Erfahrungen mit Py4A sowohl mit SL4A als auch mit Kivy

python™

Nik Klever
Hochschule Augsburg
Fakultät für Informatik

Inhalt

- Einführung und generelle Aspekte
- SL4A Grundlagen speziell für Py4A
- SL4A-Py4A API
- SL4A-Py4A Beispiele
- Kivy Grundlagen
- Kivy Python-For-Android
- Kivy Beispiele
- Kivy PyJNIus
- Ausblick

Inhalt

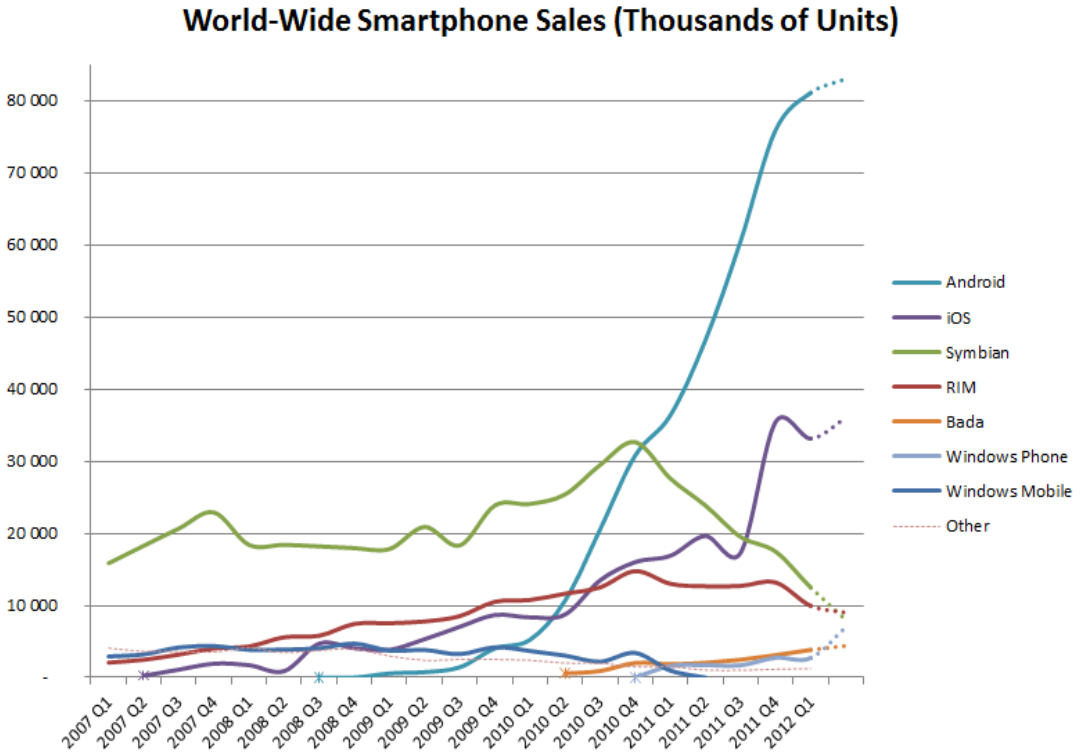
- Einführung und generelle Aspekte
- SL4A Grundlagen speziell für Py4A
- SL4A-Py4A API
- SL4A-Py4A Beispiele
- Kivy Grundlagen
- Kivy Python-For-Android
- Kivy Beispiele
- Kivy PyJNIus
- Ausblick

Motivation und Hintergrund

- Lehre im Studiengang "Interaktive Medien" seit 2001
  - "Interaktive Media" ist ein Studiengang, der zu 50% aus Informatik-Fächern und zu 50% aus Design-Fächern besteht
  - die Studierenden beginnen ihre Programmiererfahrung mit einer zweisemestrigen Java-Ausbildung
  - seit der Umstellung auf den Bachelorstudiengang (2006) mit dem Versuch, die Studierenden bereits in den Grundlagenfächern zu motivieren, Python für die Netzwerk- und Webprogrammierung (seit 2008 mit web2py) zu verwenden
- Forschungsprojekte *http://klever.hs-augsburg.de/mobile*
  - Java Projekte auf Symbian Handys seit 2003
  - einige Projekte mit Python on S60 in den Jahren 2007 und 2008
  - aber alle diese Projekte waren meist einzelne Abschlussarbeiten oder Einzelprojekte
  - und keine dieser Projekte konnte man für die Lehre in Anfängerklassen verwenden
- Anforderungen an die Lehre
  - benutzerfreundliche Bedienung und Einfachheit
  - schnelle Erfolgsumsetzung
  - Open Source und keine eingeschränkten Frameworks

Umstieg auf Android

- Android als Betriebssystem für Handys ist erstmalig 2008 veröffentlicht worden
- ein wachsender Smartphone Markt seit 2010 – fast alle Studierenden besitzen ein solches Gerät, sodass hierfür keine zusätzlichen Kosten für die Fakultät entstehen
- auf der Grundlage von Linux bringt Android diese Geräte auf Desktop Qualität



Quelle: http://en.wikipedia.org/wiki/Mobile_operating_system

Vorteile und Usability

- **Device Driven Development** vs. Emulator Driven Development
  - es wird nur ein Text Editor benötigt und es ist keine komplexe Emulator Installation notwendig
- **Warum ?**
  - die Einstellungen und Geräteabhängigkeiten einer Emulator Installation sind weniger für Anfänger denn für fortgeschrittene Programmierer geeignet
- **Wie ?**

Smartphone GUI und Beamer

- **MicroUSB – HDMI Adapter**
  - Mit diesem Adapter (Android > 4.0) lässt sich die GUI eines Smartphones sehr leicht über moderne Beamer projizieren – allerdings sind MicroUSB-HDMI-Adapter in der Regel **geräteabhängig** !
- **HDMI – VGA Adapter**
  - Mit diesem Adapter lassen sich manche ältere Beamer ebenfalls anschliessen
- **Vorsicht** – bitte auf jeden Fall **testen** !
  - Es gibt zum einen soviele Adapter der obigen Art, die meist nur zu einem einzigen Gerät kompatibel sind, sodass die entsprechende Kombination **Smartphone – HDMI – VGA – Beamer** immer vorher getestet werden muss !

WIFI vs. USB

- Wenn man einen FTP Fileserver – wie z.B. den Open Source FTP Server **SwiFTP** – auf dem Android Smartphone verwendet und einen FTP Client – wie z.B. den Open Source FTP Client **FileZilla** – auf dem Desktop, dann kann über diese WIFI basierte FTP Verbindung das gesamte Systemverzeichnis des Smartphones angesprochen werden und nicht nur die SDCard (natürlich entsprechend den eingestellten Sicherheitsmechanismen – mit Cyanogenmod entsprechend mehr)
- zusätzlich wird die SDCard während der bestehenden Verbindung nicht unmounted wie bei einer USB Verbindung
- die Bearbeitung einer Datei auf dem Desktop-PC und das anschliessende Testen auf dem Smartphone kann daher on-the-fly geschehen
- **SwiFTP APK**
  - http://github.com/ppareit/swiftp
- **FileZilla**
  - http://filezilla-project.org/

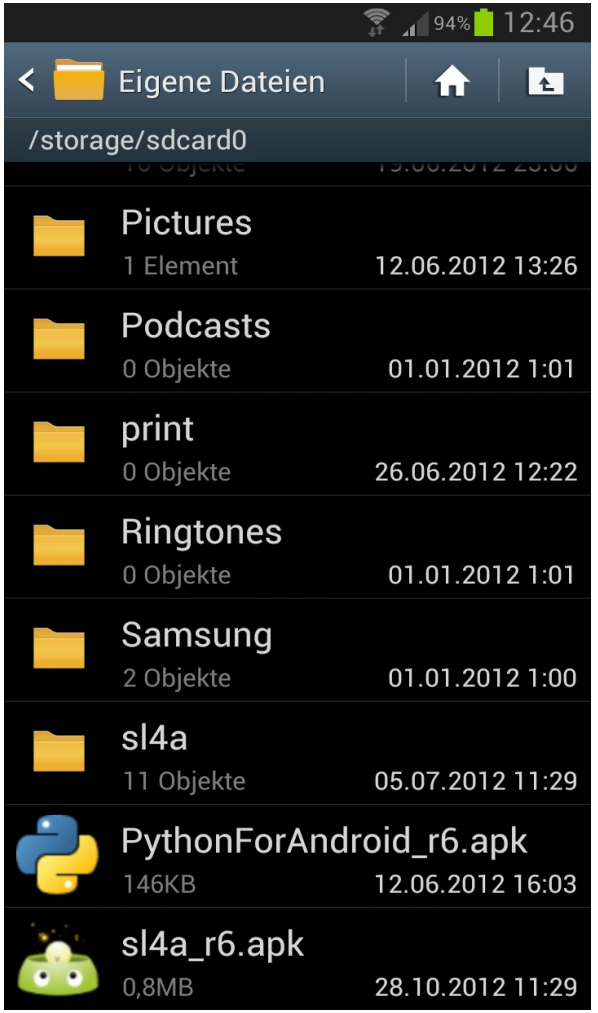
Achtung:
bei aktuelleren Betriebssystemversionen kann es im SwiFTP notwendig sein, in den „Advanced Settings“ die Angabe zu „Stay in Folder“ auf „/storage“ zu setzen

Inhalt

- Einführung und generelle Aspekte
- SL4A Grundlagen speziell für Py4A
- SL4A-Py4A API
- SL4A-Py4A Beispiele
- Kivy Grundlagen
- Kivy Python-For-Android
- Kivy Beispiele
- Kivy PyJNIus
- Ausblick

SL4A - Scripting Layer for Android

- SL4A ist ein Wrapper für die Java Android API über sogenannte Android Fassaden (facades) mittels RPC Aufrufen
  - code.google.com/p/android-scripting



- derzeitiges Release: sl4a_r6 (9.7.2012)
  - http://code.google.com/p/android-scripting/downloads/

SL4A - Scripting Layer for Android

- SL4A RPC Server

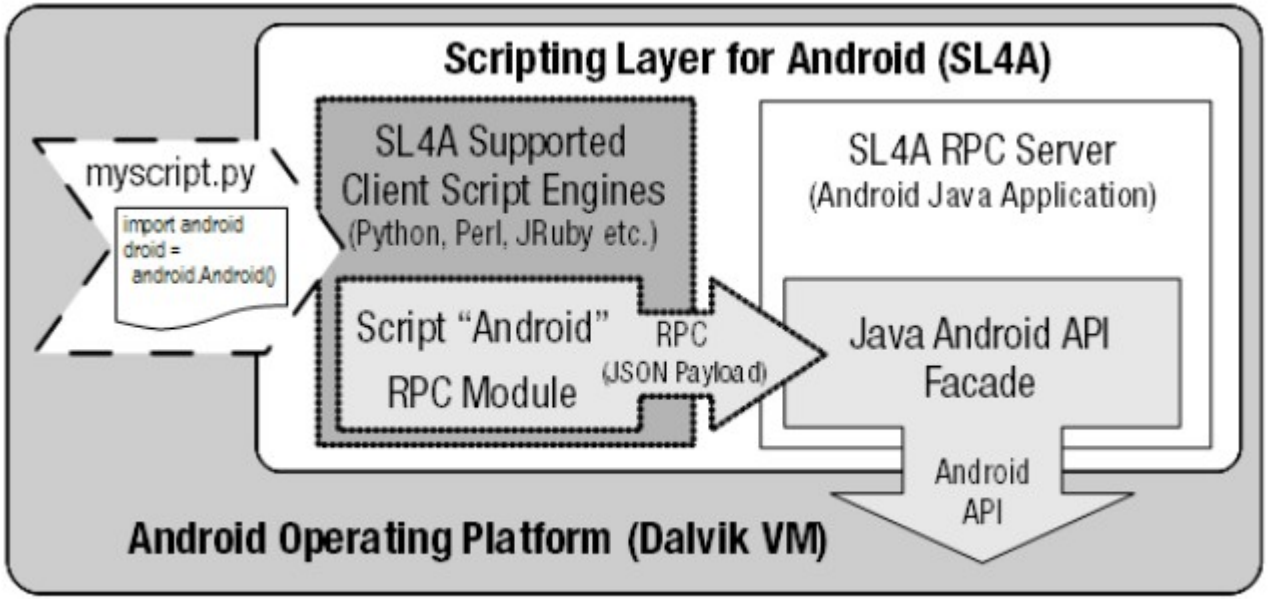


Figure 5–21. *SL4A architecture overview*

Source: Jordan/Greyling, Practical Android Projects

- SL4A benutzt die JavaScript Object Notation (JSON) für die Kommunikation zwischen dem SL4A RPC Server und seinen Clients.

SL4A Eigenschaften

- SL4A erlaubt eine **schnelle, prototypische Entwicklung von Anwendungen für Android auf dem Gerät selbst** mittels Skriptsprachen
- SL4A stellt Android Fassaden zur Verfügung über die man die **Android APIs mittels JSON RPC Aufrufen** anspricht.
- Mittels den Android Fassaden ist diese **API vorrangig auf die einfache Erstellung von Skripten ausgerichtet** als auf die Performanz und Schnelligkeit dieser Skripte.
- Python ist aktuell die cross-compilierte C Binär Version, die nativ auf Android läuft.

Quelle: siehe http://code.google.com/p/android-scripting/wiki/FAQ

Inhalt

- Einführung und generelle Aspekte
- SL4A Grundlagen speziell für Py4A
- SL4A-Py4A API
- SL4A-Py4A Beispiele
- Kivy Grundlagen
- Kivy Python-For-Android
- Kivy Beispiele
- Kivy PyJNIus
- Ausblick

SL4A API

Die SL4A API lässt sich in die folgenden Gruppen von Fassaden unterteilen:

- Subdevices (BatteryManager, Bluetooth, **Location,** SensorManager, WiFi, SignalStrength)
- Media (MediaPlayer, MediaRecorder, SpeedRecognition, TextToSpeech, Phone, Sms, **Camera**, WebCam, ToneGenerator)
- **UI**
- Contacts
- Intents (e.g. scanBarcode, search, viewHtml, viewMap, ...)
- Andere (Android, **Events**, Preferences, Settings, WakeLock, ActivityResult, ApplicationManager)

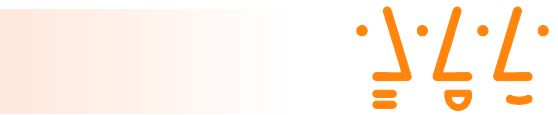
SL4A API - Liste

Liste aller Fassaden:

http://code.google.com/p/android-scripting/wiki/ApiReference

- Dort findet sich eine ausreichende Beschreibung für die meisten Methoden
- Falls nicht, hilft auch der Link zur erweiterten API Hilfe von http://www.mithril.com.au/android/doc

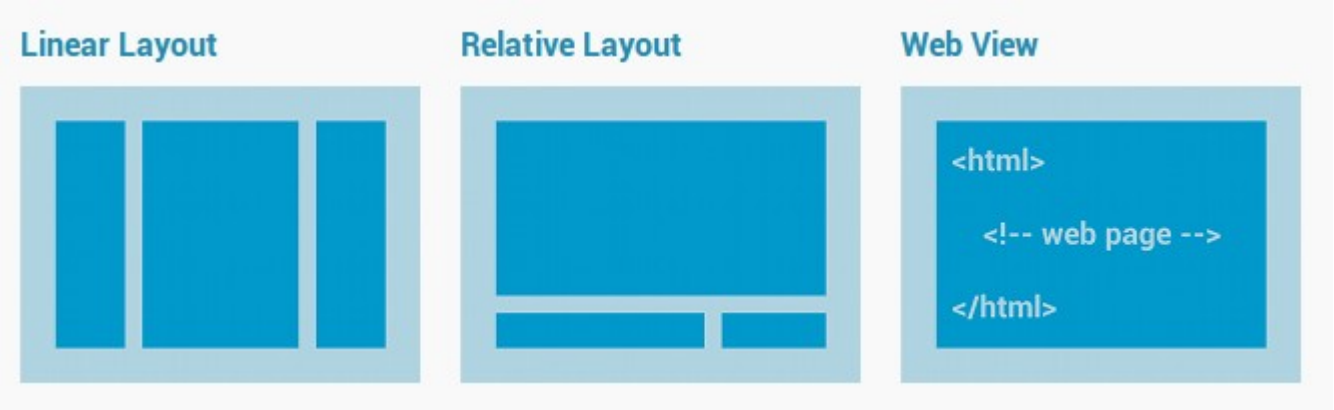
Py4A - Python for Android



- PythonForAndroid-r6.apk (Python 2.6.2)
  - http://code.google.com/p/python-for-android/
  - oder Python3ForAndroid-r6.apk (Python 3.3)
- Während der Installation werden dann neben dem Interpreter auch die aktuellen Module und Scripts aus dem obigen Download-Bereich heruntergeladen und installiert
  - Latest Versions, interpreter: 16, extras: 14, scripts: 13
- zusätzliche Dokumentation:
  - Paul Ferrill: Pro. Android Python with SL4A, 2011, Apress

SL4A – Full UI Support

- Unterstützung der Standard Android UI mittels XML Templates
  - XML Layout Schema wie auf der Android Entwickler Seite beschrieben
  - http://developer.android.com/guide/topics/ui/overview.html
  - LinearLayout, RelativeLayout



Full UI Beispiel

- adaptiert von fulluitest.py

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:id="@+id/background"
              android:orientation="vertical"
              android:layout_width="match_parent"
              android:layout_height="match_parent"
              android:background="#ff000000">
        <TextView android:layout_width="match_parent"
                  android:layout_height="wrap_content"
                  android:text="TextView"
                  android:id="@+id/textView"
                  android:textAppearance="?android:attr/textAppearanceLarge"
                  android:gravity="center_vertical|center_horizontal|center"></TextView>
        <EditText android:layout_width="match_parent"
                  android:layout_height="wrap_content"
                  android:id="@+id/editText"
                  android:tag="Tag Me"
                  android:inputType="textCapWords|textPhonetic|number">
                  <requestFocus></requestFocus></EditText>
        <Button android:id="@+id/button"
                  android:layout_width="wrap_content"
                  android:layout_height="wrap_content"
                  android:text="Ok"></Button>
        <CheckBox android:layout_height="wrap_content"
                  android:id="@+id/checkBox1"
                  android:layout_width="234dp"
                  android:text="Howdy, neighbors."
                  android:checked="true"></CheckBox>
</LinearLayout>
```

fulluitest.py


```
import android
droid=android.Android()

def eventloop():
  while True:
    event=droid.eventWait().result
    print event

    if event["name"]=="click":
      id=event["data"]["id"]
      if id=="button":
        droid.fullSetProperty("editText","text","OK has been pressed")
      elif id=="checkBox1":
        droid.fullSetProperty("textView","text","Other stuff here")
        if event["data"]["checked"] == "false":
          droid.fullSetProperty("background","backgroundColor","0xff7f0000")
        else:
          droid.fullSetProperty("background","backgroundColor","0xff000000")
    elif event["name"]=="key":
      if event["data"]["key"]=="4":
        return

layout=""" ... """

print droid.fullShow(layout)
eventloop()
print droid.fullQuery()
print "Data entered =",droid.fullQueryDetail("editText").result
droid.fullDismiss()
```

Inhalt

- Einführung und generelle Aspekte
- SL4A Grundlagen speziell für Py4A
- SL4A-Py4A API
- SL4A-Py4A Beispiele
- Kivy Grundlagen
- Kivy Python-For-Android
- Kivy Beispiele
- Kivy PyJNIus
- Ausblick

SL4A Beispiel: Web2py

- Was ist Web2py ?
  - modernes Python Web Application Framework
  - von Massimo di Pierro anfänglich erstellt um Studierende in Webprogrammierung auszubilden
  - inzwischen eine interessante Alternative für Python basierte Web Application Frameworks
- läuft **out-of-the-box**
  - integrietes SQLite 3
  - integrierter Roxen Webserver
  - integriertes jQuery
- nichtsdestotrotz modular und leicht anpassbar auf alle bekannten Datenbanken und Webserver
  - siehe auch Vortrag „web2py“ PyCon DE 2011

Smartphone Web2py

- zum einen kann das Handy wie üblich als Client mittels jQuery mobile verwendet werden
  - http://jquerymobile.com/
  - http://web2py.com/plugins/plugin_jqmobile/about
- zum zweiten aber auch als Server über Py4A und SL4A, allerdings mit den folgenden Einschränkungen:
  - web2py nutzt für sein Disk-Caching das Python Modul shelve basierend auf der bsddb, welche aber nicht in Py4A integriert ist (nur dumbdbm)
    - Workaround: http://klever.hs-augsburg.de/aktuelles#web2pyAndroid
  - die Administrationsoberfläche von web2py lässt sich aufgrund eines I/O Errors, der wiederum einen „internal error in the Python interpreter“ mit dem Hinweis „error return without exception set“ aber ohne Zeilen und Dateiangabe auswirft, nicht aufrufen
    - Workaround: Entwicklung auf dem Desktop betreiben

Sensoren

einfaches Code Beispiel für die Anbindung der Sensoren über das web2py Framework auf meinem Smartphone: http://10.20.x.y:8000/Android/default/sensors :

```
import android
droid = android.Android()

def sensors():
    result = None
    while not result:
        # 1 = all Sensors, 100 ms time between readings
        droid.startSensingTimed(1,100)
        time.sleep(2)
        result = droid.readSensors().result
    return response.render('default/sensors.html',dict(data=result))
```

Lokation

einfaches Code Beispiel für die Anbindung des Lokation Aufrufs von SL4A aus dem web2py Framework heraus http://10.20.x.y:8000/Android/default/gps :

```
import android
droid = android.Android()

def gps():
    result = None
    while not result:
        # 2 Minutes as minimum time between updates,
        # 200 m minimum distance between updates
        droid.startLocating(2*60*1000,200)
        time.sleep(2)
        result = droid.readLocation().result
    return response.render('default/sensors.html',dict(data=result))
```

Kamera Aufnahme

einfaches Code Beispiel für den Aufruf der Kamera – die Pfad Variable ist gerätespezifisch anzupassen
http://10.20.x.y:8000/Android/default/picture :

```
import android, datetime
droid = android.Android()

def picture():
    dt = datetime.datetime.now()
    path = '/mnt/sdcard/sl4a/scripts/web2py/applications/Android/static/'
    PictureName = 'pic_%s.jpg'%dt.strftime('%Y_%m_%d_%H_%M_%S')
    response = droid.cameraCapturePicture(path+PictureName)
    if response.result['takePicture']:
        return dict(data=PictureName)
    else:
        return dict(data=None)
```

SL4A Web2py Beispiele

- Aufruf des Web2py Webservers über das WLAN
  - Sensoren Beispiel
  - Lokation Beispiel
  - Kamera Beispiel

- Abändern des Layouts
  - z.B. **picture.html** in einem Editor auf dem PC ändern
  - auf das Gerät mittels **FTP** übertragen
  - reload der entsprechenden Seite genügt

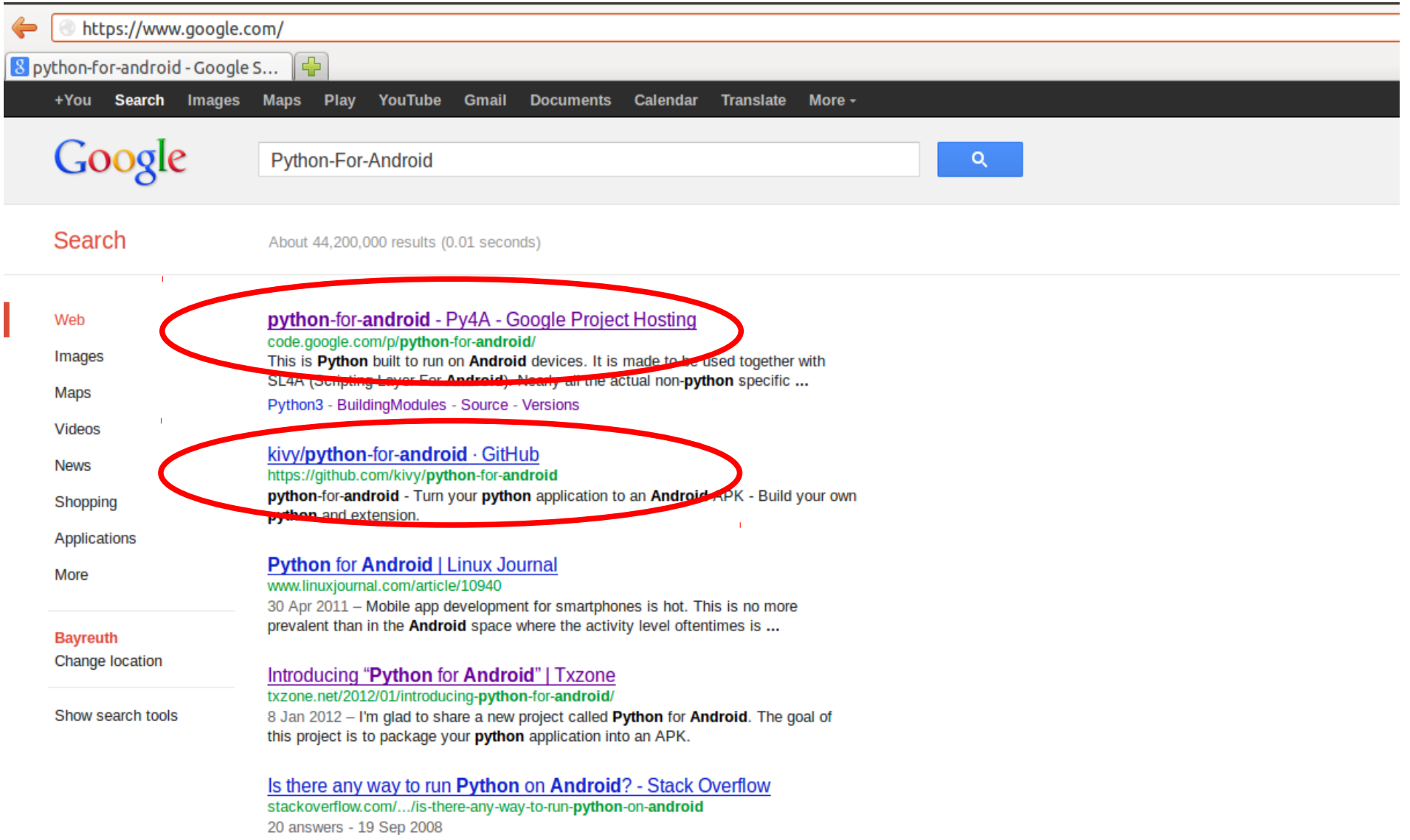
SL4A Beispiele auf dem Handy

- Aufruf des Web2py Webservers auf dem Smartphone unter Nutzung des jQuery mobile plugin
  - Sensoren Beispiel
  - Lokations Beispiel
- fulluitest.py
- Full Screen UI Wrapper
  - http://code.google.com/p/python-for-android/downloads/

Zusammenfassung SL4A

- **Zusammenfassung SL4A**
- tty Ausgabe und Dateioperationen (z.B. logging) wie man es erwartet,
- die meisten Python Standard Bibliotheksmodule sind integriert,
- tty input seit R6 ebenfalls möglich
- einfache Schnittstelle zu fast allen möglichen Subdevices und Mediengeräten eines Smartphones wie Bluetooth, Telefon, SMS, Lokation, Kamera, Beschleunigungssensor, Magnetsensor, ...
- elementares UI über XML und der Browserschnittstelle – angepasst an das Android UI Design
- **Was fehlt ?**
- das standardmäßige, touch-fähige Natural User Interface (NUI) kann infolge der RPC Architektur nicht integrativ unterstützt werden

„Python for Android“ existiert zweimal



Python for Android - Ergebnisse

- **python-for-android** - Py4A – als Google Projekt

  code.google.com/p/python-for-android

  „Python for Android“ als Py4A zusammen mit Scripting Layer for Android (SL4A), wie oben beschrieben
- kivy/**python-for-android –** als GitHub Projekt

  github.com/kivy/python-for-android

  ein weiteres „Python for Android“ zusammen mit Kivy, einem Python Framework für multitouch-fähige Geräte wie Tablets, Smartphones, etc.
- **Andere**

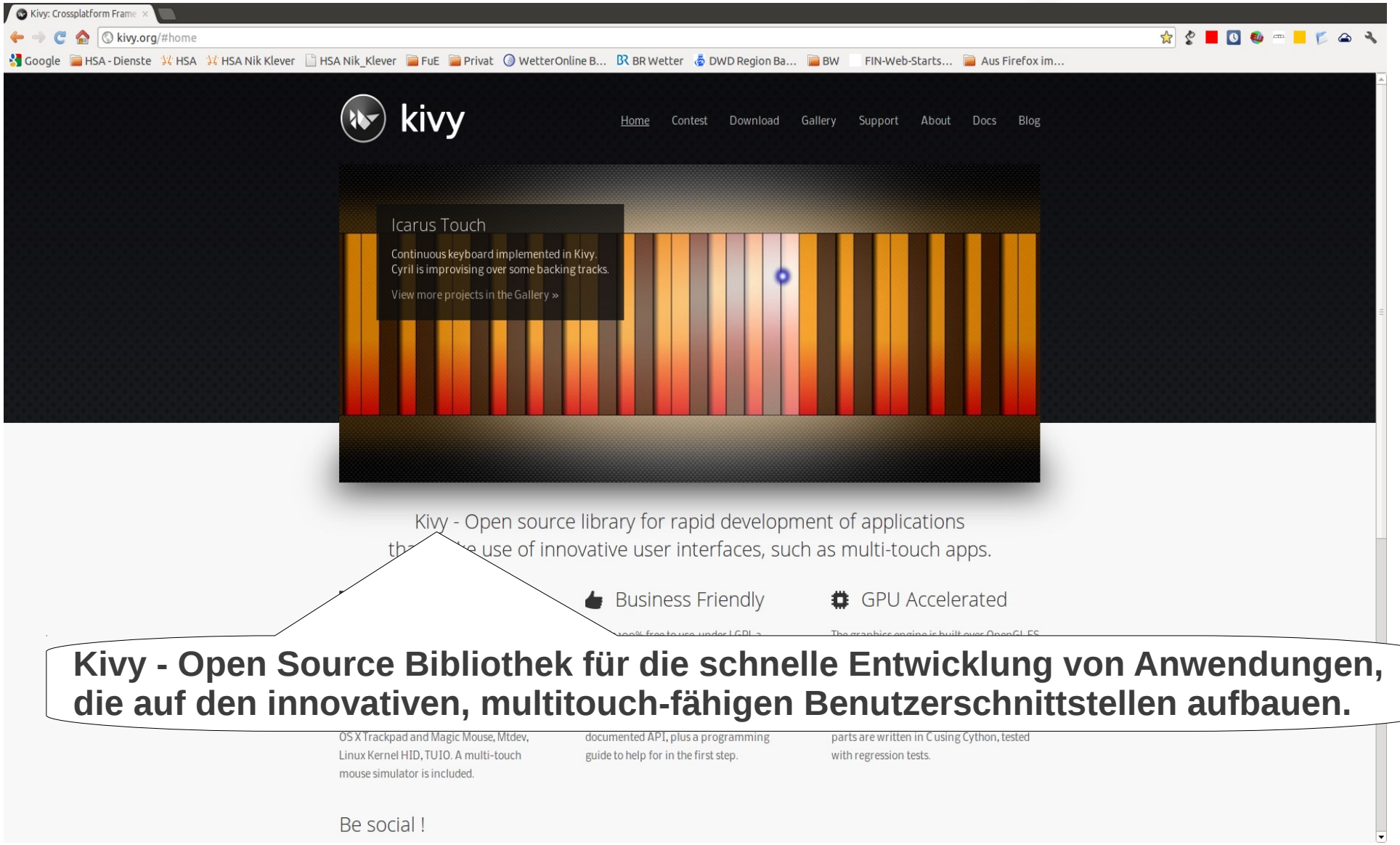
  es gibt noch einige ander Python „on“ Android Lösungen … (PySide, PyGame, PGS4A, Python-on-a-chip, …) auf die hier jedoch nicht eingegangen wird

Inhalt

- Einführung und generelle Aspekte
- SL4A Grundlagen speziell für Py4A
- SL4A-Py4A API
- SL4A-Py4A Beispiele
- Kivy Grundlagen
- Kivy Python-For-Android
- Kivy Beispiele
- Kivy PyJNIus
- Ausblick

Interaktive Medien: Kivy

Kivy – Unterstützung für NUIs – Natural User Interfaces



Kivy - Open Source Bibliothek für die schnelle Entwicklung von Anwendungen, die auf den innovativen, multitouch-fähigen Benutzerschnittstellen aufbauen.

Kivy - http://www.kivy.org

Cross Plattform

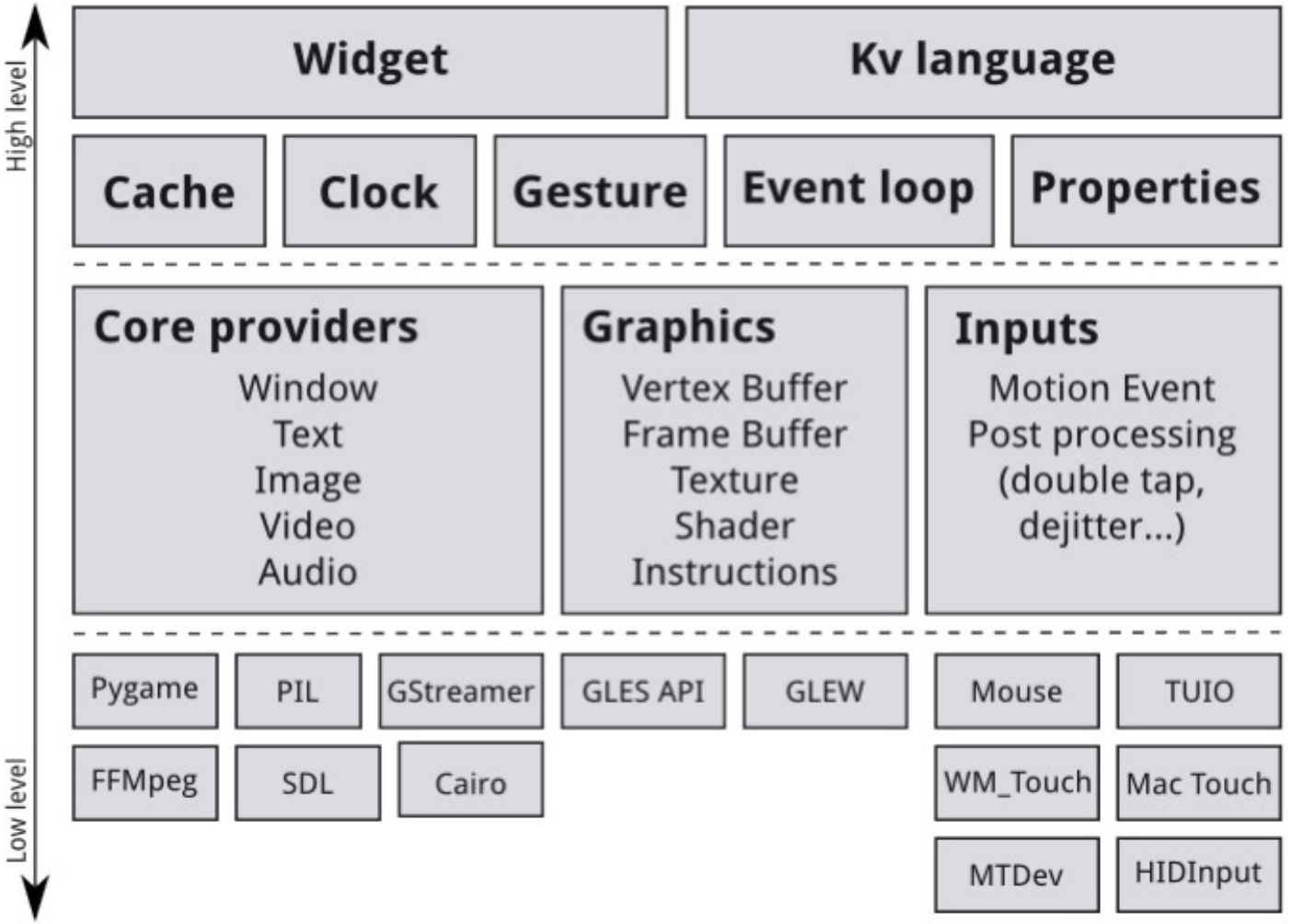
- Läuft auf Linux, Windows, MacOSX, Android und iOS
- kann nativ die meisten Eingabe Protokolle und Eingabe Geräte verwenden

GPU Beschleunigung

- die Grafik-Engine von Kivy baut auf OpenGL ES 2 auf
- derzeit sind mehr als 20 widgets integriert
- geschrieben in C und Cython

Quelle: www.kivy.org/#home

Kivy - Architektur



Quelle: www.kivy.org/docs/guide-index.html

Kivy API

- Kivy Framework (…, **App**, ...)
- Core Abstraction (Audio, Camera, Image, Text, ...)
- Extension Support
- Grafik (Basics (…, **Color, Ellipse, Line**, …), Canvas, ...)
- Input Management (Providers, Motion Event, …)
- Externe Bibliotheken (jinja2)
- Module (Inspector, Monitor, ...)
- Netzwerk Unterstützung (url)
- Widgets (..., **Widget, Button,** ...)

Inhalt

- Einführung und generelle Aspekte
- SL4A Grundlagen speziell für Py4A
- SL4A-Py4A API
- SL4A-Py4A Beispiele
- Kivy Grundlagen
- Kivy Python-For-Android
- Kivy Beispiele
- Kivy PyJNIus
- Ausblick

Kivy auf Android Geräten

- wie an der folgenden Stelle ausgeführt:
  - http://python-for-android.readthedocs.org/en/latest/android/
- wird die Java Android API in dem Android Modul von Kivy über den folgenden Weg angesprochen:
  - Cython Dateien transformieren nach JNI über
  - C JNI um die Android Java API über die JNI Layer anzusprechen
- Recipes (zusätzliche Python Module) sind erhältlich, aber noch nicht vergleichbar mit der SL4A API
- weitere Hinweise zu Python For Android über Kivy:
  - http://python-for-android.readthedocs.org/en/latest/extend/

Inhalt

- Einführung und generelle Aspekte
- SL4A Grundlagen speziell für Py4A
- SL4A-Py4A API
- SL4A-Py4A Beispiele
- Kivy Grundlagen
- Kivy Python-For-Android
- Kivy Beispiele
- Kivy PyJNIus
- Ausblick

MyPaint Beispiel


```
from random import random
from kivy.app import App
from kivy.uix.widget import Widget
from kivy.uix.button import Button
from kivy.graphics import Color, Ellipse, Line

class MyPaintWidget(Widget):
    def on_touch_down(self, touch):
        userdata = touch.ud
        userdata['color'] = c = (random(), 1, 1)
        with self.canvas:
            Color(*c, mode='hsv')
            d = 30
            Ellipse(pos=(touch.x - d/2, touch.y - d/2), size=(d, d))
            userdata['line'] = Line(points=(touch.x, touch.y))
    def on_touch_move(self, touch):
        touch.ud['line'].points += [touch.x, touch.y]

class MyPaintApp(App):
    def build(self):
        return MyPaintWidget()

if __name__ == '__main__':
    MyPaintApp().run()
```

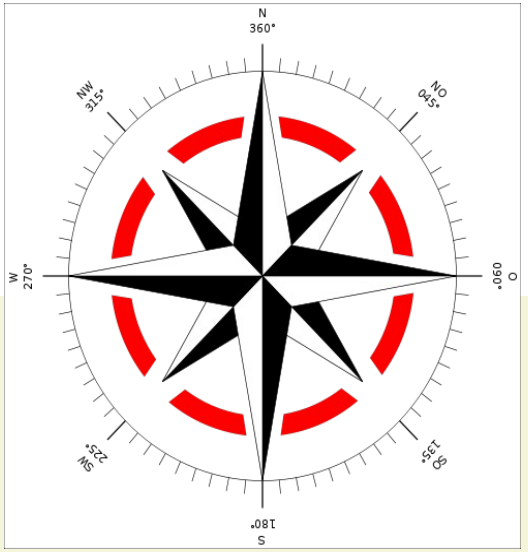
Quelle: http://kivy.org/docs/guide/firstwidget.html

Compass – Import Teil


```
import kivy
kivy.require('1.3.0')
from kivy.app import App
from kivy.uix.floatlayout import FloatLayout
from kivy.uix.button import Button
from kivy.uix.scatter import Scatter
from kivy.clock import Clock
from kivy.vector import Vector
from kivy.core.window import Window
from kivy.graphics import Color, Ellipse, Rectangle, Triangle
from kivy.logger import Logger

"""
importieren der Wrapper, die über JNI die Android Java API ansprechen
"""
from android import magneticFieldSensorEnable, magneticFieldSensorReading
```

Compass – Compass Widget




```
class CompassWidget(FloatLayout):
    """
    Compass Widget
    """
    def build(self,pos,size):
        """
        erstellt ein Grafikwidget mit einem Kreis, dessen
        Bildhintergrund eine Kompass Windrose ist
        """
        with self.canvas:
            self.pos = pos
            self.size = size
            self.windrose = Ellipse(source="500px-Windrose.svg.png",
                                    pos=pos, size=size)
```

Bildquelle:http://en.wikipedia.org/wiki/Compass_rose

Compass – Needle Widget 1


```
class NeedleWidget(Scatter):
    """
    Kompassnadel Widget, abgeleitet aus einem Scatter Widget
    """
    def __init__(self, **kwargs):
        """
        abgeleitet aus einem Scatter Widget, welches nicht interaktiv
        verschoben, skaliert und verdreht werden darf
        """
        super(NeedleWidget, self).__init__(**kwargs)
        self.do_rotation = False
        self.do_translation = False
        self.do_scale = False
        self.auto_bring_to_front = True

    def rotateNeedle(self,angle):
        """
        die Drehung der Nadel wird hier erzeugt, dabei ist der Winkel
        mit 0 Grad im Osten der Windrose - deswegen müssen 90 Grad vom
        entsprechenden Winkel abgezogen werden, um an die Windrose
        angepasst zu sein
        """
        self.rotation = angle - 90
        Logger.info('Compass: rotateNeedle angle=%s rotation=%s ' %
                    (str(angle),str(self.rotation)))
```

Compass – Needle Widget 2


```
def build(self,center):
    """
    erstellen der Kompassnadel aus zwei verschiedenfarbigen Dreiecken
    """
    needleSize = Vector(100., 60.)
    self.pos = center - needleSize/2.
    self.size = needleSize
    self.size_hint = [None, None]
    with self.canvas:
        Color(1., 0, 0)
        needleTP1 = Vector(needleSize[0]/2.,needleSize[1])
        needleTP2 = Vector(needleSize[0]/2.,0)
        needleTP3 = Vector(-needleSize[0],needleSize[1]/2.)
        needlePoints = (needleTP1[0],needleTP1[1],
                        needleTP2[0],needleTP2[1],
                        needleTP3[0],needleTP3[1])
        self.needleT1 = Triangle(points=needlePoints)
        Color(0.5, 0.5, 0.5)
        needleTP3 = Vector(2*needleSize[0],needleSize[1]/2.)
        needlePoints = (needleTP1[0],needleTP1[1],
                        needleTP2[0],needleTP2[1],
                        needleTP3[0],needleTP3[1])
        self.needleT2 = Triangle(points=needlePoints)
```

Compass – Compass App 1


```
class CompassApp(App):
    """
    Kompass Anwendung
    """
    def viewCompass(self, *largs):
        """
        Auslesen der aktuellen Magnetsensordaten und Berechnen der
        Deklination als Winkel zwischen X- und Y-Wert und Übergeben
        des Winkels an die entsprechende Funktion zur Drehung der Kompassnadel
        """
        (x, y, z) = magneticFieldSensorReading()
        d = Vector(x,y).angle((0,1))
        self.needle.rotateNeedle(d)
        Logger.info('Compass: viewCompass x=%s y=%s z=%s => d=%s'%(x,y,z,d))

    def stopApp(self,*largs):
        """
        Funktion zum Beenden der Anwendung
        """
        magneticFieldSensorEnable(False)
        Logger.info('Compass: stop largs '+str(largs))
        self.stop()
```

Compass – Compass App 2


```
def build(self):
    """
    Erstellen des Parent Widgets und dessen Hintergrundfarbe sowie
    des Kompasswidgets einschliesslich des Festlegens der Größe und
    Position des Kompasses, des Nadelwidgets und des Stopbuttons
    sowie letztendlich starten des Magnetsensors und des zeitgesteuerten
    Schedulers zum Auslesen des Magnetsensors
    """
    parent = FloatLayout(size=(500,500))
    Window.clearcolor = (1, 1, 1, 1)

    self.Compass = CompassWidget()
    parent.add_widget(self.Compass)
    CompassPos = Vector(50., 200.)
    CompassSize = Vector(400., 400.)
    self.Compass.build(pos=CompassPos,size=CompassSize)

    self.needle = NeedleWidget()
    parent.add_widget(self.needle)
    self.needle.build(center=CompassPos+CompassSize/2.)
```

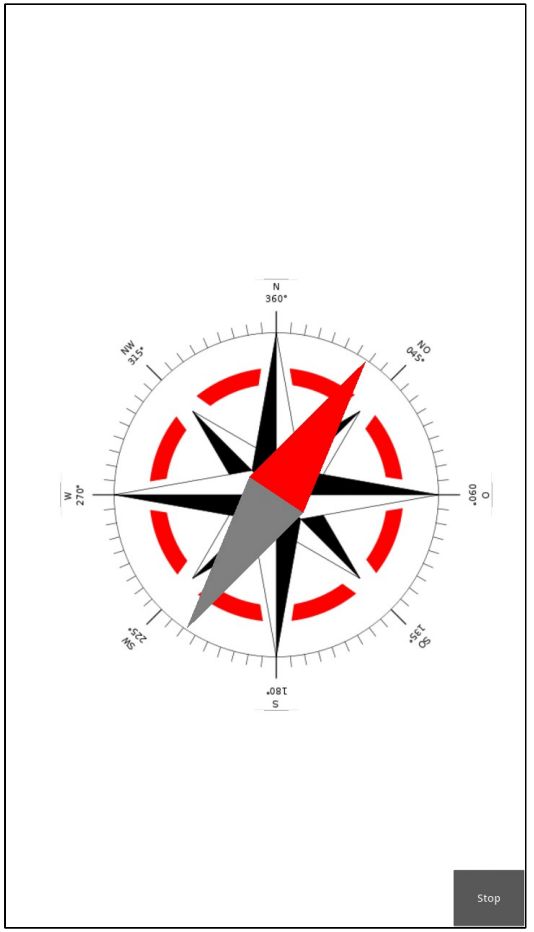
```
        self.needle = NeedleWidget()
        parent.add_widget(self.needle)
        self.needle.build(center=CompassPos+CompassSize/2.)

        self.stopButton = Button(text='Stop',
                                 pos_hint={'right':1},
                                 size_hint=(None,None),
                                 height=60)
        parent.add_widget(self.stopButton)
        self.stopButton.bind(on_press=self.stopApp)

        magneticFieldSensorEnable(True)
        Clock.schedule_interval(self.viewCompass, 1.)
        return parent

if __name__ in ('__main__', '__android__'):
    """
    Ausführen der Kompass-Anwendung
    """
    CompassApp().run()
```

Ergebnis



Sourcecode:https://github.com/kivy/kivy/tree/master/examples/android

Kivy Beispiele

- MyPaint
- Compass
- **Touchtracer** vom Kivy Team
- **Showcase** vom Kivy Team

Inhalt

- Einführung und generelle Aspekte
- SL4A Grundlagen speziell für Py4A
- SL4A-Py4A API
- SL4A-Py4A Beispiele
- Kivy Grundlagen
- Kivy Python-For-Android
- Kivy Beispiele
- Kivy PyJNIus
- Ausblick

PyJNIus

Die Java Klassen werden über **autoclass** angesprochen:

```
from jnius import autoclass

System = autoclass('java.lang.System')
System.out.println('hello world')
```

PyJNIus benutzt Java Reflection um über eine neue **autoclass**() an den Ergebnistyp heranzukommen:

```
>>> System = autoclass('java.lang.System')
>>> System
<class 'jnius.java.lang.System'>
>>> System.out
<java.io.PrintStream at 0x234df50 jclass=java/io/PrintStream jself=37921360>
>>> System.out.println
<jnius.JavaMethodMultiple object at 0x236adb8>
```

Der bisherige Weg Python → Cython → C JNI → Java API wird mit PyJNIus automatisiert

PyJNIus – Beispiel

Beispiel 1: Desktop

```
from jnius import autoclass

Stack = autoclass('java.util.Stack')
stack = Stack()
stack.push('hello')
stack.push('world')

print stack.pop()
print stack.pop()
```

Beispiel 2: Android

```
from jnius import autoclass

DisplayMetrics = autoclass('android.util.DisplayMetrics')

metrics = DisplayMetrics()

print 'DPI', metrics.getDeviceDensity()
```

Hardware.java

Auszug aus kivy/python-for-android/src/src/org/renpy/android/Hardware.java

```
public generic3AxisSensor accelerometerSensor = new generic3AxisSensor(Sensor.TYPE_ACCELEROMETER);
public generic3AxisSensor orientationSensor = new generic3AxisSensor(Sensor.TYPE_ORIENTATION);
public generic3AxisSensor magneticFieldSensor = new generic3AxisSensor(Sensor.TYPE_MAGNETIC_FIELD);

/**
 * functions for backward compatibility reasons
 */

public void accelerometerEnable(boolean enable) { accelerometerSensor.changeStatus(enable); }
public float[] accelerometerReading() { return (float[]) accelerometerSensor.readSensor(); }
public void orientationSensorEnable(boolean enable) { orientationSensor.changeStatus(enable); }
public float[] orientationSensorReading() { return (float[]) orientationSensor.readSensor(); }
public void magneticFieldSensorEnable(boolean enable) { magneticFieldSensor.changeStatus(enable); }
public float[] magneticFieldSensorReading() { return (float[]) magneticFieldSensor.readSensor(); }
```

Compass mit PyJNIus


```
from pyjnius import autoclass
```

```
def __init__(self, **kwargs):
    super(CompassApp, self).__init__(**kwargs)

    Hardware = autoclass('org.renpy.android.Hardware')
    self.hw = Hardware()
```

```
def stopApp(self,*largs):
    """
    Funktion zum Beenden der Anwendung
    """
    self.hw.magneticFieldSensorEnable(False)
```

```
def viewCompass(self, *largs):
    """
    Auslesen der aktuellen Magnetsensordaten und Berechnen der
    Deklination als Winkel zwischen X- und Y-Wert und Übergeben
    des Winkels an die entsprechende Funktion zur Drehung der Kompassnadel
    """
    (x, y, z) = self.hw.magneticFieldSensorReading()
```

Inhalt

- Einführung und generelle Aspekte
- SL4A Grundlagen speziell für Py4A
- SL4A-Py4A API
- SL4A-Py4A Beispiele
- Kivy Grundlagen
- Kivy Python-For-Android
- Kivy PyJNIus
- Kivy Beispiele
- Ausblick

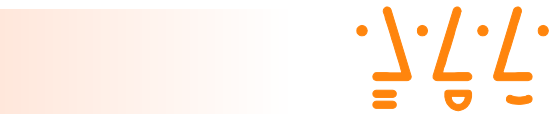
Ausblick

- angedacht ist die weitere Anbindung der Android Java API für Kivy Python-For-Android
- interessant wäre auch eine der Android Java API vergleichbare Android Python API
- … und in ferner Zukunft eine Cross-Plattform übergreifende Python API für die unterschiedlichsten Sensoren und Medien auf unterschiedlichsten Geräten ...
- … wir sollten daran arbeiten …

Danke

- speziellen Dank an
- **Massimo Di Pierro**, dem Ersteller und Pfleger von web2py
- **Mathieu Virbel, Gabriel Pettier, Thomas Hansen** einschliesslich aller weiteren Mitglieder aus dem **Kivy Team**
- **Stefan Behnel** für seine Unterstützung von Cython
- dem gesamten **PyCon DE Team**

Fragen ?



- noch Fragen ?

- http://klever.hs-augsburg.de/Aktuelles

Literatur und zusätzliche Links

- Lucas Jordan & Pieter Greyling „Practical Android Projects“, 2011, Apress
- Chapter 5 „Introducing SL4A: The Scripting Layer for Android“ in Jordan/Greyling „Practical Android Projects“, http://android-scripting.googlecode.com/files/Practical_Android_Projects_Ch05_Introducing_SL4A.pdf
- Paul Ferrill „Pro Android Python with SL4A“, 2011, Apress
- SL4A FAQ, http://code.google.com/p/android-scripting/wiki/FAQ
- Nik Klever, Python On Mobiles – Python For Android, Forschungsbericht 2012, Hochschule Augsburg, 2012