PyConPl '13
Szczyrk, 17-20 X

Writing standalone Qt & Python applications for Android

Martin Kolman
Red Hat
http://www.modrana.org/pyconpl2013
martin.kolman@gmail.com
@M4rtinK

Overview

- Android applications
- Writing Android applications with Python and Qt
- How it works
- The *pydroid* project
- Examples
- Acknowledgment

What is Android ?

- Android is a very cut-down mobile Linux distribution developed by Google Inc & co
- it uses a patched Linux kernel
- most of the usual GNU userspace is replaced by custom tools
- custom incompatible libc called Bionic
- very basic package management
- running on some *~500 million* devices globally

Android applications

- there are two basic ways types of Android applications **Java based** and **NDK based**
- **Java based** applications run on the Androids custom Dalvik Java VM and use Android specific Java classes
  - they are not very portable
- **NDK based** applications are written in C/C++ with a slim Java wrapper for the UI
  - many Android games are NDK based due to better performance
  - Android uses a cut-down C library called *Bionic*, so porting libraries from other platforms might not be straightforward

Android applications

- applications are distributed in the APK (**A**ndroid **P**ac**K**age) format
  - which is just a zip package based on the JAR file format
  - no support for cross package dependencies
- the main Android application repository is called *Google Play*
  - but there are many third party ones, such as *Fdroid*, *Samsung apps* and others
- Android *tries* to maintain binary compatibility so that packaged applications should work across different Android devices and versions

Writing Android applications with Python and Qt

- Python has been running on Android for ages
- the Qt graphics library has been available for Android since early 2011
- there have been some proof of concepts of Qt-Python bindings working on Android
- and even some proof of concepts of distributing Python-Qt based applications

Lets put it all together !

So that:

- applications can be written entirely in Python
- Qt/QML can be used for the GUI
- the applications can be easily debugged
- all binary components can be recompiled at will
- the end result is a standalone Google Play compatible APK package
- deployment is as easy as possible

How it works

the Necessitas project

- Qt4 for Android
- provides Qt libraries compiled for Android 2.2+
- most of Qt functionality is supported, including OpenGL acceleration and Qt Quick 1.1
- handles automatic library downloads & updates through the *Ministro* service

Python for Android

- provides a Python interpreter for the Python code to run in and the Python-Qt bindings are compiled against it
- there are multiple projects providing Python for Android, for this initiative, Python from the *Kivy* project was used
  - https://github.com/kivy/python-for-android

PySide Python-Qt bindings

- wraps all public Qt classes for use from Python
- compiled using the Android NDK against the Necessitas Qt libraries and Python for Android
- build scripts:

  https://github.com/M4rtinK/android-pyside-build-scripts

Optional : Qt Components

- based on *MeeGo Qt Components*
- provide high level UI components for use in QML
- modified to work with Android screen rotation
- compiled with the Android NDK
- source code available from Gitorious:

  https://qt.gitorious.org/~martink/qt-components/martinks-ineans-qt-components/commits/android

What needs to be in the package

- the Python application code & any QML code
- Python & PySide compiled for Android
- Necessitas Java boilerplate & C++ wrapper
  - gets libs from Ministro and creates application window
  - starts the embedded Python interpreter
- optionally – Qt Components
- as a result, the package has about **15 MB** :)
- *...but there are multiple ways to trim it down*

How to create the package

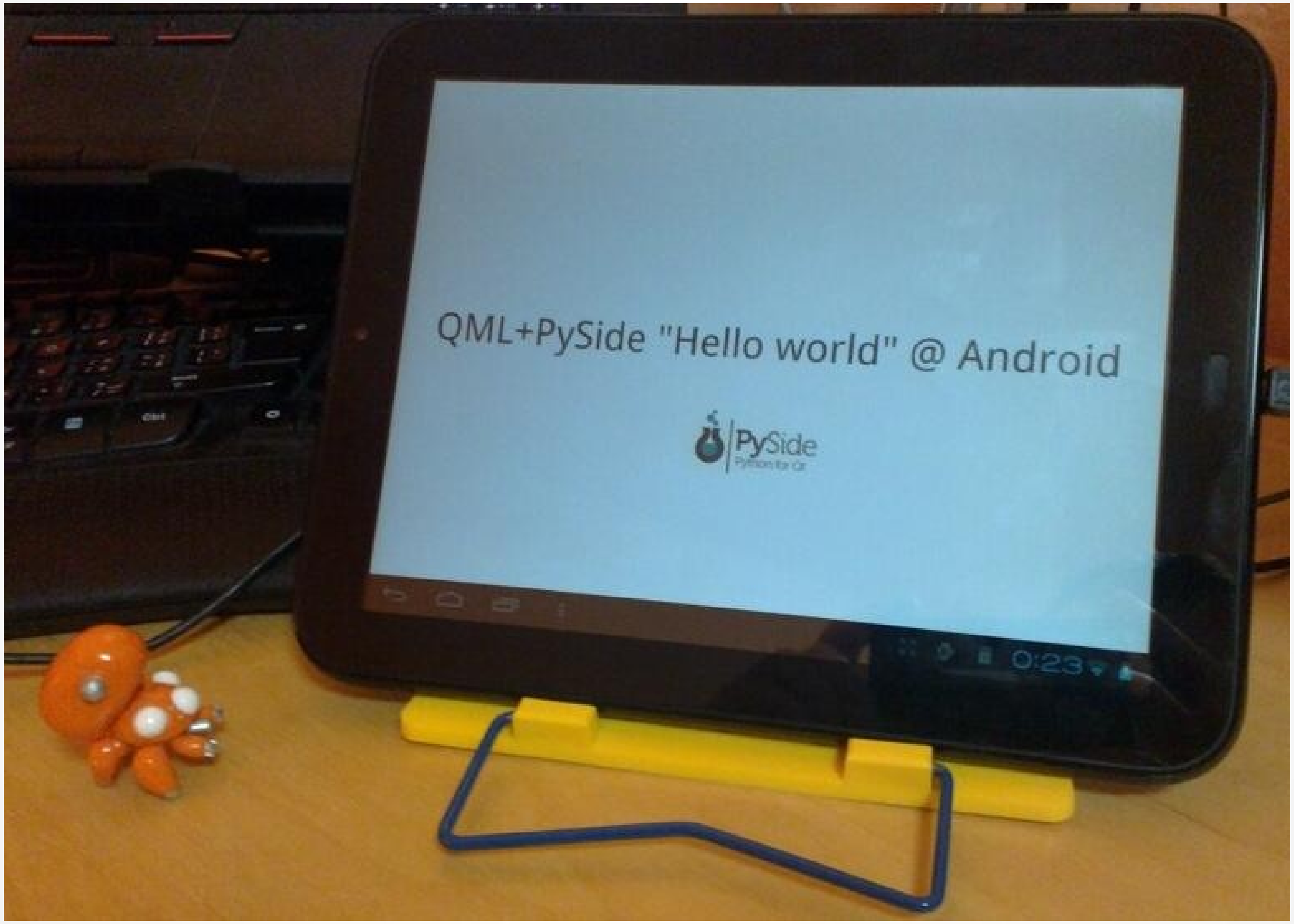
- with the Necessitas Qt creator
  - just press the **run** button
  - you can get an example project from here:

  https://github.com/M4rtinK/android-pyside-example-project
- packages can be also created from the command line using *qmake* and *ant*

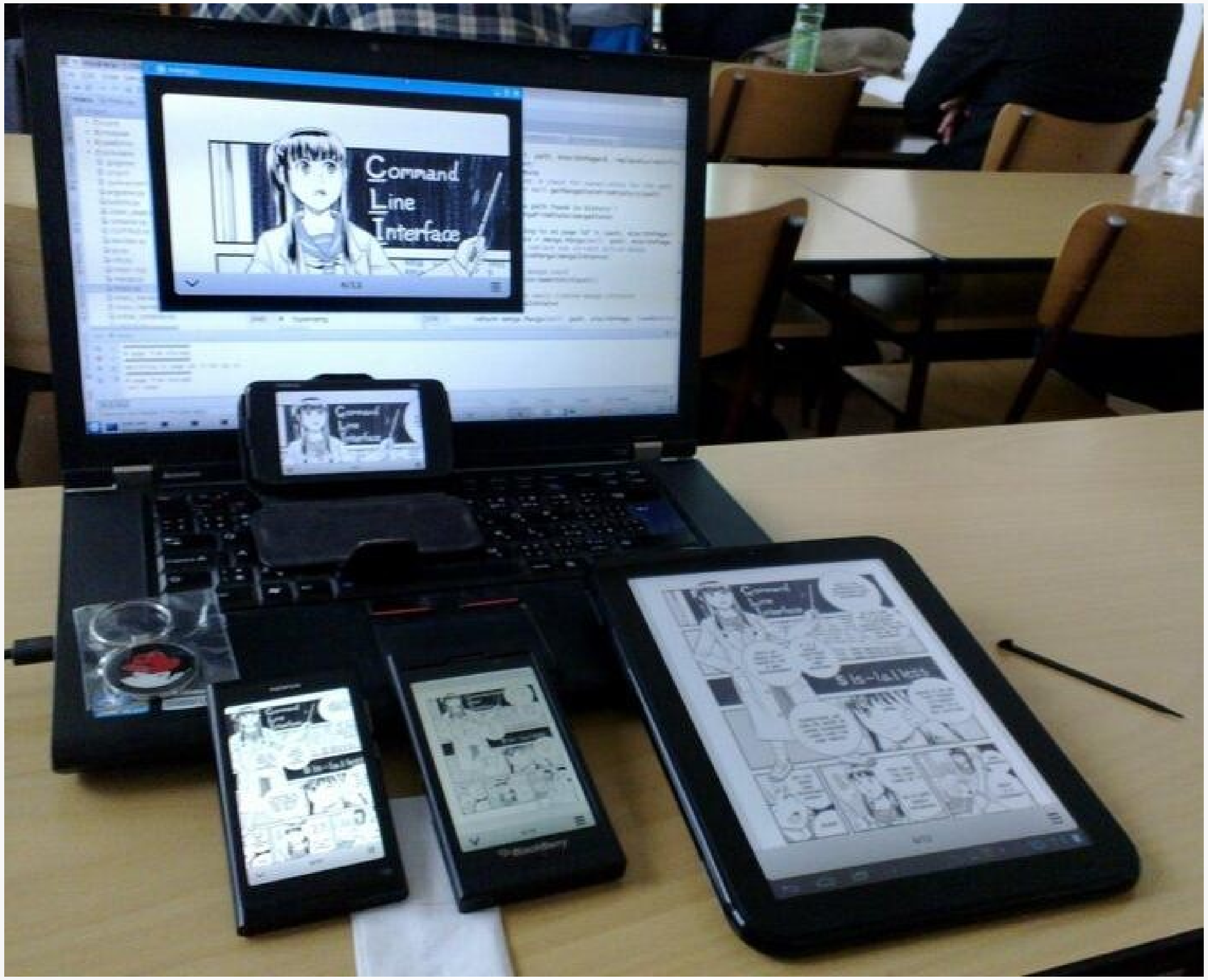
How everything gets in place on first start

- there are two zip files in the APK
  - one is for the application, the second for libraries
- the boilerplate acquires Qt libraries from Ministro and then unpacks both zip files to the application folder
- then the application is started
- all subsequent starts are as fast as for normal Android applications

How it looks like



Not only Android applications



The previous image shows

- the *Mieru* manga and comic book reader
  - Python + PySide + QML + Qt Components

    https://github.com/M4rtinK/mieru
- running on:
  - Android
  - BlackBerry 10
  - Nokia N900 & N9
  - PC with Ubuntu 12.10

PySide for Android guide

a detailed guide for building PySide and it to build Python application APKs

- available on the Qt Project Wiki

  http://qt-project.org/wiki/PySide_for_Android_guide

  – short URL:

  http://bit.ly/Zw6zHf

The pydroid project

is a really nice project developed by Aaron Richiger

- based on my PySide for Android work but even more user/developer friendly
- and **MUCH MORE POWERFUL** :)
- official website (well, a github repo :) :

  https://github.com/raaron/pydroid
- short URL:

  http://bit.ly/16CI3Kj

pydroid - features

- effortless *setup.py* based installation
- automatic project template generation
  - QWidget, QML, QtC, MWC, no-MWC, Pyjnius, etc.
- supports both QtCreator based and CLI only package generation and deployment
- automatically creates the app & lip zip bundles
- progress bar during first start/installation
- supports *pip* for adding Python modules to the project

pydroid - features

- support for logging from python directly to the android log facility
  - which is piped directly to the QtCreator console :)
- fast deployment to an Android device
  - 5 seconds from pressing the *fast deploy* button to application finishing startup on device
- bash autocomplete support
- pydroid & project diagnostics
  - just run `pydroid status`
- simple example applications is already available in Google Play
  - just search for *pydroid*

Installing pydroid

clone from Git

```
git clone https://github.com/raaron/pydroid.git
```

install

- `sudo python setup.py install`

and restart your shell, or else autocomplete will not work

- if you want to use command line deployment, don't forget to fill in the paths *~/.pydroid/deploy.conf*

Generate an example application

to generate the Qt Components example:

```
pydroid create example qt_components
```

- there are also other examples
  - QML only, Qwidget, MWC, Pyjnius, etc.

Deploying with Qt Creator

- open the *.pro* file in the Necessitas Qt Creator
- set architecture of your device
- hit “Run” (or *ctrl* + *r*)
- that's it :)

Future plans -> Qt5 !

- has much improved Android support
  - sensors, positioning, JNI interface, QtCreator support
- built-in QtComponents -> QQuick Controls
- canvas based vector drawing
- generally much better performance
- more modular
- more built-in stuff + moduarization = smaller APK

Qt5 Python bindings

- PySide
- PyOtherSide
- PyQt

PySide

- Qt4 only, Qt5 is not supported at the moment

Qt5 - PyOtherSide

- minimal asynchronous Qt5 bindings
  - developed by Thomas Perl (THP)
  - provide an interface between Python code and QtQuick 2.0
- implemented as a Python extension & Qt plugin
- very fast startup
  - Qt starts first, then starts the embedded Python interpreter
  - does not need to resolve all function symbols at startup like normal Qt bindings

Qt5 - PyOtherSide

- very small – compiled binary has ~100 **kB**
  - PySide has about 5 **MB**
- everything can be asynchronous
  - Python does it's stuff and calls callback to notify Qt
- image provider support
  - images can be loaded from Python data
- project repository:

  https://github.com/thp/pyotherside

Qt5 - PyQt

- PyQt5 added Qt5 support
- provides bindings to all Qt5 classes
  - with all the related advantages and downsides
  - access to all Qt classes VS slow startup and big size
- compiling PyQt5 for Android might be complicated
  - PyQt authors mentioned having Android support on their roadmap

Acknowledgements & sources

- Aaron Richiger for the wonderful pydroid project
- THPs PySide for Android – showing that this is possible
- Adrià Cereto-Massagué – integrated & improved THPs patches for Shiboken and PySide
- the Android-Python2.7 project – solved the APK bundling issue
- the Kivy project - provides Android-buildable Python 2.7
- the BlackBerry-Py Building PySide guide – I've used this as a base when making the Android build scripts
- the Necessitas project – made Qt on Android possible
  - also provides the Necessitas Qt Creator used for by the example project for building standalone APKs
- Qt-Project – provides the GUI toolkit :)
- PySide – provides the Python-Qt bindings Ineans Qt Components – with small modifications used in the example application & project

Thanks !

- Questions ?