will be
 * post back to the caller.
 */
- (id)performOperation;

@end
```

## Objective-C (Implementation)



```
#import "BackgroundOperation.h"
#import "NotificationCenter.h"
#import "Logger.h"

@interface BackgroundOperation ()
- (void)useRunLoop:(NSRunLoop *)loop;
- (void)postResult:(id)arg;
@end

@implementation BackgroundOperation

static NSOperationQueue *queue;

+ (id)operationWithTarget:(id)tar selector:(SEL)sel
{
    // main method have to be overriden
    [self doesNotRecognizeSelector:_cmd];
    return nil;
}

- (id)initWithTarget:(id)tar selector:(SEL)sel
{
    if (self = [self init]) {
        target = [tar retain];
        selector = sel;
    }

    return self;
}


- (void)dealloc
{
    [target release];
    [runLoop release];
    [classDependencies release];
    [super dealloc];
}
```

## Development Environment: Android

- Java
- Eclipse plus Plugins
- no User Interface Designer
- basic test support: testmonkey and android.test package
- emulator:
  - slower start time
  - closer to real hardware

- special developer devices provided by Google
  - open boot monitor which allows reflash of OS

## Platform Details: Main abstractions

Android

- Intents
- Activities
- Views
- Services
- Content Providers
- AIDL

- Intent Filters → User has to resolve ambiguities

iPhone

- UIView
- UIViewController
- UINavigationController
- Table View
- Frameworks / Shared Libs

- Popups not usually used

## Platform Details: UI description

Android

- XML description for UI
- parts are displayed while loading
- no GUI Builder
- text resources stored central and easy to reference
- listeners and interfaces used extensively

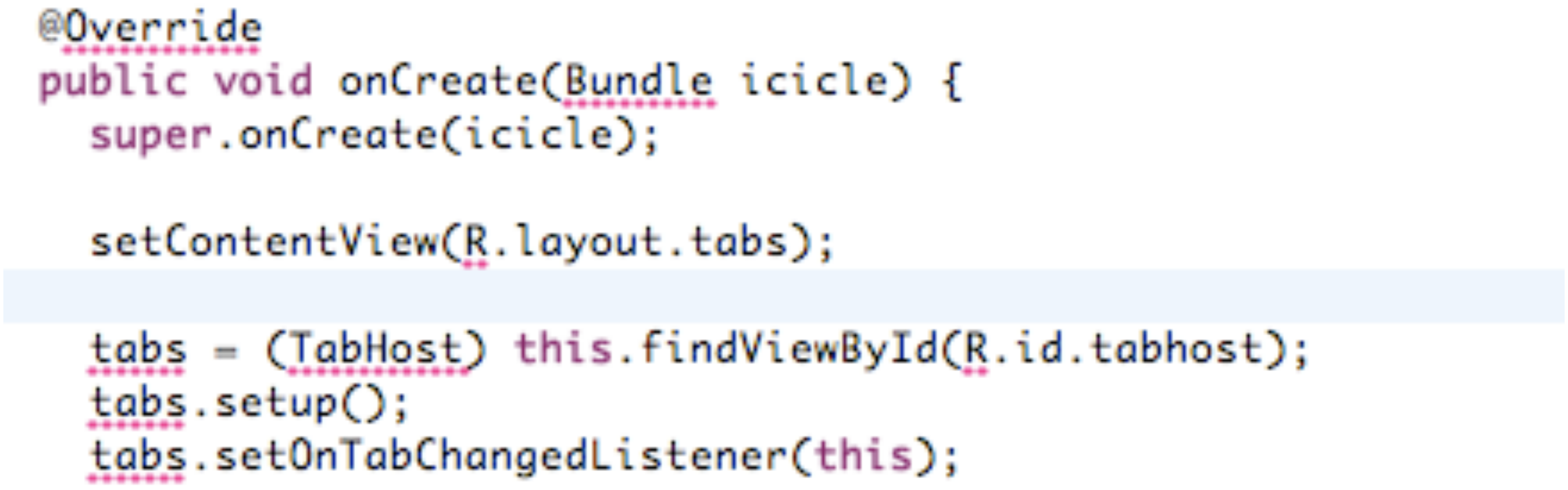
iPhone

- UI in NIB files (serialized objects)
- Default.png will be shown while loading
- Interface Builder is cool
- text resources stored in separate files
- delegates and protocols used extensively
- Animations – First Class citizens

## Android - UI

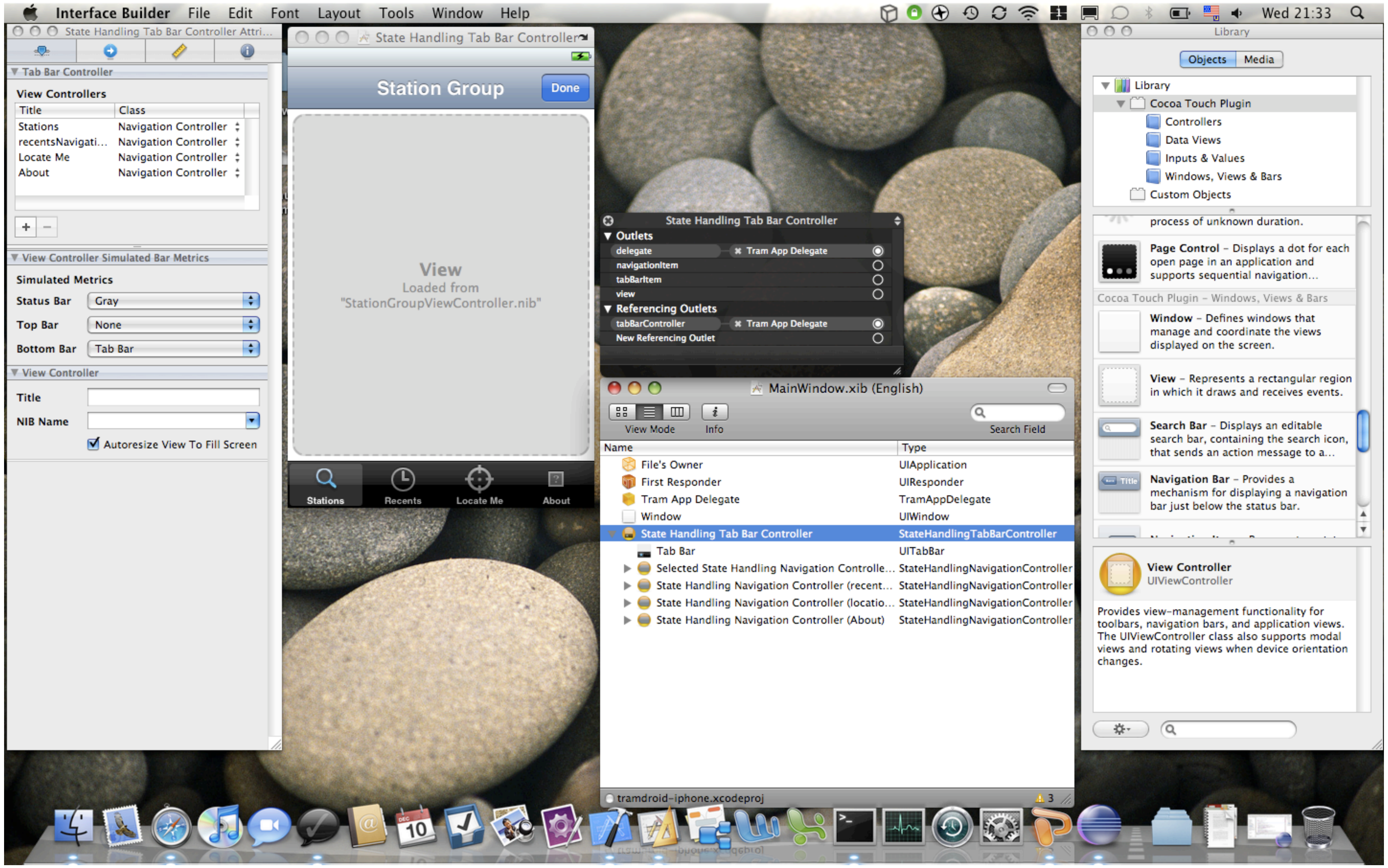
```
<!-- Join conference button -->
<AbsoluteLayout android:id="@+id/control_layout"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:paddingTop="20px"
    android:layout_below="@+id/picture_layout"
    android:layout_centerHorizontal="true">
    <AbsoluteLayout android:id="@+id/control_layout1"
        android:layout_width="300px"
        android:layout_height="wrap_content"
        android:padding="10px"
        android:background="@drawable/box_background">
        <Button android:id="@+id/button_call"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:text="Join Conference (call)"
            android:textAlign="center"
            android:textColor="@drawable/black"
            android:layout_x="0px"
            android:layout_y="0px">
        </Button>
    </AbsoluteLayout>
</AbsoluteLayout>
```

## Android UI – hooking up



```
@Override
public void onCreate(Bundle icicle) {
  super.onCreate(icicle);

  setContentView(R.layout.tabs);

  tabs = (TabHost) this.findViewById(R.id.tabhost);
  tabs.setup();
  tabs.setOnTabChangedListener(this);
```

## iPhone – Interface Builder


Interface Builder File Edit Font Layout Tools Window Help
Wed 21:33
State Handling Tab Bar Controller Attri...
State Handling Tab Bar Controller
Station Group
Done
View
Loaded from "StationGroupViewController.nib"
Stations
Recents
Locate Me
About
Library
Objects Media
tramdroid-iphone.xcodeproj

**Tab Bar Controller**

View Controllers

| Title | Class |
|---|---|
| Stations | Navigation Controller |
| recentsNavigati... | Navigation Controller |
| Locate Me | Navigation Controller |
| About | Navigation Controller |

**View Controller Simulated Bar Metrics**

Simulated Metrics

| | |
|---|---|
| Status Bar | Gray |
| Top Bar | None |
| Bottom Bar | Tab Bar |

**View Controller**

Title

NIB Name

[✓] Autoresize View To Fill Screen

**State Handling Tab Bar Controller**

Outlets

| | |
|---|---|
| delegate | Tram App Delegate |
| navigationItem | |
| tabBarItem | |
| view | |

Referencing Outlets

| | |
|---|---|
| tabBarController | Tram App Delegate |
| New Referencing Outlet | |

**MainWindow.xib (English)**

View Mode Info Search Field

| Name | Type |
|---|---|
| File's Owner | UIApplication |
| First Responder | UIResponder |
| Tram App Delegate | TramAppDelegate |
| Window | UIWindow |
| State Handling Tab Bar Controller | StateHandlingTabBarController |
| Tab Bar | UITabBar |
| Selected State Handling Navigation Controlle... | StateHandlingNavigationController |
| State Handling Navigation Controller (recent... | StateHandlingNavigationController |
| State Handling Navigation Controller (locatio... | StateHandlingNavigationController |
| State Handling Navigation Controller (About) | StateHandlingNavigationController |

**Library**

- Library
  - Cocoa Touch Plugin
    - Controllers
    - Data Views
    - Inputs & Values
    - Windows, Views & Bars
  - Custom Objects

process of unknown duration.

**Page Control** – Displays a dot for each open page in an application and supports sequential navigation...

Cocoa Touch Plugin – Windows, Views & Bars

**Window** – Defines windows that manage and coordinate the views displayed on the screen.

**View** – Represents a rectangular region in which it draws and receives events.

**Search Bar** – Displays an editable search bar, containing the search icon, that sends an action message to a...

**Navigation Bar** – Provides a mechanism for displaying a navigation bar just below the status bar.

**View Controller**
UIViewController

Provides view-management functionality for toolbars, navigation bars, and application views. The UIViewController class also supports modal views and rotating views when device orientation changes.

## Platform Details: Application Data

Android

- application local storage
- shared /sdcard files
- SQLite DB

- API to store non committed user input for next invocation

iPhone

- Bundles
- app local docs, settings, tmp
- no shared local storage
- SQLite DB

- no support for state persistency, you're on your own

## Platform Details: Hardware

Android

- ARM, ~500MHz
- 128 MB RAM
- 1GB builtin Flash
- optional SDCARD
  - complicates handling
- GPU??
- aGPS, motion sensor
- Compass!

iPhone

- ARM, ~400MHz. GPU
- 128MB RAM
  - max 50MB for App
- Unified Filesystem >= 8GB
- Hardware assisted 2D and 3D graphics
  - animations are cheap
- aGPS, motion sensor

## Platform Details: level of detail, e.g. accelerometer

Android

- more bells and whistles
- sampling rate presettings
- filtererd
- accuracy info

iPhone

- request sampling rate
- one delegate
- one event (x,y,z)
- unfiltered data (noisy)

## DRM: iPhone

- developers always need Apple signed certificates
- all developers and devices must be registered with Apple
- correct certificates must be installed in Xcode and on the device
  - this is not always obvious and may cause ‘trouble’
- every deployed App can be traced back to an individual and/or company
  - regardless whether for development or distribution

- stealing of intellectual properties is difficult
- no software pirates

## DRM: Android

- non existent
- no payment/revenue models

## How to make money: iPhone

- sell applications via App Store to end users
  - 70:30 revenue sharing
  - attention: if you want to earn money with an App, then you can only sell it!
- iTunes Affiliate Program; 5% for placing a link/logo
- sell services and/or subscriptions via web apps only
- handling of closed (and paying) user groups is not well supported yet:
  - Adhoc Profiles (limit: 100 devices)
  - Corporate Clients (limit: 500+ employees)

## How to make money: Android

- nothing available out of the box
- Android Marketplace: no payment in place

## Part III: Essence

## Essence iPhone

- Home button
- 3rd Party Apps cannot do harm to the phone and/or user
  - e.g. transfer costs, battery life
- simple and understandable UI and process model
- one App at a time
- full traceability of App providers
- Apple controlled quality, plus user feedback (App Store)
- 3rd P: ‘plug-n-play’ installations (no questions, all or nothing)
- uniform physical parameter:
  - no buttons, one screen size

- reliability and trust

## Essence Android

- Back Button works always same (well, nearly….)
  - weak user visible distinction between applications
- 3rd Party Apps have equal rights to builtins
- border between Apps is blurred
- self signed Apps
- community controlled quality (Android Marketplace)
- 3rd P: access rights are granted by user at install time
- platform for many devices:
  - different screen sizes and button sets

- features and developers attractiveness

## In the end: its philosophy

- iPhone
    - users first
    - Home Button
    - obvious and sexy UI
    - apps and developers will follow
- Android
    - developers first
    - Back Button
    - open, feature rich, sexy architecture
    - apps and users will follow

## Rules for developers

- use the devices on a daily basis to get an understanding
- learn by looking into other apps
- read and follow the Human Interface Guidelines
  - also Android developers can learn a lot from Apple docs
- write a mission statement for your App

- people use and see mobile phones differently to a PC
  - apps must be fast
  - apps must be reliable
  - the UI must be slick, obvious and sexy
  - letter typing must be avoided as long as possible