Apps für alle Plattformen entwickeln!

Plattformübergreifende App-Entwicklung mit Python und Kivy

Andreas Schreiber <andreas.schreiber@dlr.de>

FrOSCon
Free and Open Source Software Conference

DLR

Übersicht

- Vorstellung
- Python
- Kivy
- Entwicklungstools
- Demos
- Beschränkungen
- Credits

Vorstellung

Wissenschaftler,
Abteilungsleiter

DLR Deutsches Zentrum für Luft- und Raumfahrt
German Aerospace Center

Gründer,
Geschäftsführer

medando

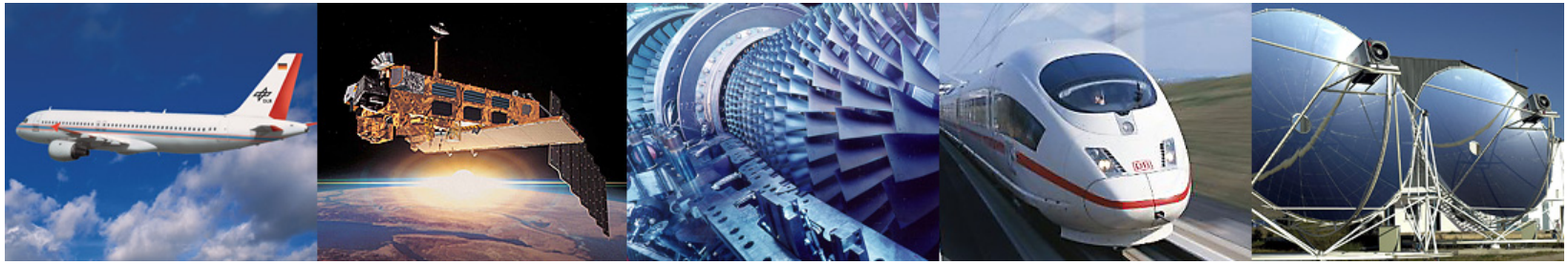
Python-begeistert

Py cologne

PyHPC2013

PyCon. DE 2013

Das DLR
Deutsches Zentrum für Luft- und Raumfahrt



- Forschungseinrichtung
- Raumfahrt-Agentur
- Projektträger

Standorte und Personal

Circa 7.400 Mitarbeiterinnen und Mitarbeiter arbeiten in 32 Instituten und Einrichtungen in

- 16 Standorten.

Büros in Brüssel, Paris, Tokio und Washington.

Python

DLR

Python

- Universelle High-Level Programmiersprache
- Objektorientiert, Aspektorientiert, Funktional
- Dynamische Typisierung
- Einfach zu lernen und klare, übersichtliche Syntax

```
def faculty(x):
    if x > 1:
        return x * faculty(x - 1)
    else:
        return 1
```

Python auf mobilen Geräten

Frühe Technologien

- PyS60 for Symbian
- Python CE for Windows Mobile

Aktuelle Technologien

- Scripting Layer for Android (SL4A)
- Python for Android (Py4A)
- PySide / Qt for Android
- WinRT / IronPython for Windows 8
- Kivy…

Kivy

DLR

Kivy

Plattformübergreifendes Python-Framework

- Android
- iOS
- Meego
- Windows
- Linux
- OS X
- (Raspberry Pi)

kivy.org

Entwicklung in Python auf allen Plattformen – keine Emulation!

Kivy Grundlagen

Framework für *Natural User Interfaces* (NUI)

- Touchscreens / Multi-Touch

GPU-beschleunigte Grafik

- Basiert auf OpenGL ES 2.0

Geeignet für Prototypen und Produkte

- Portierung auf neue Plattformen ist leicht

Kivy Software

Open Source (LGPL), 7 Core Developer

Source Code: `github.com/kivy`

Dokumentation: `kivy.org/docs`

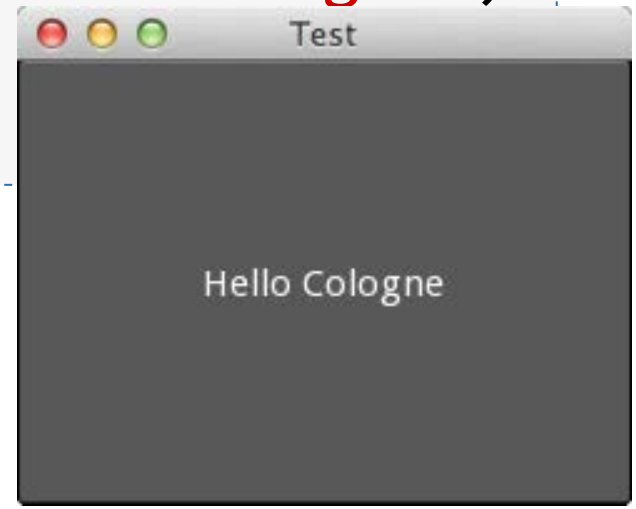
Kivy im Google Play Store:
`play.google.com/store/apps/details?id=org.kivy.pygame`

Kivy sagt Hallo!

```
from kivy.app import App
from kivy.uix.button import Button

class TestApp(App):
    def build(self):
        return Button(text='Hello Cologne')

TestApp().run()
```




Hello St. Augustin

Entwicklung mit Kivy

Python für Widgets, I/O und Programmlogik

Sprache KV für Layout und Grafik

Cython für Low-level-Zugriff auf Grafikroutinen

Widgets

Verfügbare Standard-Widgets

- Label, Button, CheckBox, Image, Slider, ProgressBar, Switch, Texteingabe
- Fünf Layouts: Grid, Box, Anchor, Float, Stack
- Komplexe Widgets: Bubble, Dropdown-Liste, Filechooser, Popup, Spinner, ListView, TabbedPanel, ...
- Screen manager und Screen-Übergänge
- Interaktionen: Scatter, Zoom, Pan, Rotate

Kivy Language (KV)

Leicht(er)es Erstellen des Benutzerinterfaces

Drei Typen

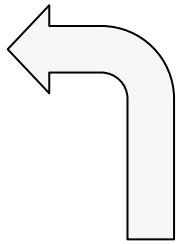
- KV Rule: „Style„, das auf Widget-Klasse angewendet wird
- Root-Widget: Erzeugt ein neues Widget
- Dynamische Klassen: Erzeugt Widgets „on-the-fly“

“Hello St. Augustin” mit KV

```
from kivy.app import App

class HelloApp(App):
    pass

HelloApp().run()
```



Datei **hello.kv**
definiert Root-Widget

```
#:kivy 1.0

Button:
    text: ‘Hello St. Augustin’
```

Beispiel: Pong

```
import kivy
from kivy.app import App
from kivy.uix.widget import Widget

class PongGame(Widget):
    pass

class PongApp(App):
    def build(self):
        return PongGame()

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

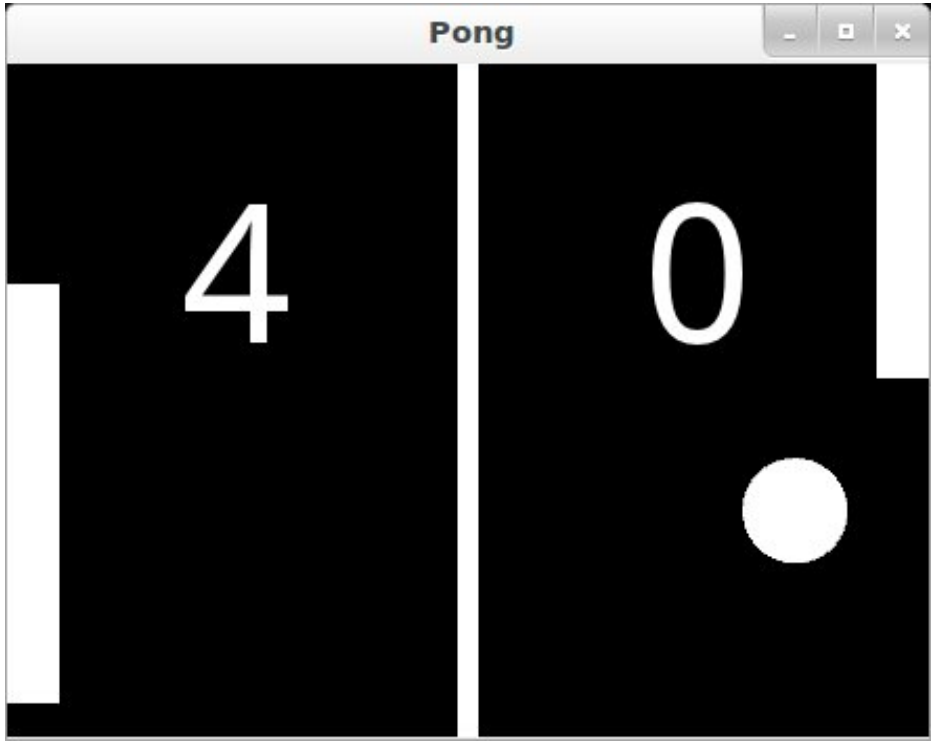
Pong Grafik

```
#:kivy 1.7.1
<PongGame>:
    canvas:
        Rectangle:
            pos: self.center_x - 5, 0
            size: 10, self.height

    Label:
        font_size: 70
        center_x: root.width / 4
        top: root.top - 50
        text: "0"

    Label:
        font_size: 70
        center_x: root.width * 3 / 4
        top: root.top - 50
        text: "0"
```

Pong



Ganzes Beispiel: **`http://kivy.org/docs/tutorials/pong.html`**

Entwicklungstools

Entwicklungstools

Tools für Android

- Python-for-android
- Cross compiler für ARM
- Android SDK & NDK
- Python und einige Python-Msodule

Tools für iOS

- Kivy-iOS
- Generierung eines Xcode-Projekts

Verstecken der Komplexität

Buildozer

Erledigt die einzelnen Schritte:

- Download
- Compilation
- Packaging

https://github.com/kivy/buildozer

```
% buildozer android debug deploy run
```

Zugriff auf APIs der Geräte

Geräte-APIs

- Kamera, Kompass, Beschleunigung, Location, …

Zugriff aus Kivy- bzw. Python-Anwendungen

- **PyJNIus** – Zugriff auf Java-Klassen von Python
- **PyOBJus** – Zugriff auf Objective-C von Python
- **Plyer** – Plattformunabhängiger Wrapper der APIs

Zugriff auf Java-Klassen von Python

PyJNIus

- Implementiert mit JNI und Java-Reflection
- **https://github.com/kivy/pyjnius**

```
from jnius import autoclass

Hardware = autoclass('org.renpy.android.Hardware')
print 'DPI is', Hardware.getDPI()
```

Wrapper für plattformspezifische APIs

Plyer

- Noch sehr frisch!
- Benutzt PyJNIus bzw. PyOBJus

- **https://github.com/kivy/plyer**

```
from plyer import notification

notification.notify(title='Hello', message='World')
```

Demos

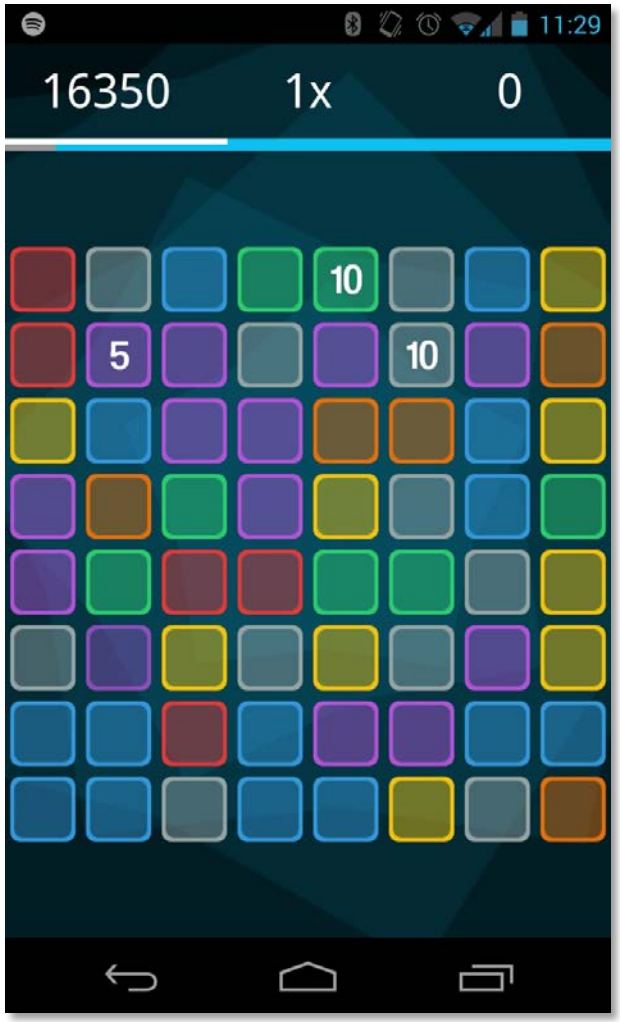
DLR

Kivy Pictures

ANDROID APP BEI Google play

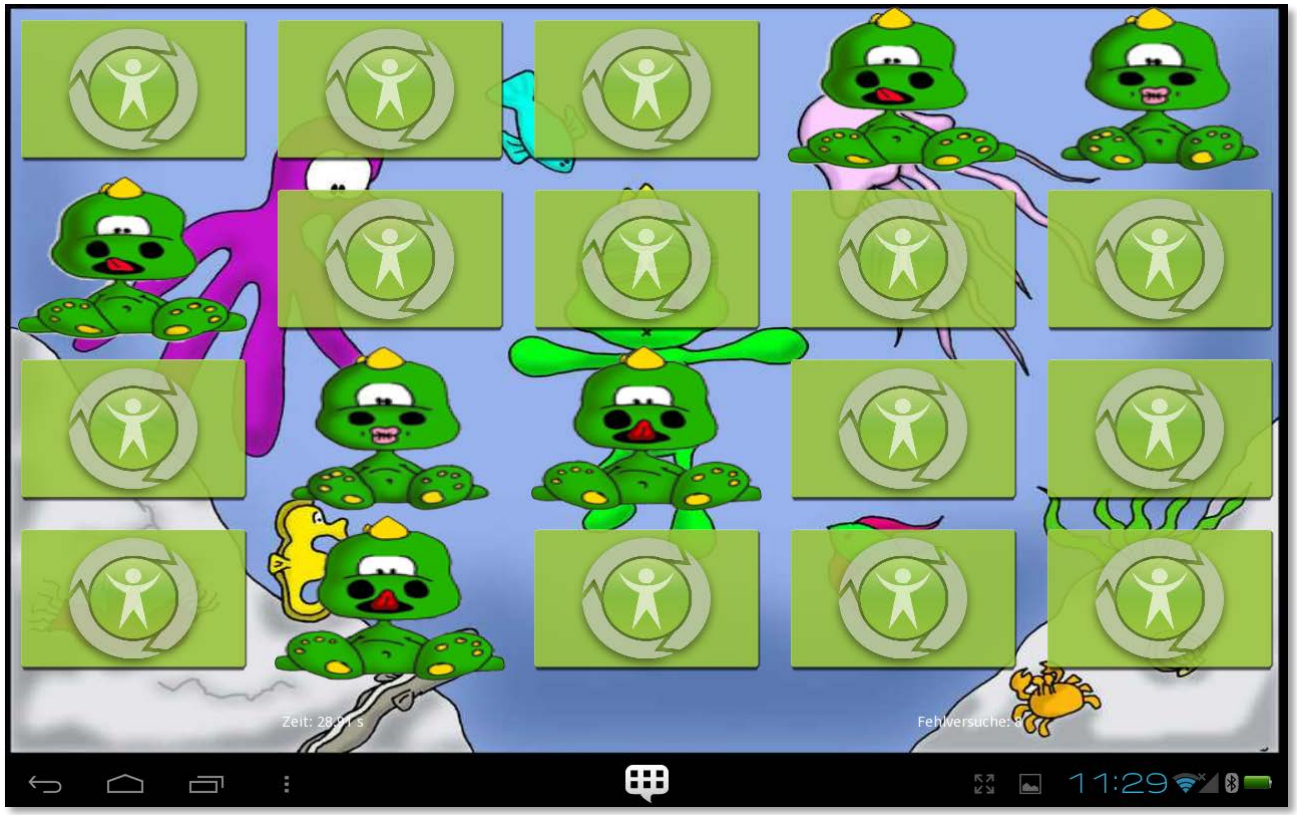
Flat Jewels

ANDROID APP BEI Google play



Kleiner Drache Luki

medando



DLR

Kleiner Drache Luki

medando

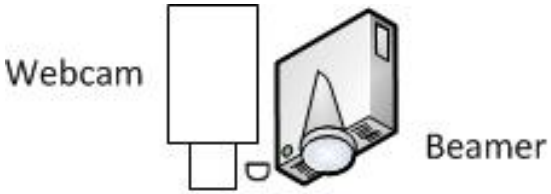
MQTT Client

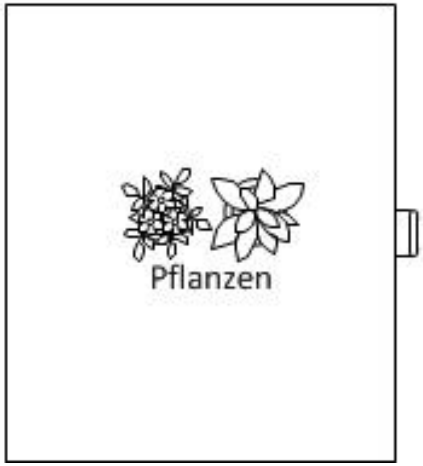

medando messaging 0.1
MQTT Topics
$SYS/broker/version 0
$SYS/broker/timestamp 0
$SYS/broker/changeset 0
$SYS/broker/uptime 0
$SYS/broker/messages/stored 0
$SYS/broker/messages/received 0
$SYS/broker/messages/sent 0
$SYS/broker/messages/dropped 0
$SYS/broker/subscriptions/count 0
$SYS/broker/retained messages/count 0
$SYS/broker/clients/total 0
$SYS/broker/clients/inactive 0
$SYS/broker/clients/active 0
$SYS/broker/clients/maximum 0
$SYS/broker/clients/expired 0
$SYS/broker/heap/current size 0
$SYS/broker/heap/maximum size 0
$SYS/broker/publish/messages/received 0
$SYS/broker/publish/messages/sent 0
$SYS/broker/publish/bytes/received 0
Test Messages
Send test callback!

Anwendung aus der Raumfahrtbiologie

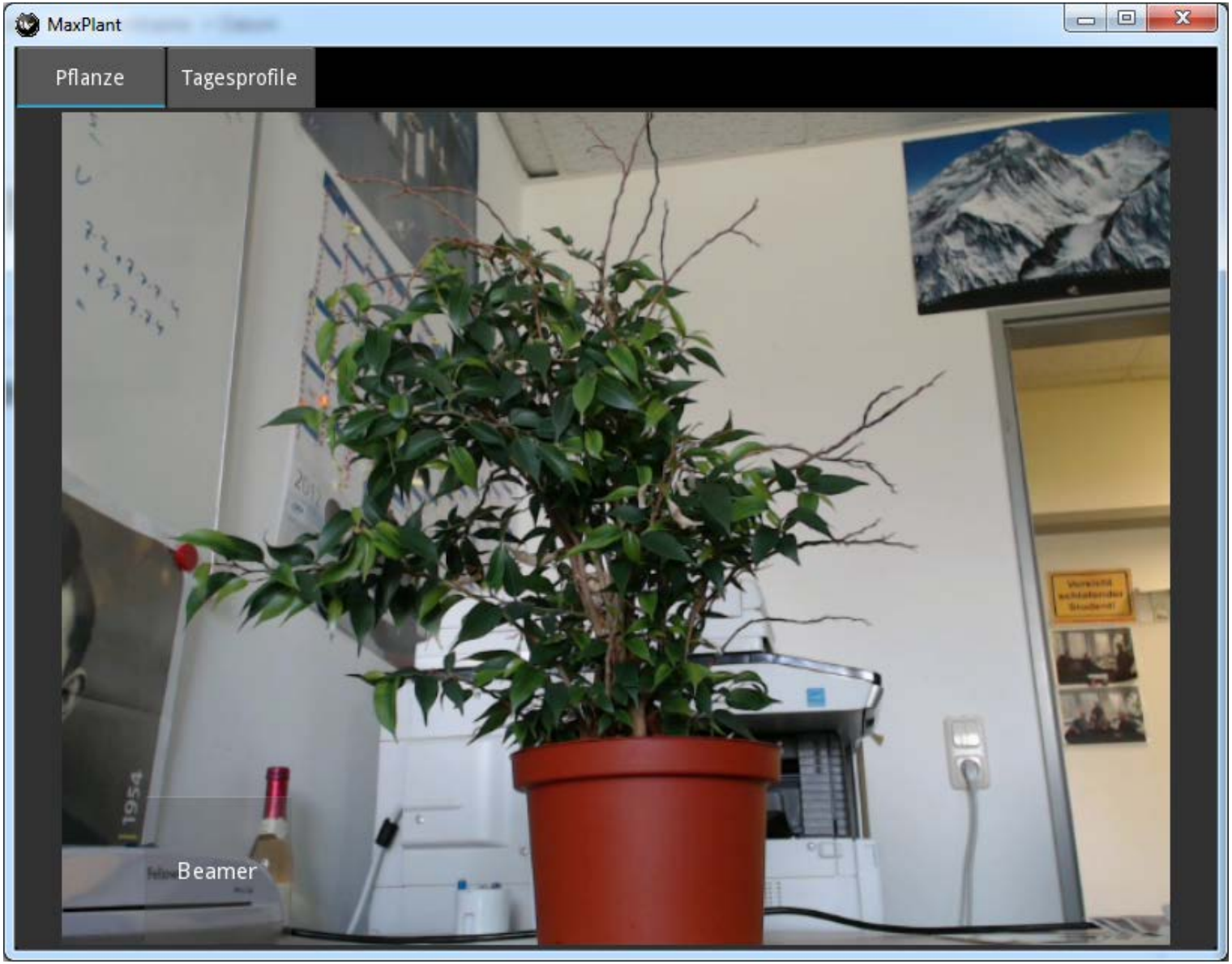
Pflanzenbeleuchtung

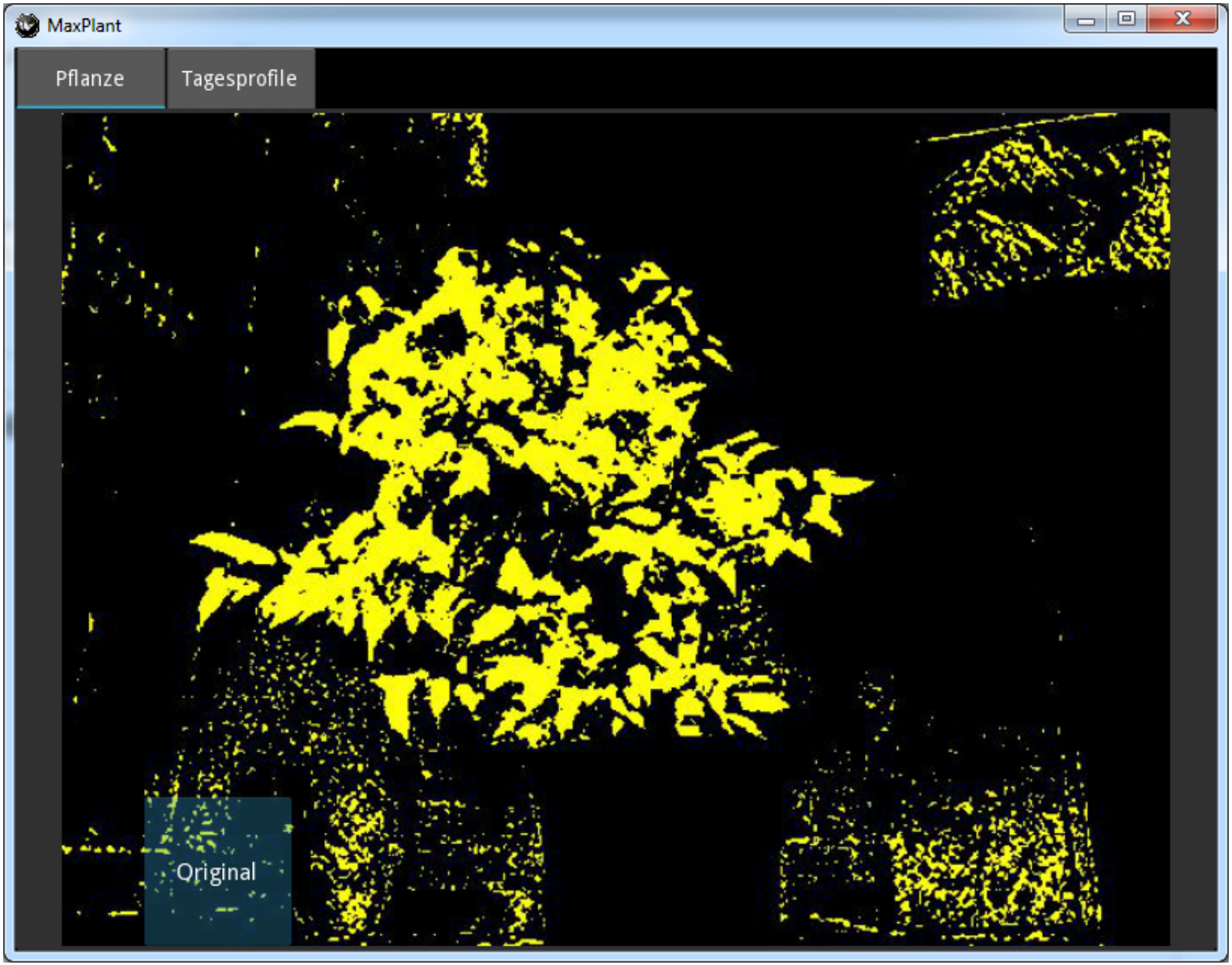
- Webcam nimmt Bild auf
- Rechner erkennt Pflanze
- Rechner berechnet anhand von Einstellungen ein Ausgabebild
- Lichtquelle (z.B. Beamer) beleuchtet die Pflanze mit dem Bild

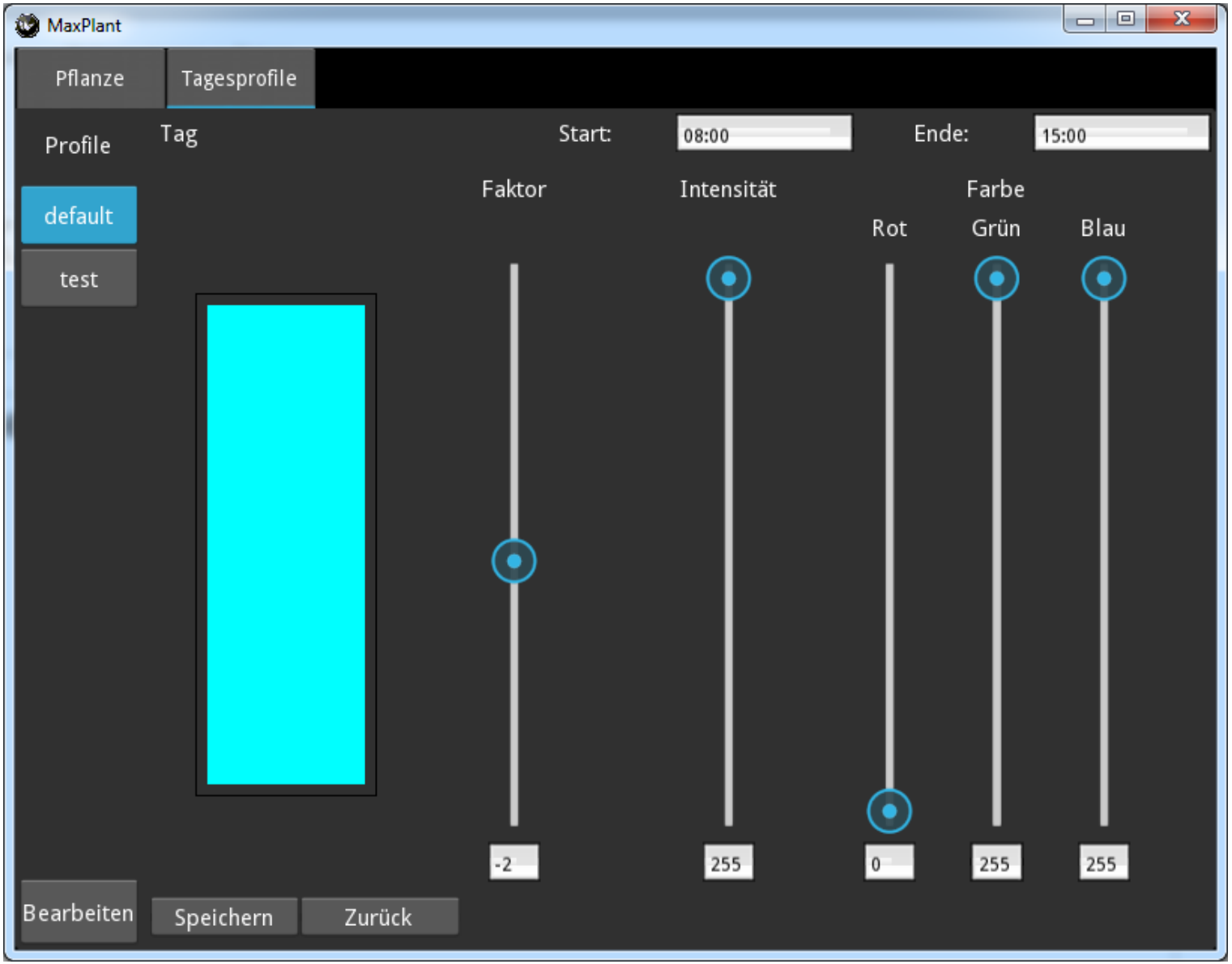




Infos: C.R.O.P.

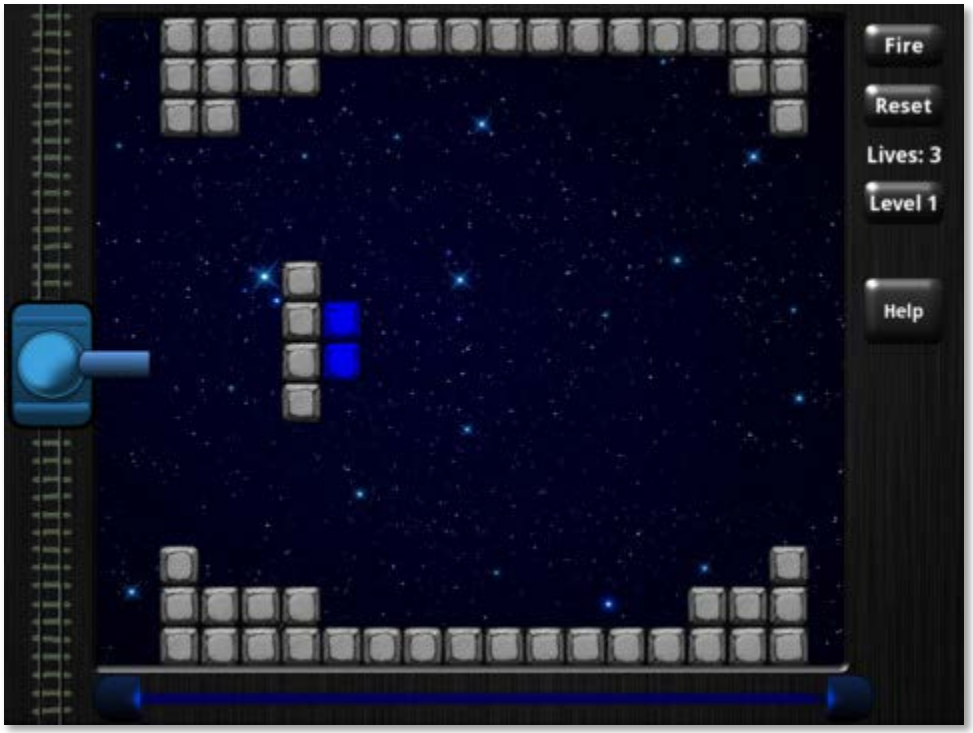

MaxPlant
Pflanze
Tagesprofile
Beamer


MaxPlant
Pflanze
Tagesprofile
Original

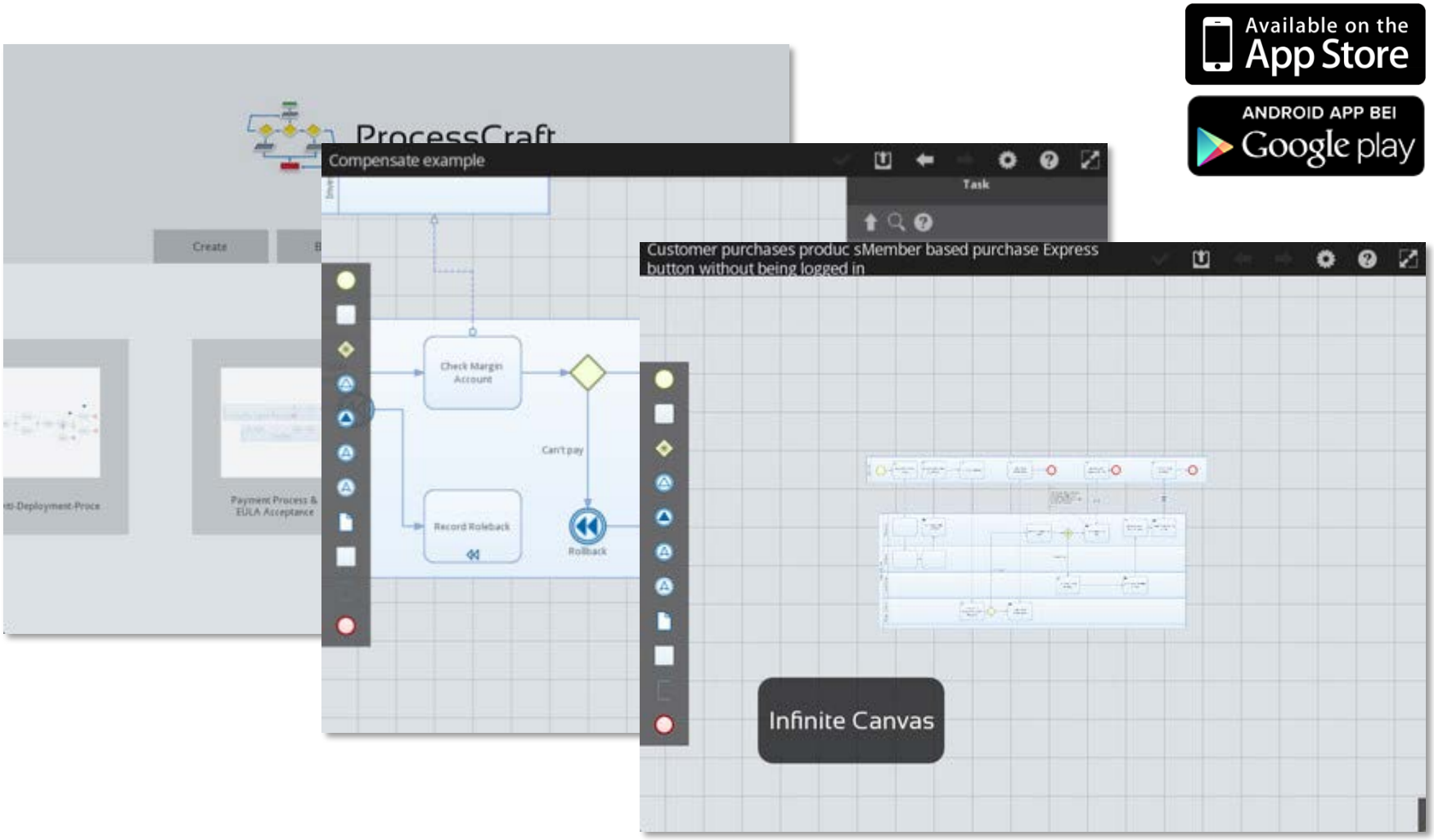

MaxPlant
Pflanze
Tagesprofile
Profile
Tag
Start:
08:00
Ende:
15:00
default
test
Faktor
Intensität
Farbe
Rot
Grün
Blau
-2
255
0
255
255
Bearbeiten
Speichern
Zurück

Deflectouch (iOS)

Available on the App Store



ProcessCraft (Android, iOS; kommerziell)



Beschränkungen

Was fehlt?

User Interface Designer

- Design-Werkzeug für die Kivy Language KV
- Verschiedene Anläufe bei GSoC

Was fehlt?

Abstraktion der mobile APIs

- Plattformunabhängige Wrapper für plattformspezifische APIs (Android, iOS, Linux/Mac/Windows, …)
- Project **Plyer** seit kurzem gestartet

Was fehlt?

Portierung auf Raspberry Pi

- Nützlich für günstige Stand-alone Anwendungen
- Finanzierung via Crowdsourcing (**`bountysource.com`**)

Credits

Thanks to the Kivy developers

- Mathieu Virbel (**@mathieuvirbel**)
- Thomas Hansen (**@hansent**)
- Gabriel Pettier (**@tshirtman**)
- and many others

Hinweis

PyCon.DE 2013

- 3. Deutsche Python-Konferenz
- KOMED, MediaPark, Köln
- 14.-19. Oktober 2013

PyCon. DE 2013

de.pycon.org

cologne

DLR Deutsches Zentrum für Luft- und Raumfahrt
German Aerospace Center

Python Software Verband e.V.

DLR

Fragen?

Zusammenfassung

- Kivy erlaubt plattformunabhängige Entwicklung von Apps für Android, iOS, Meego, Windows, OSX and Linux
- Geeignet für Multi-touch- und Grafik-Anwendungen (Kiosksysteme, Ausstellungen, Spiele, …)

Andreas Schreiber
Twitter: @onyame
http://www.dlr.de/sc